A NEURAL STOCHASTIC VOLATILITY MODEL

Rui Luo[†], **Xiaojun Xu**[‡], **Weinan Zhang**[‡], **Jun Wang**[†]
[†]University College London, [‡]Shanghai Jiao Tong University
{r.luo,j.wang}@cs.ucl.ac.uk, {xuxj,wnzhang}@apex.sjtu.edu.cn

ABSTRACT

In this paper, we show that the recent integration of statistical models with recurrent neural networks provides a new way of formulating volatility models that have been popular in time series analysis and prediction. The model comprises a pair of complementary stochastic recurrent neural networks: the generative network models the joint distribution of the stochastic volatility process; the inference network approximates the conditional distribution of the latent variables given the observable ones. Our focus in this paper is on the formulation of temporal dynamics of volatility over time under a stochastic recurrent neural network framework. Our derivations show that some popular volatility models are a special case of our proposed neural stochastic volatility model. Experiments demonstrate that the proposed model generates a smoother volatility estimation, and outperforms standard econometric models GARCH, EGARCH, GJR-GARCH and some other GARCH variants as well as MCMC-based model *stochvol* and a recent Gaussian processes based volatility model *GPVOL* on several metrics about the fitness of the volatility modelling and the accuracy of the prediction.

1 INTRODUCTION

The volatility of the price movements reflects the ubiquitous uncertainty within financial markets. It is critical that the level of risk, indicated by volatility, is taken into consideration before investment decisions are made and portfolio are optimised (Hull, 2006); volatility is substantially a key variable in the pricing of derivative securities. Hence, estimating and forecasting volatility is of great importance in branches of financial studies, including investment, risk management, security valuation and monetary policy making (Poon & Granger, 2003).

Volatility is measured typically by using the standard deviation of price change in a fixed time interval, such as a day, a month or a year. The higher the volatility, the riskier the asset. One of the primary challenges in designing volatility models is to identify the existence of latent (stochastic) variables or processes and to characterise the underlying dependences or interactions between variables within a certain time span. A classic approach has been to handcraft the characteristic features of volatility models by imposing assumptions and constraints, given prior knowledge and observations. Notable examples include autoregressive conditional heteroskedasticity (ARCH) model (Engle, 1982) and its generalisation GARCH (Bollerslev, 1986), which makes use of autoregression to capture the properties of time-variant volatility within many time series. Heston (1993) assumed that the volatility follows a Cox-Ingersoll-Ross (CIR) process (Cox et al., 1985) and derived a closed-form solution for options pricing. While theoretically sound, those approaches require strong assumptions which might involve complex probability distributions and non-linear dynamics that drive the process, and in practice, one may have to impose less prior knowledge and rectify a solution under the worst-case volatility case (Avellaneda & Paras, 1996).

In this paper, we take a fully data driven approach and determine the configurations with as few exogenous input as possible, or even purely from the historical data. We propose a neural network re-formulation of stochastic volatility by leveraging stochastic models and recurrent neural networks (RNNs). We are inspired by the recent development on variational approaches of stochastic (deep) neural networks (Kingma & Welling, 2013; Rezende et al., 2014) to a recurrent case (Chung et al., 2015; Fabius & van Amersfoort, 2014; Bayer & Osendorfer, 2014), and our formulation shows that existing volatility models such as the GARCH (Bollerslev, 1986) and the Heston model (Heston, 1993) are the special cases of our neural stochastic volatility formulation. With the hidden latent

variables in the neural networks we naturally uncover the underlying stochastic process formulated from the models.

Experiments with synthetic data and real-world financial data are performed, showing that the proposed model outperforms the widely-used GARCH model on several metrics of the fitness and the accuracy of time series modelling and prediction: it verifies our model's high flexibility and rich expressive power.

2 Related Work

A notable volatility method is autoregressive conditional heteroskedasticity (ARCH) model (Engle, 1982): it can accurately capture the properties of time-variant volatility within many types of time series. Inspired by ARCH model, a large body of diverse work based on stochastic process for volatility modelling has emerged. Bollerslev (1986) generalised ARCH model to the generalised autoregressive conditional heteroskedasticity (GARCH) model in a manner analogous to the extension from autoregressive (AR) model to autoregressive moving average (ARMA) model by introducing the past conditional variances in the current conditional variance estimation. Engle & Kroner (1995) presented theoretical results on the formulation and estimation of multivariate GARCH model within simultaneous equations systems. The extension to multivariate model allows the covariances to present and depend on the historical information, which are particularly useful in multivariate financial models. Heston (1993) derived a closed-form solution for option pricing with stochastic volatility where the volatility process is a CIR process driven by a latent Wiener process such that the current volatility is no longer a deterministic function even if the historical information is provided. Notably, empirical evidences have confirmed that volatility models provide accurate forecasts (Andersen & Bollerslev, 1998) and models such as ARCH and its descendants/variants have become indispensable tools in asset pricing and risk evaluation.

On the other hand, deep learning (LeCun et al., 2015; Schmidhuber, 2015) that utilises nonlinear structures known as deep neural networks, powers various applications. It has triumph over pattern recognition challenges, such as image recognition (Krizhevsky et al., 2012; He et al., 2015; van den Oord et al., 2016), speech recognition (Hinton et al., 2012; Graves et al., 2013; Chorowski et al., 2015), machine translation (Sutskever et al., 2014; Cho et al., 2014; Bahdanau et al., 2014; Luong et al., 2015) to name a few.

Time-dependent neural networks models include RNNs with advanced neuron structure such as long short-term memory (LSTM) (Hochreiter & Schmidhuber, 1997), gated recurrent unit (GRU) (Cho et al., 2014), and bidirectional RNN (BRNN) (Schuster & Paliwal, 1997). Recent results show that RNNs excel for sequence modelling and generation in various applications (Graves, 2013; Gregor et al., 2015). However, despite its capability as non-linear universal approximator, one of the drawbacks of neural networks is its deterministic nature. Adding latent variables and their processes into neural networks would easily make the posterori computationally intractable. Recent work shows that efficient inference can be found by variational inference when hidden continuous variables are embedded into the neural networks structure (Kingma & Welling, 2013; Rezende et al., 2014). Some early work has started to explore the use of variational inference to make RNNs stochastic (Chung et al., 2015; Bayer & Osendorfer, 2014; Fabius & van Amersfoort, 2014). Bayer & Osendorfer (2014) and Fabius & van Amersfoort (2014) considered the hidden variables are independent between times, whereas (Fraccaro et al., 2016) utilised a backward propagating inference network according to its Markovian properties. Our work in this paper extends the work (Chung et al., 2015) with a focus on volatility modelling for time series. We assume that the hidden stochastic variables follow a Gaussian autoregression process, which is then used to model both the variance and the mean. We show that the neural network formulation is a general one, which covers two major financial stochastic volatility models as the special cases by defining the specific hidden variables and non-linear transforms.

3 Preliminary: Volatility Models

Stochastic processes are often defined by stochastic differential equations (SDEs), e.g. a (univariate) generalised Wiener process is $\mathrm{d}\,x_t = \mu\,\mathrm{d}\,t + \sigma\,\mathrm{d}\,w_t$, where μ and σ denote the time-invariant rates of drift and standard deviation (square root of variance) while $\mathrm{d}\,w_t \sim \mathcal{N}(0, \mathrm{d}\,t)$ is the increment of

standard Wiener process at time t. In a small time interval between t and $t + \Delta t$, the change in the variable is $\Delta x_t = \mu \Delta t + \sigma \Delta w_t$. Let $\Delta t = 1$, we obtain the discrete-time version of basic volatility model:

$$x_t = x_{t-1} + \mu + \sigma \epsilon_t, \tag{1}$$

where $\epsilon_t \sim \mathcal{N}(0, 1)$ is a sample drawn from standard normal distribution. In the multivariate case, $\boldsymbol{\Sigma}$ represents the covariance matrix in place of σ^2. As presumed that the variables are multidimensional, we will use $\boldsymbol{\Sigma}$ to represent variance in general case except explicitly noted.

3.1 Deterministic volatility

The time-invariant variance $\boldsymbol{\Sigma}$ can be extended to be a function $\boldsymbol{\Sigma}_t = \boldsymbol{\Sigma}(\boldsymbol{x}_{<t})$ relying on history of the (observable) underlying stochastic process $\{\boldsymbol{x}_{<t}\}$. The current variance $\boldsymbol{\Sigma}_t$ is therefore determined given the history $\{\boldsymbol{x}_{<t}\}$ up to time t. An example of such extensions is the univariate GARCH(1,1) model (Bollerslev, 1986):

$$\sigma_t^2 = \alpha_0 + \alpha_1 (x_{t-1} - \mu_{t-1})^2 + \beta_1 \sigma_{t-1}^2, \tag{2}$$

where x_{t-1} is the observation from $\mathcal{N}(\mu_{t-1}, \sigma_{t-1}^2)$ at time $t - 1$. Note that the determinism is in a conditional sense, which means that it only holds under the condition that the complete history $\{\boldsymbol{x}_{<t}\}$ is presented, such as the case of 1-step-ahead forecast. otherwise the current volatility would still be stochastic as it is built on stochastic process $\{\boldsymbol{x}_t\}$. However, for multi-step-ahead forecast, we usually exploit the relation $\mathbb{E}_{t-1}[(x_t - \mu_t)^2] = \sigma_t^2$ to substitute the corresponding terms and calculate the forecasts with longer horizon in a recursive fashion, for example, $\sigma_{t+1}^2 = \alpha_0 + \alpha_1 \mathbb{E}_{t-1}[(x_t - \mu_t)^2] + \beta_1 \sigma_t^2 = \alpha_0 + (\alpha_1 + \beta_1)\sigma_t^2$. For n-step-ahead forecast, there will be n iterations and the procedure is hence also deterministic.

3.2 Stochastic volatility

Another extension is applicable for $\boldsymbol{\Sigma}_t$ from being conditionally deterministic (i.e. deterministic given the complete history $\{\boldsymbol{x}_{<t}\}$) to fully stochastic: $\boldsymbol{\Sigma}_t = \boldsymbol{\Sigma}(\boldsymbol{z}_{\leq t})$ is driven by another latent stochastic process $\{\boldsymbol{z}_t\}$ instead of the observable process $\{\boldsymbol{x}_t\}$. Heston (1993) model instantiates a continuous-time stochastic volatility model for univariate processes:

$$\mathrm{d}\, x_t = (\mu - 0.5\sigma_t^2)\,\mathrm{d}\, t + \sigma_t\,\mathrm{d}\, w_t^{\langle 1 \rangle}, \tag{3}$$

$$\mathrm{d}\, \sigma_t = a\sigma_t\,\mathrm{d}\, t + b\,\mathrm{d}\, w_t^{\langle 2 \rangle}, \tag{4}$$

where the correlation between $\mathrm{d}\, w_t^{\langle 1 \rangle}$ and $\mathrm{d}\, w_t^{\langle 2 \rangle}$ applies: $\mathbb{E}[\mathrm{d}\, w_t^{\langle 1 \rangle} \cdot \mathrm{d}\, w_t^{\langle 2 \rangle}] = \rho\,\mathrm{d}\, t$. We apply Euler's scheme of quantisation (Stoer & Bulirsch, 2013) to obtain the discrete analogue to the continuous-time Heston model (Eqs. (3) and (4)):

$$\begin{aligned} x_t &= (x_{t-1} + \mu - 0.5\sigma_t^2) + \sigma\epsilon_t \\ \sigma_t &= (1 + a)\sigma_{t-1} + bz_t \end{aligned} \quad \text{where} \quad \begin{bmatrix} \epsilon_t \\ z_t \end{bmatrix} = \mathcal{N}(\mathbf{0}, \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix}). \tag{5}$$

3.3 Volatility model in general

As discussed above, the observable variable $\boldsymbol{x}_t$ follows Gaussian distribution of which the mean and variance depend on the history of observable process $\{\boldsymbol{x}_t\}$ and latent $\{\boldsymbol{z}_t\}$. We presume in addition that the latent process $\{\boldsymbol{z}_t\}$ is an autoregressive model such that $\boldsymbol{z}_t$ is (conditionally) Gaussian distributed. Therefore, we formulate the volatility model in general as:

$$\boldsymbol{z}_t \sim \mathcal{N}(\boldsymbol{\mu}^z(\boldsymbol{z}_{<t}), \boldsymbol{\Sigma}^z(\boldsymbol{z}_{<t})), \tag{6}$$

$$\boldsymbol{x}_t \sim \mathcal{N}(\boldsymbol{\mu}^x(\boldsymbol{x}_{<t}, \boldsymbol{z}_{\leq t}), \boldsymbol{\Sigma}^x(\boldsymbol{x}_{<t}, \boldsymbol{z}_{\leq t})), \tag{7}$$

where $\boldsymbol{\mu}^z(\boldsymbol{x}_{<t}, \boldsymbol{z}_{\leq t})$ and $\boldsymbol{\Sigma}^z(\boldsymbol{x}_{<t}, \boldsymbol{z}_{\leq t})$ denote the autoregressive time-varying mean and variance of the latent variable $\boldsymbol{z}_t$ while $\boldsymbol{\mu}^x(\boldsymbol{x}_{<t}, \boldsymbol{z}_{\leq t})$ and $\boldsymbol{\Sigma}^x(\boldsymbol{x}_{<t}, \boldsymbol{z}_{\leq t})$ represent the mean and variance of observable variable $\boldsymbol{x}_t$, which depend on not only history of the observable process $\{\boldsymbol{x}_{<t}\}$ but that of the latent process $\{\boldsymbol{z}_{\leq t}\}$.

These two formulas (Eqs. (6) and (7)) abstract the generalised formulation of volatility models. Together, they represents a broad family of volatility models with latent variables, where the Heston

model for stochastic volatility is merely a special case of the family. Furthermore, it will degenerate to deterministic volatility models such as the well-studied GARCH model if we disable the latent process.

4 NEURAL STOCHASTIC VOLATILITY MODELS

In this section, we establish the *neural stochastic volatility model* (NSVM) for stochastic volatility estimation and forecast.

4.1 GENERATING OBSERVABLE SEQUENCE

Recall that the latent variable $\boldsymbol{z}_t$ (Eq. (6)) and the observable $\boldsymbol{x}_t$ (Eq. (7)) are described by autoregressive models ($\boldsymbol{x}_t$ has the exogenous input $\{\boldsymbol{z}_{\leq t}\}$.) For the distributions of $\{\boldsymbol{z}_t\}$ and $\{\boldsymbol{x}_t\}$, the following factorisation applies:

$$p_{\boldsymbol{\Phi}}(\boldsymbol{Z}) = \prod_t p_{\boldsymbol{\Phi}}(\boldsymbol{z}_t|\boldsymbol{z}_{<t}) = \prod_t \mathcal{N}(\boldsymbol{z}_t; \boldsymbol{\mu}^z_{\boldsymbol{\Phi}}(\boldsymbol{z}_{<t}), \boldsymbol{\Sigma}^z_{\boldsymbol{\Phi}}(\boldsymbol{z}_{<t})), \tag{8}$$

$$p_{\boldsymbol{\Phi}}(\boldsymbol{X}|\boldsymbol{Z}) = \prod_t p_{\boldsymbol{\Phi}}(\boldsymbol{x}_t|\boldsymbol{x}_{<t}, \boldsymbol{z}_{\leq t}) = \prod_t \mathcal{N}(\boldsymbol{x}_t; \boldsymbol{\mu}^x_{\boldsymbol{\Phi}}(\boldsymbol{x}_{<t}, \boldsymbol{z}_{\leq t}), \boldsymbol{\Sigma}^x_{\boldsymbol{\Phi}}(\boldsymbol{x}_{<t}, \boldsymbol{z}_{\leq t})), \tag{9}$$

where $\boldsymbol{X} = \{\boldsymbol{x}_t\}$ and $\boldsymbol{Z} = \{\boldsymbol{z}_t\}$ are the sequences of observable and latent variables, respectively, while $\boldsymbol{\Phi}$ represents the parameter set of the model. The full generative model is defined as the joint distribution:

$$\begin{aligned} p_{\boldsymbol{\Phi}}(\boldsymbol{X}, \boldsymbol{Z}) &= \prod_t p_{\boldsymbol{\Phi}}(\boldsymbol{x}_t|\boldsymbol{x}_{<t}, \boldsymbol{z}_{\leq t}) p_{\boldsymbol{\Phi}}(\boldsymbol{z}_t|\boldsymbol{z}_{<t}) \\ &= \prod_t \mathcal{N}(\boldsymbol{z}_t; \boldsymbol{\mu}^z_{\boldsymbol{\Phi}}(\boldsymbol{z}_{<t}), \boldsymbol{\Sigma}^z_{\boldsymbol{\Phi}}(\boldsymbol{z}_{<t})) \mathcal{N}(\boldsymbol{x}_t; \boldsymbol{\mu}^x_{\boldsymbol{\Phi}}(\boldsymbol{x}_{<t}, \boldsymbol{z}_{\leq t}), \boldsymbol{\Sigma}^x_{\boldsymbol{\Phi}}(\boldsymbol{x}_{<t}, \boldsymbol{z}_{\leq t})). \end{aligned} \tag{10}$$

It is observed that the means and variances are conditionally deterministic: given the historical information $\{\boldsymbol{z}_{<t}\}$, the current mean $\boldsymbol{\mu}^z_t = \boldsymbol{\mu}^z_{\boldsymbol{\Phi}}(\boldsymbol{z}_{<t})$ and variance $\boldsymbol{\Sigma}^z_t = \boldsymbol{\Sigma}^z_{\boldsymbol{\Phi}}(\boldsymbol{z}_{<t})$ of $\boldsymbol{z}_t$ is obtained and hence the distribution $\mathcal{N}(\boldsymbol{z}_t; \boldsymbol{\mu}^z_t, \boldsymbol{\Sigma}^z_t)$ of $\boldsymbol{z}_t$ is specified; after sampling $\boldsymbol{z}_t$ from the specified distribution, we incorporate $\{\boldsymbol{x}_{<t}\}$ and calculate the current mean $\boldsymbol{\mu}^x_t = \boldsymbol{\mu}^x_{\boldsymbol{\Phi}}(\boldsymbol{x}_{<t}, \boldsymbol{z}_{\leq t})$ and variance $\boldsymbol{\Sigma}^x_t = \boldsymbol{\Sigma}^x_{\boldsymbol{\Phi}}(\boldsymbol{x}_{<t}, \boldsymbol{z}_{\leq t})$ of $\boldsymbol{x}_t$ and determine its distribution $\mathcal{N}(\boldsymbol{x}_t; \boldsymbol{\mu}^x_t, \boldsymbol{\Sigma}^x_t)$ of $\boldsymbol{x}_t$. It is natural and convenient to present such a procedure in a recurrent fashion because of its autoregressive nature. As is known that RNNs can essentially approximate arbitrary function of recurrent form (Hammer, 2000), the means and variances, which may be driven by complex non-linear dynamics, can be efficiently computed using RNNs.

It is always a good practice to reparameterise the random variables before we go into RNN architecture. As the covariance matrix $\boldsymbol{\Sigma}$ is symmetric and positive definite, it can be factorised as $\boldsymbol{\Sigma} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^\top$, where $\boldsymbol{\Lambda}$ is a full-rank diagonal matrix with positive diagonal elements. Let $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Lambda}^{\frac{1}{2}}$, we have $\boldsymbol{\Sigma} = \boldsymbol{A}\boldsymbol{A}^\top$. Hence we can reparameterise the latent variable $\boldsymbol{z}_t$ (Eq. (6)) and observable $\boldsymbol{x}_t$ (Eq. (7)):

$$\boldsymbol{z}_t = \boldsymbol{\mu}^z_t + \boldsymbol{A}^z_t \boldsymbol{\epsilon}^z_t, \tag{11}$$

$$\boldsymbol{x}_t = \boldsymbol{\mu}^x_t + \boldsymbol{A}^x_t \boldsymbol{\epsilon}^x_t, \tag{12}$$

where $\boldsymbol{A}^z_t(\boldsymbol{A}^z_t)^\top = \boldsymbol{\Sigma}^z_t, \boldsymbol{A}^x_t(\boldsymbol{A}^x_t)^\top = \boldsymbol{\Sigma}^x_t$ and $\boldsymbol{\epsilon}^x_t \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_x), \boldsymbol{\epsilon}^z_t \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_z)$ are auxiliary variables. Note that the randomness within the variables of interest (e.g. $\boldsymbol{z}_t$) is extracted by the auxiliary variables (e.g. $\boldsymbol{\epsilon}_t$) which follow the standard distributions. Hence, the reparameterisation guarantees that gradient-based methods can be applied in learning phase (Kingma & Welling, 2013).

In this paper, the joint generative model is comprised of two sets of RNN and multilayer perceptron (MLP): $\mathrm{RNN}^z_g/\mathrm{MLP}^z_g$ for the latent variable, while $\mathrm{RNN}^x_g/\mathrm{MLP}^z_g$ for the observables. We stack these two RNN/MLP together according to the causal dependency between those variables. The

joint generative model is implemented as the generative network:

$$\{\boldsymbol{\mu}_t^z, \boldsymbol{A}_t^z\} = \text{MLP}_g^z(\boldsymbol{h}_t^z; \boldsymbol{\Phi}), \tag{13}$$

$$\boldsymbol{h}_t^z = \text{RNN}_g^z(\boldsymbol{h}_{t-1}^z, \boldsymbol{z}_{t-1}; \boldsymbol{\Phi}), \tag{14}$$

$$\boldsymbol{z}_t = \boldsymbol{\mu}_t^z + \boldsymbol{A}_t^z \boldsymbol{\epsilon}_t^z, \tag{15}$$

$$\{\boldsymbol{\mu}_t^x, \boldsymbol{A}_t^x\} = \text{MLP}_g^x(\boldsymbol{h}_t^x; \boldsymbol{\Phi}), \tag{16}$$

$$\boldsymbol{h}_t^x = \text{RNN}_g^x(\boldsymbol{h}_{t-1}^x, \boldsymbol{x}_{t-1}, \boldsymbol{z}_t; \boldsymbol{\Phi}), \tag{17}$$

$$\boldsymbol{x}_t = \boldsymbol{\mu}_t^x + \boldsymbol{A}_t^x \boldsymbol{\epsilon}_t^x, \tag{18}$$

where $\boldsymbol{h}_t^z$ and $\boldsymbol{h}_t^x$ denote the hidden states of the corresponding RNNs. The MLPs map the hidden states of RNNs into the means and deviations of variables of interest. The parameter set $\boldsymbol{\Phi}$ is comprised of the weights of RNNs and MLPs.

One should notice that when the latent variable $\boldsymbol{z}$ is obtained, e.g. by inference (details in the next subsection), the conditional distribution $p_{\boldsymbol{\Phi}}(\boldsymbol{X}|\boldsymbol{Z})$ (Eq. (9)) will involve in generating the observable $\boldsymbol{x}_t$ instead of the joint distribution $p_{\boldsymbol{\Phi}}(\boldsymbol{X}, \boldsymbol{Z})$ (Eq. (10)). This is essentially the scenario of predicting future values of the observable variable given its history. We will use the term "generative model" and will not discriminate the joint generative model or the conditional one as it can be inferred in context.

4.2 Inferencing the Latent Process

As the generative model involves latent variable $\boldsymbol{z}_t$, of which the true valus are unaccessible even we have observed $\boldsymbol{x}_t$. Hence, the marginal likelihood $p_{\boldsymbol{\Phi}}(\boldsymbol{X})$ becomes the key that bridges the model and the data. The calculation of marginal likelihood involves the posterior distribution $p_{\boldsymbol{\Phi}}(\boldsymbol{Z}|\boldsymbol{X})$, which is often intractable as complex integrals are involved. We are unable to learn the paramters or to infer the latent variables. Therefore, we consider instead a restricted family of tractable distributions $q_{\boldsymbol{\Psi}}(\boldsymbol{Z}|\boldsymbol{X})$, referred to as the approximate posterior family, as approximations to the true posterior $p_{\boldsymbol{\Phi}}(\boldsymbol{Z}|\boldsymbol{X})$ such that the family is sufficiently rich and flexible to provide good approximations (Bishop, 2006; Kingma & Welling, 2013; Rezende et al., 2014).

We define the *inference model* in accordance with the approximate posterior family we have presumed, in a similar fashion as (Chung et al., 2015), where the factorised distribution is formulated as follows:

$$q_{\boldsymbol{\Psi}}(\boldsymbol{Z}|\boldsymbol{X}) = \prod_t q_{\boldsymbol{\Psi}}(\boldsymbol{z}_t|\boldsymbol{z}_{<t}, \boldsymbol{x}_{<t}) = \prod_t \mathcal{N}(\boldsymbol{z}_t; \tilde{\boldsymbol{\mu}}_{\boldsymbol{\Psi}}^z(\boldsymbol{z}_{<t}, \boldsymbol{x}_{<t}), \tilde{\boldsymbol{\Sigma}}_{\boldsymbol{\Psi}}^z(\boldsymbol{z}_{<t}, \boldsymbol{x}_{<t})), \tag{19}$$

where $\tilde{\boldsymbol{\mu}}_{\boldsymbol{\Psi}}^z(\boldsymbol{z}_{<t}, \boldsymbol{x}_{<t})$ and $\tilde{\boldsymbol{\Sigma}}_{\boldsymbol{\Psi}}^z(\boldsymbol{z}_{<t}, \boldsymbol{x}_{<t})$ are functions of the historical information $\{\boldsymbol{z}_{<t}\}, \{\boldsymbol{x}_{<t}\}$, representing the approximated mean and variance of the latent variable $\boldsymbol{z}_t$, respectively. Note that $\boldsymbol{\Psi}$ represents the parameter set of inference model.

The inference model essentially describes an autoregressive model on $\boldsymbol{z}_t$ with exogenous input $\boldsymbol{x}_t$. Hence, in a similar fashion as the generative model, we implement the inference model as the inference network using RNN/MLP:

$$\{\tilde{\boldsymbol{\mu}}_t^z, \tilde{\boldsymbol{A}}_t^z\} = \text{MLP}_i^z(\tilde{\boldsymbol{h}}_t^z), \tag{20}$$

$$\tilde{\boldsymbol{h}}_t^z = \text{RNN}_i^z(\tilde{\boldsymbol{h}}_{t-1}^z, \boldsymbol{z}_{t-1}, \boldsymbol{x}_{t-1}), \tag{21}$$

$$\boldsymbol{z}_t = \tilde{\boldsymbol{\mu}}_t^z + \tilde{\boldsymbol{A}}_t^z \tilde{\boldsymbol{\epsilon}}_t^z, \tag{22}$$

where $\tilde{\boldsymbol{A}}_t^z (\tilde{\boldsymbol{A}}_t^z)^\top = \tilde{\boldsymbol{\Sigma}}_t^z = \tilde{\boldsymbol{\Sigma}}_{\boldsymbol{\Psi}}^z(\boldsymbol{z}_{<t}, \boldsymbol{x}_{<t})$ while $\tilde{\boldsymbol{h}}_t^z$ represents the hidden state of RNN and $\tilde{\boldsymbol{\epsilon}}_t^z \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_z)$ is an auxiliary variable to extract randomness. The inference mean $\tilde{\boldsymbol{\mu}}_t^z$ and deviation $\tilde{\boldsymbol{A}}_t^z$ is computed by an MLP from the hidden state $\tilde{\boldsymbol{h}}_t^z$. We use the subscript i instead of g to distinguish the architecture used in inference model in contrast to generative model.

4.3 Forecasting Observations in Future

In the realm of time series analysis, we usually pay more attention on forecasting over generating (Box et al., 2015). It means that we are essentially more interested in the generation procedure

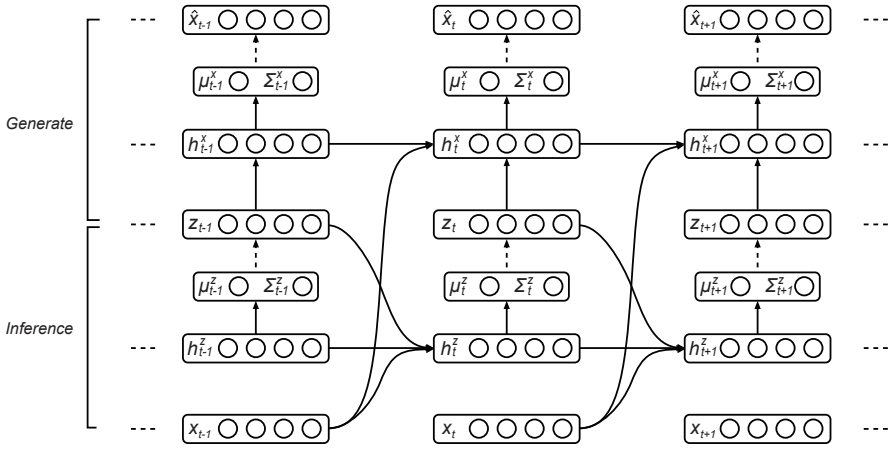


Figure 1: Forecasting the future using Neural Stochastic Volatility Model.

conditioning on the historical information rather than generation purely based on a priori belief since the observations in the past of $\boldsymbol{x}_{<t}$ influences our belief of the latent variable $\boldsymbol{z}_t$. Therefore, we apply the approximate posterior distribution of the latent variable $\boldsymbol{z}_t$ (Eq. (19)) as discussed in previous subsection, in place of the prior distribution (Eq. (8)) to build our *predictive model*.

Given the historical observations $\boldsymbol{x}_{<t}$, the predictive model infers the current value of latent variable $\boldsymbol{z}_t$ using inference network and then generates the prediction of the current observation $\boldsymbol{x}_t$ using generative network. The procedure of forecasting is shown in Fig. 1.

NSVM is learned using Stochastic Gradient Variational Bayes following (Kingma & Welling, 2013; Rezende et al., 2014). For readability, we provide the detailed derivation in Appendix A.

4.4 Links to GARCH(1,1) and Heston model

Although we refer to GARCH and Heston as volatility models, the purposes of them are quite different: GARCH is a predictive model used for volatility forecasting whereas Heston is more of a generative model of the underlying dynamics which facilitate closed-form solutions to SDEs in option pricing. The proposed NSVM has close relations to GARCH(1,1) and Heston model: both of them can be regarded as a special case of the neural network formulation. Recall Eq. (2), GARCH(1,1) is formulated as $\sigma_t^2 = \alpha_0 + \alpha_1(x_{t-1} - \mu_{t-1})^2 + \beta_1\sigma_{t-1}^2$, where μ_{t-1} is the trend estimate of $\{x_t\}$ at time step t calculated by some mean models. A common practice is to assume that μ_t follows the ARMA family (Box et al., 2015), or even simpler, as a constant that $\mu_t \equiv \mu$. We adopt the constant trend for simplicity as our focus is on volatility estimation.

We define the hidden state as $\boldsymbol{h}_t^x = [\mu, \sigma_t]^\top$, and disable the latent variable $z_t \equiv 0$ as the volatility modelled by GARCH(1,1) is conditionally deterministic. Hence, we instantiate the generative network (Eqs. (16), (17) and (18)) as follows:

$$\{\mu, \sigma_t\} = \mathrm{MLP}_g^x(\boldsymbol{h}_t^x; \boldsymbol{\Phi}) = \{[1,0]\boldsymbol{h}_t^x, [0,1]\boldsymbol{h}_t^x\}, \tag{23}$$

$$\begin{aligned}\boldsymbol{h}_t^x &= \mathrm{RNN}_g^x(\boldsymbol{h}_{t-1}^x, x_{t-1}; \boldsymbol{\Phi}) \\ &= \sqrt{\begin{bmatrix}0\\ \alpha_0\end{bmatrix} + \begin{bmatrix}0\\ \alpha_1\end{bmatrix}(x_{t-1} - [1,0]\boldsymbol{h}_{t-1}^x)^2 + \begin{bmatrix}1 & 0\\ 0 & \beta_1\end{bmatrix}(\boldsymbol{h}_{t-1}^x)^2},\end{aligned} \tag{24}$$

$$x_t = \mu + \sigma_t\epsilon_t \qquad \text{where } \epsilon_t \sim \mathcal{N}(0,1). \tag{25}$$

The set of generative parameters is $\boldsymbol{\Phi} = \{\mu, \alpha_0, \alpha_1, \beta_1\}$.

Next, we show the link between NSVM and (discrete-time) Heston model (Eq. (5)). Let $\boldsymbol{h}_t^x = [x_{t-1}, \mu, \sigma_t]^\top$ be the hidden state and z_t be i.i.d. standard Gaussian instead of autoregressive vari-

able, we represent the Heston model in the framework of NSVM as:

$$\begin{bmatrix}\epsilon_t \\ z_t\end{bmatrix} = \mathcal{N}(\mathbf{0}, \begin{bmatrix}1 & \rho \\ \rho & 1\end{bmatrix}), \tag{26}$$

$$\{\mu_t, \sigma_t\} = \text{MLP}_g^x(\boldsymbol{h}_t^x; \boldsymbol{\Phi}) = \{[1,1,0]\boldsymbol{h}_t^x - [0,0,0.5](\boldsymbol{h}_t^x)^2, [0,0,1]\boldsymbol{h}_t^x\}, \tag{27}$$

$$\begin{aligned}\boldsymbol{h}_t^x &= \text{RNN}_g^x(\boldsymbol{h}_{t-1}^x, x_{t-1}, z_t; \boldsymbol{\Phi}) \\ &= \begin{bmatrix}0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1+a\end{bmatrix}\boldsymbol{h}_{t-1}^x + \begin{bmatrix}1 \\ 0 \\ 0\end{bmatrix} x_{t-1} + \begin{bmatrix}0 \\ 0 \\ b\end{bmatrix} z_t,\end{aligned} \tag{28}$$

$$x_t = \mu_t + \sigma_t \epsilon_t. \tag{29}$$

The set of generative parameters is $\boldsymbol{\Phi} = \{\mu, a, b\}$.

One should notice that, in practice, the formulation may change in accordance with the specific architecture of neural networks involved in building the model, and hence a closed-form representation may be absent.

5 Experiments

In this section, we present our experiments[1] both on the synthetic and real-world datasets to validate the effectiveness of NSVM.

5.1 Baselines and Evaluation Metrics

To evaluate the performance of volatility modelling, we adopt the standard econometric model GARCH(1,1) Bollerslev (1986) as well as its variants EGARCH(1,1) Nelson (1991), GJR-GARCH(1,1,1) Glosten et al. (1993), ARCH(5), TARCH(1,1,1), APARCH(1,1,1), AGARCH(1,1,1), NAGARCH(1,1,1), IGARCH(1,1), IAVGARCH(1,1), FIGARCH(1,d,1) as baselines, which incorporate with the corresponding mean model AR(20). We would also compare our NSVM against a MCMC-based model "stochvol" and the recent Gaussian-processes-based model "GPVOL" Wu et al. (2014), which is a non-parametric model jointly learning the dynamics and hidden states via online inference algorithm. In addition, we setup a naive forecasting model as an alternative baseline referred to as *NAIVE*, which maintains a sliding window of size 20 on the most recent historical observations and forecasts the current values of mean and volatility by the average mean and variance of the window.

For synthetic data experiments, we take *four* metrics into consideration for performance evaluation: 1) the negative log-likelihood (NLL) of observing the test sequence with respect to the generative model parameters; 2) the mean-squared error (MSE) between the predicted mean and the ground truth (μ-MSE), 3) MSE of the predicted variance against the true variance (σ-MSE); 4) smoothness of fit, which is the standard deviation of the differences of succesive variance estimates. As for the real-world scenarios, the trend and volatility are implicit such that no ground truth is accessible to compare with, we consider only NLL and smoothness as the metrics for evaluation on real-world data experiment.

5.2 Model Implementation

The implementation of NSVM in experiments is in accordance with the architecture illustrated in Fig. 1: it consists of two neural networks, namely inference network and generative network. Each network comprises a set of RNN/MLP as we have discussed above: the RNN is instantiated by stacked LSTM layers whereas the MLP is essentially a 1-layer fully-connected feedforward network which splits into two equal-sized sublayers with different activation functions – one sublayer applies exponential function to impose the non-negativity and prevents overshooting of variance estimates while the other uses linear function to calculate mean estimates. During experiment, the model is structured by cascading the inference network and generative network as depicted in Fig. 1. The input layer is of size 20, which is the same as the embedding dimension D_E; the layer on the

[1] Repeatable experiment code: `https://github.com/xxj96/nsvm`

interface of inference network and generative network – we call it *latent variable layer* – represents the latent variable z, where its dimension is 2. The output layer has the same structure as the input one, therefore the latent variable layer acts as a bottleneck of the entire architecture which helps to extract the key factor. The stacked layers between input layer, latent variable layer and output layer are the hidden layers of either inference network or generative network, it consists of 1 or 2 LSTM layers with size 10, which contains recurrent connection for temporal dependencies modelling.

State-of-the-art learning techniques have been applied: we introduce Dropout (Zaremba et al., 2014) into each LSTM recurrent layer and impose L2-norm on the weights of each fully-connected feed-forward layer as regularistion; *NADAM* optimiser (Dozat, 2015) is exploited for fast convergence, which is a variant of *ADAM* optimiser (Kingma & Ba, 2014) incorporated with Nesterov momentum; stepwise exponential learning rate decay is adopted to anneal the variations of convergence as time goes.

For econometric models, we utilise several widely-used packages for time series analysis: *statsmodels* (`http://statsmodels.sourceforge.net/`), *arch* (`https://pypi.python.org/pypi/arch/3.2`), *Oxford-MFE-toolbox* (`https://www.kevinsheppard.com/MFE_Toolbox`), *stochvol* (`https://cran.r-project.org/web/packages/stochvol`) and *fGarch* (`https://cran.r-project.org/web/packages/fGarch`). The implementation of GPVOL is retrived from `http://jmhl.org` and we adopt the same hyperparameter setting as in Wu et al. (2014).

5.3 Synthetic Data Experiment

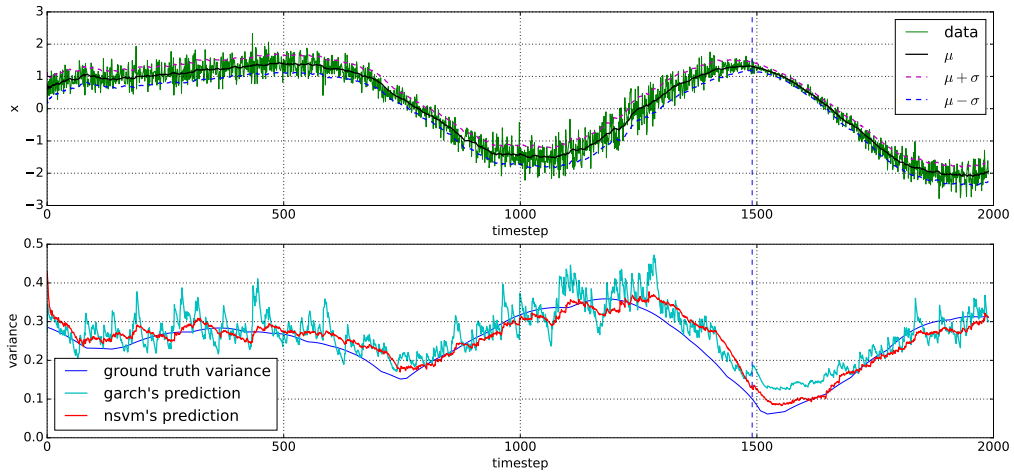
We build up the synthetic dataset by generating 256 heteroskedastic univariate time series, each with 2000 data points i.e. 2000 time steps. At each time step, the observation is drawn from a Gaussian distribution with pre-determined mean and variance, where the tendency of mean and variance is synthesised as linear combinations of sine functions. Specifically, for the trend and variance, we synthesis each using 3 sine functions with randomly chosen amplitudes and frequencies; then the value of the synthesised signal at each timestep is drawn from a Gaussian distribution with the corresponding value of trend and variance at that timestep. A sampled sequence is shown in Fig. 2a. We expect that this limited dataset could well simulate the real-world scenarios: one usually has very limited chances to observe and collect a large amount of data from time-invariant distributions. In addition, it seems that every observable or latent quantity within time series varies from time to time and seldom repeats the old patterns. Hence, we presume that the tendency shows long-term patterns and the period of tendency is longer than observation. In the experiment, we take the former 1500 time steps as the training set whereas the latter 500 as the test set.

For the synthetic data experiment, we simplify the recurrent layers in both inference net and generative net as single LSTM layer of size 10. The actual input $\{\vec{\boldsymbol{x}}_t\}$ fed to NSVM is D_E-dimensional time-delay embedding (Kennel et al., 1992) of raw univariate observation $\{x_t\}$ such that $\vec{\boldsymbol{x}}_t = [x_{t+1-D_E}, \ldots, x_t]$. 2-dimensional latent variable $\boldsymbol{z}_t$ is adopted to capture the latent process, and enforces an orthogonal representation of the process by using diagonal covariance matrix. At each time step, 30 samples of latent variable $\boldsymbol{z}_t$ are generated via reparameterisation (Eq. (22)).

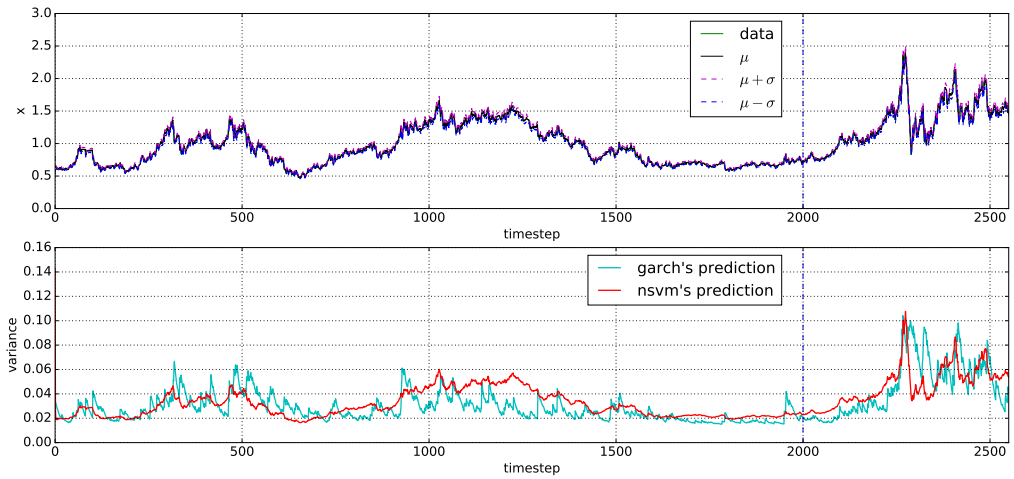
5.4 Real-world Data Experiment

We select 162 out of more than 1500 stocks from Chinese stock market and collect the time series of their daily closing prices from 3 institutions in China. We favour those with earlier listing date of trading (from 2006 or earlier) and fewer suspension days (at most 50 suspension days in total during the period of observation) so as to reduce the noise introduced by insufficient observation or missing values, which has significant influences on the performance but is essentially irrelevant to the purpose of volatility forecasting. More specifically, the dataset obtained contains 162 time series, each with 2552 data points (7 years). A sampled sequence is shown in Fig. 2b. We divide the whole dataset into two subsets: the training subset consists of the first 2000 data points while the test subset contains the rest 552 data points.

Similar model configuration is applied to the real-world data experiment: time-delay embedding of dimension D_E on the raw univariate time series; 2-dimensional latent variable with diagonal



(a) Synthetic time series prediction. (up) The data and the predicted μ^x and bounds $\mu^x \pm \sigma^x$. (down) The groundtruth data variance and the corresponding prediction from GARCH(1,1) and NSVM.



(b) Real-world stock price prediction. (up) The data and the predicted μ^x and bounds $\mu^x \pm \sigma^x$. (down) The variance prediction from GARCH(1,1) and NSVM. The prediction of NSVM is more smooth and stable than that of GARCH(1,1), also yielding smaller NLL.

Figure 2: A case study of time series prediction.

covariance matrix; 30 sampling for the latent variable at each time step. Instead of single LSTM layers, here we adopt stacked LSTM layers composed of 2×10 LSTM cells.

5.5 Result and Discussion

The overall performance of NSVM and baselines is listed in details in Table 1 and case studies on synthetic data and real-world financial data are illustrated in Fig. 2. The results show that NSVM has higher accuracies for modelling heteroskedastic time series on various metrics: NLL shows the fitness of the model under likelihood measure; the smoothness indicates that NSVM obtains more robust representation of the latent volatility; μ-MSE and σ-MSE in synthetic data experiment imply the ability of recognising the underlying patterns of both trend and volatility, which in fact verifies our claim of NSVM's high flexibility and rich expressive power for volatility (as well as trend) modelling and forecasting compared with the baselines. Although the improvement comes at the cost of longer training time before convergence, it can be mitigated by applying parallel computing techniques as well as more advanced network architecture or training procedure.

	SYNTHETIC DATA				STOCK DATA	
	NLL	μ-MSE	σ-MSE	smoothness	NLL	smoothness
NSVM	**3.932e-2**	**2.393e-3**	**6.178e-4**	4.322e-3	**-2.184**	3.505e-3
GARCH(1,1)	6.905e-2	7.594e-3*	8.408e-4	4.616e-3	-1.961	6.659e-3
GJRGARCH(1,1,1)	6.491e-2	7.594e-3*	7.172e-4	4.426e-3	-2.016	4.967e-3
EGARCH(1,1)	5.913e-2	7.594e-3*	8.332e-4	4.546e-3	-2.001	5.451e-3
ARCH(5)	7.577e-2	7.594e-3*	1.610e-3	5.880e-3	-1.955	7.917e-3
TARCH(1,1,1)	6.365e-2	7.594e-3*	7.284e-4	4.727e-3	-2.012	3.399e-3
APARCH(1,1,1)	6.187e-2	7.594e-3*	9.115e-4	4.531e-3	-2.014	4.214e-3
AGARCH(1,1)	6.311e-2	7.594e-3*	9.543e-4	4.999e-3	-2.008	5.847e-3
NAGARCH(1,1,1)	1.134e-1	7.594e-3*	9.516e-4	4.904e-3	-2.020	5.224e-3
IGARCH(1,1)	6.751e-2	7.594e-3*	9.322e-4	4.019e-3	-1.999	4.284e-3
IAVGARCH(1,1)	6.901e-2	7.594e-3*	7.174e-4	4.282e-3	-1.984	4.062e-3
FIGARCH(1,d,1)	6.666e-2	7.594e-3*	1.055e-3	5.045e-3	-2.002	5.604e-3
MCMC-stochvol	0.368	7.594e-3*	3.956e-2	**6.421e-4**	-0.909	**1.511e-3**
GPVOL	1.273	7.594e-3*	6.457e-1	4.142e-2	-2.052	5.739e-3
NAIVE	2.037e-1	8.423e-3	3.515e-3	2.708e-2	-0.918	7.459e-3

*the same results obtained from AR(20) mean models

Table 1: Results of the experiments.

The newly proposed NSVM outperforms standard econometric models GARCH(1,1), EGARCH(1,1), GJR-GARCH(1,1,1) and some other variants as well as the MCMC-based model "stochvol" and the recent GP-based model "GPVOL". Apart from the higher accuracy NSVM obtained, it provides us with the ability to simply generalise univariate time series analysis to multivariate cases by extending network dimensions and manipulating the covariance matrices. Furthermore, it allows us to implement and deploy a similar framework on other applications, for example signal processing and denoising. The shortcoming of NSVM comparing to GPVOL is that the training procedure is offline: for short-term prediction, the experiments have shown the accuracy, but for long-term forecasting, the parameters need retraining, which will be rather time consuming. The online algorithm for inference will be one of the work in the future.

Specifically, our NSVM outperforms GARCH(1,1) on 142 out of 162 stocks on the metric of NLL. In particular, NSVM obtains -2.111, -2.044, -2.609 and -1.939 on the stocks corresponding to Fig2(b), Fig 4(a), (b) and (c) respectively, each of which is better than the that of GARCH (0.3433, 0.589, 0.109 and 0.207 lower on NLL).

6 CONCLUSION

In this paper, a novel volatility model NSVM has been proposed for stochastic volatility estimation and forecast. We integrated statistical models and RNNs, leveraged the characteristics of each model, organised the dependences between random variables in the form of graphical models, implemented the mappings among variables and parameters through RNNs, and finally established a powerful stochastic recurrent model with universal approximation capability. The proposed architecture comprises a pair of complementary stochastic neural networks: the generative network and inference network. The former models the joint distribution of the stochastic volatility process with both observable and latent variables of interest; the latter provides with the approximate posterior i.e. an analytical approximation to the (intractable) conditional distribution of the latent variables given the observable ones. The parameters (and consequently the underlying distributions) are learned (and inferred) via variational inference, which maximises the lower bound for the marginal log-likelihood of the observable variables. Our NSVM has presented higher accuracy compared to GARCH(1,1), EGARCH(1,1) and GJR-GARCH(1,1,1) as well as GPVOL for volatility modelling and forecasting on synthetic data and real-world financial data. Future work on NSVM would be to incorporate well-established models such as ARMA/ARIMA and to investigate the modelling of seasonal time series and correlated sequences.

As we have known, for models that evolve explicitly in terms of the squares of the residuals ($e_t^2 = (x_t - \mu_t)^2$), e.g. GARCH, the multi-step-ahead forecasts have closed-form solutions, which means

that those forecasts can be efficiently computed in a recursive fashion due to the linear formulation of the model and the exploitation of relation $E_{t-1}[e_t^2] = \sigma_t^2$.

On the other hand, for models that are not linear or do not explicitly evolve in terms of e^2, e.g. EGARCH (linear but not evolve in terms of e^2), our NSVM (nonlinear and not evolve in terms of e^2), the closed-form solutions are absent and thus the analytical forecast is not available. We will instead use simulation-based forecast, which uses random number generator to simulate draws from the predicted distribution and build up a pre-specified number of paths of the variances at 1 step ahead. The draws are then averaged to produce the forecast of the next step. For n-step-ahead forecast, it requires n iterations of 1-step-ahead forecast to get there.

NSVM is designed as an end-to-end model for volatility estimation and forecast. It takes the price of stocks as input and outputs the distribution of the price at next step. It learns the dynamics using RNN, leading to an implicit, highly nonlinear formulation, where only simulation-based forecast is available. In order to obtain reasonably accurate forecasts, the number of draws should be relatively large, which will be very expensive for computation. Moreover, the number of draws will increase exponentially as the forecast horizon grows, so it will be infeasible to forecast several time steps ahead. We have planned to investigate the characteristics of NSVM's long-horizontal forecasts and try to design a model specific sampling method for efficient evaluation in the future.

REFERENCES

Torben G Andersen and Tim Bollerslev. Answering the skeptics: Yes, standard volatility models do provide accurate forecasts. *International economic review*, pp. 885–905, 1998.

Marco Avellaneda and Antonio Paras. Managing the volatility risk of portfolios of derivative securities: the lagrangian uncertain volatility model. *Applied Mathematical Finance*, 3(1):21–52, 1996.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Justin Bayer and Christian Osendorfer. Learning stochastic recurrent networks. *arXiv preprint arXiv:1411.7610*, 2014.

Christopher M Bishop. Pattern recognition. *Machine Learning*, 128, 2006.

Tim Bollerslev. Generalized autoregressive conditional heteroskedasticity. *Journal of econometrics*, 31(3):307–327, 1986.

George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. Attention-based models for speech recognition. In *Advances in Neural Information Processing Systems*, pp. 577–585, 2015.

Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. In *Advances in neural information processing systems*, pp. 2980–2988, 2015.

John C Cox, Jonathan E Ingersoll Jr, and Stephen A Ross. A theory of the term structure of interest rates. *Econometrica: Journal of the Econometric Society*, pp. 385–407, 1985.

Timothy Dozat. Incorporating nesterov momentum into adam. 2015.

Robert F Engle. Autoregressive conditional heteroscedasticity with estimates of the variance of united kingdom inflation. *Econometrica: Journal of the Econometric Society*, pp. 987–1007, 1982.

Robert F Engle and Kenneth F Kroner. Multivariate simultaneous generalized arch. *Econometric theory*, 11(01):122–150, 1995.

Otto Fabius and Joost R van Amersfoort. Variational recurrent auto-encoders. *arXiv preprint arXiv:1412.6581*, 2014.

Marco Fraccaro, Søren Kaae Sønderby, Ulrich Paquet, and Ole Winther. Sequential neural models with stochastic layers. *arXiv preprint arXiv:1605.07571*, 2016.

Lawrence R Glosten, Ravi Jagannathan, and David E Runkle. On the relation between the expected value and the volatility of the nominal excess return on stocks. *The journal of finance*, 48(5): 1779–1801, 1993.

Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pp. 6645–6649. IEEE, 2013.

Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. Draw: A recurrent neural network for image generation. *arXiv preprint arXiv:1502.04623*, 2015.

Barbara Hammer. On the approximation capability of recurrent neural networks. *Neurocomputing*, 31(1):107–123, 2000.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

Steven L Heston. A closed-form solution for options with stochastic volatility with applications to bond and currency options. *Review of financial studies*, 6(2):327–343, 1993.

Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

John C Hull. *Options, futures, and other derivatives*. Pearson Education India, 2006.

Matthew B Kennel, Reggie Brown, and Henry DI Abarbanel. Determining embedding dimension for phase-space reconstruction using a geometrical construction. *Physical review A*, 45(6):3403, 1992.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

Daniel B Nelson. Conditional heteroskedasticity in asset returns: A new approach. *Econometrica: Journal of the Econometric Society*, pp. 347–370, 1991.

Ser-Huang Poon and Clive WJ Granger. Forecasting volatility in financial markets: A review. *Journal of economic literature*, 41(2):478–539, 2003.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015.

Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

Josef Stoer and Roland Bulirsch. *Introduction to numerical analysis*, volume 12. Springer Science & Business Media, 2013.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pp. 3104–3112, 2014.

Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*, 2016.

Yue Wu, José Miguel Hernández-Lobato, and Zoubin Ghahramani. Gaussian process volatility model. In *Advances in Neural Information Processing Systems*, pp. 1044–1052, 2014.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014.

A COMPLEMENTARY DISCUSSIONS OF NSVM

In this appendix section we present detailed derivations of NSVM, specifically, the parameters learning and calibration, and covariance reparameterisation.

A.1 LEARNING PARAMETERS / CALIBRATION

Given the observations $\boldsymbol{X}$, the objective of learning is to maximise the marginal log-likelihood of $\boldsymbol{X}$ given $\boldsymbol{\Phi}$, where the posterior is involved. However, as we have discussed in the previous subsection, the true posterior is usually intractable, which means exact inference is difficult. Hence, approximate inference is applied instead of rather than exact inference by following (Kingma & Welling, 2013; Rezende et al., 2014). We represent the marginal log-likelihood of $\boldsymbol{X}$ in the following form:

$$\begin{aligned}\ln p_{\boldsymbol{\Phi}}(\boldsymbol{X}) &= \mathbb{E}_{q_{\boldsymbol{\Psi}}(\boldsymbol{Z}|\boldsymbol{X})}\Big[\ln \frac{p_{\boldsymbol{\Phi}}(\boldsymbol{X},\boldsymbol{Z})}{p_{\boldsymbol{\Phi}}(\boldsymbol{Z}|\boldsymbol{X})}\Big] = \mathbb{E}_{q_{\boldsymbol{\Psi}}(\boldsymbol{Z}|\boldsymbol{X})}\Big[\ln \frac{p_{\boldsymbol{\Phi}}(\boldsymbol{X},\boldsymbol{Z})}{q_{\boldsymbol{\Psi}}(\boldsymbol{Z}|\boldsymbol{X})}\frac{q_{\boldsymbol{\Psi}}(\boldsymbol{Z}|\boldsymbol{X})}{p_{\boldsymbol{\Phi}}(\boldsymbol{Z}|\boldsymbol{X})}\Big] \\ &= \mathbb{E}_{q_{\boldsymbol{\Psi}}(\boldsymbol{Z}|\boldsymbol{X})}[\ln p_{\boldsymbol{\Phi}}(\boldsymbol{X},\boldsymbol{Z}) - \ln q_{\boldsymbol{\Psi}}(\boldsymbol{Z}|\boldsymbol{X})] + KL[q_{\boldsymbol{\Psi}}(\boldsymbol{Z}|\boldsymbol{X})\|p_{\boldsymbol{\Phi}}(\boldsymbol{Z}|\boldsymbol{X})] \\ &\geq \mathbb{E}_{q_{\boldsymbol{\Psi}}(\boldsymbol{Z}|\boldsymbol{X})}[\ln p_{\boldsymbol{\Phi}}(\boldsymbol{X},\boldsymbol{Z}) - \ln q_{\boldsymbol{\Psi}}(\boldsymbol{Z}|\boldsymbol{X})] \qquad (\text{as } KL \geq 0),\end{aligned} \tag{30}$$

where the expectation term $\mathbb{E}_{q_{\boldsymbol{\Psi}}(\boldsymbol{Z}|\boldsymbol{X})}[\ln p_{\boldsymbol{\Phi}}(\boldsymbol{X},\boldsymbol{Z}) - \ln q_{\boldsymbol{\Psi}}(\boldsymbol{Z}|\boldsymbol{X})]$ is referred to as the variational lower bound $\mathcal{L}[q;\boldsymbol{X},\boldsymbol{\Phi},\boldsymbol{\Psi}]$ of the approximate posterior $q_{\boldsymbol{\Psi}}(\boldsymbol{Z}|\boldsymbol{X},\boldsymbol{\Psi})$. The lower bound is essentially a functional with respect to distribution q and parameterised by observations $\boldsymbol{X}$ and parameter sets $\boldsymbol{\Phi},\boldsymbol{\Psi}$ of both generative and inference model. In theory, the marginal log-likelihood is maximised by optimisation on the lower bound $\mathcal{L}[q;\boldsymbol{X},\boldsymbol{\Phi},\boldsymbol{\Psi}]$ with respect to $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$.

We apply the factorisations in Eqs. (10) and (19) to the integrand within expectation of Eq. (30):

$$\begin{aligned}\ln p_{\boldsymbol{\Phi}}(\boldsymbol{X},\boldsymbol{Z}) - \ln q_{\boldsymbol{\Psi}}(\boldsymbol{Z}|\boldsymbol{X}) = \sum_t \Big[&\ln \mathcal{N}(\boldsymbol{x}_t;\boldsymbol{\mu}^x_{\boldsymbol{\Phi}}(\boldsymbol{x}_{<t},\boldsymbol{z}_{\leq t}),\boldsymbol{\Sigma}^x_{\boldsymbol{\Phi}}(\boldsymbol{x}_{<t},\boldsymbol{z}_{\leq t})) \\ &+ \ln \mathcal{N}(\boldsymbol{z}_t;\boldsymbol{\mu}^z_{\boldsymbol{\Phi}}(\boldsymbol{z}_{<t}),\boldsymbol{\Sigma}^z_{\boldsymbol{\Phi}}(\boldsymbol{z}_{<t})) - \ln \mathcal{N}(\boldsymbol{z}_t;\tilde{\boldsymbol{\mu}}^z_{\boldsymbol{\Psi}}(\boldsymbol{z}_{<t},\boldsymbol{x}_{<t}),\tilde{\boldsymbol{\Sigma}}^z_{\boldsymbol{\Psi}}(\boldsymbol{z}_{<t},\boldsymbol{x}_{<t}))\Big].\end{aligned} \tag{31}$$

As there is usually no closed-form solution for the expecation (Eq. (30)), we have to estimate the expectation by applying sampling methods to latent variable $\boldsymbol{z}_t$ through time in accordance with the causal dependences. We utilise the reparameterisation of $\boldsymbol{z}_t$ as shown in Eq. (22) such that we sample the corresponding auxiliary standard variable $\tilde{\boldsymbol{\epsilon}}_t$ rather than $\boldsymbol{z}_t$ itself and compute the value of $\boldsymbol{z}_t$ on the fly. This ensures that the gradient-based optimisation techniques are applicable as the reparameterisation isolates the model parameters of interest from the sampling procedure. By sampling N sample paths, the estimator of the lower bound is defined as the average of paths:

$$\begin{aligned}\widehat{\mathcal{L}} = -\frac{1}{2N}\sum_t \Big[&\ln\det \boldsymbol{\Sigma}^z_t + (\tilde{\boldsymbol{\mu}}^z_t + \tilde{\boldsymbol{A}}^z_t\tilde{\boldsymbol{\epsilon}}^z_t - \boldsymbol{\mu}^z_t)^\top(\boldsymbol{\Sigma}^z_t)^{-1}(\tilde{\boldsymbol{\mu}}^z_t + \tilde{\boldsymbol{A}}^z_t\tilde{\boldsymbol{\epsilon}}^z_t - \boldsymbol{\mu}^z_t) \\ &+ \ln\det \boldsymbol{\Sigma}^x_t + (\boldsymbol{x}_t - \boldsymbol{\mu}^x_t)^\top(\boldsymbol{\Sigma}^x_t)^{-1}(\boldsymbol{x}_t - \boldsymbol{\mu}^x_t) - \ln\det\tilde{\boldsymbol{\Sigma}}_t\Big] + \text{const},\end{aligned} \tag{32}$$

where $\tilde{\boldsymbol{A}}^z_t(\tilde{\boldsymbol{A}}^z_t)^\top = \tilde{\boldsymbol{\Sigma}}^z_t$ and $\tilde{\boldsymbol{\epsilon}}^z_t \sim \mathcal{N}(\boldsymbol{0},\boldsymbol{I}_z)$ is parameter-independent and considered as constant when calculating derivatives.

A.2 COVARIANCE PARAMETERISATION

As is known, it entails a computational complexity of $\mathcal{O}(M^3)$ to maintain and update the full-size covariance $\boldsymbol{\Sigma}$ with M dimensions (Rezende et al., 2014). In the case of very high dimensions, the full-size covariance matrix would be too computationally expensive to afford. Hence, we use instead the covariance matrices with much fewer parameters for efficiency. The simplest setting is to use diagonal precision matrix (i.e. the inverse of covariance matrix) $\boldsymbol{\Sigma}^{-1} = \boldsymbol{D}$. However, it draws very strong restrictions on representation of the random variable of interest as the diagonal precision matrix (and thus diagonal covariance matrix) indicates independence among the dimensions. Therefore, the tradeoff becomes low-rank perturbation on diagonal matrix: $\boldsymbol{\Sigma}^{-1} = \boldsymbol{D} + \boldsymbol{V}\boldsymbol{V}^\top$, where $\boldsymbol{V} = \{\boldsymbol{v}_1,\ldots,\boldsymbol{v}_K\}$ denotes the perturbation while each $\boldsymbol{v}_k$ is a M-dimensional column vector.

The corresponding covariance matrix and its determinant is obtained using *Woodbury identity* and *matrix determinant lemma*:

$$\boldsymbol{\Sigma} = \boldsymbol{D}^{-1} - \boldsymbol{D}^{-1}\boldsymbol{V}(\boldsymbol{I} + \boldsymbol{V}^\top \boldsymbol{D}^{-1}\boldsymbol{V})^{-1}\boldsymbol{V}^\top \boldsymbol{D}^{-1} \tag{33}$$

$$\ln\det \boldsymbol{\Sigma} = -\ln\det\left(\boldsymbol{D} + \boldsymbol{V}\boldsymbol{V}^\top\right) = -\ln\det D - \ln\det\left(\boldsymbol{I} + \boldsymbol{V}^\top \boldsymbol{D}^{-1}\boldsymbol{V}\right) \tag{34}$$

To calculate the deviation $\boldsymbol{A}$ for the factorisation of covariance matrix $\boldsymbol{\Sigma} = \boldsymbol{A}\boldsymbol{A}^\top$, we first consider the rank-1 perturbation where $K = 1$. It follows that $\boldsymbol{V} = \boldsymbol{v}$ is a column vector, and $\boldsymbol{I} + \boldsymbol{V}^\top \boldsymbol{D}^{-1}\boldsymbol{V} = 1 + \boldsymbol{v}^\top \boldsymbol{D}^{-1}\boldsymbol{v}$ is a real number. A particular solution of $\boldsymbol{A}$ is obtain:

$$\boldsymbol{A} = \boldsymbol{D}^{-\frac{1}{2}} - [\gamma^{-1}(1 - \sqrt{\eta})]\boldsymbol{D}^{-1}\boldsymbol{v}\boldsymbol{v}^\top \boldsymbol{D}^{-\frac{1}{2}} \tag{35}$$

where $\gamma = \boldsymbol{v}^\top \boldsymbol{D}^{-1}\boldsymbol{v}$, $\eta = (1+\gamma)^{-1}$. The computational complexity involved here is merely $\mathcal{O}(M)$.

Observe that $\boldsymbol{V}\boldsymbol{V}^\top = \sum_{k=1}^{K} \boldsymbol{v}_k \boldsymbol{v}_k^\top$, the perturbation of rank K is essentially the superposition of K perturbations of rank 1. Therefore, we can calculate the deviation $\boldsymbol{A}$ iteratively, an algorithm is provided to demonstrate the procedure of calculation. The computational complexity for rank-K perturbation remains to be $\mathcal{O}(M)$ given $K \ll M$.

Algorithm 1 gives the detailed calculation scheme.

Algorithm 1 Calculation of rank-K perturbation of precision matrices

Input: The original diagonal matrix $\boldsymbol{D}$; The rank-K perturbation $\boldsymbol{V} = \{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_K\}$
Output: $\boldsymbol{A}$ such that the factorisation $\boldsymbol{A}\boldsymbol{A}^\top = \boldsymbol{\Sigma} = (\boldsymbol{D} + \boldsymbol{V}\boldsymbol{V}^\top)^{-1}$ holds

1: $\boldsymbol{A}_{(0)} = \boldsymbol{D}^{-\frac{1}{2}}$
2: $i = 0$
3: **while** $i < K$ **do**
4: $\quad \gamma_{(i)} = \boldsymbol{v}_{(i)}^\top \boldsymbol{A}_{(i)} \boldsymbol{A}_{(i)}^\top \boldsymbol{v}_{(i)}$
5: $\quad \eta_{(i)} = (1 + \gamma_{(i)})^{-1}$
6: $\quad \boldsymbol{A}_{(i+1)} = \boldsymbol{A}_{(i)} - [\gamma_{(i)}^{-1}(1 - \sqrt{\eta_{(i)}})]\boldsymbol{A}_{(i)}\boldsymbol{A}_{(i)}^\top \boldsymbol{v}_{(i)}\boldsymbol{v}_{(i)}^\top \boldsymbol{A}_{(i)}$
7: $\boldsymbol{A} = \boldsymbol{A}_{(K)}$

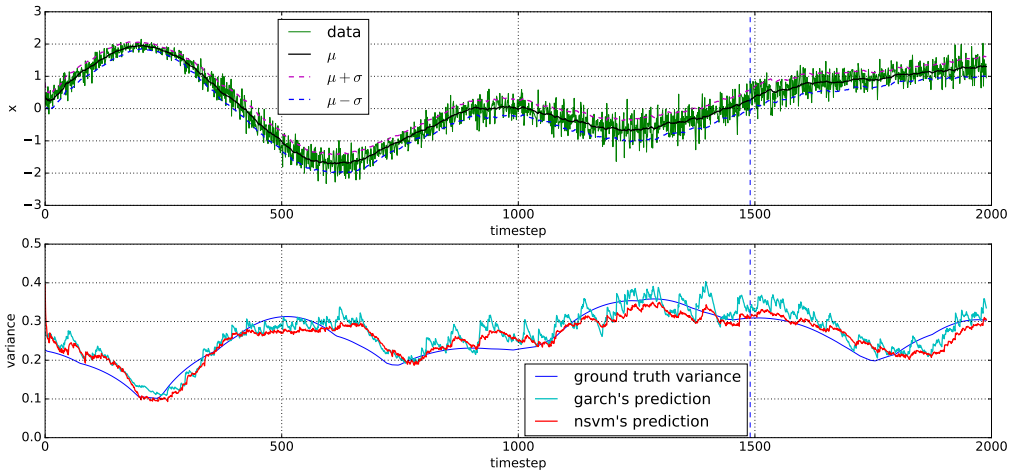
B MORE CASE STUDIES

In this appendix section we add more case studies of NVSM performance on both synthetic data and real-world stock data.

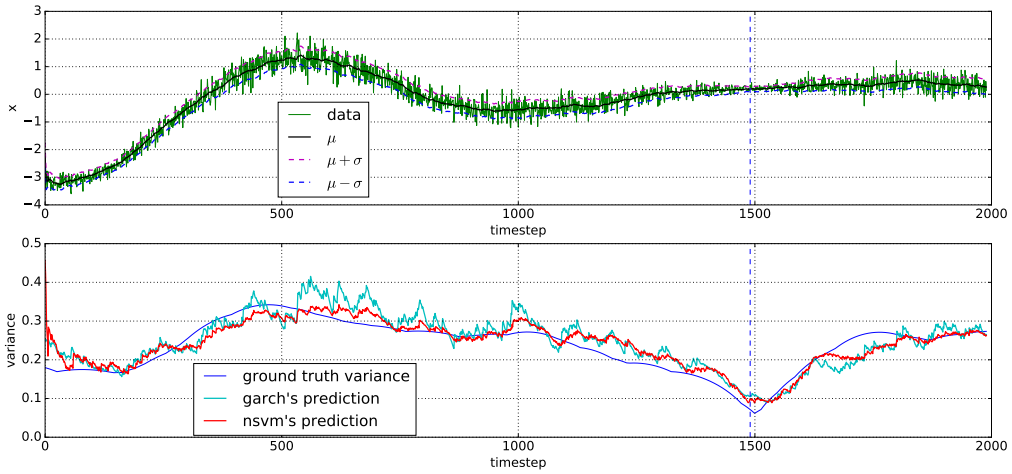
NSVM obtains -2.044, -2.609 and -1.939 on the stocks corresponding to Fig 4(a), (b) and (c) respectively, each of which is better than the that of GARCH (0.589, 0.109 and 0.207 lower on NLL).

The reason of the drops in Fig 4(b) and (c) seems to be that NSVM has captured the jumps and drops of the stock price using its nonlinear dynamics and modelled the sudden changes as part of the trend: the estimated trend "mu" goes very close to the real observed price even around the jumps and drops (see the upper figure of Fig 4(b) and (c) around step 1300 and 1600). The residual (i.e. difference between the real value of observation and the trend of prediction) therefore becomes quite small, which lead to a lower volatility estimation.

On the other hand, for the baselines, we adopt AR as the trend model, which is a relatively simple linear model compared with the nonlinear NSVM. AR would not capture the sudden changes and leave those spikes in the residual; GARCH then took the residuals as input for volatility modelling, resulting in the spikes in volatility estimation.

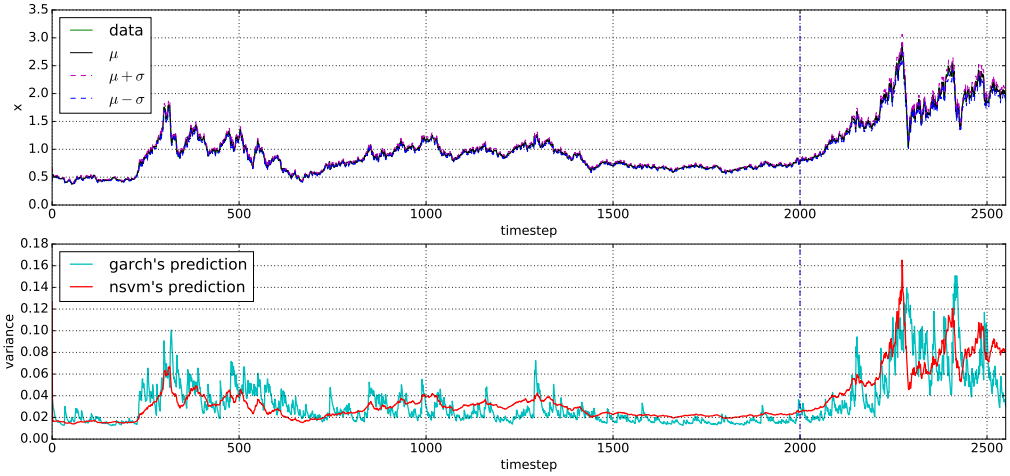


(a) Synthetic time series prediction II. (up) The data and the predicted μ^x and bounds $\mu^x \pm \sigma^x$. (down) The groundtruth data variance and the corresponding prediction from GARCH(1,1) and NSVM.

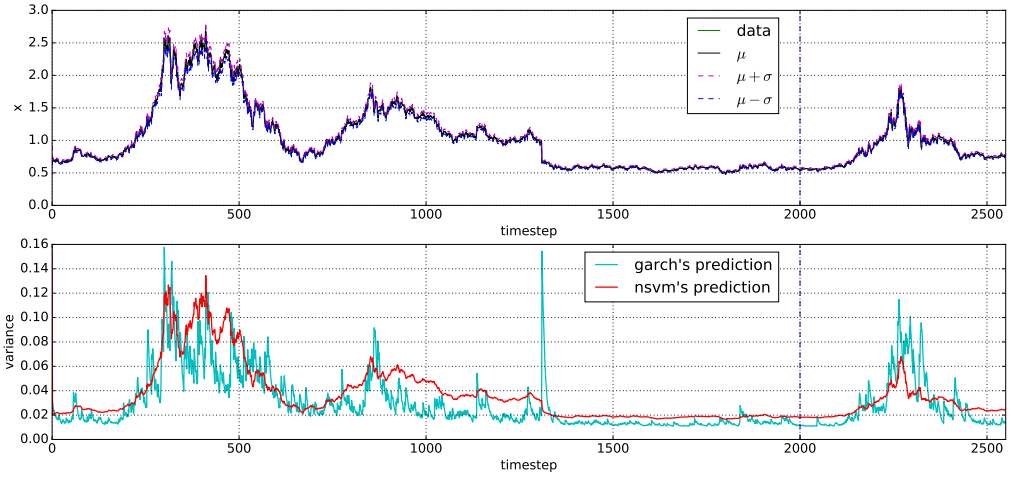


(b) Synthetic time series prediction IV. (up) The data and the predicted μ^x and bounds $\mu^x \pm \sigma^x$. (down) The groundtruth data variance and the corresponding prediction from GARCH(1,1) and NSVM.

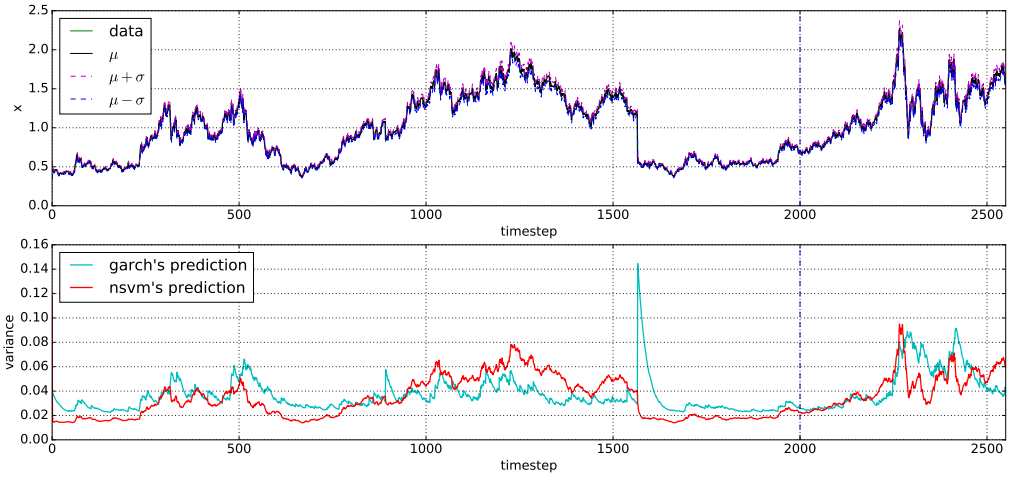
Figure 3: A case study of synthetic time series prediction.



(a) Real-world stock price prediction II. (up) The data and the predicted μ^x and bounds $\mu^x \pm \sigma^x$. (down) The variance prediction from GARCH(1,1) and NSVM. The prediction of NSVM is more smooth and stable than that of GARCH(1,1), also yielding smaller NLL.



(b) Real-world stock price prediction III. (up) The data and the predicted μ^x and bounds $\mu^x \pm \sigma^x$. (down) The variance prediction from GARCH(1,1) and NSVM. The prediction of NSVM is more smooth and stable than that of GARCH(1,1), also yielding smaller NLL.



(c) Real-world stock price prediction IV. (up) The data and the predicted μ^x and bounds $\mu^x \pm \sigma^x$. (down) The variance prediction from GARCH(1,1) and NSVM. The prediction of NSVM is more smooth and stable than that of GARCH(1,1), also yielding smaller NLL.

Figure 4: A case study of real-world stock time series prediction.