Quantitative Finance

ISSN: (Print) (Online) Journal homepage: www.tandfonline.com/journals/rquf20

A two-step framework for arbitrage-free prediction of the implied volatility surface

Wenyong Zhang, Lingfei Li & Gongqiu Zhang

To cite this article: Wenyong Zhang, Lingfei Li & Gongqiu Zhang (2023) A two-step framework for arbitrage-free prediction of the implied volatility surface, Quantitative Finance, 23:1, 21-34, DOI: 10.1080/14697688.2022.2135454

To link to this article: https://doi.org/10.1080/14697688.2022.2135454

Quantitative Finance, 2023
Vol. 23, No. 1, 21–34, https://doi.org/10.1080/14697688.2022.2135454

A two-step framework for arbitrage-free prediction of the implied volatility surface

WENYONG ZHANG†, LINGFEI LI*† and GONGQIU ZHANG‡

†Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong, Hong Kong, Hong Kong SAR
‡School of Science and Engineering, The Chinese University of Hong Kong, Shenzhen, People's Republic of China

(*Received 14 June 2021; accepted 7 October 2022; published online 31 October 2022*)

In this study, we propose a two-step framework to predict the implied volatility surface (IVS) in a manner that excludes static arbitrage. First, we select features to represent the surface and predict them. Second, we use the predicted features to construct the IVS using a deep neural network (DNN) model by incorporating constraints that can prevent static arbitrage. We consider three methods to extract features from the implied volatility data: principal component analysis, variational autoencoder, and sampling the surface. We predict these features using the long short-term memory model. Additionally, we use a long time series of implied volatility data for S&P500 index options to train our models. We find that two feature construction methods (i.e. sampling the surface and variational autoencoders combined with DNN for surface construction) are the best performers in the out-of-sample prediction. Furthermore, both of them substantially outperform a popular regression model. We also find that the DNN model for surface construction not only removes static arbitrage but also significantly reduces the prediction error compared with a standard interpolation method.

Keywords: Implied volatility surface; Static arbitrage; Prediction; Deep learning; Variational autoencoder

1. Introduction

The implied volatility of an option with the given strike K and time to maturity τ is the volatility level that makes the option price from the Black-Scholes formula equal to the observed option price. In practice, traders quote implied volatility for the price of call and put options. The implied volatility surface (IVS) is the collection of implied volatilities as a function of K and τ. It is a fundamental input in various applications, such as derivatives pricing and hedging, volatility trading, and risk management.

Many studies have been conducted on the IVS. As only a finite set of options are traded on each day, an important problem in practice is to interpolate and extrapolate the implied volatilities of these options to obtain the entire surface. Another important problem is predicting the IVS. In this study, we focus on the latter problem. Unlike predicting financial variables such as stock prices (i.e. a single value), predicting the IVS requires the entire surface, which is a bivariate function. Furthermore, the predicted surface must satisfy certain restrictions so that it is free of arbitrage, but such concern is not applicable for predicting fundamental financial variables.

1.1. Related literature

Various methods can be used to solve the interpolation and extrapolation problem of the IVS, and they mainly fall into three categories:

(a) Parametric models: Gatheral (2004) proposes a well-known stochastic volatility inspired (SVI) model for single-maturity implied volatility skews. Gatheral and Jacquier (2014) further generalize the model to obtain a surface that is free of static arbitrage.
(b) Splines: Fengler (2009), Orosi (2015), and Fengler and Hin (2015) develop spline-based models that are free of static arbitrage.
(c) Machine learning models: Neural networks have been utilized to construct the IVS. Refer to Zheng *et al.* (2019) for a neural network model that combines several single-layer neural networks and Ackerer *et al.* (2020) for a deep neural network model that has

*Corresponding author. Email: lfli@se.cuhk.edu.hk

multiple layers. Both papers formulate no-static arbitrage constraints as penalties in their loss functions to obtain a surface that is free of static arbitrage. Bergeron *et al.* (2021) employ a variational autoencoder for constructing the IVS. This method first extracts latent factors for observed implied volatilities through a neural network called encoder. Then, it obtains the surface through another neural network called decoder by using the latent factors, together with moneyness and time to maturity as inputs. Almeida *et al.* (2022) boost the performance of classical parametric option pricing models by fitting a neural network to the residuals of these models. Horvath *et al.* (2021) use a deep neural network to approximate the pricing function of the rough stochastic volatility model in Bayer *et al.* (2016) to facilitate its calibration to implied volatility data.

Many papers have analyzed the dynamics of the IVS. Skiadopoulos *et al.* (2000), Fengler *et al.* (2003), and Cont and Da Fonseca (2002) apply the principal component analysis (PCA) to the IVS data. The first two papers perform PCA for smiles of different maturities, while the third reference performs PCA directly on the surface. Cont and Da Fonseca (2002) find that the first three eigenmodes, which are solutions to the generalized eigenvalue problem in the PCA analysis, already explain most of the variations in their data; these eigenmodes are associated with the level, skewness, and convexity of the IVS. The authors approximate the IVS on each day using the linear combination of the first three eigenmodes. They also modeled the dynamics of each coefficient in the combination by using a first-order autoregressive model. Fengler *et al.* (2007) develop a semiparametric factor model for the IVS, where they assume that the IVS is given by a sum of basis functions, each of which uses moneyness and time to maturity as inputs. Factor loadings are modeled by a vector autoregressive process. The basis functions and the loadings are estimated by weighted least squares with the weight given by a kernel function. Bloch (2019) proposes a general modeling framework. He uses several risk factors to represent the surface and models each parameter (corresponding to a risk factor) using a neural network. In particular, he provides three ways to obtain the risk factors: using the first three eigenmodes from PCA in Cont and Da Fonseca (2002), the parameters of a polynomial model for the IVS, or the parameters of a stochastic volatility-type model (e.g. the SVI model). Explanatory variables, such as the spot price, volume, the volatility index (VIX), are used as inputs to the neural networks for these parameters. Cao *et al.* (2020) model the relationship between the expected daily change in the implied volatility of S&P500 index options by a multilayer feedforward neural network and they use the daily index return, VIX, moneyness, and time to maturity as inputs. They find that, by adding the index return and VIX as features, their model can significantly improve the model in Hull and White (2017). One can use all the models mentioned in this paragraph to predict the IVS of a future date. However, their predicative performance is not assessed in these papers. Furthermore, all these papers do not show how to obtain the dynamics of the IVS that is free of arbitrage.

Several papers directly address the dynamic prediction problem. Dellaportas and Mijatović (2014) predict the implied volatilities of a single maturity by forecasting the parameters in the SABR model (see Hagan *et al.* 2002) from a time series model. As the SABR model is arbitrage-free, the predicted implied volatility smile has no arbitrage. However, their approach does not apply to the entire surface because the SABR model cannot fit the surface well. Goncalves and Guidolin (2006) and Bernales and Guidolin (2014) model the IVS as a polynomial of time-adjusted moneyness and time to maturity. To predict the IVS of a future date, they use the forecasted coefficients of the polynomial from a time series model. Audrino and Colangelo (2010) develop a regression tree model through boosting to predict the IVS. Chen and Zhang (2019) apply the long short-term memory (LSTM) model with the attention mechanism to predict the IVS. Bloch and Böök (2020) predict the IVS by predicting the risk factors that drive the dynamics of the IVS using temporal difference backpropagation models. Zeng and Klabjan (2019) use tick data on options to construct the IVS at high frequency by using a support vector regression model. Almeida *et al.* (2022) predict the IVS by using parametric option pricing models boosted by neural networks. A common issue with all these papers is that the predicted IVS is not necessarily arbitrage-free. Recently, Ning *et al.* (2021) introduce an interesting approach to simulate arbitrage-free IVSs over time. They first calibrate an arbitrage-free stochastic model to the IVS data and then generate the model parameters of a future date from a variational autoencoder. The future IVS is obtained under the stochastic model using the generated model parameters. However, they do not use their approach to predict the IVS.

1.2. Our contributions

The contributions of this paper are twofold. First, we provide a general framework to predict and simulate the IVS in a manner that excludes static arbitrage. This is a new feature provided by our approach compared with existing methods for predicting the IVS. Second, we show how to construct features to represent the IVS and develop some successful prediction models in this framework.

Our framework has two steps:

- *Step 1*: We select features to represent the IVS and predict them. The predicted features are mapped to a discrete set of implied volatilities. If the task is simulation, we simulate the features in this step.
- *Step 2*: We construct the entire IVS from the discrete set of implied volatilities in Step 1 through a deep neural network (DNN) model by considering the constraints that prevent static arbitrage.

This framework is completely flexible as users can construct features and predict them in their own ways. Furthemore, any results obtained in Step 1 can be converted to a surface that is free of static arbitrage through the DNN model in Step 2. However, the predicted surface from our framework may admit dynamic arbitrage opportunities as we do not enforce constraints that prevent dynamic arbitrage in our model. Doing so would be difficult in our framework because

it is data driven and assumes no stochastic model for the underlying asset.

The accuracy of predicting the IVS clearly hinges on the selected features and the model for prediciting them. In general, one can use features extracted from the IVS data, together with exogenous variables to represent the surface. This paper focuses on how to extract features. We consider three approaches: using the eigenmodes from the PCA analysis of Cont and Da Fonseca (2002), applying the variational autoencoder (VAE) to extract latent factors for the IVS, and directly sampling the IVS on a discrete grid of moneyness and time to maturity. PCA can be regarded as a parametric approach that approximates the change of the log IVS using a linear combination of eigenmodes. The VAE approach is more general than the PCA as it offers a flexible nonlinear factor representation of the surface. The sampling approach is completely nonparametric.

To predict the extracted features, we utilize the long short-term memory (LSTM) model, which is a popular deep learning model for sequential data. Our choice is motivated by the success of deep learning in a range of prediction problems in finance. (See, e.g. Borovykh *et al.* 2017, Sezer *et al.* 2020 for various financial time series, Sirignano 2019, Sirignano and Cont 2019 for limit order books, Sadhwani *et al.* 2021 for mortgage risk, and Yan *et al.* 2018 for tail risk in asset returns.)

By training our models using data of 9.5 years and putting them to test in a period of 2.5 years, we find that both the sampling and the VAE approach are quite successful in predicting the IVS. The error of the PCA approach is almost three times of the other two. This indicates that prediction based on a linear combination of eigenmodes is not accurate.

Another important finding is that the DNN model in Step 2 not only serves the purpose of constructing an arbitrage-free surface but also crucial for prediction. Compared with a standard interpolation method for the IVS, using the DNN model can reduce the out-of-sample prediction error substantially for the sampling and VAE approach.

Our paper is related to Bloch (2019), which also provides a general and appealing framework, but they differ in many ways. First, Bloch (2019) does not consider how to obtain arbitrage-free surfaces. Second, he constructs features for the IVS using PCA and parametric models (e.g. the polynomial or SVI model). We provide two different approaches for feature construction in this paper. In general, predicting features from parametric models can be problematic (see Remark 1). Our empirical results also suggest that the prediction model based on features from PCA underperforms the models based on features constructed using our methods. Third, he proposes to model each feature by a neural network. We model these features jointly by one model (LSTM in our implementation). Having separate neural network models for the features may fail to capture potential dependence among them unless they are independent. Finally, he does not provide any empirical study to validate his models.

The rest of the paper is organized as follows. Section 2 provides background information on the IVS, including the definition of implied volatility, conditions ensuring that there is no static arbitrage, an interpolation method, and our data. Section 3 presents the two-step framework for the prediction and simulation. Section 4 shows the empirical results and compares different models. Finally, Section 5 concludes with remarks for future research.

2. IVS

We provide some background knowledge on the IVS in this section.

2.1. Implied volatility

Consider a European call option on a dividend paying asset S_t with maturity date T and strike price K. Set $\tau = T - t$, which is the time to maturity. Denote the risk-free rate by r and the dividend yield by d. Let $F_t(\tau)$ be the forward price at t for time to maturity τ. It is given by $F_t(\tau) = S_t \mathrm{e}^{(r-q)\tau}$. We will write F_t below to simplify the notation.

Under the Black-Scholes model, the call option price at time t is given by

$$C(F_t, K, \tau, \sigma) = \mathrm{e}^{-r\tau}\left(F_t N(d_1) - K N(d_2)\right),$$

where

$$\begin{aligned} d_1 &= \frac{-m + \frac{1}{2}\sigma^2\tau}{\sigma\sqrt{\tau}}, \quad d_2 = \frac{-m - \frac{1}{2}\sigma^2\tau}{\sigma\sqrt{\tau}}, \\ m &= \ln\left(\frac{K}{F_t}\right), \end{aligned} \tag{1}$$

and $N(\cdot)$ is the cumulative distribution function of the standard normal distribution. The variable m defined in (1) is known as the log forward moneyness.

It is well-known that the Black-Scholes model is misspecified. To use it in practice, one looks for the level of volatility to match an observed option price; that is, we solve σ from the equation

$$C(F_t, K, \tau, \sigma) = C_{mkt},$$

where C_{mkt} is the observed market price for the call option. The solution is called the implied volatility.

The IVS at a time point is the collection of implied volatilities of options with different K and τ. We prefer to consider the IVS as a function of m and τ because m is a relative coordinate. Hereinafter, the IVS at time t is denoted by $\sigma_t(m, \tau)$. Practitioners also prefer to quote implied volatilities using the Black-Scholes delta (denoted by δ) and τ, where $\delta = \mathrm{e}^{-q\tau}N(d_1)$.

2.2. Static arbitrage-free conditions

Conditions that ensure the IVS is free of static arbitrage have been well investigated in the literature (see, e.g. Roper 2010, Gulisashvili 2012). We summarize them in the proposition below.

PROPOSITION 1 *Consider an IVS $\sigma(m, \tau)$, and suppose that the following conditions are satisfied:*

(1) *(Positivity) $\sigma(m, \tau) > 0$ for every (m, τ).*

(2) *(Twice Differentiability) For every* $\tau > 0, m \to \sigma(m,\tau)$ *is twice differentiable.*
(3) *(Monotonicity) For every* $m \in \mathbb{R}, \tau \to \sigma(m,\tau)^2\tau$ *is increasing, or equivalently*

$$\ell_{\text{cal}}(m,\tau) = \sigma(m,\tau) + 2\tau\partial_\tau\sigma(m,\tau) \geq 0. \quad (2)$$

(4) *(Durrleman's Condition) For every* (m,τ),

$$\ell_{\text{but}}(m,\tau) = \left(1 - \frac{m\partial_m\sigma(m,\tau)}{\sigma(m,\tau)}\right)^2 - \frac{(\sigma(m,\tau)\tau\partial_m\sigma(m,\tau))^2}{4} + \tau\sigma(m,\tau)\partial_{mm}\sigma(m,\tau) \geq 0. \quad (3)$$

(5) *(Large Moneyness Behavior) For every* τ, $\sigma^2(m,\tau)$ *is linear as* $|m| \to +\infty$.

Then, $\sigma(m,\tau)$ *is free of static arbitrage.*

Condition 3 implies that $\sigma(m,\tau)$ is free of calendar spread arbitrage, and condition 4 guarantees the absence of butterfly arbitrage. In Section 3.4, we implement these conditions to achieve an arbitrage-free surface.

2.3. Interpolation for the IVS

On a given day, implied volatilities can only be calculated for a discrete set of (m,τ) pairs, which correspond to options that are listed on that day. Suppose that we are given $\{\sigma(m_i,\tau_i) : i = 1,\ldots,n\}$ on a day. These given points can be interpolated using various approaches to obtain the entire IVS. Here, we consider a simple and popular parametric model proposed in Dumas *et al.* (1998); hereinafter it will simply be called DFW. This model assumes

$$\bar{\sigma}(m,\tau) = \max(0.01, a_0 + a_1 m + a_2\tau + a_3 m^2 + a_4\tau^2 + a_5 m\tau), \quad (4)$$

where a floor of 0.01 is set to prevent the implied volatility of the model from becoming too small or even negative. The coefficients $a_0,\ldots,a_5$ can be estimated by regression.

Another popular non-parametric approach to estimate the entire implied volatility surface uses the Nadaraya–Watson (NW) estimator (Härdle 1990), which is given by

$$\bar{\sigma}(m,\tau) = \frac{\sum_{i=1}^n \sigma(m_i,\tau_i)\, g(m - m_i, \tau - \tau_i)}{\sum_{i=1}^n g(m - m_i, \tau - \tau_i)},$$

where

$$g(x,y) = \frac{1}{2\pi}\exp\left(-\frac{x^2}{2h_1}\right)\exp\left(-\frac{y^2}{2h_2}\right)$$

is a Gaussian kernel. The estimator involves two hyperparameters h_1 and h_2, which are bandwidths, and they determine the degree of smoothing. If the parameters are too small, a bumpy surface is generated. If they are too large, important details in the observed data are lost after smoothing. When implementating this approach, on each day, we apply the five fold cross-validation to the implied volatility data of this day to select the best pair of (h_1,h_2) from a grid of values for them.

Table 1. Average RMSE of NW and DFW.

	DFW	NW
RMSE	0.018	0.026

Table 1 presents the average root mean squared error (RMSE) of interpolation using these two methods, where the average is taken over days in our training period. The DFW model is more accurate and is our choice for further study. The larger error of the NW estimator is very likely caused by applying the same bandwidth (h_1,h_2) to all points, whereas our implied volatility data is non-uniformly distributed in the (m,τ) space.

2.4. Data

The dataset used for this paper contains IVSs for the S&P500 index options from January 1, 2009, to December 31, 2020. We obtained the data from OptionMetrics through the Wharton Research Data Services. On each day, we have implied volatilities of a set of (δ,τ) pairs with δ going from 0.1 to 0.9 with an increment of 0.05 and $\tau = 10, 30, 60, 91, 122, 152, 182, 273, 365, 547, 730$ calendar days. As we consider the IVS as a function of m and τ, we convert δ to m using

$$m = \tfrac{1}{2}\sigma^2\tau - \sigma\sqrt{\tau}N^{-1}\left(e^{q\tau}\delta\right).$$

This results in implied volatilies on different days for different grids of moneyness, but the same grid of τ. We denote by $\mathcal{I}_{d,t}$ the set of (m,τ) pairs for observed implied volatilities at time t. In total, we have data on 3021 days and on each day 374 points are observed from the IVS (i.e. the size of $\mathcal{I}_{d,t}$ is 374). To demonstrate salient features of the IVS for index options, we calculate the average of $\sigma_t(\delta,\tau)$ over t for all observed (δ,τ) pairs, and plot the average values as a surface in figure 1(a) in terms of δ and τ. In figure 1(b), we show the implied volatility curves for different maturities as functions of the log forward moneyness. A volatility skew is clearly observed for each τ and remains quite steep even for large maturities.

As the methods that we apply later cannot be used if the (m,τ)-grid changes daily, we have to fix it. We use the following grid for (m,τ):

$$\mathcal{I}_0 = \{(m,\tau) : m \in \mathcal{M}_0, \tau \in \mathcal{T}_0\}, \quad (5)$$

where

$$\mathcal{M}_0 = \{\log(x) : x \in \{0.6, 0.8, 0.9, 0.95, 0.975, 1, 1.025, 1.05, 1.1, 1.2, 1.3, 1.5, 1.75, 2\}\},$$
$$\mathcal{T}_0 = \{i/365 : i \in \{10, 30, 60, 91, 122, 152, 182, 273, 365, 547, 730\}\}.$$

As the set of τ is fixed over time in the data, we simply adopt the set as the grid for τ, but change the time unit to year

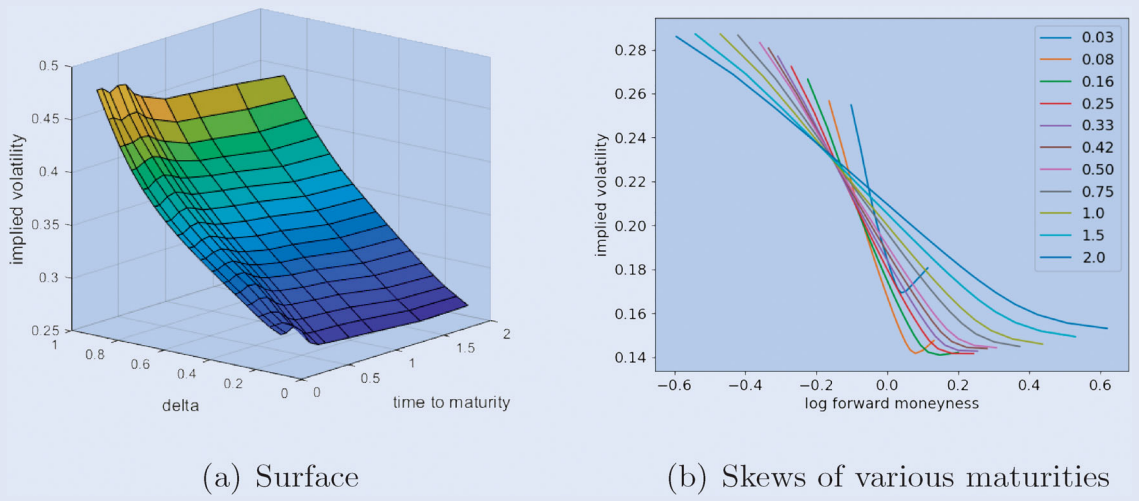

Figure 1. Average implied volatility surface of S&P500 index options with maturity up to 2 years and the skews of different maturities. In plot (b), we show the skews as functions of the log forward moneyness. (a) Surface and (b) Skews of various maturities.

(τ was quoted in days initially). For the grid of moneyness, we first obtain the minimum value and maximum value of m in our dataset and set $[\log(0.6), \log(2)]$ as the range, which is slightly wider than the range from the minimum to the maximum. Then, we create a non-uniform grid on this range so that the grid is denser near ATM. As $\mathcal{I}_0$ is different from the observed grid for (m, τ) on a day, we use the DFW model given in (4) to obtain the implied volatilities on $\mathcal{I}_0$. Eventually, at every t, we have a set of 154 implied volatilities

$$\bar{\Sigma}_t = \{\bar{\sigma}_t(m, \tau) : (m, \tau) \in \mathcal{I}_0\}, \tag{6}$$

which can be deemed as a sample of the IVS.

3. Two-step framework

Consider the IVS process $\{\sigma_t(m, \tau), t \geq 0\}$. We intend to predict $\sigma_{T+1}(m, \tau)$ (the entire surface) given the information available at T. In reality, we do not observe the entire IVS on a day, but only the implied volatilities of a finite number of (m, τ) pairs. Furthermore, the observed (m, τ) pairs can vary daily. Another important problem is how we can ensure that the predicted surface is free of static arbitrage. We propose a two-step framework to address these problems:

- *Step 1*: We select a feature vector Z_t to represent $\sigma_t(m, \tau)$ for every t. Given $\{Z_0, \ldots, Z_T\}$, we predict Z_{T+1} and the predictor is denoted by $\hat{Z}_{T+1}$.
- *Step 2*: We map $\hat{Z}_{T+1}$ to F_{T+1}, a discrete set of implied volatilities at $T + 1$, using some function h; that is, $F_{T+1} = h(\hat{Z}_{T+1})$. We predict the IVS at $T + 1$ as $\hat{\sigma}_{T+1}(m, \tau) = f(m, \tau, F_{T+1})$, where f is a deep neural network (DNN) that outputs an IVS that is free of static arbitrage.

We illustrate the framework using a flowchart in figure 2.

In this study, we focus on the day-ahead prediction. However, our framework can clearly be applied to predict the IVS for any time horizon by replacing $T + 1$ with $T + m$, where m is the length of the prediction horizon. The framework is flexible enough to accommodate various features and different prediction models for them. We explore some choices in this study. Function h is determined according to the selected features.

3.1. *Feature extraction*

We consider several methods to extract features from the implied volatility data.

Method 1 (SAM) We directly use the sampled implied volatility set $\bar{\Sigma}_t$ (see (6)) to represent the entire implied volatility surface. Thus, the feature vector $Z_t = \bar{\Sigma}_t$, which is a 154-dimensional vector in our data. We call this method as the sampling approach (SAM). Here, function h is the identity function, i.e. $F_{T+1} = \hat{Z}_{T+1}$, because the predicted $\hat{Z}_{T+1}$ is a set of implied volatilities.

While having a high-dimensional feature vector can better approximate the surface, predicting it may be more difficult. Thus, natually we can consider some dimension reduction techniques to extract features, leading us to the following two methods.

Method 2 (PCA) Cont and Da Fonseca (2002) applied the surface principle component analysis (PCA) to dissect the dynamics of IVSs. We follow their approach here. As a fixed (m, τ)-grid is required, we consider $\{\bar{\Sigma}_t, t \geq 0\}$. Define $X_t(m, \tau) = \ln \bar{\sigma}_t(m, \tau)$, where $\bar{\sigma}_t(m, \tau) \in \bar{\Sigma}_t$ and

$$U_t(m, \tau) = \ln \bar{\sigma}_t(m, \tau) - \ln \bar{\sigma}_{t-1}(m, \tau) \quad \text{for } (m, \tau) \in \mathcal{I}_0.$$

Then, we perform PCA on $\{U_t(m, \tau), (m, \tau) \in \mathcal{I}_0\}$, which is a 154-dimensional random vector in our data. Let $\mathbf{K}$ be the covariance matrix with $K(x_1, x_2) = \text{cov}(U(x_1), U(x_2))$, $x_1, x_2 \in \mathcal{I}_0$. We solve the eigenvalue problem

$$\mathbf{K} f_k = v_k f_k,$$

where $v_k \geq 0$ is the kth eigenvalue and f_k is the associated normalized eigenvector. We sort the eigenvalues in descending order and use the linear combination of the first K

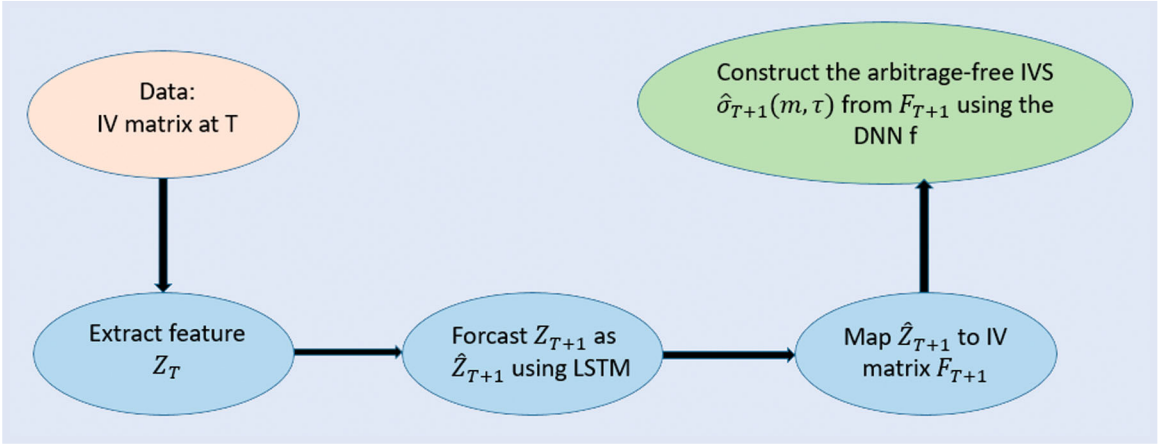

Figure 2. Workflow of the two-step framework.

eigenvectors to approximate $X_t(m,\tau) - X_0(m,\tau)$, which is

$$X_t(m,\tau) - X_0(m,\tau) \approx \sum_{k=1}^{K} x_k(t) f_k(m,\tau),$$

where the coefficient

$$x_k(t) = \langle X_t - X_0, f_k \rangle,$$

the inner product between the vectors $X_t - X_0$ and f_k. Consequently, we have

$$\bar{\sigma}_t(m,\tau) \approx \bar{\sigma}_0(m,\tau) \exp\left(\sum_{k=1}^{K} x_k(t) f_k(m,\tau)\right).$$

Thus, we have $Z_t = (x_1(t), \ldots, x_K(t))$ as the feature vector for $\sigma_t(m,\tau)$. Typically, a small K already explains most of the variation in the data, so the feature vector is low dimensional. Let $\hat{x}_k(T+1)$ be the predicted kth coefficient at $T+1$. In this approach, we have

$$F_{T+1} = h(\hat{Z}_{T+1}) = \bar{\sigma}_0(m,\tau) \exp\left(\sum_{k=1}^{K} \hat{x}_k(T+1) f_k(m,\tau)\right). \tag{7}$$

Method 3 (VAE) The VAE is proposed in Kingma and Welling (2013). This approach extracts latent factors to represent given data through an encoder and then attempts to generate synthetic data through a decoder to resemble the given data. Specifically, the method works as follows (see figure 3 for the graphical illustration):

- Let Y be the input data vector and H be the vector of d latent variables. The components of H are independent, and H follows a multivariate normal distribution with mean vector $\mu(Y)$ and standard deviation vector $\sigma(Y)$.
- The encoder is modeled by a feedfoward neural network (FNN), denoted by N_E. $\mu(Y)$ and $\sigma(Y)$ are ouputs of N_E.
- $H = \mu(Y) + \sigma(Y) \odot \epsilon$, where $\epsilon \sim \mathcal{N}(0, I_d)$ with I_d as the d-by-d identity matrix and $\odot$ is the Hadamard product.
- The decoder is modeled by another FNN, denoted by N_D, with H as the input. The output $\hat{Y} = N_D(H)$.

The loss function for training the VAE has two parts. The first part is the mean squared loss between the synthetic and original data given by

$$\text{RE} = \frac{1}{M} \sum_{i=1}^{M} \left(Y_i - \hat{Y}_i\right)^2,$$

where M is the batch size. The second part is the Kullback-Leibler (KL) divergence between the parameterized normal distribution and $N(0, I)$ given by

$$\text{KL} = \frac{1}{2} \sum_{k=1}^{d} \left(-1 - \log \sigma_k^2 + \sigma_k^2 + \mu_k^2\right),$$

where μ_k and σ_k are the mean and standard deviation of the kth latent variable, respectively. The loss function is defined as

$$\mathcal{L}(\theta_{VAE}, F) = \text{RE} + \beta \text{KL}.$$

Adding the KL divergence term encourages the model to encode a distribution that is as close to normal as possible, and the hyperparameter β measures the extent of regularization.

In our problem, $Y_t = \bar{\Sigma}_t$, and we set $Z_t = \mu(Y_t)$. With the predicted $\hat{Z}_{T+1}$, $F_{T+1} = h(\hat{Z}_{T+1}) = N_D(\hat{Z}_{T+1})$; that is, the h function is given by the decoder FNN.

Remark 1 A natural technique to extract features from the IVS data is using a model for the interpolation and extrapolation problem surveyed at the beginning of Section 1. For example, one can treat the parameters in parametric models such as the surface SVI model in Gatheral and Jacquier (2014) as features for the IVS. One advantage of using such a parametric model is that its parameters have intuitive meanings that are easily understood by traders (Bloch 2019). In our study, we calibrate the surface SVI model to our training data. However, there are two problems in predicting these calibrated parameters. First, they seem to be too volatile to be predicted well in our long training period. Second, certain constraints ensuring absence of arbitrage are not satisfied after prediction. The second one is probably a lesser issue as no

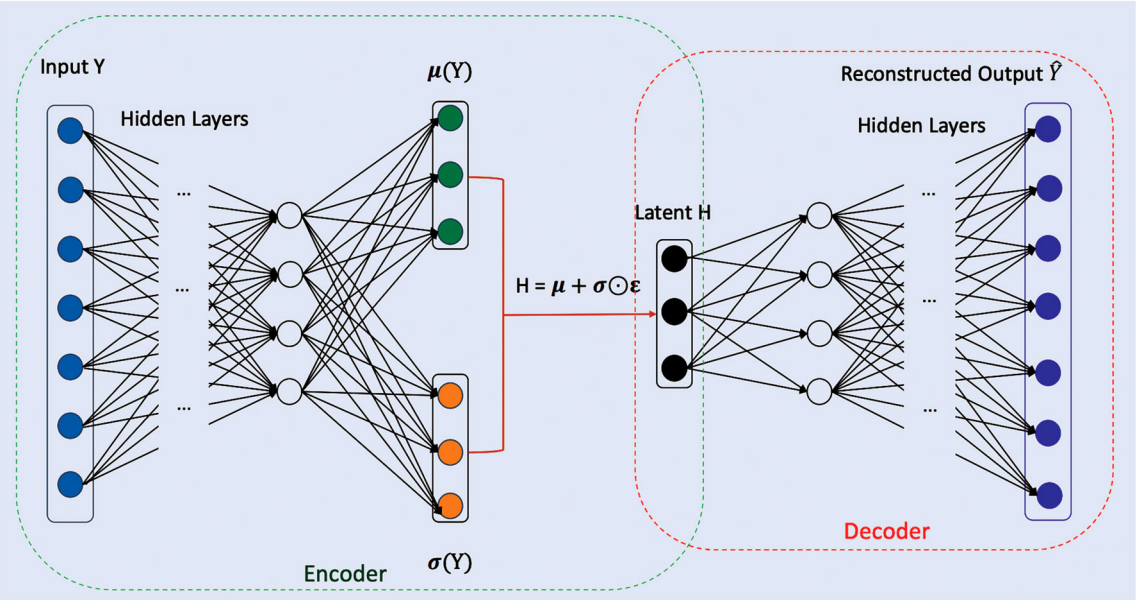

Figure 3. Structure of VAE.

arbitrage can be restored using the DNN model in Step 2 of our framework. However, one cannot solve the first problem easily. There might also be catches in using other interpolation and extrapolation models. For instance, the B-spline model of Fengler and Hin (2015) is accurate for interpolating and extrapolating the surface on a single day. One can use the control net of the B-spline model as features. However, it may vary considerably per day, making it difficult to predict. Therefore, we do not pursue these ideas for extracting features in this study.

3.2. *Feature prediction*

To predict Z_{T+1} from $\{Z_0, \ldots, Z_T\}$, one can consider all types of models. In our experiment, we use the LSTM model (Hochreiter and Schmidhuber 1997), which is a popular deep learning model for sequential data prediction, and its success has been demonstrated in many problems. We use the model in the following way for our problem:

- For any T, consider

$$Z_T^1 = \frac{1}{22}\sum_{t=T-21}^{T} Z_t, \quad Z_T^2 = \frac{1}{5}\sum_{t=T-4}^{T} Z_t, \quad Z_T^3 = Z_T.$$

  The first two represent the monthly and weekly moving averages at T, respectively. We predict Z_{T+1} using Z_T^1, Z_T^2, Z_T^3, which can be regarded as long, medium, and short-term features. A similar approach is taken by Corsi (2009) and Chen and Zhang (2019).
- Let h_j be a hidden state that represents a summary of information from $\{Z_T^1, \ldots, Z_T^j\}$. Set $h_0 = 0$. For $j = 1, 2, 3$, calculate

$$r_j = \sigma_g\left(W_r Z_T^j + U_r h_{j-1} + b_r\right),$$
$$i_j = \sigma_g\left(W_i Z_T^j + U_i h_{j-1} + b_i\right),$$
$$o_j = \sigma_g\left(W_o Z_T^j + U_o h_{j-1} + b_o\right),$$
$$g_j = \sigma_h\left(W_g Z_T^j + U_g h_{j-1} + b_g\right),$$
$$c_j = r_j \odot c_{j-1} + i_j \odot g_j,$$
$$h_j = o_j \odot \sigma_h\left(c_j\right),$$
$$y_j = \sigma_h\left(W_y h_j + b_y\right).$$

  All W, U, b are parameters, and σ_g, σ_h are the sigmoid function $1/(1+\exp(-x))$ and the tanh function $(1-\exp(-2x))/(1+\exp(-2x))$, respectively, for activation. At j, i_j, r_j and o_j represent the input, forget, and output gates.
- Finally, we predict Z_{T+1} as

$$\hat{Z}_{T+1} = \sigma_{out}(W_{out} y_3 + b_{out}).$$

  The range of Z_{T+1} varies in our framwork depending on the feature extraction method. For SAM, we use the ReLU function $\max(x, 0)$ for σ_{out} as Z_{T+1} is positive. For VAE and PCA, as Z_{T+1} can take any real value, we do not use any nonlinear activitation function and simply set σ_{out} as the identity function.

Then, we write the feature prediction model as

$$\hat{Z}_{T+1} = p(Z_T^1, Z_T^2, Z_T^3; \theta_{\mathcal{P}}),$$

where $\theta_{\mathcal{P}}$ is the vector of parameters involved.

3.3. *DNN model for surface construction*

With $F_{T+1} = h(\hat{Z}_{T+1})$, we construct the entire IVS from F_{T+1} using a DNN illustrated in figure 4. The neural network is a feedforward one with inputs F_{T+1}, m, τ. The output is an

implied volatility of the input (m, τ) pair. We use the Softplus function $\ln(1 + \exp(x))$ as the activation function of the output layer because it makes the output nonnegative and twice differentiable so that the first two no-arbitrage conditions in Proposition 1 are fulfilled.

3.4. Loss functions and no-arbitrage conditions

Suppose that the time horizon in our data is given by $\mathcal{T} = \{1, 2, \ldots, T\}$. Let q_t be the number of observed implied volatilities on the surface at t (in our data $q_t = 374$ for all t, but in general it could change over time). The loss function of the feature prediction part is given by

$$\mathcal{L}_{\mathcal{P}}(\theta_{\mathcal{P}}) = \frac{1}{T}\sum_{t=1}^{T} \left\| z_t - p\left(Z_T^1, Z_T^2, Z_T^3; \theta_{\mathcal{P}}\right)\right\|^2.$$

We minimize $\mathcal{L}_{\mathcal{P}}(\theta_{\mathcal{P}})$ to train the LSTM model. For the construction of the IVS, one can set the loss function as

$$\mathcal{L}_{\mathcal{S}}(\theta_{\mathcal{S}}) = \frac{1}{T}\sum_{t=1}^{T}\frac{1}{q_t}\sum_{i=1}^{q_t}(f\,(m_i, \tau_i, F_t; \theta_{\mathcal{S}}) - \sigma_t\,(m_i, \tau_i))^2.$$

However, minizing $\mathcal{L}_{\mathcal{S}}(\theta_{\mathcal{S}})$ to train the surface construction model cannot guarantee that the output surface is arbitrage-free. By design the output of the DNN model satisfies the first two conditions in Proposition 1, but does not necessarily fulfil the other three ones. Inspired by Zheng *et al.* (2019) and Ackerer *et al.* (2020), we incorporate conditions 3, 4, and 5 for no arbitrage into our training by formulating them as penalties in the loss function.

First, we create the following synthetic grids to faciliate the calculation of the penalty functions:

$$\begin{aligned}\mathcal{I}_{\mathrm{C34}} &= \Big\{(m, \tau) : m \in \Big\{x^3 : x \in \big[-(-2m_{\min})^{1/3}, \\ &\qquad (2m_{\max})^{1/3}\big]_{40}\Big\}, \tau \in \mathcal{T}_1\Big\}, \\ \mathcal{I}_{\mathrm{C5}} &= \{(m, \tau) : m \in \{6m_{\min}, 4m_{\min}, 4m_{\max}, 6m_{\max}\}, \\ &\qquad \tau \in \mathcal{T}_1\},\end{aligned}$$

where $m_{\min} = \log(0.6)$, $m_{\max} = \log(2)$, $[a, b]_{40}$ means a uniform grid over the interval $[a, b]$ with it divided into 40 equal parts,

$$\mathcal{T}_1 = \left\{\exp(x) : x \in [\log(1/365), \max\left(\log\left(\tau_{\max} + 1\right)\right)]_{40}\right\},$$

and $\tau_{\max} = 730/365$. The grid $\mathcal{I}_{\mathrm{C34}}$ is used for the penalty calculation associated with conditions 3 and 4, while $\mathcal{I}_{\mathrm{C5}}$ is used for condition 5. These grids are different from the (m, τ)-grid in (5) used for sampling. In particular, $\mathcal{I}_{\mathrm{C34}}$ has 1600 points, which are much higher than the 154 points on the sampling grid, and it covers a much wider range for both m and τ. We use such a dense grid on a wide region to reduce the chance of missing points on the surface at which a significant violation of the no-arbitrage conditions occurs. As condition 5 considers the large moneyness behavior, we analyze the moneyness levels that are extremely negative or positive.

We denote by $\mathcal{L}_{\mathrm{C}j}(\theta_{\mathcal{S}})$ the penalty function for the jth condition ($j = 3, 4, 5$). For conditions 3 and 4, they are given by

$$\begin{aligned}\mathcal{L}_{\mathrm{C3}}(\theta_{\mathcal{S}}) &= \frac{1}{T}\sum_{t=1}^{T}\frac{1}{|\mathcal{I}_{\mathrm{C34}}|} \\ &\quad\times \sum_{(k_i,\tau_i)\in\mathcal{I}_{\mathrm{C34}}} \max\left(0, -\ell_{\mathrm{cal}}\left(m_i, \tau_i, F_t; \theta_{\mathcal{S}}\right)\right), \\ \mathcal{L}_{\mathrm{C4}}(\theta_{\mathcal{S}}) &= \frac{1}{T}\sum_{t=1}^{T}\frac{1}{|\mathcal{I}_{\mathrm{C34}}|} \\ &\quad\times \sum_{(k_i,\tau_i)\in\mathcal{I}_{\mathrm{C34}}} \max\left(0, -\ell_{\mathrm{but}}\left(m_i, \tau_i, F_t; \theta_{\mathcal{S}}\right)\right),\end{aligned}$$

where $\ell_{\mathrm{cal}}(m_i, \tau_i, F_t; \theta_{\mathcal{S}})$ and $\ell_{\mathrm{but}}(m_i, \tau_i, F_t; \theta_{\mathcal{S}})$ are defined as in (2) and (3) with σ replaced f. For condition 5, it is equivalent to the case wherein the second-order derivative of $\sigma^2(m, \tau)$ goes to zero as $|m| \to \infty$, where $\partial^2_{mm}\sigma^2(m, \tau) = \sigma(m, \tau)\partial^2_{mm}\sigma(m, \tau) + (\partial_m\sigma(m, \tau))^2$. Hence, the penalty is

$$\begin{aligned}\mathcal{L}_{\mathrm{C5}}(\theta_{\mathcal{S}}) = \frac{1}{T}\sum_{t=1}^{T}\frac{1}{|\mathcal{I}_{\mathrm{C5}}|}\sum_{(k_i,\tau_i)\in\mathcal{I}_{\mathrm{C5}}} \Big|f\,(m_i, \tau_i, F_t; \theta_{\mathcal{S}})\,\partial^2_{mm} \\ \times f\,(m_i, \tau_i, F_t; \theta_{\mathcal{S}}) + (\partial_m f\,(m_i, \tau_i, F_t; \theta_{\mathcal{S}}))^2\Big|.\end{aligned}$$

Finally, we obtain our loss function for training the DNN model as

$$\mathcal{L}_{\mathrm{C}}(\theta_{\mathcal{S}}) = \mathcal{L}_{\mathrm{S}}(\theta_{\mathcal{S}}) + \lambda(\mathcal{L}_{\mathrm{C3}}(\theta_{\mathcal{S}}) + \mathcal{L}_{\mathrm{C4}}(\theta_{\mathcal{S}}) + \mathcal{L}_{\mathrm{C5}}(\theta_{\mathcal{S}}))$$

for some $\lambda > 0$. One could use a separate penalization parameter for each penalty, but for simplicy, we assume that they are the same. We minimize $\mathcal{L}_{\mathrm{C}}(\theta_{\mathcal{S}})$ to train the DNN model. In our implementation, we choose $\lambda = 1$, which is used in Ackerer *et al.* (2020) in their penalized loss function. We also used other values for λ and found that using $\lambda = 1$ results in the smallest error for the IVS on the training data, and the penalties converge to zero quickly.

Remark 2 To prevent static arbitrage, conditions 3 and 4 in Proposition 1 must hold for every pair of (m, τ). However, in the implementation, we cannot check them at every point in the (m, τ) space. Hence, we consider a dense grid over a wide region (see $\mathcal{I}_{\mathrm{C34}}$). Condition 5 specifies the limiting behavior of $\sigma^2(m, \tau)$ for $|m| \to \infty$. In our implementation, we can only check this condition for very large values of $|m|$ (see $\mathcal{I}_{\mathrm{C5}}$). It is very unlikely that the surface from our DNN model violates these constraints at points not in $\mathcal{I}_{\mathrm{C34}}$ or $\mathcal{I}_{\mathrm{C5}}$ (see figure 6 for the values of these penalties on the test data, which are zero if the DNN model has been trained for a sufficient number of epochs). Nevertheless, one can say that our DNN model yields an IVS that is almost free of static arbitrage.

3.5. Simulation

Our framework can also be used to simulate the IVS over time. We can write the feature transition equation as

$$Z_{T+1} = p(Z_T^1, Z_T^2, Z_T^3; \theta_{\mathcal{P}}) + \varepsilon_{T+1}, \tag{8}$$

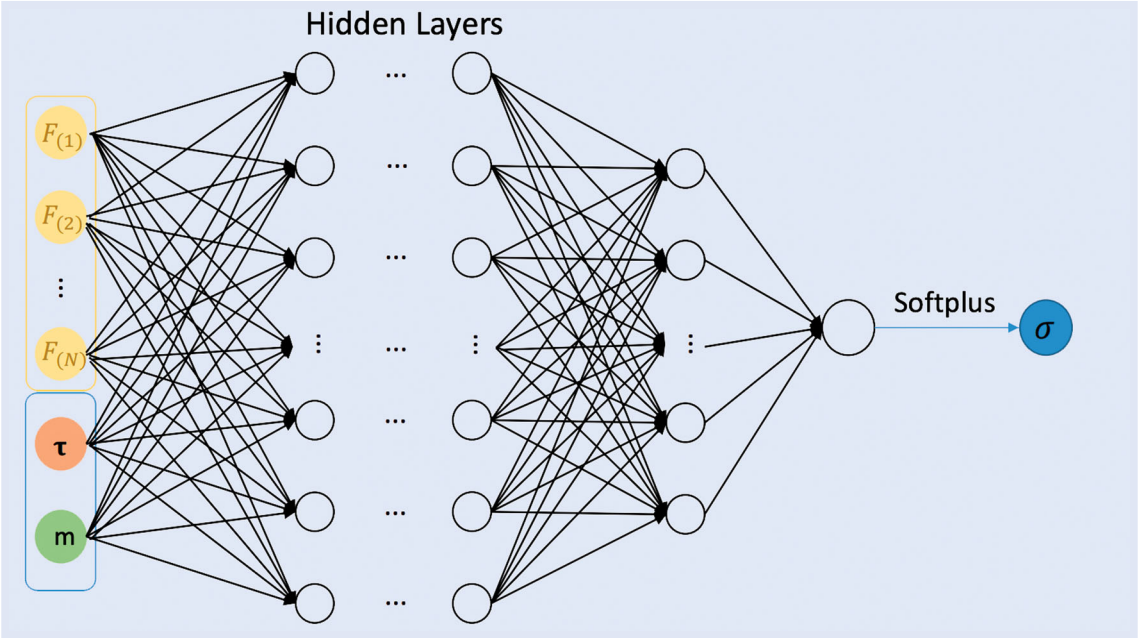

Figure 4. DNN model for implied volatility surface construction.

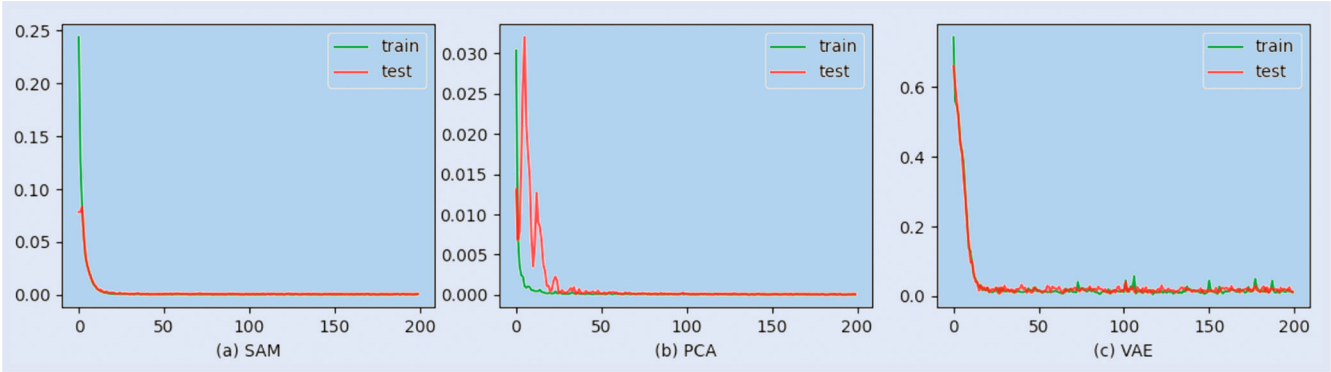

Figure 5. Loss function of the LSTM model. The x variable in each plot is the epoch index.

or

$$Z_{T+1} = p(Z_T^1, Z_T^2, Z_T^3; \theta_{\mathcal{P}}) \exp(\varepsilon_{T+1}), \tag{9}$$

where ε_{T+1} is the error vector at $T+1$. In (8), we assume additive errors, and in (9) we assume multiplicative errors. The multiplicative formulation is more convenient to use than the additive one when the positivity of Z_{T+1} is required.

We assume the error process $\varepsilon_1, \varepsilon_2, \ldots$ is an i.i.d. white noise with mean zero and covariance matrix Σ_ε. After obtaining the estimate of $\theta_{\mathcal{P}}$ by minimizing the loss function $\mathcal{L}_{\mathcal{P}}(\theta_{\mathcal{P}})$, one can calculate the error vector on each day and hence obtain a sample of the errors. We can assume that the error vector follows a multivariate parametric distribution F_η with parameter vector η (e.g. Gaussian) and estimate η from the error sample.

The simulation of Z_{T+1}, given the available information at T, consists of the following steps:

- *Step 1*: Calculate $p(Z_T^1, Z_T^2, Z_T^3; \theta_{\mathcal{P}})$.
- *Step 2*: Simulate ε_{T+1} from F_η or by bootstrapping from the error sample.
- *Step 3*: Calculate Z_{T+1} by (8) or (9).
- *Step 4*: Calculate $\sigma_{T+1}(m,\tau) = f(m,\tau,h(Z_{T+1}))$.

The DNN model f ensures that the output IVS is free of static arbitrage.

4. Empirical results

Recall that our dataset consists of daily implied volatilities of S&P 500 index options from January 1, 2009, to December 31, 2020, with a total of 3021 trading days. We split the data into training and test sets. The training dataset is from January 1, 2009, to June 27, 2018 (about 9.5 years), while the test dataset is from June 28, 2018, to December 31, 2020 (about 2.5 years). In particular, the US stock market crash in 2020 due to the COVID-19 pandemic is included in the test period. On each day, we observe implied volatilities of 374 pairs of (m,τ). Due to the limited amount of training data (about 2390 days), we do not further partition it to create a validation set for hyperparameter tuning.

4.1. *Feature extraction*

The details of the three feature extraction methods can be found in Section 3.1. For each day in the dateset, we extract

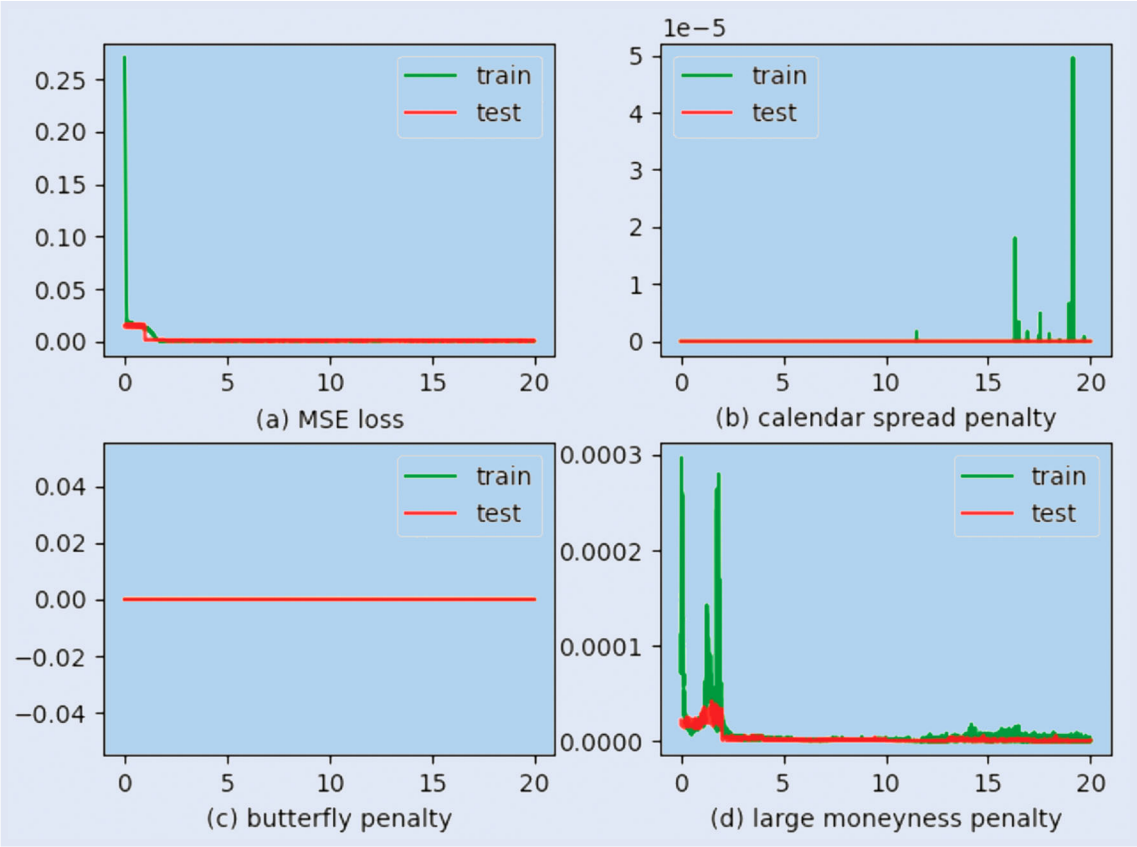

Figure 6. MSE loss and penalties for the three no-arbitrage conditions in the DNN model with features extracted from the sampling approach. The x variable in each plot is the epoch index.

features using these three methods, and some details are as follows:

- For SAM, we use $\bar{\Sigma}_t$ as the feature vector (see (6)), which is a set of implied volatilities on a (m, τ)-grid with 154 points, to represent the entire surface at t.
- For PCA, we follow Cont and Da Fonseca (2002) to use the first three eigenmodes (i.e. $K = 3$ in (7)), which already explain over 96% of the variations in our data.
- For VAE, the FNNs for the encoder and decoder both have three hidden layers with 128 nodes per layer. We use five values for the latent dimension d: 2, 5, 10, 15, 20. Their performances on the test data are presented in table 2, and the difference is small, indicating that the performance of the VAE model is quite robust to the choice of d. The VAE model with $d = 10$ achieves the smallest out-of-sample prediction error.

4.2. *Training of the LSTM and DNN models*

We use the LSTM model to predict the extracted features, as discussed in Section 3.2. For the DNN model for surface construction, we use three hidden layers with 50 neurons on each layer. In the training of both models, we perform the following:

- We initialize the parameters using Xavier initialization (Glorot and Bengio 2010), which can prevent initial weights in a deep network from being either too large or too small. This method sets the weight of the jth layer to follow a uniform distribution given by

$$W^j \sim U\left[-\frac{1}{\sqrt{n_j}}, \frac{1}{\sqrt{n_j}}\right],$$

  where n_j is the number of neurons on the jth layer.
- We use the Adam optimizer with minibatches (Kingma and Ba 2014) to minimize the loss function. Calculating the gradient of the loss function using all the samples can be computationally expensive. Therefore, in each iteration, we only use a minibatch (i.e. a subset) of samples for the gradient evaluation. The Adam optimizer is a popular gradient-descent algorithm, which utilizes the exponentially weighted average of the gradients to accelerate convergence to the minimum.
- We apply batch normalization to the inputs of the neural network (Ioffe and Szegedy 2015). For all the samples in a minibatch, we first estimate the mean and standard deviation of each input in this minibatch. Then, we normalize the input by subtracting its estimated mean and dividing by its estimated standard deviation.

The values of the hyperparameters associated with training and the hidden sizes of the models (i.e. the number of neurons on a hidden layer) are displayed in table 3. We train LSTM and

Table 2. RMSE and MAPE of the VAE-DNN model with different latent dimensions.

	$d = 2$	$d = 5$	$d = 10$	$d = 15$	$d = 20$
Training set					
RMSE	0.0208	0.0207	**0.0205**	0.0210	0.0231
MAPE	7.90%	7.71%	**7.60%**	8.00%	9.20%
Test set					
RMSE	0.0262	0.0258	**0.0248**	0.0253	0.0268
MAPE	9.98%	9.76%	**9.46%**	9.75%	10.33%

Note: The smallest value on a row is highlighted in bold.

Table 3. Hyperparameters in the LSTM and DNN model.

	Epochs	Batch size	Hidden size	Learning rate
LSTM	200	128	12	0.01
DNN	20	1024	50	0.001

DNN for 200 and 20 epochs, respectively. An epoch consists of all the iterations required to work through all the samples in the training set. Thus, it is given by the size of the training data divided by the size of a minibatch.

Figures 5 and 6 show the results of loss on the training and test data as the number of epochs increases. In figure 6, we only plot the DNN results for the model with the features extracted from the sampling approach, and the results of the other two feature extraction approaches are similar. As shown in figure 5, there is no overfitting in the LSTM model, as the test loss is close to the training loss. Similarly, there is no overfitting in the DNN model, as shown in figure 6(a). The values of the penalties for these three no-arbitrage conditions also become zero eventually on the test data. Thus, there is no violation of these conditions on the synthetic grids. It should be noted that, although there are some spikes in the calender spread penalty on the training data, the largest value remains very small, indicating the insignificance of the violation.

4.3. Out-of-sample prediction and model comparison

Let $\hat{\theta}_{\mathcal{P}}$ and $\hat{\theta}_{\mathcal{S}}$ be the estimated parameters from the training data for the LSTM model and the DNN model, respectively. Suppose that the last day in the training period and the entire dataset is denoted by T_{train} and T_{total}, respectively. Set $T_{\text{test}} = T_{\text{total}} - T_{\text{train}}$, which is the number of days in the test period. We perform the out-of-sample test as follows: for every $t > T_{\text{train}}$,

- obtain $\hat{Z}_t = p(Z^1_{t-1}, Z^2_{t-1}, Z^3_{t-1}; \hat{\theta}_{\mathcal{P}})$ and $F_t = h(\hat{Z}_t)$, and
- calculate $\hat{\sigma}_t(m,\tau) = f(m,\tau,F_t)$ for $(m,\tau) \in \mathcal{I}_{d,t}$.

Here, $\mathcal{I}_{d,t}$ is the set of (m,τ)-pairs in the observed implied volatility data at t, which contains 374 points. It is important to note that it is different from $\mathcal{I}_0$, the set of (m,τ)-pairs used for sampling the surface, which has only 154 points. The error for a pair of (m,τ) is given by

$$\hat{\sigma}_t(m,\tau) - \sigma_t(m,\tau),$$

where $\sigma_t(m,\tau)$ is the observed implied volatility at t for this pair (the ground truth). The error not only reflects the prediction error of the LSTM model for the features, but also the interpolation error of the DNN model.

To evaluate the overall out-of-sample prediction performance, we consider two commonly used error measures: root mean squared error (RMSE) and the mean absolute percentage error (MAPE). They are defined as

$$\text{RMSE} = \sqrt{\frac{1}{\sum_{t=1}^{T_{\text{test}}} |\mathcal{I}_{d,t}|} \sum_{t=1}^{T_{\text{test}}} \sum_{(m,\tau)\in\mathcal{I}_{d,t}} \left(\sigma_t(m,\tau) - \hat{\sigma}_t(m,\tau)\right)^2},$$

$$\text{MAPE} = \frac{1}{\sum_{t=1}^{T_{\text{test}}} |\mathcal{I}_{d,t}|} \sum_{t=1}^{T_{\text{test}}} \sum_{(m,\tau)\in\mathcal{I}_{d,t}} \left|\frac{\sigma_t(m,\tau) - \hat{\sigma}_t(m,\tau)}{\sigma_t(m,\tau)}\right|.$$

In our data, $\mathcal{I}_{d,t}$ contains different (m,τ) pairs for a different t, but $|\mathcal{I}_{d,t}| = 374$ for all t.†

We examine various models. In the first step, there are three feature extraction approaches: SAM, PCA and VAE. In the second step, we consider two methods, namely, the DNN model and the DFW model in (4), applied to F_t to predict $\sigma_t(m,\tau)$ at time t. The DNN model yields an arbitrage-free surface, whereas the DFW interpolation model cannot. This leads to six models for comparison: SAM-DNN, SAM-DFW, PCA-DNN, PCA-DFW, VAE-DNN, and VAE-DFW. We also consider a classical benchmark given by the DFW model. At time T, we simply forecast the IVS at $T + 1$ from the DFW model with its coefficients given by their estimates at T.

We display the performances of these models on the test dataset in table 4. For any pair of models, we also perform the Diebold-Mariano (DM) test (Diebold and Mariano 2002) to assess the statistical significance of the difference in the forecast performance as measured by RMSE, and the p-value of the test is presented in table 5. Consider Models 1 and 2. In the DM test, the null hypothesis is that the forecast errors of these two models are equal, while the alternative hypothesis is that the forecast error of Model 1 is less than that of Model 2. Table 5 should be read in the following manner. For any entry of the table, the model on the row is Model 1, and that on the column is Model 2. We consider 1% as the significance level.

Several observations can be made from tables 4 and 5. (1) The best performers in the out-of-sample prediction are SAM-DNN and VAE-DNN. The DM test shows that their

† One should be cautious in comparing the errors reported in different papers. Some papers only evaluate the error on a limited set of (m,τ) pairs. For example, Chen and Zhang (2019) only consider the errors at 45 points in the (m,τ) space.

Table 4. RMSE and MAPE of the six models.

	SAM-DNN	SAM-DFW	PCA-DNN	PCA-DFW	VAE-DNN	VAE-DFW	DFW
Training set							
RMSE	**0.0202**	0.0288	0.0527	0.0608	0.0205	0.0633	0.0346
MAPE	7.98%	11.65%	27.65%	27.88%	**7.60%**	34.21%	14.83%
Test set							
RMSE	**0.0245**	0.0312	0.0544	0.0745	0.0248	0.0647	0.0366
MAPE	9.90%	12.88%	28.93%	30.41%	**9.46%**	32.75%	15.83%

Note: The results of VAE-DNN and VAE-DFW are for $d = 10$. The smallest value on a row is highlighted in bold.

Table 5. p-value of the Diebold-Mariano test with RMSE as the error measure.

	SAM-DNN	SAM-DFW	PCA-DNN	PCA-DFW	VAE-DNN	VAE-DFW	DFW
SAM-DNN	–	2.7e−04	2.2e−16	2.2e−16	0.5639	2.2e−16	4.9e−05
SAM-DFW	0.9997	–	2.2e−16	2.2e−16	0.9999	2.2e-16	0.1495
PCA-DNN	0.9999	0.9999	–	2.2e−16	0.9999	5.3e−08	0.9999
PCA-DFW	0.9999	0.9999	0.9999	–	0.9999	0.9999	0.9999
VAE-DNN	0.4361	1.8e−13	2.2e−16	2.2e−16	–	2.2e−16	3.2e−07
VAE-DFW	0.9999	0.9999	0.9999	1.2e−12	0.9999	–	0.9999
DFW	0.9999	0.8505	2.2e−16	2.2e−16	0.9999	2.2e−16	–

difference is statistically insignificant; hence, they can be considered as equally good. Both of them outperfom the other models with overwhelmingly strong statistical evidence. In particular, these two models constructed in the proposed two-step framework dominate the classical DFW model by a large margin. (2) The out-of-sample errors of SAM-DNN and VAE-DNN are only about one third of the error of PCA-DNN. This result highlights the importance of feature selection in Step 1 for predicting the IVS. While the PCA approach is good for understanding the main factors that drive the IVS movements, the approximation based on a linear combination of eigenmodes is not sufficiently accurate for predicting the IVS. In contrast, the VAE approach is more flexible as it combines the latent factors in a nonlinear manner. Thus, its improvement over PCA can be expected, as confirmed by the results. The sampling approach can be deemed as a nonparametric technique to represent the surface, which can lead to a high-dimensional feature vector (in our data, its dimension is 154). Owing to the power of LSTM, we are able to predict it quite accurately. Using a powerful model such as LSTM for feature prediction is key to the success of the sampling approach. (3) The DNN model for IVS construction in Step 2 also makes significant contributions to improving the prediction accuracy. For each feature extraction method, using the DNN model outperforms the DFW model for surface construction in Step 2 with statistical significance. The improvement in prediction accuracy is already considerable for SAM and even more substantial for VAE. (4) It is worth noting that the SAM-DFW model also performs quite well in the prediction. The SAM-DFW model is simpler than the SAM-DNN model as it uses the simple DFW model, instead of the complex DNN model, for surface construction in Step 2.

We further plot the RMSE and MAPE of each day in the test period for the four models in figure 7. Both SAM-DNN and VAE-DNN are better than PCA-DNN throughout the period, and they also outperform DFW on most days. However, the errors of all four models increase in March 2020, during which the US stock market suffered a meltdown due to the pandemic. The relatively large error signals a shift in the market regime in that period. The features we use in all the models are extracted from the implied volatility data, which do not provide a direct representation of the market regime. To improve their performances, one can further augment the feature vector with exogenous variables such as the index return and VIX, which are proxies of the market regime.

Table 6. Violation of no-arbitrage conditions for calendar spread and butterfly arbitrage.

	SAM-DNN	PCA-DNN	VAE-DNN	DFW
L^-_{cal}	0.0	0.0	0.0	−0.1142
L^-_{but}	0.0	0.0	0.0	−0.0002

For the predicted IVS on the test days, we check for violation of the constraints on the calendar spread arbitrage and butterfly arbitrage. Consider

$$L^-_{cal} = \frac{1}{\sum_{t=1}^{T_{\text{test}}} |\mathcal{I}_{d,t}|} \sum_{t=1}^{T_{\text{test}}} \sum_{(m,\tau)\in\mathcal{I}_{d,t}} \min(\ell_{\text{cal}}(m,\tau), 0),$$

$$L^-_{but} = \frac{1}{\sum_{t=1}^{T_{\text{test}}} |\mathcal{I}_{d,t}|} \sum_{t=1}^{T_{\text{test}}} \sum_{(m,\tau)\in\mathcal{I}_{d,t}} \min(\ell_{\text{but}}(m,\tau), 0).$$

The no-arbitrage constraints require $\ell_{\text{cal}}(m,\tau)$ and $\ell_{\text{but}}(m,\tau)$ to be nonnegative for any (m,τ). In the above quantities, we check $\ell_{\text{cal}}(m,\tau)$ and $\ell_{\text{but}}(m,\tau)$ on the (m,τ) grid for the observed implied volatilities on each test day. A negative value at a pair of (m,τ) indicates violation at this point, and we calculate the average of the negative values over all test days. The results are reported for four models in table 6. While no violation is detected for the three models that use DNN in Step 2, prediction under the DFW model yields IVSs with arbitrage opportunities and the calendar spread arbitrage in particular.

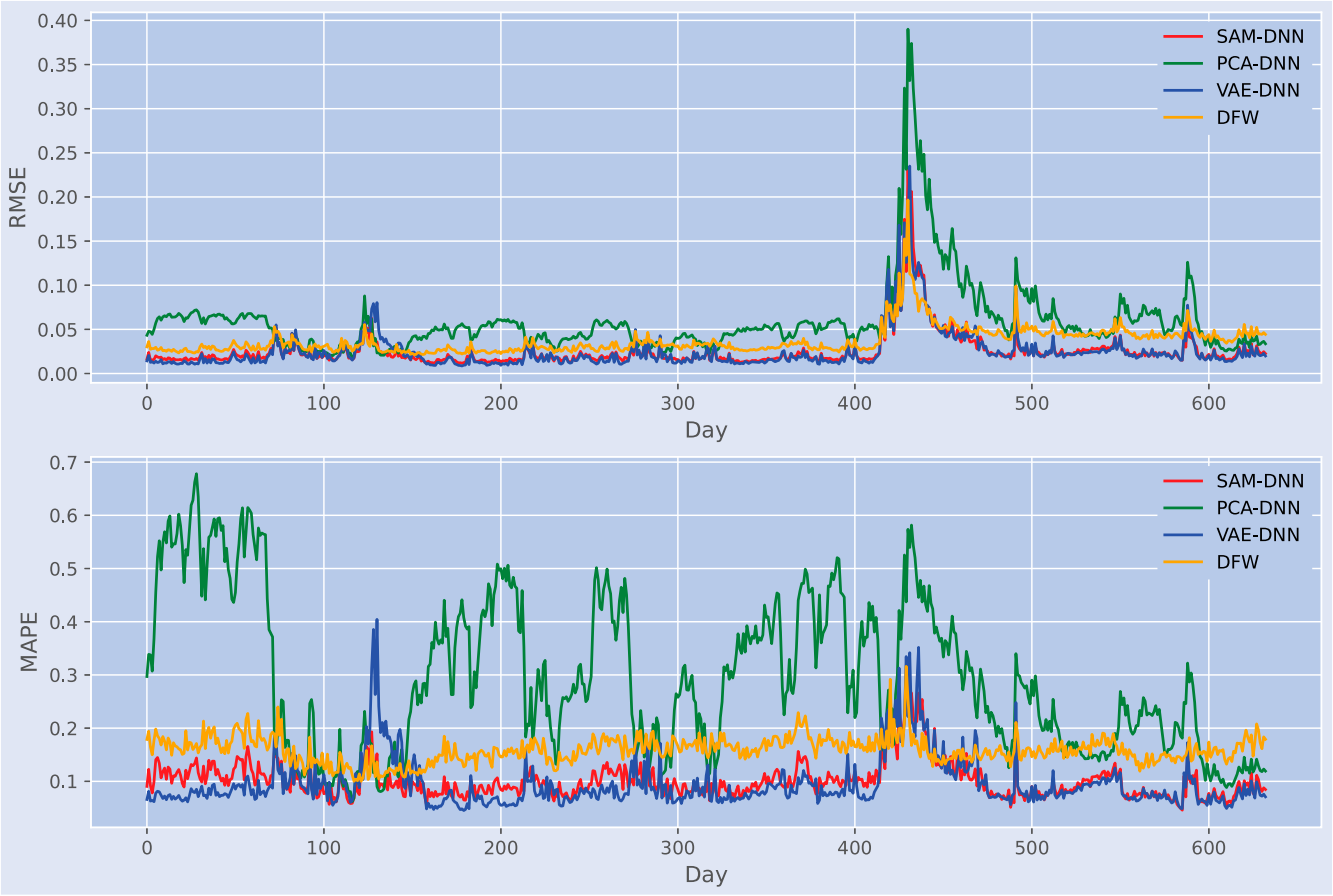

Figure 7. Daily RMSE and MAPE in the test period.

Table 7. Performance of the PCA+DFW model using LSTM and AR(1) for feature prediction.

	PCA(LSTM)+DFW	PCA(AR)+DFW
Training set		
RMSE	0.0608	0.0649
MAPE	27.88%	28.83%
Test set		
RMSE	0.0745	0.0765
MAPE	30.41%	30.47%

Remark 3 For the PCA approach, we have also attempted predicting each expansion coefficient using a separate AR(1) model (autoregressive model of order one), which is used in Cont and Da Fonseca (2002) for modeling the dynamics of these coefficients. One can regard PCA(AR)+DFW as a simple model constructed based on classical statistical techniques without using machine learning. Table 7 shows that predicting the PCA coefficients using the AR(1) model results in similar errors of predicting the IVS, compared with predicting the PCA coefficients using the LSTM model.

5. Conclusion

We developed a flexible two-step framework to predict and simulate the IVS in a manner that prevents static arbitrage. First, we constructed features to represent the IVS and then predicted/simulated them. Second, we constructed an IVS that is free of static arbitrage by using a deep neural network model from the predicted/simulated features. Using this framework, we developed two models that are quite successful in predicting the IVS. One of them extracts features by directly sampling the IVS data on a grid, and the other one extracts latent factors through the encoder neural network in the variational autoencoder. Both models significantly outperform the popular regression model of Dumas *et al.* (1998).

The prediction accuracy of our models can be further improved in the following ways. First, we can add exogenous variables, such as the index return and VIX, to the feature vector to represent the market regime. We expect that these features would boost the prediction power of our models when the market is under stress. Second, we can use more sophisticated deep learning models to predict features, which might be particularly helpful if the feature vector is high dimensional and exhibits complex behavior. In future research, we also plan to apply our models to financial applications, such as hedging, risk management, and options trading.

Acknowledgments

We are grateful to the anonymous reviewers whose comments led to significant improvement in the paper.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

Lingfei Li was supported by Hong Kong Research Grant Council General Research Fund Grant 14206020. Gongqiu Zhang was supported by the National Science Foundation of China Grant 12171408 and by Shenzhen Basic Research Program Project JCYJ20190813165407555 .

References

Ackerer, D., Tagasovska, N. and Vatter, T., Deep smoothing of the implied volatility surface. In *Advances in Neural Information Processing Systems*, edited by H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan and H. Lin, Vol. 33, pp. 11552–11563, 2020 (Curran Associates, Inc: Vancouver, Canada).

Almeida, C., Fan, J., Freire, G. and Tang, F., Can a machine correct option pricing models? *J. Bus. Econom. Statist.*, 2022, forthcoming.

Audrino, F. and Colangelo, D., Semi-parametric forecasts of the implied volatility surface using regression trees. *Stat. Comput.*, 2010, **20**(4), 421–434.

Bayer, C., Friz, P. and Gatheral, J., Pricing under rough volatility. *Quant. Finance*, 2016, **16**(6), 887–904.

Bergeron, M., Fung, N., Poulos, Z., Hull, J.C. and Veneris, A., Variational autoencoders: A hands-off approach to volatility, 2021. Available online at: https://dx.doi.org/10.2139/ssrn.3827447.

Bernales, A. and Guidolin, M., Can we forecast the implied volatility surface dynamics of equity options? Predictability and economic value tests. *J. Bank. Financ.*, 2014, **46**, 326–342.

Bloch, D.A., Neural networks based dynamic implied volatility surface, 2019. Available online at: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3492662.

Bloch, D.A. and Böök, A., Predicting future implied volatility surface using TDBP-learning, 2020. Available online at: https://dx.doi.org/10.2139/ssrn.3739514.

Borovykh, A., Bohte, S. and Oosterlee, C.W., Conditional time series forecasting with convolutional neural networks, 2017. Available online at: https://arxiv.org/pdf/1703.04691v1.pdf.

Cao, J., Chen, J. and Hull, J., A neural network approach to understanding implied volatility movements. *Quant. Finance*, 2020, **20**(9), 1405–1413.

Chen, S. and Zhang, Z., Forecasting implied volatility smile surface via deep learning and attention mechanism, 2019. Available online at: https://dx.doi.org/10.2139/ssrn.3508585.

Cont, R. and Da Fonseca, J., Dynamics of implied volatility surfaces. *Quant. Finance*, 2002, **2**, 45–60.

Corsi, F., A simple approximate long-memory model of realized volatility. *J. Financ. Econom.*, 2009, **7**(2), 174–196.

Dellaportas, P. and Mijatović, A., Arbitrage-free prediction of the implied volatility smile, 2014. Available online at: https://arxiv.org/abs/1407.5528.

Diebold, F.X. and Mariano, R.S., Comparing predictive accuracy. *J. Bus. Econom. Statist.*, 2002, **20**(1), 134–144.

Dumas, B., Fleming, J. and Whaley, R.E., Implied volatility functions: Empirical tests. *J. Finance*, 1998, **53**(6), 2059–2106.

Fengler, M.R., Arbitrage-free smoothing of the implied volatility surface. *Quant. Finance*, 2009, **9**(4), 417–428.

Fengler, M.R. and Hin, L.-Y., Semi-nonparametric estimation of the call-option price surface under strike and time-to-expiry no-arbitrage constraints. *J. Econom.*, 2015, **184**(2), 242–261.

Fengler, M.R., Härdle, W.K. and Villa, C., The dynamics of implied volatilities: A common principal components approach. *Rev. Deriv. Res.*, 2003, **6**(3), 179–202.

Fengler, M.R., Härdle, W.K. and Mammen, E., A semiparametric factor model for implied volatility surface dynamics. *J. Financ. Econom.*, 2007, **5**(2), 189–218.

Gatheral, J., A parsimonious arbitrage-free implied volatility parameterization with application to the valuation of volatility derivatives. Presentation at Global Derivatives & Risk Management, Madrid, 2004.

Gatheral, J. and Jacquier, A., Arbitrage-free SVI volatility surfaces. *Quant. Finance*, 2014, **14**(1), 59–71.

Glorot, X. and Bengio, Y., Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 249–256, 2010 (JMLR Workshop and Conference Proceedings).

Goncalves, S. and Guidolin, M., Predictable dynamics in the S&P 500 index options implied volatility surface. *J. Bus.*, 2006, **79**(3), 1591–1635.

Gulisashvili, A., *Analytically Tractable Stochastic Stock Price Models*, 2012 (Springer Science & Business Media: Berlin, Germany).

Hagan, P.S., Kumar, D., Lesniewski, A.S. and Woodward, D.E., Managing smile risk. *Wilmott Mag.*, 2002, **1**, 249–296.

Härdle, W., *Applied Nonparametric Regression*, 1990 (Cambridge University Press: Cambridge).

Hochreiter, S. and Schmidhuber, J., Long short-term memory. *Neural Comput.*, 1997, **9**(8), 1735–1780.

Horvath, B., Muguruza, A. and Tomas, M., Deep learning volatility: A deep neural network perspective on pricing and calibration in (rough) volatility models. *Quant. Finance*, 2021, **21**(1), 11–27.

Hull, J. and White, A., Optimal delta hedging for options. *J. Bank. Financ.*, 2017, **82**, 180–190.

Ioffe, S. and Szegedy, C., Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pp. 448–456, 2015 (PMLR).

Kingma, D.P. and Ba, J., Adam: A method for stochastic optimization, 2014. Preprint, arXiv:1412.6980. Available online at: https://doi.org/10.1016/j.asoc.2020.106181.

Kingma, D.P. and Welling, M., Auto-encoding variational Bayes, 2013. Available online at: https://arxiv.org/abs/1312.6114.

Ning, B., Jaimungal, S., Zhang, X. and Bergeron, M., Arbitrage-free implied volatility surface generation with variational autoencoders, 2021. Preprint, arXiv:2108.04941.

Orosi, G., Arbitrage-free call option surface construction using regression splines. *Appl. Stoch. Models Bus. Ind.*, 2015, **31**(4), 515–527.

Roper, M., Arbitrage free implied volatility surfaces, 2010. Available online at: https://talus.maths.usyd.edu.au/u/pubs/publist/preprints/2010/roper-9.pdf.

Sadhwani, A., Giesecke, K. and Sirignano, J., Deep learning for mortgage risk. *J. Financ. Econom.*, 2021, **19**(2), 313–368.

Sezer, O.B., Gudelek, M.U. and Ozbayoglu, A.M., Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. *Appl. Soft. Comput.*, 2020, **90**, 106181.

Sirignano, J.A., Deep learning for limit order books. *Quant. Finance*, 2019, **19**(4), 549–570.

Sirignano, J. and Cont, R., Universal features of price formation in financial markets: Perspectives from deep learning. *Quant. Finance*, 2019, **19**(9), 1449–1459.

Skiadopoulos, G., Hodges, S. and Clewlow, L., The dynamics of the S&P 500 implied volatility surface. *Rev. Deriv. Res.*, 2000, **3**(3), 263–282.

Yan, X., Zhang, W., Ma, L., Liu, W. and Wu, Q., Parsimonious quantile regression of financial asset tail dynamics via sequential learning. In *Advances in Neural Information Processing Systems*, edited by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi and R. Garnett, Vol. 31, 2018 (Curran Associates, Inc: Montréal, Canada).

Zeng, Y. and Klabjan, D., Online adaptive machine learning based algorithm for implied volatility surface modeling. *Knowl. Based Syst.*, 2019, **163**, 376–391.

Zheng, Y., Yang, Y. and Chen, B., Gated deep neural networks for implied volatility surfaces, 2019. Available online at: https://arxiv.org/pdf/1904.12834.pdf.