Proceedings of the 2020 Winter Simulation Conference
K.-H. Bae, B. Feng, S. Kim, S. Lazarova-Molnar, Z. Zheng, T. Roeder, and R. Thiesing, eds.

ONLINE RISK MEASURE ESTIMATION VIA NATURAL GRADIENT BOOSTING

Xiaoting Cai

School of Management
Shanghai University
99 Shangda Road, Baoshan District
Shanghai, 200444, CHINA

Yang Yang

School of Data Science
The Chinese University of Hong Kong, Shenzhen
No. 2001 Longxiang Blvd.
Shenzhen, Guangdong, 518172, CHINA

Guangxin Jiang

School of Management
Harbin Institute of Technology
2 Yikuang Street, Nangang District
Harbin, Heilongjiang, 150001, CHINA

ABSTRACT

Estimating risk measures of a portfolio with financial derivatives in real time is an important yet challenging task. Since it costs some time to conduct simulation experiments, traditional nested simulation methods may not be well applied. In this paper, we propose to use a natural gradient boosting (NGBoost) approach to estimate the conditional probability density function of the loss (or the value) of the portfolio under the framework of offline simulation online application, and then estimate the real-time risk measures based on the conditional probability density function. By training only one learning model, the NGBoost approach can obtain various real-time risk measures, which measure the risk from different aspects and provides a comprehensive understanding of risk. Numerical examples show the effectiveness of the NGBoost approach.

1 INTRODUCTION

Risk measures, such as the probability of large loss (PLL), Value-at-Risk (VaR), conditional Value-at-Risk (CVaR), are important indicators of risk of financial portfolios and even stability of financial institutions. To estimate these risk measures, simulation is commonly used since it can provide accurate estimates as long as the stochastic processes of the underlying risk factors can be simulated, see Hong et al. (2014) for a survey. However, simulation methods are in general very time consuming, especially when the portfolios contain derivative securities like options and swaps whose prices need to be estimated via additional simulation efforts. This is known as the nested simulation, see Lee and Glynn (2003), Gordy and Juneja (2010). Some enhanced methods are proposed to improve the computational efficiency of the nested simulation. For example, Liu and Staum (2010) used stochastic kriging to improve the estimation efficiency; Broadie et al. (2015) proposed a regression method and Hong et al. (2017) derived a kernel estimation method to avoid the inner simulation.

In practice, the real-time turbulence in the financial market often leads to dramatic changes in risk factors, and at the same time affects the risk measures. In this regard, risk managers urgently need to estimate real-time risk measures to determine whether the risks of the financial products are under control. For example, when a risk manager monitors the risk measures of a portfolio that contains a lot of stock

options during some major risk events (such as Brexit), the underlying stock prices may change frequently, leading that the risk measures of this portfolio may change frequently and sometimes exceed the preset risk level. So, at this moment, the portfolio manager needs to adjust the portfolio immediately to ensure the safety of this portfolio. To achieve such a goal, the portfolio manager needs to know risk measures of the portfolio in real time based on the changing underlying risk factors (like stock prices). However, almost all the methods mentioned above cannot be applied in real-time estimation, since they need to observe the current risk factors first, and then conduct time-consuming simulation procedures. Recently, Hong and Jiang (2019) proposed a simulation framework called offline simulation online application (OSOA) which attempts to adapt the simulation methods from offline design tools to online decision tools. Particularly, Jiang et al. (2019) proposed a simulation analytics approach based on this framework to estimate the real-time PLL (also known as exceedance probability). Logistic regression is applied and the coefficients of the logistic model are estimated by the data generated in the past nested simulation experiments. By this approach, a good real-time estimation of the PLL and a good real-time classifier of risk levels are provided.

However, the approach proposed in Jiang et al. (2019) may not be applied to estimate other risk measures like VaR and CVaR directly. That is, the logistic regression is good to model the PLL, but it may not be a good model for VaR and CVaR, where quantile regression may be more suitable. In other words, estimating different risk measures requires different learning models, and it makes the OSOA framework complicated and limited in the process of risk measurement. In this paper, we propose a method that can estimate risk measures (PLL, VaR, CVaR) in real time by training only one learning model. Noticing that almost all the risk measures are statistics of the loss of a portfolio at a given time (Broadie et al. (2015)), so that if we know the probability density function of the loss, we can estimate various types of risk measures. In the classical nested simulation, after generating different risk factor scenarios and estimating the loss on each scenario, we actually obtain the empirical probability distribution of the loss of the portfolio, so that we can easily obtain the PLL, VaR, and CVaR by doing additional basic operations (like calculating the mean of the indicators and summation of tails). Similarly, if we can obtain the probability distribution of the loss of the portfolio conditional on risk factors, we can also obtain the risk measures conditional on these risk factors (we call these risk measures the *real-time risk measures*, since they can be used to monitor the risk in real time). Moreover, by observing the probability distribution, portfolio managers can have a comprehensive understanding of the loss of the portfolio so that they can make smarter decisions to control the risk, which is difficult to be achieved through a single risk measure. Therefore, in order to obtain the different types of risk measures in real time and better understand the risk of the portfolio, we need to estimate the conditional probability distribution of the loss of the portfolio (or the conditional probability distribution of the value of the portfolio).

For estimating conditional distribution, there are some methods in the statistical literatures, such as the kernel method (Rosenblatt (1969), Fan et al. (1996)), the spline method (Stone (1994), Maacsse and Truong (1999)), and Bayesian density regression (Dunson et al. (2007)). Recently, machine learning-based methods have been applied in conditional density estimation (Lakshminarayanan et al. (2017), Cousins and Riondato (2019), Duan et al. (2019)). In this paper, we apply the natural gradient boosting (NGBoost) method proposed by Duan et al. (2019) to estimate the conditional probability density function of the loss (or the value) of the portfolio based on the data generated in the nested simulation experiment. Specifically, we use NGBoost to estimate the distribution parameters in a preset distribution family that models the probability distribution of the loss (or the value) of the portfolio, and the natural gradient is used in the gradient boosting to ensure the good and stable learning efficiency. For more details about gradient boosting, one may refer to Friedman (2001).

The main contributions of this article are summarized as follows: (a) Based on the paper of Jiang et al. (2019), this paper proposes that using only one learning model can effectively estimate all risk measures, which makes OSOA method more convenient and universal; (b) This paper proposes an effective machine learning method, NGBoost, to estimate the conditional density function of the loss (or the value) of the

portfolio; (c) It is proposed that risk managers can analyze the overall risk distribution intuitively with the help of probability distribution, which is more conducive to risk prevention and decision-making. The rest of the paper is organized as follows. In Section 2, we review the nested simulation and the OSOA framework in estimating the real-time risk measures. In Section 3, we derive the NGBoost approach to estimate the conditional probability distributions, and then to estimate the real-time risk measures. Numerical results are provided in Section 4, followed by conclusions and discussions in Section 5.

2 PRELIMINARIES

In this section, we first review the nested simulation in estimating risk measures of a portfolio with financial derivatives, and then introduce the OSOA framework in estimating the real-time probability of large loss in Jiang et al. (2019). Suppose that $\mathbf{S}(t) = (S_1(t), S_2(t), \ldots, S_m(t))^\top$ is a vector of the risk factors that influences risk measures of a portfolio, and $\mathbf{S}(t)$ is a Markov process defined on a probability space $(\Omega, \mathscr{F}, \mathbf{P})$. The portfolio has q financial derivatives, whose values are $V_i(t), i = 1, 2, \ldots, q$ at t, and the number of the ith financial derivative is $w_i, i = 1, 2, \ldots, q$. So the value of the portfolio at time t is $\Phi(t) = \sum_{i=1}^{q} w_i V_i(t)$. Throughout, we assume that $V_i(t)$ has no closed-formula and needs to be estimated via Monte Carlo simulation. Let $L(t) \triangleq \Phi(0) - \Phi(t)$ be the loss of the portfolio at time t.

Now we are at time 0 and there is a fixed future time T at which portfolio loss $L(T)$ needs to be considered, and risk measures are derived based on the distribution of $L(T)$. When the given portfolio loss threshold is Q, PLL indicates the probability that the portfolio loss is more than or equal to Q, and is given by

$$\text{PLL} \triangleq \Pr\{L(T) \geq Q\}. \tag{1}$$

VaR represents the possible loss of the financial product at future time T under the given confidence level α, and is given by

$$\text{VaR}_\alpha \triangleq \inf\{Q : \Pr\{L(T) \geq Q\} \leq 1 - \alpha\}. \tag{2}$$

That is, the $1-\alpha$ quantile of the loss of the portfolio. CVaR represents the corresponding loss expectation when the loss of the financial product exceeds VaR, where the given confidence level is α. And it is given by

$$\text{CVaR}_\alpha \triangleq \mathbb{E}[L(T) \cdot \mathbf{1}\{L(T) \geq \text{VaR}_\alpha\}], \tag{3}$$

where $\mathbf{1}\{\cdot\}$ is the indictor function. In the portfolio, we assume that the expiration dates of these financial derivatives are the same and given by τ with $\tau > T$. In this paper, we want to know the real-time risk measures (4) - (6) (real-time PLL, real-time VaR, and real-time CVaR) given the observed risk factor scenario $\mathbf{S}(u) = s_u$ at time $0 < u < T$ instantly.

$$\text{PLL}(s_u) \triangleq \Pr\{L(T) \geq Q | \mathbf{S}(u) = s_u\}, \tag{4}$$

$$\text{VaR}_\alpha(s_u) \triangleq \inf\{Q : \Pr\{L(T) \geq Q\} \leq 1 - \alpha | \mathbf{S}(u) = s_u\}. \tag{5}$$

$$\text{CVaR}_\alpha(s_u) \triangleq \mathbb{E}[L(T) \cdot \mathbf{1}\{L(T) \geq \text{VaR}_\alpha\} | \mathbf{S}(u) = s_u]. \tag{6}$$

2.1 Nested Simulation

We first review how to use the nested simulation to estimate risk measures (1)-(3). The nested simulation consists of the outer and the inner simulation. The outer simulation is used to generate risk factor scenarios $\mathbf{S}_1(T), \mathbf{S}_2(T), \ldots$, and the inner simulation is used to estimate corresponding portfolio loss $L(\mathbf{S}_i(T)), i = 1, 2, \ldots$. The specific procedure is shown in Figure 1 and presented as follows:

Outer Simulation: Under the real probability measure, simulate m sample paths of the underlying risk factor scenarios, and record the scenarios at time T as $\{\mathbf{S}_i(T)\}_{i=1}^m$.

Inner Simulation:

1. For each scenario $\mathbf{S}_i(T)$, simulate n sample paths $\{\mathbf{S}_i^k(t), t \in [T,\tau]\}_{l=1}^n$ from time T to time τ;
2. Based on these inner simulation sample paths, estimate the value of the financial derivative $\widehat{\mathbf{V}}_i = \{\widehat{V}_{ij}(T)\}_{j=1}^q$ with $\{\widehat{V}_{ij}(T) = 1/n\sum_{l=1}^n G_j\left(\mathbf{S}_i^k(t), t \in [T,\tau]\right)\}_{j=1}^q$ under the risk-neutral measure, where $G_j(\cdot)$ represents the payoff function of the jth financial derivative;
3. Estimate the corresponding portfolio value under scenario $\mathbf{S}_i(T)$ as $\widehat{\Phi}_i(T) = \sum_{j=1}^q w_j V_{ij}(T)$;
4. Estimate portfolio loss L under scenario $\mathbf{S}_i(T)$ as $\widehat{L}_i(T) = \Phi(0) - \widehat{\Phi}_i(T)$.

Estimation: Estimate risk measures (1)-(3) based on $\{\widehat{L}_i(T)\}_{i=1}^m$.

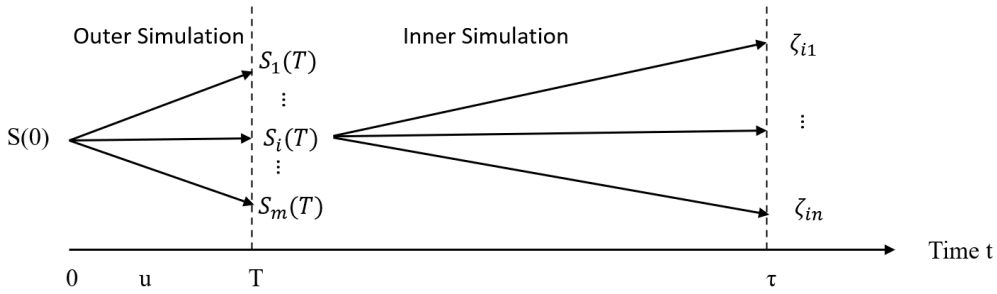

Figure 1: Nested simulation to estimate risk measures.

Notice that based on $\{\mathbf{S}_i(T)\}_{i=1}^m$ and $\{\widehat{L}_i(T)\}_{i=1}^m$, we actually know the empirical distribution of the portfolio loss $L(T)$, and we then can estimate the risk measures. For example, the estimated PLL is given by

$$\widehat{\mathrm{PLL}} = \frac{1}{m}\sum_{i=1}^m \mathbf{1}\{\widehat{L}_i(T) \geq Q\}.$$

In practice, the number of the financial derivatives in a portfolio may be very large, and the maturities of the financial derivatives may be very long, even decades. So we need to set large m (the number of risk factor scenarios) and n (the number of simulation sample paths in pricing the financial derivative) to estimate the risk measures accurately. As a result, it will cost too much time to conduct the nested simulation, leading that the nested simulation cannot meet our needs for real-time estimation of risk measures. Although some scholars have proposed methods to speed up the nested simulation from the perspective of optimizing experimental design and replacing the inner simulation (Broadie et al. (2015), Hong et al. (2017)), etc., it is still difficult to meet the requirement of real-time estimation.

2.2 Online Risk Monitoring

Recently, Jiang et al. (2019) have proposed to use logistic regression to model the real-time (conditional) probability of large loss based on the data generated in the past nested simulation experiment. After obtaining the estimated $\widehat{\mathrm{PLL}}(s_u)$, the observed real-time risk factors s_u can be plugged into the function to calculate the real-time PLL simultaneously. Such a framework to conduct simulation is also called offline simulation online application, see Hong and Jiang (2019) for more details.

Figure 2 interprets the way to estimate the real-time PLL in Jiang et al. (2019). First, a nested simulation is conducted to estimate $\mathrm{PLL} = \Pr\{L(T) \geq Q\}$ in advance, and all the data including $\{\mathbf{S}_1(t), \mathbf{S}_2(t), \ldots, \mathbf{S}_m(t)\}$ for $t \in (0,T]$ and $\{L_1(T), L_2(T), \ldots, L_m(T)\}$ are stored. Notice that, in Jiang et al. (2019), other intermediate data like $\{\hat{V}_{ij}(T)\}_{j=1}^q$ for $i = 1,2,\ldots,m$, may be stored to improve the accuracy of the estimation by using some enhancing techniques, which are not considered in this paper. Let $\Upsilon(T) = \mathbf{1}\{L(T) \geq Q\}$, so we have

$$\mathrm{PLL}(\mathbf{S}(u)) = \mathbb{E}[\Upsilon(T)|\mathbf{S}(u)].$$

According to Jiang et al. (2019), suppose that

$$\mathrm{PLL}(\mathbf{S}(u)) = \varphi(\chi(\mathbf{S}(u)), \beta(u)) \triangleq \exp\left(\beta(u)^\top \chi(\mathbf{S}(u))\right) / \left(1 + \exp\left(\beta(u)^\top \chi(\mathbf{S}(u))\right)\right), \tag{7}$$

where $\beta(u)$ represents the corresponding coefficient vector, which needs to be estimated, and $\chi(\mathbf{S}(u))$ represents the preset basis function computed from $\mathbf{S}(u)$ to make the logistic model more flexible. Both $\chi(\mathbf{S}(u))$ and $\beta(u)$ depend on u. Denote $\{\Upsilon_1(T), \Upsilon_2(T), \ldots, \Upsilon_m(T)\} \triangleq \{\mathbf{1}\{L_1(T) \geq Q\}, \mathbf{1}\{L_2(T) \geq Q\}, \ldots, \mathbf{1}\{L_m(T) \geq Q\}\}$. For logistic regression model (7), the maximum likelihood estimation (MLE) is usually used to estimate the coefficients. According to the posterior probability of Bernoulli distribution and the training data pairs $\{(\chi(\mathbf{S}_i(u)), \Upsilon_i(T))\}_{i=1}^m$, the log-likelihood function is derived by

$$L_m(\beta(u)) = \frac{1}{m}\sum_{i=1}^{m} \{\Upsilon_i(T)\log(\varphi(\chi(\mathbf{S}_i(u)), \beta(u))) + (1 - \Upsilon_i(T))\log(1 - \varphi(\chi(\mathbf{S}_i(u)), \beta(u)))\}, \tag{8}$$

and the estimated coefficient vector is given by $\hat{\beta}(u) = \arg\max_\beta L_m(\beta(u))$ for fixed u. Then the real-time PLL estimator is given by

$$\widehat{\mathrm{PLL}}(\mathbf{S}^r(u)) = \varphi(\chi(\mathbf{S}^r(u)), \hat{\beta}(u)). \tag{9}$$

That is, when we observe a real-time risk factor scenario $\mathbf{S}^r(u)$ in online application, we plug it into Equation (9) to calculate the corresponding real-time PLL.

According to the approach above, the requirement for estimating risk measures in real time can be met. However, we found that if we want to estimate different risk measures, we need to train different learning models. For example, if we want to calculate the real-time VaR, logistic model may not be a good model, and quantile model and quantile regression may be used. In the next section, we provide a general-purpose approach to estimate real-time risk measures.

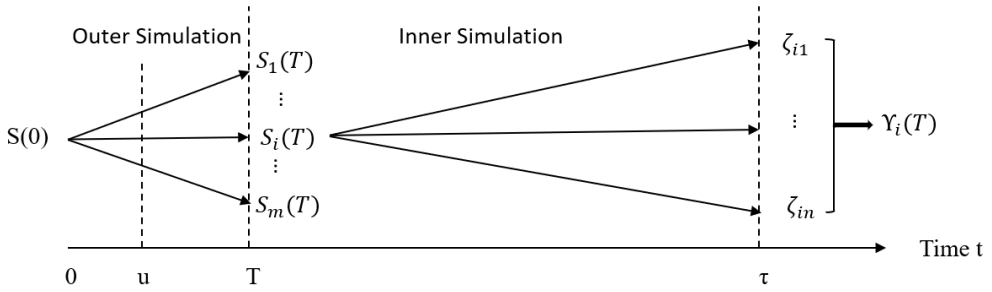

Figure 2: Online risk monitoring via the nested simulation data.

3 CONDITIONAL DENSITY ESTIMATION WITH NGBOOST

In this section, we propose a method to estimate real-time risk measures (specifically, we consider the real-time PLL, the real-time VaR, and the real-time CVaR) with only one learning model under the offline simulation online application framework. We know that all risk measures are statistics of the loss of the portfolio (Broadie et al. 2015). From this perspective, if we know the conditional probability density function of the portfolio value Φ, which is equivalent to know the conditional probability density function of the loss of the portfolio, then we can estimate various types of real-time risk measures. In this paper, we assume that the conditional probability distributions of the portfolio value Φ given $\mathbf{S}(u)$ are within a distribution family. We use p_θ to denote the conditional density function, where parameter θ determines the unique probability distribution, and θ depends on $\mathbf{S}(u)$. For example, when p_θ is probability density function of normal distribution, $\theta = (\mu, \sigma)$, where μ and σ are the mean and the variance of the normal

distribution, respectively. Therefore, our goal boils down to determine θ according to the given risk factor scenario $\mathbf{S}(u)$, i.e., to determine $\theta(\mathbf{S}(u))$. For ease of notation and in accordance with the notation in conditional density estimation area, we use $\mathbf{x}$ to represent the risk factor scenario $\mathbf{S}(u)$ and drop the subscript u for convenience.

Boosting is a commonly used and effective ensemble method for improving the model predictions of given learning models. The basic idea is to form a strong learner with the additive combination of several base learners (weak learners) like stumps, see Freund and Schapire (1997), Collins et al. (2002) and Hastie et al. (2011) for more details. However, almost all traditional boosting methods and learning models only fit training data to predict an estimator of the scalar function $\mathbb{E}[y|\mathbf{x}]$, without considering to output a conditional probability density function $p(y|\mathbf{x})$, where $\mathbf{x}$ is a vector of observed features (risk factor scenario in this paper) and y is the prediction target (value of the portfolio in this paper). Recently, Duan et al. (2019) have proposed an approach based on boosting method to estimate conditional probability distribution. Specifically, they assumed that $p(y|\mathbf{x})$ was a parametric conditional probability density function, and the distribution parameter θ depended on the observed feature vector $\mathbf{x}$. So the conditional probability density function can be rewritten as $p_{\theta(\mathbf{x})}(y)$.

Using traditional boosting methods to estimate $\theta(\mathbf{x})$ will inevitably encounter difficulties brought by the process of parameterization and multi-parameter boosting. Duan et al. (2019) improved Gradient Boosting (Friedman 2001) whose output value is $\mathbb{E}[y \mid \mathbf{x}]$, by replacing the ordinary gradient with the natural gradient. The natural gradient represents the steepest direction in Riemann space. When the proper scoring rule is $\mathscr{C}(\theta, y)$, according to Amari (1998), the natural gradient is given by

$$g = I_{\mathscr{C}}(\theta)^{-1} \nabla \mathscr{C}(\theta, y),$$

where $I_{\mathscr{C}}(\theta)$ denotes Riemannian metric of the statistical manifold at θ induced by $\mathscr{C}$, and $\nabla \mathscr{C}(\theta, y)$ denotes the ordinary gradient with respect to θ of $\mathscr{C}(\theta, y)$. Here we assume that both $I_{\mathscr{C}}(\theta)^{-1}$ and $\nabla \mathscr{C}(\theta, y)$ exist. The natural gradient has the advantage of being invariant to parameterization, and it is stable and effective in estimating parameters. Duan et al. (2019) named the gradient boosting with natural gradient the *NGBoost*, which can better solve the problems mentioned above in estimating $\theta(\mathbf{x})$.

In this paper, we focus on the NGBoost to estimate parameter $\theta(\mathbf{S}(u))$ in our setting, and obtain the probability density function of the value of the portfolio $p_{\theta(\mathbf{S}(u))}(\Phi(T))$ and then calculate real-time risk measures. Next, we will introduce the steps of the NGBoost to estimate the parameter vector in a given distribution family.

3.1 NGBoost

In NGBoost, we can choose different basic learners. In this paper, we set the base learner as decision trees, which is similar to Duan et al. (2019) and let $\mathscr{C}(\theta, y)$ as the scoring rule (e.g., the log-likelihood in MLE), and it should be noted that scoring rule must meet certain conditions for obtaining natural gradients, see Amari (1998) and Duan et al. (2019) for more details.

Suppose there is a training set containing m input-output pairs $D = \{\mathbf{x}_i, y_i\}_{i=1}^m$. The procedure of NGBoost is given as follows.

Initialization: Set base learners f, parametric probability density function p_θ, proper scoring rule $\mathscr{C}$, boosting iterations N and learning rate η.

Step 1: Calculate initial $\theta^{(0)} = \arg\min_\theta \sum_{i=1}^m \mathscr{C}(\theta, y_i)$.

Step 2: Perform N natural gradient boosting stages. Note that at stage h ($h = 1, 2, \cdots, N$), each component of $\theta^{(h)}$ corresponds to one base learner. That is, when p_θ is set as normal density function, $\theta = (\mu, \sigma)$ and $f^{(h)} = \left(f_\mu^{(h)}, f_\sigma^{(h)}\right)$, where $f_\mu^{(h)}$ and $f_\sigma^{(h)}$ represent the base learners of parameters μ and σ at stage h, respectively. The specific process at stage h is as follows:

Step 2.h.1: Calculate generalized natural gradient $\{g_i^{(h)} = I_{\mathscr{C}}\left(\theta_i^{(h-1)}\right)^{-1} \nabla_\theta \mathscr{C}\left(\theta_i^{(h-1)}, y_i\right)\}_{i=1}^m$;

Step 2.h.2: Taking $\{\mathbf{x}_i\}_{i=1}^m$ as input, $\{g^{(h)}\}_{i=1}^m$ as output, fit $f^{(h)}$. For example, when fitting decision trees, in order to ensure efficiency, usually use sample variance as splitting criterion;

Step 2.h.3: Let single scalar $\rho^{(h)} = \arg\min_\rho \sum_{i=1}^m \mathscr{C}\left(\theta_i^{(h-1)} + \rho \cdot f^{(h)}(\mathbf{x}_i), y_i\right)$ through the form of line search;

Step 2.h.4: Calculate $\{\theta_i^{(h)} = \theta_i^{(h-1)} - \eta\left(\rho^{(h)} \cdot f^{(h)}(\mathbf{x}_i)\right)\}_{i=1}^m$.

Output: $\theta^{(0)}$ and $\left\{\rho^{(h)}, f^{(h)}\right\}_{h=1}^N$.

According to $\theta^{(0)}$ and $\left\{\rho^{(h)}, f^{(h)}\right\}_{h=1}^N$ obtained from the above training process, the parameter θ given $\mathbf{x}$ can be estimated as $\widehat{\theta}(\mathbf{x}) = \theta^{(0)} - \eta\sum_{h=1}^N \rho^{(h)} \cdot f^{(h)}(\mathbf{x})$, and then get $\widehat{p}_{\theta(\mathbf{x})}(y)$ accordingly.

3.2 Offline Simulation Online Monitoring With NGBoost

Continuing the problem mentioned in Section 2, we want to build an approach to estimate real-time risk measures by training only one learning model. As mentioned earlier, if we want to achieve this goal, we need to get the conditional probability density function of portfolio value $p_{\theta(\mathbf{S}^r(u))}(\Phi(T))$ under the real-time scenario $\mathbf{S}^r(u)$ observed in the real market.

In the offline simulation, we conduct a nested simulation similar to Section 2.2 to generate the data to train the NGBoost model. That is, we have a set of data pairs composed of different simulation sample paths of the underlying risk factors $\{\mathbf{S}_i(u)\}_{i=1}^m, u \in [0,T]$, and the corresponding portfolio values $\{\Phi_i(T)\}_{i=1}^m$.

After preparing $\{\mathbf{S}_i(u), \widehat{\Phi}_i(T)\}_{i=1}^m, u \in [0,T]$, we start to train NGBoost model at time u. Based on the market experience and characteristics of the distribution of portfolio value, set base learner f, parametric probability density function p_θ, scoring rule $\mathscr{C}$, boosting iterations N and learning rate η. The specific algorithm of NGBoost is shown in Algorithm 1:

Algorithm 1: NGBoost for estimating probability density function $p_{\theta(\mathbf{S}^r(u))}(\Phi(T))$

Input: Dataset $\{\mathbf{S}_i(u), \widehat{\Phi}_i(T)\}_{i=1}^m$, base learner f, parametric probability density distribution p_θ, scoring rule $\mathscr{C}$, boosting iterations N and learning rate η.

Initialize: $\theta^{(0)} = \arg\min_\theta \sum_{i=1}^m \mathscr{C}\left(\theta, \widehat{\Phi}_i(T)\right)$;

for $h = 1 : N$ **do**

  for $i = 1 : m$ **do**

    $g_i^{(h)} = I_{\mathscr{C}}\left(\theta_i^{(h-1)}\right)^{-1} \nabla_\theta \mathscr{C}\left(\theta_i^{(h-1)}, \widehat{\Phi}_i(T)\right)$;

  $f^{(h)} \leftarrow \text{fit}\left(\left\{\mathbf{S}_i(u), g_i^{(h)}\right\}_{i=1}^m\right)$;

  $\rho^{(h)} = \arg\min_\rho \sum_{i=1}^m \mathscr{C}\left(\theta_i^{(h-1)} + \rho \cdot f^{(h)}(\mathbf{S}_i(u)), \widehat{\Phi}_i(T)\right)$;

  for $i = 1 : m$ **do**

    $\theta_i^{(h)} = \theta_i^{(h-1)} - \eta\left(\rho^{(h)} \cdot f^{(h)}(\mathbf{S}_i(u))\right)$;

Output: $\{\rho^{(h)}, f^{(h)}\}_{h=1}^N$ and $\theta^{(0)}$.

Remark 1. *When p_θ belongs to a normal distribution family and the scoring rule is maximizing likelihood, $\mathscr{C}(\theta, \widehat{\Phi}_i(T)) = -\log p_\theta(\widehat{\Phi}_i(T))$, where $\theta = (\mu, \sigma)$.*

In the online application, we plug the real-time scenario $\mathbf{S}^r(u)$ observed in the real market into this trained model, and get

$$\widehat{\theta}(\mathbf{S}^r(u)) = \theta^{(0)} - \eta \sum_{h=1}^{N} \rho^{(h)} \cdot f^{(h)}(\mathbf{S}^r(u)),$$

and then obtain $\widehat{p}_{\theta(\mathbf{S}^r(u))}(\Phi(T))$. In addition, because $L(T) = \Phi(0) - \Phi(T)$ where $\Phi(0)$ is a constant, we only need to change the horizontal axis of $\widehat{p}_{\theta(\mathbf{S}^r(u))}(\Phi(T))$, i.e., the axis representing the value of the portfolio, to $L(T)$ accordingly, and we get the distribution $\widehat{p}_{\theta(\mathbf{S}^r(u))}(L(T))$. Then, we can estimate various types of risk measures. Here we introduce methods for calculating some risk measures.

- **Real-time PLL**

$$\begin{aligned} \mathrm{PLL}(\mathbf{S}^r(u)) &= \int_Q^{+\infty} p_{\theta(\mathbf{S}^r(u))}(L(T))\mathrm{d}L(T) \\ &\approx \int_Q^{+\infty} \widehat{p}_{\theta(\mathbf{S}^r(u))}(L(T))\mathrm{d}L(T) \qquad (10) \\ &\triangleq \widehat{\mathrm{PLL}}(\mathbf{S}^r(u)). \end{aligned}$$

- **Real-time VaR**

$$\begin{aligned} \mathrm{VaR}_\alpha(\mathbf{S}^r(u)) &= \inf\left\{ a \in R \mid \int_a^{+\infty} p_{\theta(\mathbf{S}^r(u))}(L(T)) \le (1-\alpha) \right\} \\ &\approx \inf\left\{ a \in R \mid \int_a^{+\infty} \widehat{p}_{\theta(\mathbf{S}^r(u))}(L(T)) \le (1-\alpha) \right\} \qquad (11) \\ &\triangleq \widehat{\mathrm{VaR}}_\alpha(\mathbf{S}^r(u)), \qquad \alpha \in (0,1). \qquad (12) \end{aligned}$$

- **Real-time CVaR**

$$\begin{aligned} \mathrm{CVaR}_\alpha(\mathbf{S}^r(u)) &= \frac{1}{1-\alpha}\int_{1-\alpha}^{1} VaR_v(\mathbf{S}^r(u))dv \\ &\approx \frac{1}{1-\alpha}\int_{1-\alpha}^{1} \widehat{\mathrm{VaR}}_v(\mathbf{S}^r(u))dv \qquad (13) \\ &\approx \frac{1}{1-\alpha}\sum_{v=\lceil (1-\alpha)M \rceil}^{M-1} \widehat{\mathrm{VaR}}_{\frac{v}{M}}(\mathbf{S}^r(u)) \qquad (14) \\ &\triangleq \widehat{\mathrm{CVaR}}_\alpha(\mathbf{S}^r(u)), \qquad (15) \end{aligned}$$

where M is a very large number and $\lceil a \rceil$ is a ceiling function that maps to the least integer no less than a.

Remark 2. *In this paper, we assume that p_θ is in a normal distribution family. The purpose of Equation (14) is to convert the integral into a numerical integral, because we cannot directly integrate $\widehat{VaR}_v(\mathbf{S}^r(u))$ analytically. On the contrary, in eqaution (10) and (11), we can use the probability density function of normal distribution to do the integral.*

4 NUMERICAL EXPERIMENTS

In this section, we consider a similar example in Jiang et al. (2019). Suppose that there is a portfolio with underlying assets consisting of 40 independent stocks whose prices are modeled by geometric Brownian motions (GBMs). The portfolio longs a European call option and a European put option based on each underlying stock, so it totally contains 40 European call options and 40 European put options. To simplify the problem, we assume that the risk factors are stock prices. The related parameters are given as follow:

- The initial stock price $\mathbf{S}(0) = (50, 50, \ldots, 50)^\top$, i.e., each component $S_j(0) = 50 (j = 1, 2, \ldots, 40)$;
- The drifts of the GBMs $\mu_j = 0.02$ $(j = 1, 2, \ldots, 10), \mu_j = 0.03$ $(j = 11, \ldots, 20), \mu_j = 0.04$ $(j = 21, 22, \ldots, 30)$ and $\mu_j = 0.05$ $(j = 31, 32, \ldots, 40)$;
- The risk-free interest rate $r_f = 0.02$;
- The volatility $\sigma_j = 0.1 (\ j = 1, 2, \ldots, 20)$ and $\sigma_j = 0.2$ $(j = 21, \ldots, 40)$;
- The strike price of the European call option $K_j^c = 35.5 + 0.5j$ $(j = 1, 2, \ldots, 40)$, and the strike price of the European put option is $K_j^p = 45.5 + 0.5j$ $(j = 1, 2, \ldots, 40)$;
- The fixed future time $T = 0.6$ and the expiration dates of the financial derivatives $\tau_j = 1$ $(j = 1, 2, \ldots, 40)$.

When generating the outer simulation sample paths, we discretize the time by 0.01, i.e., we set the time as $t = 0.01, 0.02, \ldots, 0.6$, and denote the outer simulation sample paths as $\{\mathbf{S}_i(t), t = 0.01, 0.02, \ldots, 0.6\}_{i=1}^m$ with $m = 10^4$. That is, we have 10^4 risk factor scenarios at T. For each scenario $\mathbf{S}_i(T)$, the number of inner simulation sample paths to estimate the value of the portfolio (loss) is $n_i = 100$, and

$$\widehat{\Phi}_i(T) = \frac{1}{100} \sum_{k=1}^{100} \sum_{j=1}^{40} \left((S_{ij}^k(\tau) - K_j^c, 0)^+ + (K_j^p - S_{ij}^k(\tau), 0)^+ \right) \exp^{-r_f \cdot (\tau - T)},$$

where $S_{ij}^k(\tau)$ represents the kth sample of the jth component (the jth risk factor) of the ith scenario $\mathbf{S}_i(\tau)$. Here, we set the threshold $Q = 17$. Besides, in the setting of NGBoost model, we choose decision tree as base learner, MLE as scoring rule, and normal distribution as preset probability density distribution, and let boosting iterations $N = 500$, learning rate $\eta = 0.02$.

First, we compare our approach in estimating real-time PLL with the logistic regression in Jiang et al. (2019). Specifically, we select two types of basis functions $\chi_1(\mathbf{S}(u))$ and $\chi_2(\mathbf{S}(u))$, which are linear and quadratic basis functions, respectively, in Equation (7), and monitor the real-time PLL from $u = 0.01$ to $u = 0.12$.

Test data set. In order to evaluate the estimated accuracy of different approaches, we generate 10 test sets for each time $u = 0.01, 0.02, ..., 0.12$. For the lth $(l = 1, 2, \cdots, 10)$ test set of time u, we simulate 100 scenario paths from time 0 to u and record $\{\mathbf{S}_i^l(u)\}_{i=1}^{100}$ as input value of test set for the NGBoost model at time u. Then calculate $\{\chi_1(\mathbf{S}_i^l(u))\}_{i=1}^{100}$ and $\{\chi_2(\mathbf{S}_i^l(u))\}_{i=1}^{100}$ as input value of test data for linear logistic regression model and quadratic logistic regression model at time u, respectively. We evaluate the true risk measures of each test point by running 10000 outer simulation from u to T, and for each risk factor scenario, we use Black-Scholes formula to calculate the value of the portfolio. We then calculate the RMSE of the estimated real-time PLL by the NGBoost model and PLL estimated by linear logistic regression model and quadratic logistic regression model. The RMSE is calculated by

$$\text{RMSE}(u) = \sqrt{\frac{1}{L} \sum_{l=1}^{L} \left[\frac{1}{100} \sum_{i=1}^{100} (\hat{\alpha}(\mathbf{S}_i^l(u)) - \alpha^{true}(\mathbf{S}_i^l(u))) \right]^2},$$

where the number of test sets is $L = 10$, $\hat{\alpha}(\cdot)$ and $\alpha^{true}(\cdot)$ represent the estimated value and the true value of a risk measure, respectively.

Then, we obtain Figure 3, which indicates that the NGBoost approach can obtain an accurate estimation of the real-time PLL, and it is comparable with the logistic regression approach in Jiang et al. (2019). Specifically, the accuracy of NGBoost is better than that of both 1order-LR and 2order-LR. According to the result in Jiang et al. (2019), a good choice of basis function can produce a good estimate of the real-time PLL. In our setting, we may choose other distribution families to improve the accuracy.

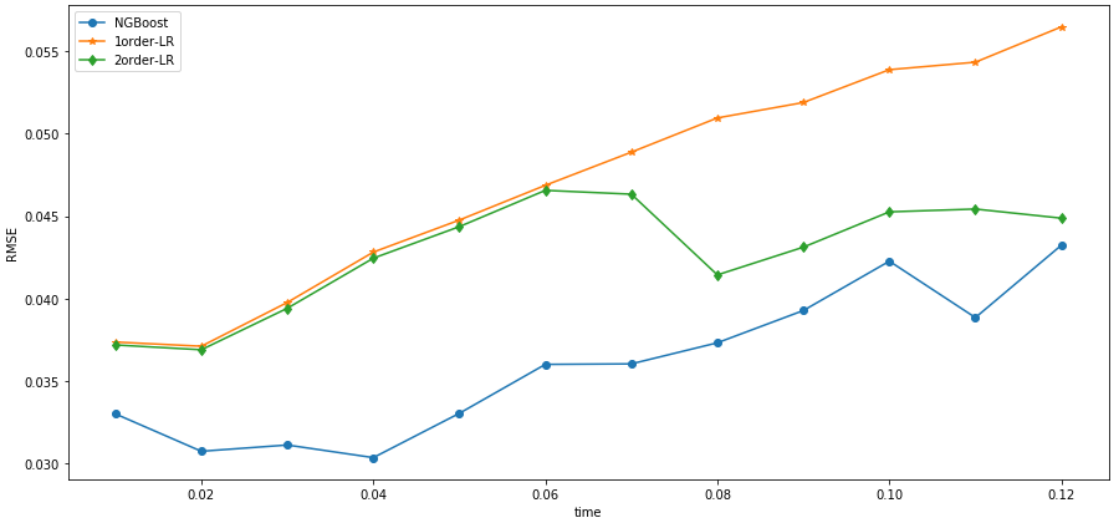

Figure 3: RMSE for the real-time PLL for NGBoost, 1order-LR (linear logistic regression) and 2order-LR (quadratic logistic regression).

Besides the real-time PLL, the real-time VaR and the real-time CVaR can also be obtained according to Equations (12) and (15), respectively. According to Equations (2) and (3), we can use a similar method in estimating the true real-time PLL to evaluate the true real-time VaR and CVaR. Specifically, we consider confidence level $\alpha = 0.99$, but given the huge differences in their benchmarks, RMSE alone can not reflect the accuracy of predicted value. In order to better illustrate the prediction results of NGBoost model, we further provide the relative root mean squared-error (RRMSE) of VaR and CVaR. The RRMSE is calculated as follows

$$\text{RRMSE}(u) = \sqrt{\frac{1}{L}\sum_{l=1}^{L}\left[\frac{1}{100}\sum_{i=1}^{100}\left(\frac{\hat{\alpha}(\mathbf{S}_i^l(u)) - \alpha^{true}(\mathbf{S}_i^l(u))}{\alpha^{true}(\mathbf{S}_i^l(u))}\right)\right]^2}.$$

The Figure (4) shows the box-plot of RRMSE of VaR and CVaR derived by NGBoost approach, and it indicates that the accuracies of the real-time VaR and CVaR are both quite good, but as the time u increases, the prediction effect is worse.

5 CONCLUSION

As the characteristics of diversification and frequent changes of financial risk factors, risk managers are increasingly eager to be able to monitor the risk of portfolios in real time. Recently, Jiang et al. (2019) proposed an offline simulation online application approach, which estimates real-time probability of large loss by training metamodels to learn the conditional relationship between risk measures and risk factors. However, through analysis, the method proposed in Jiang et al. (2019) needs to train different metamodels when estimating different risk measures, which shows the limitation of the method in risk measurement. In order to fix this issue, this paper proposes to directly estimate the conditional probability density function of the value (loss) of a portfolio using the NGBoost approach under the framework of offline simulation online application. When the conditional probability density function is obtained, the real-time risk measures can

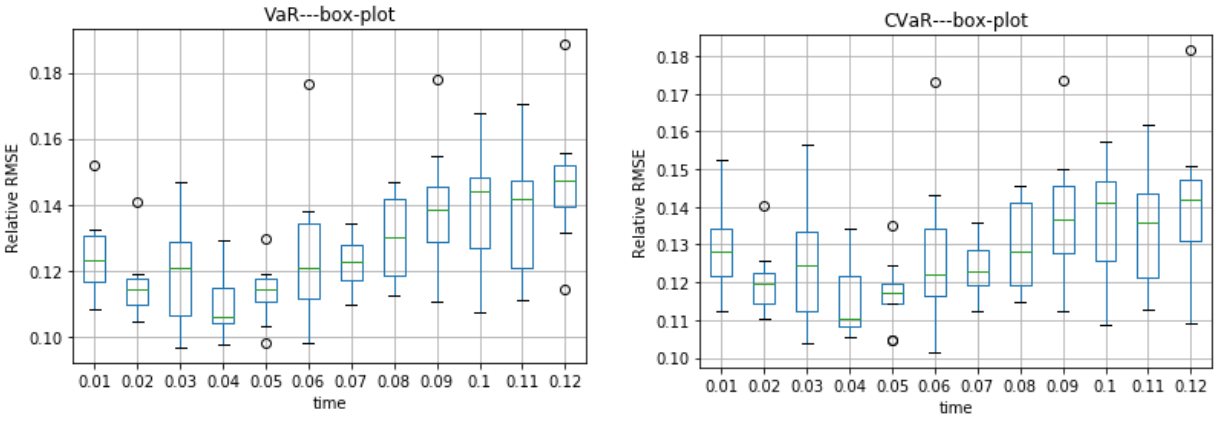

Figure 4: Left panel is box-plot of RRMSE of VaR and right panel show box-plot of RRMSE of CVaR.

be obtained by doing additional basic operations. Moreover, numerical experiments show the effectiveness of this method in estimating various real-time risk measures.

In future research, we may use more distribution families (e.g. heavy-tailed distribution) to improve the prediction accuracy, and also try other density estimation methods to estimate the conditional density function of the loss of the portfolio.

ACKNOWLEDGMENTS

The authors are grateful to Professor Qiang Ye, from the School of Management at Harbin Institute of Technology, for his helpful discussions. He helps us improve the efficiency of the algorithm and points out the important application background of our proposed methods. The research reported in this paper is supported by the National Natural Science Foundation of China [Grants 71801148].

REFERENCES

Amari, S.-i. 1998. "Natural Gradient Works Efficiently in Learning". *Neural Computation* 10(2):251–276.

Broadie, M., Y. Du, and C. C. Moallemi. 2015. "Risk estimation via regression". *Operations Research* 63(5):1077–1097.

Collins, M., R. E. Schapire, and Y. Singer. 2002. "Logistic Regression, AdaBoost and Bregman distances". *Machine Learning* 48(1/2/3):253–285.

Cousins, C., and M. Riondato. 2019. "CaDET: interpretable parametric conditional density estimation with decision trees and forests". *Machine Learning* 108(8-9):1613–1634.

Duan, T., A. Avati, D. Y. Ding, S. Basu, A. Y. Ng, and A. Schuler. 2019. "NGBoost: Natural Gradient Boosting for Probabilistic Prediction". *arXiv preprint arXiv:1910.03225*.

Dunson, D. B., N. Pillai, and J. H. Park. 2007. "Bayesian density regression". *Journal of the Royal Statistical Society: Series B* 69:163–183.

Fan, J., Q. Yao, and H. Tong. 1996. "Estimation of conditional densities and sensitivity measures in nonlinear dynamical systems". *Biometrika* 83:189–206.

Freund, Y., and R. Schapire. 1997. "A decision-theoretic generalization of on-line learning and an application to boosting". *Journal of Computer and System Sciences* 55(1):119–139.

Friedman, J. H. 2001. "Greedy function approximation: A gradient boosting machine". *The Annals of Statistics* 29(5):1189–1232.

Gordy, M. B., and S. Juneja. 2010. "Nested simulation in portfolio risk measurement". *Management Science* 56(10):1833–1848.

Hastie, T., R. Tibshirani, and J. Friedman. 2011. *The Elements of Statistical Learning*. Second ed. New York: Springer-Verlag.

Hong, L. J., Z. Hu, and G. Liu. 2014. "Monte Carlo methods for Value-at-Risk and conditional Value-at-Risk: A review". *ACM Transactions on Modeling and Computer Simulation* 24(4):Article 22.

Hong, L. J., and G. Jiang. 2019. "Offline simulation online application: A new framework of simulation-based decision making". *Asia-Pacific Journal of Operational Research* 36(6):No.1940015.

Hong, L. J., S. Juneja, and G. Liu. 2017. "Kernel smoothing for nested estimation with application to portfolio risk measurement". *Operations Research* 65(3):657–673.

Jiang, G., L. J. Hong, and B. L. Nelson. 2019. "Online Risk Monitoring Using Offline Simulation". INFORMS Journal on Computing. https://doi.org/10.1287/ijoc.2019.0892.

Lakshminarayanan, B., A. Pritzel, and C. Blundell. 2017. "Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles". In *NIPS*, edited by I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, 6402–6413. Long Beach, California: Neural Information Processing Systems Foundation, Inc.

Lee, S.-H., and P. W. Glynn. 2003. "Computing the distribution function of a conditional expectation via Monte Carlo: Discrete conditioning spaces". *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 13(3):238–258.

Liu, M., and J. Staum. 2010. "Stochastic kriging for efficient nested simulation of expected shortfall". *Journal of Risk* 12(3):3–27.

Maacsse, B. R., and Y. K. Truong. 1999. "Conditional logspline density estimation". *Canadian Journal of Statistics* 27:819–832.

Rosenblatt, M. 1969. "Multivariate Analysis". In *Conditional probability density and regression estimates*, edited by P. R. Krishnaiah, 25–31. New York: Academic Press.

Stone, C. J. 1994. "The use of polynomial splines and their tensor products in multivariate function estimation". *The Annals of Statistics* 22:118–171.

AUTHOR BIOGRAPHIES

XIAOTING CAI is a master student in management science and engineering at Shanghai University. She received her double degree in Engineering Management and Finance from Dongbei University of Finance and Economics. Her research interests include but are not limited to simulation, real-time risk measurement and data science. Her email address is cxt95@shu.edu.cn.

YANG YANG is a graduate student of School of Data Science in the Chinese University of Hong Kong (Shenzhen). He has earned a bachelor's degree in Management from Shanghai University. His research interests involve stochastic models and simulation, machine learning, financial engineering, etc. His email address is y970614739@shu.edu.cn (Corresponding author).

GUANGXIN JIANG is a professor in the School of Management at Harbin Institute of Technology. From February 2017 to December 2019, he served as a lecturer and associate professor at the School of Management of Shanghai University. From February 2015 to February 2017, he was a postdoctoral fellow at the Business School of City University of Hong Kong. He earned his Ph.D. in Applied Mathematics from Tongji University. His research interests include stochastic models and simulation, machine learning, financial engineering and risk management, FinTech, etc. His email address is gxjiang@hit.edu.cn. His website is http://homepage.hit.edu.cn/jiangguangxin/.