Learning conditional expectation, value-at-risk, and expected shortfall by randomized neural networks

Hoang-Dung NGUYEN[*]
Supervision : Prof. Stéphane Crépey

September 27, 2020

Abstract

Alongside conditional expectation for the purpose of pricing, conditional value-at-risk (VaR) and expected shortfall (ES) are two well-known risk measures in the framework of financial enterprise risk management. This report deals with the problem of estimating conditional VaR and ES based on two approaches: Rockafellar & Uryasev's two-step and Fissler & Ziegel's joint approach. The convergence of the two-step approach can be analyzed in terms of Rademacher complexity. As a family of non-parametric models that are used for practical implementation, we investigate neural network regression and its fast computational paradigm, randomized neural network, in which the majority of connections are pre-selected and the training process can be reduced to a linear regression. Our numerical case studies emphasize the efficiency of randomized neural networks in solving quadratic loss but not quantile loss (for training conditional VaR) nor the complex loss of the joint approach. The most efficient approach for conditional VaR estimation is to use standard neural networks learning Huber loss, a smooth approximation of quantile loss.

Keywords: *Value-at-Risk*, *Expected Shortfall*, *Quantile regression*, *Machine Learning*, *Neural networks*, *Randomized neural networks*, *Extreme Learning.*

Acknowledgement: Thanks to Marc Chataigner and Bouazza Saadeddine for very useful discussions on the algorithms and their implementation, and to David Barrera for his reading of the convergence analysis.

[*] Master 2 Data Science, Paris-Saclay.

Table of contents

1 Introduction

In the aftermath of the 2008 financial crisis, banks have been required to quantify the counterparty risk and its capital and funding implications via a family of X-valuation adjustments (XVAs). Since the advent of these metrics, XVA analysis has become a major quantitative mission for investment banks. Advanced XVA metrics involve not only conditional expectation but also conditional Value-at-risk (VaR) and Expected Shortfall (ES) (see e.g. Albanese, Crépey, Hoskinson & Saadeddine (2020)).

With this XVA motivation in mind, we can state the two main objectives of this work: (i) we look for stable and accurate conditional VaR and ES estimation, and (ii) given that these risk measures must be reevaluated at each valuation steps in XVA framework, it is essential to accelerate the computational paradigm of these metrics (as well as of conditional expectation, corresponding to future prices in the financial application, in the first place).

Regarding the first objective, the estimation of conditional VaR is a particular case of quantile regression, a well established statistical problem with rich developments since 100+ years nicely surveyed in Koenker et al. (2017). The estimation of conditional ES can be realized based on two views: Rockafellar & Uryasev (2000) expressed the ES as a function of VaR and Fissler & Ziegel (2016) later discovered the joint elicitability of the pair VaR/ES.

Regarding the learning machine tools that can be used to the benefit of the second objective, we investigate neural networks that are reported to be workable for XVA analysis in Albanese, Crépey, Hoskinson & Saadeddine (2020, Section 4.4 and 5). The second objective leads us to explore the class of randomized neural networks in which only the output layer of the network is trained whereas the hidden layers are randomly fixed. For these networks, Wang & Li (2017) emphasized the importance of hidden layers designs. Gallicchio & Scardapane (2020) reviewed several ways of determining the hidden layers. Huang et al. (2006) and Igelnik & Pao (1995) focused on the universal approximation capabilities of these networks and on the optimization of the output layer by linear regression in order to achieve an extremely high speed training process, which they then call extreme learning machine.

1.1 Training Paradigm

We provide the basic notation which is used throughout the report.

Functional methods search in a predefined (hypothesis) space the function mapping the input to the output under certain criteria. These criteria are usually defined as a loss objective function we aim to minimize, and the predefined space represents the model's approximation capacity and flexibility. A canonical example is the least squares criterion, which is perfectly suited to the estimation of the conditional expectation.

Let $(\Omega, \mathcal{A}, \mathrm{P})$ be a probability space. We consider a set of covariates $X : \Omega \to \mathbb{R}^d$, with a positive integer d, and a response variable $Y : \Omega \to \mathbb{R}$. We want to estimate an arbitrary function $F : \mathbb{R}^d \to \mathbb{R}$ such that there exists a loss function l taking F as its minimizer. The *M-estimator* estimator $F^\star$ in a so-called hypothesis space of functions $\mathcal{H}$ is then obtained by minimizing the following expected loss:

$$F^\star \in \underset{F \in \mathcal{H}}{\arg\min}\, \mathrm{E}\left[l(Y, F(X))\right]. \tag{1}$$

Example 1.1. *By plugging the least squares loss $l_2(x, y) = \|x - y\|^2$ into* (1)*, $F^\star$ becomes an estimator of the conditional expectation of Y given X, while the least absolute loss $l_1(x, y) = |x - y|$ allows one to estimate the conditional median.*

In addition, the hypothesis space $\mathcal{H}$ needs to have enough structure to guarantee the feasibility of the optimization procedure. At least, the functions in $\mathcal{H}$ need to have the same domain and range as the target F (Torossian et al. 2020) (e.g. a classifier should not be used for a regression problem). We then link the choice of $\mathcal{H}$ to machine learning models such as linear regression (when $\mathcal{H}$ is the space of linear functions), polynomial regression, kernel regression, support vector machines, or neural networks.

1.2 Outline and Contributions

This report is organised as follows. Section 2 presents the two-step and joint functional representations of the conditional VaR and ES based on (Rockafellar & Uryasev 2000) and (Fissler & Ziegel 2016), respectively. Section 3 studies the convergence analysis for the two-step estimation procedure to the conditional VaR and ES. Section 4 reviews neural networks model for regression. Section 5 explores the state-of-the-art of randomized neural networks in order to accelerate the training of the model. Sections 6 and 7 set up our case studies and provide illustrative numerical results.

The main contributions of our research are (i) a variant of the convergence analysis of Barrera et al. (2020) for ES in the two-step approach, which is concluded in Theorem 3.7, and (ii) the empirical work regarding the application of the randomized neural networks in regression problem.

2 Conditional Value-at-risk and Expected Shortfall

Conditional Value-at-risk can be estimated by quantile regression referring to a group of high confidence level quantiles. Quantile regression, mostly contributed by Koenker et al. (2017), is a well-known statistical approach providing a more complete statistical view than mean regression. It has been increasingly used in many different academic fields. Since its first introduction over 40 years ago, nowadays a wide class of models ranging from parametric to non-parametric have been proposed to determine conditional quantile. To be more specific: parametric linear setup based on the simplex method, building on seminal contributions by Bošković - a Jesuit Catholic priest from Dubrovnik - in the 18th century, interior point methods (Koenker & Ng 2005), and non-parametric quantile regression approaches including kernel regression (Takeuchi et al. 2006), nearest neighbors (Bhattacharya & Gangopadhyay 1990), smoothing splines (Koenker et al. 1994) and neural networks (White 1992) methods.

In addition, conditional expected shortfall can be estimated by non-parametric methods based on two characterizations of VaR and ES: Rockafellar & Uryasev (2000) wrote the training loss function of ES based on VaR, while Fissler & Ziegel (2016) investigated that the pair VaR/ES is jointly elicitable with respect to a suitable class of loss functions.

2.1 Conditional VaR and ES Definition

Value-at-risk (VaR) and expected shortfall (ES) are the risk measures that are the most commonly used by financial firms and regulators to assess risk and compute capital charges. They are typically estimated at a given confidence level $\alpha = (1-\epsilon)$, for some small (and positive) ϵ (e.g. 2.5% or 1%), and represent the amount of capital needed to cover their highest loss (quantile for the VaR and conditional tail expectation beyond the VaR for the ES) at the corresponding confidence level.

Definition 2.1 (Value-at-risk and Expected Shortfall (cf. Definition 2.1, Remark 2.1 and 2.2 in Barrera et al. (2020)))**.** *Let Y represent the future loss random variable assumed to have a continuous cumulative distribution function for simplicity. Value-at-risk is the left quantile at level $(1-\epsilon)$ of Y, i.e.*

$$\mathrm{VaR}(Y) = \min\{q \in \mathbb{R} : \mathrm{P}(Y \leq q) = 1-\epsilon\} \tag{2}$$

With a probability $(1-\epsilon)$, the loss Y will not exceed the level q.

Assuming integrability of Y, Expected Shortfall is the average loss given the latter exceeds the VaR at the confidence level $(1-\epsilon)$, i.e.

$$\begin{aligned} \mathrm{ES}(Y) &= \mathrm{E}\left[Y \mid Y \geq \mathrm{VaR}(Y)\right] \\ &= \epsilon^{-1}\mathrm{E}\left[Y \mathbb{1}_{\{Y \geq \mathrm{VaR}(Y)\}}\right] \\ &= \epsilon^{-1}\mathrm{E}\left[(Y - \mathrm{VaR}(Y))^{+}\right] + \mathrm{VaR}(Y) \\ &(\geq \mathrm{VaR}(Y)) \end{aligned} \tag{3}$$

First proposed by RiskMetrics[1] in 1996, VaR quickly became the main risk measure adopted by Basel II−2004 to set up capital requirements. However, there were intense debates around the lack of sub-additivity of VaR (which only holds "in most cases"). Besides, the global financial 2008 revealed practical shortcomings of the VaR. Subsequently, the Basel Committee changed to use expected shortfall as the main risk measure instead of Value-at-risk. Obviously, ES overcame the lack of risk diversification effect (i.e. sub-additivity) of VaR, as a coherent risk measure (cf. see Definition 9.1)

Furthermore, in practice, we are actually interested in the VaR and ES of a future loss based on the information of related risk factors. Thus, we introduce the following definition:

Definition 2.2 (Conditional Value-at-risk and Expected Shortfall)**.** *Let Y be future loss random variable that depends on multivariate random variable X representing risk factors. The conditional risk measures of Y given X are mathematically defined by*

$$\begin{aligned} \mathrm{VaR}(Y|X) &= \min\{q \in \mathbb{R} : \mathrm{P}(Y \leq q|X) = 1-\epsilon\} \\ \mathrm{ES}(Y|X) &= \mathrm{E}\left[Y|X, Y \geq \mathrm{VaR}(Y|X)\right] \end{aligned} \tag{4}$$

[1] Established in 1989 by J.P. Morgan. The aim of RiskMetrics was to create a benchmark for measuring market risks.

As detailed in Barrera et al. (2020), the conditional VaR and ES of a loss Y given risk factors X can be represented by Borel measurable functions of X, which we respectively denote hereafter as $q(X)$ and $s(X)$.

2.2 Conditional VaR and ES Functional Representations

Even if estimating conditional Value-at-risk is a complicated problem in comparison with conditional expectation, related statistical schemes have been developed and surveyed for a long time. By contrast, until recently, learning conditional expected shortfall has been a challenge. Rockafellar & Uryasev (2000) found out the functional representation of ES through VaR, opening the gate to advanced non-parametric approaches for that matter. Thereafter, Fissler et al. (2015) established that the pair VaR/ES is jointly elicitable with respect to a specific class of loss functions. Dimitriadis et al. (2019) then proposed linear regression of the conditional VaR and ES building on their joint loss function, following which Albanese et al. (2020) used neural networks.

A. Rockafellar and Uryasev (RU) representation

Theorem 2.3 (Theorem 1 in Rockafellar & Uryasev (2000)). *Let*

$$\ell(y,q)=\epsilon^{-1}(y-q)^{+}+q. \tag{5}$$

Given a loss random variable Y satisfying the assumptions in Definition 2.1, the expected shortfall of Y can be determined from the formula

$$\mathrm{ES}(Y)=\mathrm{E}\left[\ell(Y,\mathrm{VaR}(Y))\right]=\min_{q}\mathrm{E}\left[\ell(Y,q)\right] \tag{6}$$

and

$$\mathrm{VaR}(Y)\in\arg\min_{q}\mathrm{E}[\ell(Y,q)]. \tag{7}$$

Remark 2.4. *From the optimization viewpoint, the loss function* (5) *is equivalent to the pinball loss, which is commonly used to estimate a quantile at confidence level $\alpha\,(=1-\epsilon)$,*

$$\begin{aligned}\rho_\alpha(y,q)&=(y-q)\left(\alpha-\mathbb{1}_{(y-q<0)}\right)=\alpha(y-q)\mathbb{1}_{(y>q)}-(1-\alpha)(y-q)\mathbb{1}_{(y<q)}\\&=\alpha(y-q)^{+}+(1-\alpha)(y-q)^{-}.\end{aligned} \tag{8}$$

In fact, starting from that $\rho_\alpha(y,q)$ by equation (8) *can be rewritten as*

$$\begin{aligned}\rho_\alpha(y,q)&=(y-q)\left(1-\epsilon-\mathbb{1}_{(y-q<0)}\right)\\&=(y-q)\left(\mathbb{1}_{(y-q>0)}-\epsilon\right)\\&=(y-q)^{+}-\epsilon(y-q),\end{aligned}$$

the loss can be divided by ϵ and then the data point y of the second term can be ignored in the optimisation phase because they are constant. This leads to the loss (5).

B. Fissler and Ziegel (FZ) representation

A statistical functional, e.g. the mean or the median, is said to be elicitable if there exists a loss function such that the correct estimator of the functional is the unique minimizer of the expected loss (cf. the estimator $F^\star$ is the unique solution of (1) in $\mathcal{H}$). The elicitability allows us to compare transparently the hypothesis spaces (i.e. machine learning methods) for a given problem.

VaR at the confidence level α is elicitable for random variables which have a unique α-quantile, which allows comparing different training models. By contrast, ES is not elicitable, at least not on a standalone basis. Yet Fissler et al. (2015) pointed out the joint elicitability of the pair VaR and ES.

Corollary 2.5 (Corollary 5.5 in Fissler & Ziegel (2016) [2]). *Let ι and ν be functions $\mathbb{R} \to \mathbb{R}$, $\dot{\nu} = \nu'$, where ι is increasing and ν is [strictly] increasing and [strictly] concave, e.g.*

$$i(\cdot) = \cdot\ , \qquad \nu(\cdot) = -\ln\left(1 + e^{-\cdot}\right)$$

and

$$\begin{aligned}\lambda(y, q) &= \epsilon^{-1}\left[\iota(y) - \iota(q)\right]^+ + \iota(q)\\ \varrho(y, q, s) &= \nu(s) - \dot{\nu}(s)\left(s - q - \epsilon^{-1}(y - q)^+\right)\end{aligned}$$

If the law of Y has a unique α-quantile and $\iota(Y)\mathbb{1}_{Y \geq q}$ is integrable for all $q \in \mathbb{R}$, then

$$(\mathrm{VaR}(Y), \mathrm{ES}(Y)) = \underset{(q,s)}{\arg\min}\, \mathrm{E}\left[\lambda(Y, q) + \varrho(Y, q, s)\right]. \tag{9}$$

C. The objective functions

We summarize training algorithms for conditional VaR and ES. For any convex spaces $\mathcal{F}^*$ and $\mathcal{G}^*$ of functions : $\mathbb{R}^d \to \mathbb{R}$ containing the true $q(\cdot)$ and $s(\cdot)$, and $\mathcal{H}^*$ of functions : $\mathbb{R}^d \to \mathbb{R}^2$ containing the true $(q(\cdot), s(\cdot))$, respectively:

- **Two-step approach** based on RU representation

$$\begin{aligned}q(X) &\in \underset{f \in \mathcal{F}^*}{\arg\min}\, \mathrm{E}\ell(Y, f(X))\\ s(X) &\in \underset{g \in \mathcal{G}^*}{\arg\min}\, \mathrm{E}\left[(\ell(Y, q(X)) - g(X))^2\right]\end{aligned} \tag{10}$$

- **Joint approach** based on FZ representation

$$(q(X), s(X)) = \underset{q,s \in \mathcal{H}^*}{\arg\min}\, \mathrm{E}[\lambda(Y, f(X)) + \varrho(Y, f(X), g(X))] \tag{11}$$

[2] We slightly reformulate this corollary in order to provide an introduction consistent with our previous treatment of the RU representation

These functional representations open the door to conditional VaR and ES training algorithms, using stochastic gradient descent (SGD) as a scalable optimization tool. In our numerical training process, the expectation E is replaced by sample (empirical) average and true spaces $\mathcal{F}^*, \mathcal{G}^*, \mathcal{H}^*$ are replaced by hypothesis spaces $\mathcal{F}, \mathcal{G}, \mathcal{H}$, for which we would like to turn our interest to the class of neural networks later on in this report (Section 4).

2.3 Overview of Quantile Regression

We conclude this section by a review of already known estimation methods for conditional VaR known as quantile regression, based on the pinball loss function (8). Accordingly, the quantile regression function is defined by

$$q(\cdot) \in \underset{q \in \mathcal{H}}{\arg\min}\, \mathrm{E}\left[\rho_\alpha(Y, q(X))\right] \tag{12}$$

Many solutions were proposed for solving (12): linear quantile regression based on the simplex method (Bošković), interior point methods (Koenker & Ng 2005), non-parametric quantile regression methods based on stochastic gradient descent (White 1992, Zheng 2011 and Koenker et al. 2017, Section 5.5). However, unlike the square loss, the pinball loss is a convex but non-smooth function (its second derivative is not continuous, and neither is its first derivative). This leads to the poor performance of both interior point methods and the family of gradient descents. Hence, to overcome this drawback, we usually add a regularization term, which not only avoids over-fitting of the hypothesis space $\mathcal{H}$, but also increases the smoothness of the objective function, in order to accelerate the training (optimisation) step. Thus, the estimator of α-quantile q is the minimizer of the following loss:

$$q(X) \in \underset{q \in \mathcal{H}}{\arg\min}\, \mathrm{E}\left[\rho_\alpha(Y, q(X))\right] + \lambda h(q), \tag{13}$$

for some regularization term h weighted by λ. Koenker et al. (1994) introduced the p-norm for h and (13) is then called *quantile smoothing spline* (QSS). For $p = 2$, QSS training can be implemented by quadratic programming (Chakravarti 2019). In addition, Takeuchi et al. (2006) used the RKHS (Reproducing Kernel Hilbert Space) norm as a regularizer, which is tested to over-perform QSS but be more computationally intensive in Chakravarti (2019).

Torossian et al. (2020) reviewed various models of quantile regression, in three groups: (i) approaches based on statistical order: K nearest-neighbors (Bhattacharya & Gangopadhyay 1990) and Random Forest (Meinshausen 2006), (ii) functional approaches: neural networks (Cannon 2011) and regression in RKHS (Takeuchi et al. 2006), and (iii) Bayesian approaches based on Gaussian processes. They benchmarked these models on several simulation problems of different data shapes as well as on a real-world data of weather records. Based on their experiments, the functional model that would be the most suitable to our motivating problem (i.e. XVA calculations), where the number of observations is large and more than fifty times the features dimension, is neural networks. Besides the work in section 4.4 and 5 of Albanese et al. (2020), these important remarks of Torossian et al. (2020) also motivate us to investigate the class of neural networks later on (Section 4).

3 Convergence Analysis for the Two-step Approach

We now discuss on the convergence analysis for the estimators of conditional VaR and ES based on RU's representation (i.e. equation (10)) (for the moment, the convergence analysis for the joint approach is work in process). In particular, we state the final bound of the error related to conditional VaR estimates in Barrera et al. (2020). Inspired by their work, we develop our own convergence analysis for conditional ES estimates which can be considered as a variant of Barrera et al. (2020)'s work. In fact, Barrera et al. (2020) analyzed the convergence of ES estimates via the difference between ES and VaR estimates, while in this report we work directly on the error of ES estimates.

In what follows, we use the notations q and s for true conditional VaR and ES (at confidence level α) of variable $Y \in \mathbb{R}$ given features $X \in \mathbb{R}^d$. Set the fixed hypothesis spaces $\mathcal{F}$ and $\mathcal{G}$ for q and s, respectively. Furthermore, let P_X denote the distribution of X, $\mathrm{P}_{X,Y}$ denote the joint distribution (X, Y) and F_X denote the distribution of Y given X.

Definition 3.1 (p-norm). *We define the p-norm of an arbitrary function h on the vector space X as*

$$\|h(X)\|_{\mathrm{P}_X,p} = \left[\mathrm{E}\left(|h(X)|^p\right)\right]^{1/p}$$

In particular, when $p = 2$, we get the Euclidean norm. Throughout our analysis, we sometimes use only $\|\cdot\|$ (dropping p and P_X) for the Euclidean norm. Moreover, the infinity norm is defined as

$$\|h(X)\|_{\mathrm{P}_X,\infty} = \inf\{B : \mathrm{P}(|h(X)| \leq B) = 1\}.$$

Using also the notation $\mathcal{D}_n = \{(x_i, y_i)|i = 1, \ldots, n\}$ for a data set of n i.i.d. samples of (X, Y), we are interested in analyzing the error between the true metrics and their empirical estimators denoted by $\hat{q}$ and $\hat{s}_{\hat{q}}$. For this purpose, we introduce $\tilde{q}$ as the estimator in $\mathcal{F}$ using the true expectation, $\tilde{s}_{\hat{q}}$ and $\tilde{s}_{\tilde{q}}$ as the estimators in $\mathcal{G}$ using the true expectation and the respective estimators $\hat{q}$ and $\tilde{q}$ for q (even if the parameter is not specified for q, s and their estimators, we refer to functions of the variable x). To be more specific,

$$\begin{aligned}
&\tilde{q} \in \operatorname*{arg\,min}_{f \in \mathcal{F}} \mathrm{E}\ell(Y, f(X)), && \hat{q} \in \operatorname*{arg\,min}_{f \in \mathcal{F}} \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, f(x_i)) \\
&\tilde{s}_{\tilde{q}} \in \operatorname*{arg\,min}_{g \in \mathcal{G}} \mathrm{E}\left[(\ell(Y, \tilde{q}(X)) - g(X))^2\right], && \tilde{s}_{\hat{q}} \in \operatorname*{arg\,min}_{g \in \mathcal{G}} \mathrm{E}\left[(\ell(Y, \hat{q}(X)) - g(X))^2\right] \\
&\hat{s}_{\hat{q}} \in \operatorname*{arg\,min}_{g \in \mathcal{G}} \frac{1}{n}\sum_{i=1}^{n}(\ell(y_i, \hat{q}(x_i)) - g(x_i))^2
\end{aligned} \tag{14}$$

3.1 The approximation error of $\hat{q}$

The approximation error of $\hat{q}$ is upper bounded as follows:

$$\|q - \hat{q}\| \leq \underbrace{\|q - \tilde{q}\|}_{\text{bias}} + \underbrace{\|\tilde{q} - \hat{q}\|}_{\text{variance}} \tag{15}$$

To dig into the inequality (15), we provide two additional assumptions:

Assumption 1. *With no loss of generality for VaR, both the response Y and the hypothesis space $\mathcal{F}$ are bounded by B:*

$$\max\left\{\|Y\|_\infty, \sup_{f\in\mathcal{F}}\|f(X)\|_{\mathrm{P}_{X,Y},\infty}\right\} \le B.$$

Assumption 2. *There exists Borel measurable functions a, b such that, for some $0 < M_1 \le M_2 < \infty$,*

$$M_1 \le F'_X(y) \le M_2, \quad \textit{for every } y \in [a(X), b(X)]$$

and, for every $f \in \mathcal{F}$,

$$[f(X), q(X)] \cup [q(X), f(X)] \subset [a(X), b(X)], \quad \textit{except on a set of } \mathrm{P}-\textit{measure zero} \tag{16}$$

Theorem 3.2 (Bias for VaR (cf. Lemma 6.1 and Remark 6.2 in Barrera et al. (2020)))**.** *Under the above assumptions,*

$$\|q - \tilde{q}\|^2_{\mathrm{P}_X,2} \le \frac{1}{M_1}\left[\left(2(2-\alpha)\inf_{f\in\mathcal{F}}\|q - f\|_{\mathrm{P}_X,1}\right) \wedge \left(M_2 \inf_{f\in\mathcal{F}}\|q - f\|^2_{\mathrm{P}_X,2}\right)\right]. \tag{17}$$

Thus, for the VaR, the bias is controlled by the L^1 and L^2 projection errors on the hypothesis space $\mathcal{F}$. The bias error shows the capabilities of the hypothesis space to approximate the true q with respect to the distribution P_X.

The variance term in (15) needs to be analyzed via Rademacher estimates. In computational learning theory, Rademacher complexity measures the richness of a class of real-valued functions according to a given probability distribution.

Definition 3.3 (Rademacher Complexity)**.** *Given a space Z with a fixed distribution P_Z, let $\mathcal{S}_n = \{z_1, \dots, z_n\}$ be i.i.d samples drawn from P_Z. Furthermore, let $\mathcal{H}$ be a class of functions $h : Z \to \mathbb{R}$.*

For $U_1, \dots, U_n$ independent random (Rademacher) variables uniformly chosen from $\{-1, 1\}$, the empirical Rademacher complexity of $\mathcal{H}$ on the sample $\mathcal{S}_n$ is defined by

$$\hat{\mathcal{R}}(\mathcal{H}, \mathcal{S}_n) = \mathrm{E}\left[\sup_{h\in\mathcal{H}}\left(\frac{1}{n}\sum_{i=1}^n U_i h(z_i)\right)\right];$$

the Rademacher complexity of $\mathcal{H}$ is defined by

$$\mathcal{R}_n(\mathcal{H}) = \mathrm{E}_{\mathcal{S}_n\sim\mathrm{P}_Z}\left[\hat{\mathcal{R}}(\mathcal{H}, \mathcal{S}_n)\right].$$

The smaller the Rademacher complexity of a function class, the easier this class can be learnt.

Barrera et al. (2020) found out an upper bound for the variance of the error as stated in Theorem 3.4 below.

Theorem 3.4 (Statistical error for VaR (cf. Lemma 6.2 in Barrera et al. (2020)) [3]). *Under the hypotheses of Theorem 3.2 and for every $\delta \in (0,1)$, we have*

$$\|\tilde{q} - \hat{q}\|^2_{\mathrm{P}_X,2} \leq \frac{(1-\alpha)}{M_1} 2^3 \left(\left(\frac{2-\alpha}{1-\alpha} \right) B \sqrt{\frac{\log(2/\delta)}{2n}} + \mathcal{R}_n \left(\ell \circ \mathcal{F} \right) \right) \tag{18}$$

with probability at least $(1-\delta)$. The notation $\ell \circ \mathcal{F}$ is the class of functions of the form

$$(x,y) \mapsto \ell(y, f(x)) \quad \textit{for } f \in \mathcal{F}.$$

The statistical error explains how well the hypothesis space fits the loss objective function over a set of samples.

3.2 The approximation error of $\hat{s}_{\hat{q}}$

In what follows, we denote, for an arbitrary function h,

$$\mathrm{E}_{Y|X}\left[h(X,Y)\right] = \mathrm{E}\left[h(X,Y)|X\right]$$

Before starting, we provide two following inequalities

Theorem 3.5. *For a, b are two real-valued functions of X,*

$$\|a^+ - b^+\| \leq \|a - b\| \tag{19}$$

Proof. By definition 3.1, the inequality (19) is equivalent to

$$\begin{aligned} & \mathrm{E}\left[(a^+ - b^+)^2\right] \leq \mathrm{E}\left[(a-b)^2\right] \\ \Leftrightarrow \quad & \mathrm{E}\left[(a^+)^2 + (b^+)^2 - 2a^+b^+\right] \leq \mathrm{E}\left[a^2 + b^2 - 2ab\right]. \end{aligned}$$

The above inequality even holds (for sure) without the expectation. Indeed, we can consider when $ab \geq 0$ and $ab < 0$ and show that the inequality holds in both cases. □

Theorem 3.6. *Given 2 variables X, Y and a real-valued function h taking X, Y as its parameters, we have, for $p \geq 1$,*

$$\|\mathrm{E}_{Y|X} h(X,Y)\|_{\mathrm{P}_X,p} \leq \|h(X,Y)\|_{\mathrm{P}_{X,Y},p}.$$

[3] The presentation of this theorem slightly differs from the one in Barrera et al. (2020) because of their different Rademacher Complexity definition (using the sum instead of the average).

Proof. From definition 3.1,

$$\|\mathrm{E}_{Y|X}h(X,Y)\|_{\mathrm{P}_X,p} = [\mathrm{E}_X\, |\mathrm{E}_{Y|X}h(X,Y)|^p]^{1/p} \tag{20}$$

On the other hand,

$$\|h(X,Y)\|_{\mathrm{P}_{X,Y},p} = [\mathrm{E}_X\mathrm{E}_{Y|X}\, |h(X,Y)|^p]^{1/p} \tag{21}$$

For $p \geq 1$, $\phi : \mathbb{R} \ni z \mapsto |z|^p$ is a Borel convex function and without loss of generality, suppose that $\mathrm{E}|Z| < \infty$. By Jensen's inequality, we infer that

$$\phi(\mathrm{E}(Z)) \leq \mathrm{E}\phi(Z). \tag{22}$$

Theorem 3.6 is then proven by combining (20), (21) and (22). □

Based on the RU representation in Theorem 2.3 and on Definition 2.2, we infer

$$s(X) = \mathrm{E}_{Y|X}\left[\ell(Y, q(X))\right].$$

The error associated to the empirical estimator of ES is then upper bounded by

$$\begin{aligned} \|s - \hat{s}_{\hat{q}}\| &\leq \|\mathrm{E}_{Y|X}\ell(Y,q) - \mathrm{E}_{Y|X}\ell(Y,\hat{q})\| + \|\mathrm{E}_{Y|X}\ell(Y,\hat{q}) - \hat{s}_{\hat{q}}\| \\ &\leq \underbrace{\|\ell(Y,q) - \ell(Y,\hat{q})\|_{\mathrm{P}_{X,Y},2}}_{\mathrm{I}} \quad + \underbrace{\|\ell(Y,\hat{q}) - \hat{s}_{\hat{q}}\|_{\mathrm{P}_{X,Y},2}}_{\mathrm{II}}, \end{aligned} \tag{23}$$

where Theorem 3.6 is applied to both two terms in the right hand side.

From equation (5), the first term I of (23) is equal to

$$\begin{aligned} \mathrm{I} &= \left\|\epsilon^{-1}(Y-q)^+ + q - \epsilon^{-1}(Y-\hat{q})^+ - \hat{q}\right\|_{\mathrm{P}_{X,Y},2} \\ &\leq \epsilon^{-1}\left\|(Y-q)^+ - (Y-\hat{q})^+\right\|_{\mathrm{P}_{X,Y},2} + \|q - \hat{q}\| \\ &\leq \epsilon^{-1}\|q-\hat{q}\| + \|q-\hat{q}\| \\ &= (1+\epsilon^{-1})\|q-\hat{q}\| \end{aligned} \tag{24}$$

where the second to third step follows from triangle inequality and Theorem 3.5. Hence, the bound of term I is associated with the approximation error of $\hat{q}$.

For the second term II, we recall the definition of $\hat{s}_{\hat{q}}$:

$$\hat{s}_{\hat{q}} \in \arg\min_{g\in\mathcal{G}} \frac{1}{n}\sum_{i=1}^{n}(\ell(y_i, \hat{q}(x_i)) - g(x_i))^2$$

One can see that $\ell(Y,\hat{q})$ is a target function and $\hat{s}_{\hat{q}}$ is its empirical estimator in space $\mathcal{G}$. For the sake of simplicity, we denote hereafter $\hat{g} \equiv \hat{s}_{\hat{q}}$.

By plugging $Z = (X, Y)$ in Theorem 9.3 (in Appendix 9.2) and assuming furthermore

$$T = \sup_{g\in\mathcal{G}} \|\ell(Y,\hat{q}) - g\|_{\mathrm{P}_{X,Y},\infty},$$

the following inequality holds, with probability at least $1-\delta$,

$$\|\ell(Y,\hat{q})-\hat{g}\|_{\mathrm{P}_{X,Y},2}^2 \leq \frac{1}{n}\sum_{i=1}^{n}|\ell(y_i,\hat{q}(x_i))-\hat{g}(x_i)|^2 + 4T\hat{\mathcal{R}}(\mathcal{G},\mathcal{D}_n) + 3T^2\sqrt{\frac{\log(2/\delta)}{2n}} \tag{25}$$

We now investigate the empirical quadratic loss. For this purpose, let us use the following notations:

$$\hat{L}(g) = \frac{1}{n}\sum_{i=1}^{n}|\ell(y_i,\hat{q}(x_i))-g(x_i)|^2 \qquad \text{empirical loss (risk)} \tag{26}$$

$$L(g) = \mathrm{E}\left[|\ell(Y,\hat{q})-g|^2\right] = \mathrm{E}\left[\hat{L}(g)\right] \quad \text{expected loss (risk)} \tag{27}$$

As $\hat{g}$ is the empirical risk minimizer, one can deduce that

$$\hat{L}(\hat{g}) \leq \hat{L}(g) \quad \text{for } \forall g \in \mathcal{G} \tag{28}$$

On another note, for finite hypothesis space $\mathcal{G}$, a fixed $g \in \mathcal{G}$ and i.i.d samples $|\ell(y_i,\hat{q}(x_i))-g(x_i)|^2$ bounded by T^2, we infer from Theorem 9.5 (in Appendix 9.2) that

$$\mathrm{P}\left[\hat{L}(g)-L(g)\geq\varepsilon\right] \leq \exp\left(-\frac{2n\varepsilon^2}{T^4}\right)$$

The next step then follows from a union bound:

$$\begin{aligned}\mathrm{P}\left[\forall g\in\mathcal{G}, \hat{L}(g)-L(g)\geq\varepsilon\right] &\leq \sum_{g\in\mathcal{G}}\mathrm{P}\left[\hat{L}(g)-L(g)\geq\varepsilon\right]\\ &\leq \sum_{g\in\mathcal{G}}\exp\left(-\frac{2n\varepsilon^2}{T^4}\right) = |\mathcal{G}|\exp\left(-\frac{2n\varepsilon^2}{T^4}\right)\end{aligned}$$

This is equivalent to

$$\mathrm{P}\left[\forall g\in\mathcal{G}, \hat{L}(g)-L(g)\leq\varepsilon\right] \geq 1-|\mathcal{G}|\exp\left(-\frac{2n\varepsilon^2}{T^4}\right) \tag{29}$$

By setting $\delta = |\mathcal{G}|\exp\left(-\frac{2n\varepsilon^2}{T^4}\right)$ and from (28), with probability at least $1-\delta$, we have

$$\begin{aligned}\hat{L}(\hat{g}) \leq \hat{L}(g) &\leq L(g) + \underbrace{T^2\sqrt{\frac{\log|\mathcal{G}|+\log(1/\delta)}{2n}}}_{\varepsilon} \quad \text{for } \forall g\in\mathcal{G}\\ &\leq \inf_{g\in\mathcal{G}} L(g) + \varepsilon. \end{aligned} \tag{30}$$

Hence, with probability at least $1-\delta$,

$$\begin{aligned}\hat{L}(\hat{g}) &\leq \inf_{g\in\mathcal{G}}\left\|\epsilon^{-1}(Y-\hat{q})^+ + \hat{q} - g\right\|_{\mathrm{P}_{X,Y},2}^2 + \varepsilon\\ &\leq \left\|\epsilon^{-1}(Y-\hat{q})^+ + \hat{q} - q\right\|_{\mathrm{P}_{X,Y},2}^2 + \inf_{g\in\mathcal{G}}\|q-g\|^2 + \varepsilon\\ &\leq \epsilon^{-1}\left\|(Y-q)^+ + (q-\hat{q})^+\right\|_{\mathrm{P}_{X,Y},2}^2 + \|\hat{q}-q\|^2 + \inf_{g\in\mathcal{G}}\|q-g\|^2 + \varepsilon\\ &\leq \epsilon^{-1}\left\|(Y-q)^+\right\|_{\mathrm{P}_{X,Y},2}^2 + \epsilon^{-1}\|q-\hat{q}\|^2 + \|\hat{q}-q\|^2 + \inf_{g\in\mathcal{G}}\|q-g\|^2 + \varepsilon\\ &\leq \epsilon^{-1}\left\|(Y-q)^+\right\|_{\mathrm{P}_{X,Y},2}^2 + (1+\epsilon^{-1})\|q-\hat{q}\|^2 + \inf_{g\in\mathcal{G}}\|q-g\|^2 + \varepsilon,\end{aligned} \tag{31}$$

where the first step follows from (27) and Definition 3.1, the third step follows from inequality $(a+b)^+ \leq a^+ + b^+$, and the second and fourth steps follow from the triangle inequality and from inequality $\|a^+\| \leq \|a\|$.

Hence, by a combination of (23), (24), (25) and (31), and note that inequality $\sqrt{a^2+b^2} \leq |a| + |b|$ allows us to get rid of the square, we finally conclude the following theorem.

Theorem 3.7 (Approximation error for ES). *Given an estimator $\hat{q}$, loss ℓ and finite hypothesis space $\mathcal{G}$, let*

$$T = \sup_{g \in \mathcal{G}} \|\ell(Y, \hat{q}) - g\|_{\mathrm{P}_{X,Y},\infty} .$$

The approximation error of ES is bounded by

$$\begin{aligned} \|s - \hat{s}_{\hat{q}}\| \leq & \inf_{g \in \mathcal{G}} \|q - g\| \\ & + 2(1+\epsilon^{-1}) \|q - \hat{q}\| \\ & + \epsilon^{-1} \left\|(Y-q)^+\right\|_{\mathrm{P}_{X,Y},2} \\ & + \sqrt{4T\hat{\mathcal{R}}(\mathcal{G}, \mathcal{D}_n) + T^2\sqrt{\frac{\log|\mathcal{G}| + \log(1/\delta)}{2n}} + 3T^2\sqrt{\frac{\log(2/\delta)}{2n}}} \end{aligned} \tag{32}$$

with probability at least $1-\delta$.

Consequently the approximation error for ES is controlled by a combination of factors corresponding to the different lines in (32): (i) the L^2 projection error of the true VaR on hypothesis space, (ii) the approximation error of numerical VaR estimates, (iii) the own distribution of Y and X, and (iv) the goodness of fit of the hypothesis space, loss function and sample data.

A. Bounds for infinite hypothesis spaces

We use the notation $\ell_{es}(x, y, g)$ for training loss of ES estimates using $\hat{q}$, which is defined by

$$\mathbb{R}^d \times \mathbb{R} \times \mathcal{G} \ni (x, y, g) \mapsto \ell_{es}(x, y, g) = [\ell(y, \hat{q}(x)) - g(x)]^2 \tag{33}$$

Thus, the empirical loss $\hat{L}(g)$ and expected loss $L(g)$ are written as

$$\begin{aligned} \hat{L}(g) &= \frac{1}{n}\sum_{i=1}^{n} \ell_{es}(x_i, y_i, g) \quad \text{for } (x_i, y_i) \in \mathcal{D}_n \\ L(g) &= \mathrm{E}\left[\ell_{es}(X, Y, g)\right] = \mathrm{E}\left[\hat{L}(g)\right] \end{aligned}$$

Without loss of generality, we assume that the hypothesis space $\mathcal{G}$ is bounded by B.

Corollary 3.8. *Under Assumption 1 and the assumption about the bound of $\mathcal{G}$,*

$$\sup_{g \in \mathcal{G}} \|\ell_{es}(g)\|_{\mathrm{P}_{X,Y},\infty} \leq \left(\left(2 + \epsilon^{-1}\right) B\right)^2$$

Proof. For $g \in \mathcal{G}$,

$$\|\ell_{es}(g)\|_{\mathrm{P}_{X,Y},\infty} = \left\|(\epsilon^{-1}(Y-\hat{q})^{+} + \hat{q} - g)^2\right\|_{\mathrm{P}_{X,Y},\infty} \tag{34}$$

As both the target Y and hypothesis space $\mathcal{F}$ of VaR estimates are bounded by B (i.e. Assumption 1), one can deduce that

$$\left\|\epsilon^{-1}(Y-\hat{q})^{+} + \hat{q}\right\|_{\mathrm{P}_{X,Y},\infty} \leq \sup_{f\in\mathcal{F}} \left\|\epsilon^{-1}(Y-f)^{+} + f\right\|_{\mathrm{P}_{X,Y},\infty} \leq \left(1+\epsilon^{-1}\right)B \tag{35}$$

The conclusion follows from (34), (35) and the bound B of $\mathcal{G}$. □

Thus, L and $\hat{L}$ take value in $\left[0, \left(\left(2+\epsilon^{-1}\right)B\right)^2\right]$.

Corollary 3.9. *$\ell_{es}(g)$ is $\left(2\left(2+\epsilon^{-1}\right)B\right)$-Lipschitz with respect to the l_2 norm on the domain $\mathcal{G}$. This is equivalent to*

$$|\ell_{es}(g) - \ell_{es}(g')| \leq 2(2+\epsilon^{-1})B|g-g'| \tag{36}$$

for $g, g' \in \mathcal{G}$.

Proof. For $g, g' \in \mathcal{G}$,

$$\begin{aligned}
|\ell_{es}(g) - \ell_{es}(g')| &= |g^2 - (g')^2 + 2\left(\epsilon^{-1}(Y-\hat{q})^{+} + \hat{q}\right)(g'-g)| \\
&= |g + g' - 2(\epsilon^{-1}(Y-\hat{q})^{+} + \hat{q})| \times |g-g'| \\
&\leq \left(|g - \epsilon^{-1}(Y-\hat{q})^{+} + \hat{q}| + |g' - \epsilon^{-1}(Y-\hat{q})^{+} + \hat{q}|\right)|g-g'| \\
&\leq 2\left(2+\epsilon^{-1}\right)B|g-g'|
\end{aligned}$$

where the first and second inequalities follow from the triangle inequality and Corollary 3.8. □

The square of term II of (23) is indeed $L(\hat{g})$ which can be rewritten as follows:

$$L(\hat{g}) = L(\hat{g}) - L(g^*) + L(g^*) = L(\hat{g}) - L(g^*) + \inf_{g\in\mathcal{G}} L(g) \tag{37}$$

with g^* and $\hat{g}$ is the minimizer of expected loss $L(g)$ and of empirical loss $\hat{L}(g)$, respectively (i.e. $\tilde{s}_{\hat{q}}$ and $\hat{s}_{\hat{q}}$ in (14)). We then concentrate in the first two terms of (37):

$$\begin{aligned}
L(\hat{g}) - L(g^*) &= \left[L(\hat{g}) - \hat{L}(\hat{g})\right] + \left[\hat{L}(\hat{g}) - \hat{L}\left(g^*\right)\right] + \left[\hat{L}\left(g^*\right) - L\left(g^*\right)\right] \\
&\leq \left|L(\hat{g}) - \hat{L}(\hat{g})\right| + 0 + \left|\hat{L}\left(g^*\right) - L\left(g^*\right)\right| \\
&\leq 2\sup_{g\in\mathcal{G}} \left|L(g) - \hat{L}(g)\right| \qquad (38)
\end{aligned}$$

Here we want to apply the Rademacher complexity bounds in Theorem 9.2 and 9.3 for (38), but these theorems concern one-sided bounds. To deal with the two sides of (38), we first introduce risks L' and $\hat{L}'$ defined by the negative loss $\ell'_{es} = -\ell_{es}$ and then use the symmetry of Rademacher complexity (cf. Theorem 9.4) to find out a bound of (38). Thus, L' and $\hat{L}'$ are bounded in $\left[-\left(\left(2+\epsilon^{-1}\right)B\right)^2, 0\right]$. Given a positive ε, (38) implies:

$$\begin{aligned} \mathrm{P}\left[L(\hat{g}) - L(g^*) \geq \varepsilon\right] &\leq \mathrm{P}\left[\sup_{g\in\mathcal{G}} \left|L(g) - \hat{L}(g)\right| \geq \frac{\varepsilon}{2}\right] \\ &\leq \mathrm{P}\left[\sup_{g\in\mathcal{G}} \left(L(g) - \hat{L}(g)\right) \geq \frac{\varepsilon}{2}\right] + \mathrm{P}\left[\sup_{g\in\mathcal{G}} \left(L'(g) - \hat{L}'(g)\right) \geq \frac{\varepsilon}{2}\right] \end{aligned} \tag{39}$$

By applying Theorem 9.3 (in Appendix 9.2) and Corollary 3.8, the following inequality holds with probability at least[4] $1-\delta/2$, for all $g \in \mathcal{G}$:

$$L(g) - \hat{L}(g) \leq 4\left(2+\epsilon^{-1}\right)B\hat{\mathcal{R}}(\mathcal{G},\mathcal{D}_n) + 3\left(\left(2+\epsilon^{-1}\right)B\right)^2\sqrt{\frac{\log(4/\delta)}{2n}}$$

This is equivalent to,

$$\mathrm{P}\left[\forall g \in \mathcal{G}, L(g) - \hat{L}(g) \geq 4\left(2+\epsilon^{-1}\right)B\hat{\mathcal{R}}(\mathcal{G},\mathcal{D}_n) + 3\left(\left(2+\epsilon^{-1}\right)B\right)^2\sqrt{\frac{\log(4/\delta)}{2n}}\right] \leq \frac{\delta}{2} \tag{40}$$

By applying Theorem 9.2 (in Appendix 9.2) and the bound of L' and $\hat{L}'$, we infer the following inequality for all $g \in \mathcal{G}$ and with probability at least $1-\delta/2$:

$$\begin{aligned} L'(g) - \hat{L}'(g) &\leq 2\hat{\mathcal{R}}(-\ell_{es}\circ\mathcal{G},\mathcal{D}_n) + 3\left(\left(2+\epsilon^{-1}\right)B\right)^2\sqrt{\frac{\log(4/\delta)}{2n}} \\ &= 2\hat{\mathcal{R}}(\ell_{es}\circ\mathcal{G},\mathcal{D}_n) + 3\left(\left(2+\epsilon^{-1}\right)B\right)^2\sqrt{\frac{\log(4/\delta)}{2n}} \\ &\leq 4\left(2+\epsilon^{-1}\right)B\hat{\mathcal{R}}(\mathcal{G},\mathcal{D}_n) + 3\left(\left(2+\epsilon^{-1}\right)B\right)^2\sqrt{\frac{\log(4/\delta)}{2n}} \end{aligned}$$

where the second step follows from Theorem 9.4 and the third step uses Corollary 3.9 and Talagrand's lemma (see the proof of Theorem 9.3). This is equivalent to,

$$\mathrm{P}\left[\forall g \in \mathcal{G}, L'(g) - \hat{L}'(g) \geq 4\left(2+\epsilon^{-1}\right)B\hat{\mathcal{R}}(\mathcal{G},\mathcal{D}_n) + 3\left(\left(2+\epsilon^{-1}\right)B\right)^2\sqrt{\frac{\log(4/\delta)}{2n}}\right] \leq \frac{\delta}{2} \tag{41}$$

Combining (39), (40) and (41), we deduce, for $\delta \in [0,1]$,

$$\mathrm{P}\left[L(\hat{g}) - L(g^*) \leq 8\left(2+\epsilon^{-1}\right)B\hat{\mathcal{R}}(\mathcal{G},\mathcal{D}_n) + 3\left(\left(2+\epsilon^{-1}\right)B\right)^2\sqrt{\frac{2\log(4/\delta)}{n}}\right] \geq 1-\delta \tag{42}$$

Hence, the following theorem is inferred from a combination of (23), (24), (37), (42) and the same demonstration of (31).

[4] We shift δ in Theorem 9.3 to $\delta/2$.

Theorem 3.10. *Under the above-stated assumptions, the approximation error of ES is bounded by*

$$\begin{aligned}\|s-\hat{s}_{\hat{q}}\| \leq & \inf_{g\in\mathcal{G}} \|q-g\| \\ & + 2(1+\epsilon^{-1})\,\|q-\hat{q}\| \\ & + \epsilon^{-1}\left\|(Y-q)^{+}\right\|_{\mathrm{P}_{X,Y},2} \\ & + \sqrt{8\,(2+\epsilon^{-1})\,B\hat{\mathcal{R}}(\mathcal{G},\mathcal{D}_n)+3\left((2+\epsilon^{-1})\,B\right)^{2}\sqrt{\frac{2\log(4/\delta)}{n}}}\end{aligned} \tag{43}$$

with probability at least $1-\delta$.

4 Neural Network Model

In this section, we explore the class of feedforward neural networks, one of the most popular algorithms in machine learning world. After providing the basic notation, we discuss on its universal approximation capabilities and its training procedure via backpropagation of errors.

4.1 Architecture of Feedforward Neural Network

For d, m, h, u positive integers and an element-wise scalar function σ, let $\mathcal{NN}_{d,m,h,u,\sigma}$ denote a family of functions that take a vector $x \in \mathbb{R}^d$ as input and return a vector $z^{h+1} \in \mathbb{R}^m$ through the following sequential mapping

$$\begin{aligned} a^0 &= z^0 = x \\ a^l &= \sigma(z^l) = \sigma\left(w^l a^{l-1} + b^l\right) \quad ,l=1,\dots,h \\ z^{h+1} &= w^{h+1}a^h + b^{h+1} \end{aligned} \tag{44}$$

This family of functions is parameterized by the weights $w^1 \in \mathbb{R}^{u\times d}, \dots, w^l \in \mathbb{R}^{u\times u}, \dots, w^{h+1} \in \mathbb{R}^{m\times u}$ and biases $b^1 \in \mathbb{R}^u, \dots b^l \in \mathbb{R}^u, \dots, b^{h+1} \in \mathbb{R}^m$. Thus, we denote the set of trainable parameters as $\theta = \left(w^1, \dots, w^{h+1}, b^1, \dots, b^{h+1}\right)$. The vectors z^l and a^l are the pre-activated and activated output of layer l, respectively.

Therefore, $\mathcal{NN}_{d,m,h,u,\sigma}$ is the set of neural networks of h hidden layers, u units per hidden layer and σ activation function, respectively. These parameters called hyperparameters of neural networks, must be defined before training process. By which, the hypothesis space would be set up.

If $h = 0$ then $\mathcal{NN}_{d,m,0,,}$ is equivalent to the linear regression and if $h > 2$ the network is said to be deep neural network. Additionally, nonlinear activation functions play a crucial role in establishing the complex mapping between inputs and outputs of the network. In regards to regression, there are several commonly selections for nonlinear activation functions:

$$\begin{aligned} &\mathrm{Sigmoid}(x) = \left(1+e^{-x}\right)^{-1}, && \mathrm{ReLU}(x) = x^{+}, \\ &\mathrm{Softplus}(x) = \ln\left(1+e^{x}\right) \quad \text{and} && \mathrm{ELU}(x) \;= x^{+} + \min(0, \exp(x)-1). \end{aligned}$$

The transformation through activated (multi-)hidden-layer(s) creates the strong approximation capabilities of neural networks, which is practically observed in many applications and also theoretically proven by universal approximation theorems (UATs).

Theorem 4.1 (A variant of universal approximation theorem (based on Pinkus (1999)))**.** *Set a continuous function but not polynomial one $\sigma : \mathbb{R} \to \mathbb{R}$ and positive integers d, b. For every continuous target function $f : \mathbb{R}^d \to \mathbb{R}^m$, every compact subset K of $\mathbb{R}^d$ and every $\epsilon > 0$, there exists a continuous function $f_\epsilon \in \mathcal{NN}_{d,m,1,u,\sigma}$ (i.e. 1-hidden layer neural networks with arbitrary hidden units u) such that the following approximation bound holds*

$$\sup_{x \in K} \|f(x) - f_\epsilon(x)\| < \epsilon.$$

In simple language, single hidden layer networks can approximate any desired continuous function with the approximation error depending on the number of hidden units (Fujita 1998). In contrast to this version of UTAs, Sontag (1991) and Leshno et al. (1993) stated that single hidden layer networks fail in solving certain problems, e.g. the inverse of a continuous but non one-to-one function f. However, 2-hidden layers networks can overcome these drawbacks. Therefore, in our later numerical applications, we consider up to 2-hidden layers networks.

4.2 Training Process via Backpropagation

Hereafter, given a data set $\mathcal{D}_n = \{(x_i, y_i) | i = 1, \dots, n\}$, the numerical estimator of function f by ANN model is the solution of the empirical loss:

$$\hat{f} \in \underset{g \in \mathcal{NN}_{d,b,h,u,\sigma}}{\arg\min} \frac{1}{n} \sum_{i=1}^{n} l(y_i, g(x_i)). \tag{45}$$

A. Loss functions

Loss objective functions determine the target that we aim to approximate. Besides the quadratic loss for conditional mean, the pinball loss for conditional quantile (cf. equation and the RU and FZ loss function for conditional expected shortfall (cf. equation (10) and (11)), we introduce here the quantile Huber loss, an approximation for conditional quantile loss (Cannon 2011).

The Huber norm $h(u)$ is a hybrid norm in which the use of l_1 and l_2 norms is switched by a given threshold of the absolute inputs

$$h(u) = \begin{cases} \frac{u^2}{2\tau} & \text{if } 0 \leq |u| \leq \tau \\ |u| - \frac{\tau}{2} & \text{if } |u| > \tau \end{cases} \tag{46}$$

The Huber norm can be used to replace the absolute term in the pinball loss

$$\rho_\alpha^H(u) = \begin{cases} \alpha h(u) & \text{if } u \geq 0 \\ (\alpha - 1) h(u) & \text{if } u < 0 \end{cases} \tag{47}$$

This formulation provides a smooth loss objective function, that help both to accelerate and to improve some optimisation algorithms, including gradient methods. When the threshold τ tends to 0, loss $\rho_\alpha^H(u)$ converges to the quantile loss. Hence, Cannon (2011) proposed a fixed schedule of τ values: $\tau = 2^i$ for $i = -8, -9, \ldots, -32$. We also take in account this schedule in our practical applications.

B. Optimisation algorithm

To simplify the next expressions, we turn problem (45) to find the optimal set of parameters (weights and biases) $\hat{\theta}$ on space Θ equivalently to the $\mathcal{NN}_{d,b,h,u,\sigma}$ search space, with the empirical average loss of n samples denoted by $\mathcal{L}(\theta, x_{1:n}, y_{1:n})$

$$\hat{\theta} \in \operatorname*{arg\,min}_{\theta \in \Theta} \mathcal{L}(\theta, x_{1:n}, y_{1:n}) \tag{48}$$

There are a considerable number of numerical methods for solving (48) and most of them are derived from gradient descent. Gradient descent is a local optimisation algorithm in which parameter θ is iteratively adjusted in the direction of the negative gradient of objective function. For a positive learning rate lr, the update at step k can be written as

$$\theta_{k+1} = \theta_k - lr \nabla_\theta \mathcal{L}(\theta_k, x_{1:n}, y_{1:n}) \tag{49}$$

Even though the calculations of gradient can be simply implemented by automatic differentiation in practice (Paszke et al. 2017), computing gradient from a huge data set is still costly in terms of computation time. In addition, gradient descent can easily get stuck at an undesired local minimum point.

For these reasons, gradient descent is practically replaced by stochastic gradient descent (SGD). The (vanilla) SGD can be seen as a stochastic approximation of gradient descent, since instead of using the entire data set, it computes the gradient from a random subset of the data (or even a single sample). Against gradient descent, SGD achieves faster computation at each iteration in trade for a lower convergence rate. Thus, the standard training process of neural networks by stochastic gradient descent is detailed in Algorithm 1.

Algorithm 1. Standard training process of neural networks by stochastic gradient descent

```
input: Data set $\mathcal{D}_n = \{(x_i, y_i)|i = 1, \dots, n\}$ and Networks hyperparameters: $h, u, \sigma$
Set learning rate $lr$, tolerance level $\epsilon$, maximum epoch $K$, batchsize $m$;
Initialize network parameters [a] $\hat{\theta} = \theta_0 = (w_0^1, \dots, w_0^{h+1}, b_0^1, \dots, b_0^{h+1})$ ;
Initialize optimal loss $\hat{\mathcal{L}} = \infty$;
Initialize optimisation step count index $r = 0$ and epoch count index $k = 0$;
Partition $\mathcal{D}_n$ into $\lfloor \frac{n}{m} \rfloor$ batches of size $m$ and note $\left(x_{\{j\}}, y_{\{j\}}\right)$ for $j$-th batch;
while $k < K$ and $\hat{\mathcal{L}} < \epsilon$ do // Do until convergence or maximum epoch reached
    for $j = 1, \dots, \lfloor \frac{n}{m} \rfloor$ do // Loop over batches
        for $l = h + 1, \dots, 1$ do // Update networks parameters
            Compute gradients $\nabla_{w^l}\mathcal{L}\left(\theta_r, x_{\{j\}}, y_{\{j\}}\right)$, $\nabla_{b^l}\mathcal{L}\left(\theta_r, x_{\{j\}}, y_{\{j\}}\right)$ ; // Use
             backpropagation as explained in part 9.4 below
            $w_{r+1}^l \leftarrow w_r^l - lr\nabla_{w^l}\mathcal{L}\left(\theta_r, x_{\{j\}}, y_{\{j\}}\right)$;
            $b_{r+1}^l \leftarrow w_r^l - lr\nabla_{b^l}\mathcal{L}\left(\theta_r, x_{\{j\}}, y_{\{j\}}\right)$;
        end
        $r \leftarrow r + 1$;
    end
    if $\hat{\mathcal{L}} < \mathcal{L}(\theta_r, x_{1:n}, y_{1:n})$ then // Keep track of best parameters
        $\hat{\mathcal{L}} \leftarrow \mathcal{L}(\theta_r, x_{1:n}, y_{1:n})$;
        $\hat{\theta} \leftarrow \theta^r$;
    end
    $k = k + 1$
end
output: Best calibrated parameters $\hat{\theta}$
```

[a] Several initialization strategies for network parameters in practice can be found in https://pytorch.org/docs/master/nn.init.html.

Additionally, there exists several variations of SGD which differ by the learning rate or/and the update direction[5]. By now, the most used algorithm as well as the one applied in this study is Adam (short for Adaptive Moment Estimation) optimizer. The algorithm modifies both the learning rate and the update direction based on the first and second moment estimation (of gradient). Adam optimizer was first proposed by Kingma & Ba (2014) with the complete process illustrated in Algorithm 2 in Appendix.

C. Regularization

This term can be alternatively added into the loss target function with the weight λ, in order to avoid the over-fitting of estimators space Θ. Considering l_2 regularizator, optimisation problem

[5] A quick overview can be found in https://towardsdatascience.com/10-gradient-descent-optimisation-algorithms-86989510b5e9

(48) turns to

$$\hat{\theta} \in \arg\min_{\theta \in \Theta} \mathcal{L}(\theta, x_{1:n}, y_{1:n}) + \frac{\lambda}{n} \|\theta\|_2^2 \tag{50}$$

D. Hyperparameter tuning

This task, i.e. determination of ANN architecture, must be implemented separately from parameter training. The objective of the hyperparameter tuning is to find a hypothesis space wherein the optimal estimator of target function f is also the best one globally. In particular, we determine the neural networks design which is the most reasonable to our problems.

At the moment, the most widely used methods are Grid search, Random search and Bayesian optimization. Grid search is the naive one which enumerates exhaustively the Cartesian product of the considered sets of all hyperparameters in order to find out the best combination based on a benchmark (e.g. loss objective on an independent validation test). Random search, which selects randomly several combinations to be tested, is proven better than Grid search regarding to both the performance and the computation time (Bergstra & Bengio 2012). Lastly, Bayesian optimisation sets a prior probability model of function mapping from hyperparameters to the benchmark and then iteratively search promising hyperparameters values based on the information revealed from observations.

Regarding to the architecture (44), we have three hyperparameters: number of hidden layers h, hidden units u and activation function σ. In the training process, the learning rate[6], batch size (cf. SGD) and weight of regularization term (λ) can be considered as other hyperparameters. For our numerical applications, Random Search is employed

5 Randomized Neural Networks

In order to accelerate the computation time in the context of XVA analysis where the regression model must be retrained at each valuation steps (can attain up to hundreds of steps), we investigate the randomized neural networks (RNN). Contrary to the standard (fully trainable) neural networks, these model classes freeze the majority of connections (i.e. network parameters), which are initially set by either a stochastic or a deterministic design. Hence, such systems go for a (possibly extremely) fast training process by trading off a (possibly acceptable) proportion of their accuracy.

We study the typical examples of RNN applying in the feedforward networks where only the (linear) output layer of the network is trained (so-called readout layer) and the hidden layer(s) is given beforehand and fixed (so-called randomized layers). Such architectures are known as Random vector functional-link (RVFL) networks.

In this section, we first provide the basic notations and discuss training algorithms for RVFL networks. We then examine two setups for randomized layers among several ones surveyed in the work of Gallicchio & Scardapane (2020).

[6] Other parameters of Adam optimizer are kept as default in PyTorch

5.1 Training Algorithm

Notations Let $\widetilde{\mathcal{NN}}_{d,m,h,u,\sigma}$ denote the set of RVFL networks that belong to $\mathcal{NN}_{d,m,h,u,\sigma}$, but is only parameterized by the last layer (i.e. w^{h+1}, b^{h+1}). As its definition, the RVFL networks can be rewritten as a composite function of the randomized part and the readout part

$$\widetilde{\mathcal{NN}}_{d,m,h,u,\sigma} = \left\{ \mathbb{R}^d \ni x \to w\tilde{a}(x) + b \in \mathbb{R}^m \,\middle|\, w \in \mathbb{R}^{m \times u}, b \in \mathbb{R}^m, \tilde{a} \,:\, \mathbb{R}^d \to \mathbb{R}^u \right\} \tag{51}$$

with $\tilde{a}$ is a stack of h (activated by σ) hidden layers transformation. All parameters of the mapping $\tilde{a}$ are pre-selected (e.g. by randomization) and excluded in the training process. As of now, we call $\tilde{a}$ as randomized layer(s). The matrix w and vector b represent the linear readout and they are trainable parameters of the model. As a consequence, all the training algorithms for linear model can be applied basically. For example, the least mean square algorithm using matrix inversion for conditional expectation and linear programming (e.g. simplex, interior point methods) or Iterative Reweighted Least Squares (IRLS) method for conditional quantile (cf. Appendix 9.5).

According to Igelnik & Pao (1995), Huang et al. (2006), this class of networks theoretically retains the universal approximation capabilities of fully trainable neural networks. Additionally, Wang & Li (2017) noticed that the practical performance of RVFL networks depends strongly on the selection of the fixed coefficients (i.e. randomized layer).

Training the networks We discuss now some training algorithms for RVFL networks. Recall that in comparison with fully trainable networks, RNNs significantly accelerate the computation by dropping an acceptable percentage of accuracy. Hence, achieving the optimal accuracy/ computational efficiency trade-offs (ratios) is the primary challenge of the randomized fashions as well as of the optimization of trainable parameters.

Similar to the standard neural networks, RNNs can be learned by gradient methods (via backpropagation). Given the RVFL networks of the form (51), instead of performing backpropagation at the optimization step, solely gradients (of the loss function) for readout layer parameters (w and b) are required.

In addition, as mentioned above, we can use linear optimization methods for training the readout layer. Retaining the data set $\mathcal{D}_n = \{(x_i, y_i) | i = 1, \dots, n\}$, let the boldface notations for row concentrated matrices: $\mathbf{y}$ for output and $\tilde{\mathbf{A}}$ for output matrix of randomized layer $\tilde{a}(x_i)$ column stacking with one. The model parameter β is a column concentrated matrix of w and b. Thus, the numerical training procedure of RVFL networks for conditional mean -the most common machine learning task- can be expressed as follows:

$$\hat{\beta} = \arg\min_{\beta} \|\tilde{\mathbf{A}}\beta - \mathbf{y}\|^2 + \lambda\|\beta\|^2 \tag{52}$$

with l_2 regularization term weighted by λ.

Although the matrix $\tilde{\mathbf{A}}^T\tilde{\mathbf{A}}$ may be not invertible, by adding the l_2 penalty to the cost function, problem (52) can always be solved by following equation [7]

$$\hat{\beta} = \left(\tilde{\mathbf{A}}^T\tilde{\mathbf{A}} + \lambda\mathbf{I}\right)^{-1}\tilde{\mathbf{A}}^T\mathbf{y} \tag{53}$$

with $\mathbf{I}$ is a $n \times n$ identity matrix.

For the medium-sized data sets, the inverse of matrix solution performs much more efficiently than gradient methods. Even for big scale problems, many approximations and algorithmic advances are available to compute (53) and they are still less costly than the vanilla stochastic gradient descent (Gallicchio & Scardapane 2020).

Otherwise, for conditional quantile (i.e. conditional VaR), the optimization problem can be solved via Iterative Reweighted Least Squares method (cf. Appendix 9.5).

Summing up, the computation of RVFL networks is rapid not only because the majority of parameters (randomized layer) are fixed, but also because of the efficient accessibility for training the remaining parameters (readout layer). In the next subsections, we will concentrate on several popular setups for randomized layer.

5.2 Extreme Learning Machines

Extreme learning machines (ELM), as named by its main promotor Huang et al. (2006), can be considered as a class of RVFL networks, where the randomized layers are assigned by random fashions (e.g. standard Gaussian) and the readout layer is learned (usually in one step) by linear models. Throughout its development, ELM research has mainly focused on a single randomized hidden layer with a large number of design variants such as random projection, kernel methods, Laplacian transform, Wavelet transform or even some classical unsupervised techniques such as principal component analysis (PCA), etc. Hence, for the sake of clarity, **we refer to the class of RVFL networks assigning randomized layer by random projections as ELM networks in our study**, denoted by

$$\mathcal{ELM}_{d,m,u,\sigma,R} = \left\{\mathbb{R}^d \ni x \to w\sigma(\tilde{w}x + \tilde{b}) + b \in \mathbb{R}^m \,\middle|\, w \in \mathbb{R}^{m\times u}, b \in \mathbb{R}^m, \tilde{w} \sim R^{u\times d}, \tilde{b} \sim R^u\right\} \tag{54}$$

where $R^{u\times d}$ is random $u\times d$ matrices with entries independently drawn from a given distribution R. Here are several choices of R:

- standard normal distribution $\mathcal{N}(0,1)$
- distribution taking values $+1$ and -1 with probability $\frac{1}{2}$ for each (Achlioptas 2003)
- distribution taking values $+\sqrt{3}, 0$ and $-\sqrt{3}$ with probability $\frac{1}{6}, \frac{2}{3}, \frac{1}{6}$ respectively (Achlioptas 2003).

For numerical applications, we name these distributions by Gaussian, Rademacher and Achlioptas, respectively.

In fact, random projections have been widely used as a pre-processing in linear regression, for reducing the dimensionality of the dataset. The main motivation of random projection is

[7] Equation (53) is the solution in the case that the number of observations is greater than the dimension of input.

given by the Johnson-Lindenstrauss lemma, which states that points in high dimensional vector space may be projected into a suitable lower dimensional space with the distances between the points being approximately preserved.

Lemma 5.1 (Johnson–Lindenstrauss lemma)**.** *Given $\varepsilon \in (0,1)$, a data set X of n samples in $\mathbb{R}^d$ and a number $k > 8\ln(n)/\varepsilon^2$, there exists a linear map $f : \mathbb{R}^d \to \mathbb{R}^k$ such that*

$$(1-\varepsilon)\|u-v\|^2 \leq \|f(u)-f(v)\|^2 \leq (1+\varepsilon)\|u-v\|^2$$

for all $u, v \in X$.

Readers may refer to Maillard & Munos (2012) for an analysis of random projection in the context of linear regression.

However, in the context of Extreme Learning Machines, random projection is not only used as a dimensionality reduction. Actually, it can also be used to map the input into a higher dimensional space (number of units in the hidden layer greater than the original input dimension). Used in combination with the application of a nonlinear activation function to the transformed inputs, this plays an important role in the approximation capabilities of ELM. Huang et al. (2006) proved two universal approximation theorems of ELM. The first one states that the single randomized layer networks of n hidden units (i.e $\mathcal{ELM}_{d,b,n,\sigma,R}$) can learn a target with zero squared error from n distinct samples (cf. Huang et al. (2006, Theorem 2.1)). The second one broadly says that an ELM can learn a target with the error smaller than a given small positive ϵ depending on the number of hidden units:

Theorem 5.2 (Universal approximation theorem of ELM (Theorem 2.2 in Huang et al. (2006)))**.** *Given any small positive value ε and a nonlinear activation function σ which is infinitely differentiable in any interval, there exists $u \leq n$ such that for n arbitrary distinct samples $\mathcal{D}_n = \{(\mathbf{x},\mathbf{y})\} = \{(x_i,y_i)|i = 1,\ldots,n\}$, where $x_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}^m$, then the following inequality holds for all $f \in \mathcal{ELM}_{d,m,u,\sigma,R}$*

$$\|f(\mathbf{x})-\mathbf{y}\| \leq \varepsilon$$

with probability equals to one and $f(\mathbf{x})$ is the row concentrated matrix of $f(x_i)$.

5.3 Radial Basis Function Networks

One alternative to generate the randomized layer is the use of radial basis function (RBF). RBF is a popular kernel function which is commonly used in supporting vector machine. This function takes the distance of two points (e.g. Euclidean distance) as input and it returns a real value which can be interpreted as a measure quantifying the similarity of two points. Taking this idea, Broomhead & Lowe (1988) formulated the radial basis function network as the set of functions of the form

$$\mathcal{RBF}_{d,m,u,\phi} = \left\{ \mathbb{R}^d \ni x \to w \begin{bmatrix} \phi(\|x-c_1\|) \\ \vdots \\ \phi(\|x-c_u\|) \end{bmatrix} \in \mathbb{R}^m \;\middle|\; w \in \mathbb{R}^{m\times u}, c_i \in \mathbb{R}^d \text{ for } i = 1,\ldots,u \right\} \tag{55}$$

where c_i are center vectors of the RBF functions in the hidden layer. They can be trainable parameters of the model. However, as part of randomized neural network, we use some predefined setups for center vectors (e.g. k-means clustering in our experiments). The number of center vectors is also the number of hidden units in RBF networks. Besides, ϕ is a radial basis function which playing a role as a nonlinear activation function, with some common choices as below:

- Gaussian: $\phi(r) = \exp -(\kappa r)^2$
- Inverse quadratic: $\phi(r) = (1+(\kappa r)^2)^{-1}$
- Multiquadric: $\phi(r) = \sqrt{1+(\kappa r)^2}$
- Thin plate spline: $\phi(r) = r^2 \ln(r)$

The approximation capabilities of RBF networks have been widely studied in the context of kernel methods. We state one in the theorem below.

Theorem 5.3 (Universal approximation theorem of RBF networks (Theorem 1 and Corollary 1 in Park & Sandberg (1991))). *Let $\phi(\|\cdot\|) : \mathbb{R}^d \mapsto \mathbb{R}$ be an integrable bounded function such that $\phi(\|\cdot\|)$ is continuous almost everywhere and $\int_{\mathbb{R}^d} \phi(\|x\|)dx \neq 0$. Then $\mathcal{RBF}_{d,b,u,\phi}$ are universal approximators on a compact subset of $\mathbb{R}^d$, i.e. $\mathcal{RBF}_{d,b,u,\phi}$ are dense in $L^p(\mathbb{R}^d)$ for $p \geq 1$.*

6 Experimental Design

With the motivation to prepare for XVA computations, we study two financial problems: (i) pricing a portfolio of calls in the Black-Scholes (B&S) framework, i.e. estimation of Conditional Expectation and (ii) VaR and ES regression with normal distributed features.

6.1 Pricing Problem

Model setup We consider a portfolio of d calls with the following parameters:

- Constant risk-free rate $r = 0.025$
- Time to maturity $T = 0.5$
- For $i = 1, \ldots, d$,
  - Call strikes: $K^i \overset{\text{iid}}{\sim} Uniform([80, 120])$
  - Call volatilities: $\sigma^i \overset{\text{iid}}{\sim} Uniform([0.1, 0.5])$
  - Call weights: $\omega^i \overset{\text{iid}}{\sim} Uniform([0, 1])$.

The input is the vector of initial underlying prices $S_0 = [S_0^1 \ldots S_0^d]$ such that $S_0^{i=1,\ldots,d} \sim Uniform([10+5i, 180+5i])$. The output value is the portfolio price calculated by discounted payoff and Black-Scholes stock prices:

$$\pi = e^{-rT} \sum_{i=1}^{d} \omega^i \left(S_T^i - K^i\right)^+ \tag{56}$$

where

$$S_T^i = S_0^i \exp\left(rT - \frac{1}{2}(\sigma^i)^2 T + \sigma^i \sqrt{T}\epsilon\right) \quad \text{for} \quad \epsilon \sim \mathcal{N}(0,1)$$

For the sake of efficiency in neural network framework, the input is normalized by its empirical mean and standard error over training set and the output is scaled by its standard error of training set. In addition, two shapes of traing (also test) data are examined for pricing problem: $n = 20,000 \times d = 40$ and $n = 50,000 \times d = 10$.

Benchmarking Training algorithms are benchmarked by the normalized root-mean-square error

$$NRMSE(\hat{y}_{1:n}, y_{1:n}) = \frac{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}}{std(y_{1:n})} \tag{57}$$

where n denotes the number of test sample, y and $\hat{y}$ are the targets and its predictions. The target that we want to approach in the pricing problem is the analytical B&S call prices defined as follows:

$$\pi^{BS} = \sum_{i=1}^{d} \omega^i \left(S_0^i \Phi(d_1^i) - K_i e^{-rT} \Phi(d_2^i) \right) \tag{58}$$

where $\Phi(\cdot)$ is the cumulative distribution function of the standard normal distribution and

$$\begin{aligned} d_1^i &= \frac{1}{\sigma^i \sqrt{T}} \left[\ln\left(\frac{S_0^i}{K^i}\right) + \left(r + \frac{(\sigma^i)^2}{2}\right) T \right] \\ d_2^i &= d_1^i - \sigma^i \sqrt{T} \end{aligned}$$

Note that the B&S price (58) is the true conditional expectation which will not be neither used for training nor for tuning models. Training step uses observations of target generated by (56).

6.2 VaR and ES Case Study

Model setup Let the dimension $d = 40$, we suppose the input vector $X \sim \mathcal{N}(0, \mathbf{I}_d)$ [8] and $Y \mid X \sim \mathcal{N}\left(P_1(X), (P_2(X))^2\right)$ where P_1 and P_2 are multivariate polynomials of degree 2. For every $x = (x_1, \ldots, x_d) \in \mathbb{R}^d$, and coefficients λ and μ following a uniform distribution between -1 and 1,

$$\begin{aligned} P_1(x) &= \sum_{i=1}^{d} \lambda_i x_i + \sum_{1 \le i,j \le d} \lambda_{i,j} x_i x_j \\ P_2(x) &= \sum_{i=1}^{d} \mu_i x_i + \sum_{1 \le i,j \le d} \mu_{i,j} x_i x_j \end{aligned}$$

Each of training and test sets includes $50,000$ samples drawn from the model setting. In this case study, the data does not need to be scaled or normalized.

Benchmarking As the conditional probability distribution of Y given X is known, that allows us to compute the true conditional VaR and ES of Y given X at confidence level $(1-\epsilon)$ $(= 97.5\%)$. Denoting by Φ the cumulative distribution function of the standard normal distribution and by φ its density, we have:

$$\begin{aligned} \mathrm{VaR}(Y \mid X) &= P_1(X) - P_2(X)\Phi^{-1}(\epsilon) \\ \mathrm{ES}(Y \mid X) &= P_1(X) + \frac{1}{\epsilon} P_2(X) \varphi\left(\Phi^{-1}(\epsilon)\right) \end{aligned}$$

[8] $\mathbf{I}_d$ is a $d \times d$ identity matrix.

We then benchmark training models by NRMSE between the prediction and the true metrics. Note that the true metrics will be neither used for training nor for tuning models. The training process is implemented by using simulated input and output via functional methods: two-step and joint approach.

Otherwise, for both two problems, we are highly interested in the model computation time.

7 Numerical Results

Our neural networks are trained by PyTorch and cuML on Python. Both packages allow us to perform parallel computation on GPU. In particular, cuML is a new GPU-accelerated machine learning platform developed by NVIDIA team, which does the linear regression (i.e. matrix inversion) much faster than Scikit-learn and more stably than PyTorch. Our entire work is implemented on Google Colab with the selection of GPU. A free google account has access to a machine with Intel(R) Xeon(R) CPU @ 2.20GHz, 13GB RAM, 35GB disk and a GPU NVIDIA Tesla T4.

We refer to the training model time as computation time in our report. The numerical results are reported below.

7.1 Pricing Problem

Table 1 and Figure 1 compare the performance of model classes for Pricing problem. The hyperparameters for each model are tuned to the best choice (by Random search) which can be found in Table 2. In addition, both ELM and RBF networks are optimized by matrix inversion algorithm, while standard networks are trained by Adam optimizer over 300 epochs (with early stop at $1\text{E} - 8$).

Table 1 – Average accuracy and computation time from 10 trials for Pricing problem

	Data shape 20,000 × 40				Data shape 50,000 × 10			
	NRMSE		Computational Efficiency		NRMSE		Computational Efficiency	
	in-sample	out-of-sample	Trainable parameters	Computation time (s)	in-sample	out-of-sample	Trainable parameters	Computation time (s)
NN	0.1894	0.1922	16,801	0.9605	0.0875	0.0877	1,201	0.6265
ELM	0.2189	0.2231	601	**0.1669**	**0.0817**	**0.0821**	201	**0.1442**
RBF	**0.1553**	**0.1517**	**400**	0.6083	0.1706	0.1709	**50**	0.2956

Notes: For data shape $20,000 \times 40$, NN has 1 hidden layer of 400 units, ELM and RBF have 600 and 400 units, respectively. For data shape $50,000 \times 10$, NN has 1 hidden layer of 100 units, ELM and RBF have 200 and 50 units, respectively (see Table 2 for more details).

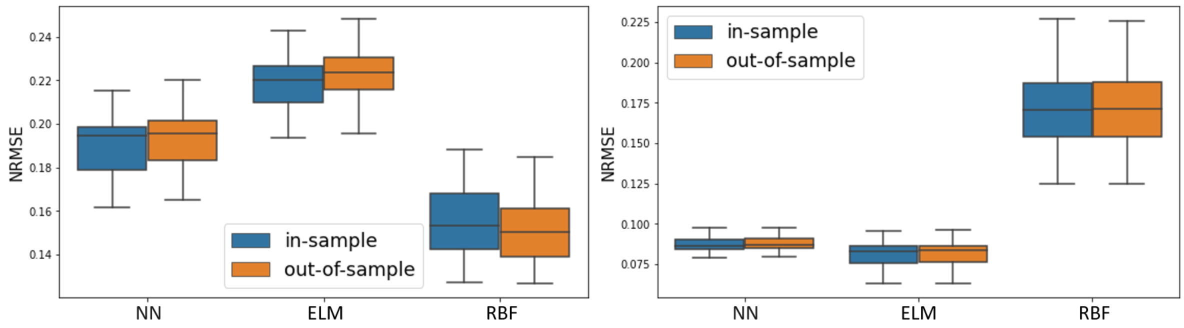

Figure 1 – Box plot from 10 trials of NRMSE for Pricing problem on data shape $20,000 \times 40$ (*Left*) and $50,000 \times 10$ (*Right*).

For both data shapes, one class of randomized neural networks (RNNs) learns more efficiently in term of computation and also achieves better accuracy than the standard networks: RBF networks perform better on higher dimensional data, while ELM networks work well on large data points. However, the first and third quantiles of RBF in the box plot (cf. Figure 1) show that the errors of this model are less stable than the others.

We also remark that the hyperparameter tunning has more impact on the class of RNNs (particularly on RBF design) than on standard networks. This is consistent with the observation made in Wang & Li (2017) about the importance of randomized layer design.

7.2 VaR and ES Case Study

A. Two-step approach

Figure 2 illustrates the accuracy and computation time of learned VaR. The density plot of the predictions is deferred to the appendix (see Figure 5).

The basic neural networks learning Huber loss (NN_Huber) perform the best among all. They attain the similar accuracy to standard NN learning quantile loss but they record also better computation time. ELM networks run extremely fast as usual but they are significantly less accurate in this case study, while RBF networks do not work (NRMSE > 1).

One can also note that for the application of RNNs in the quantile regression, the optimization by iteratively reweighted least squares algorithm is numerically better than by stochastic gradient.

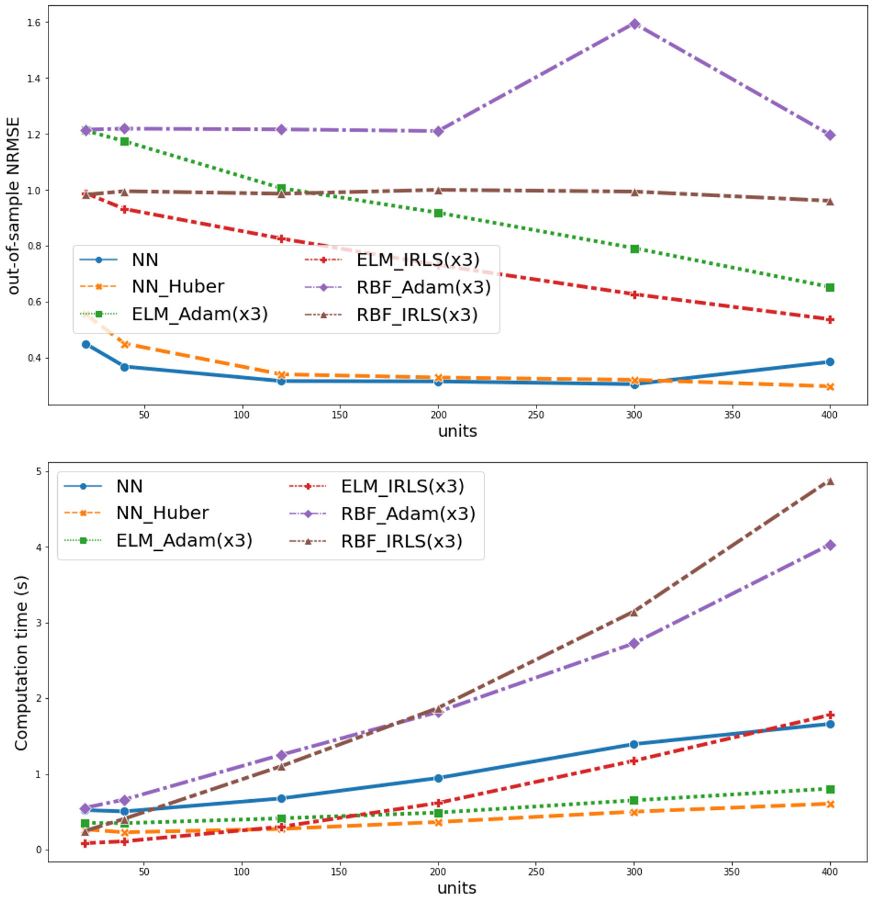

Figure 2 – (*Top*) RMSE of learned versus true VaR and (*Bottom*) Computation time in second. For models which are trained by Adam optimizer, the learning rate and number of epochs are set at 0.1 and 300, respectively. Each randomized layer has three times more (x3) units than hidden layer of standard networks. The model information is also detailed as below (regularization weight = 0 if not specified):

- NN: Standard neural networks learning RU quantile loss
- NN_Huber: Standard neural networks learning Huber loss
- ELM_Adam: ELM (Gaussian projection) networks using Adam optimizer
- ELM_IRLS: ELM (Achlioptas projection) networks using IRLS algorithm
- RBF_Adam: RBF (Multiquadric with shape parameter = 1) networks using Adam optimizer (regularization weight = 1)
- RBF_IRLS: RBF (Multiquadric with shape parameter = 1) networks using IRLS algorithm (regularization weight = 1).

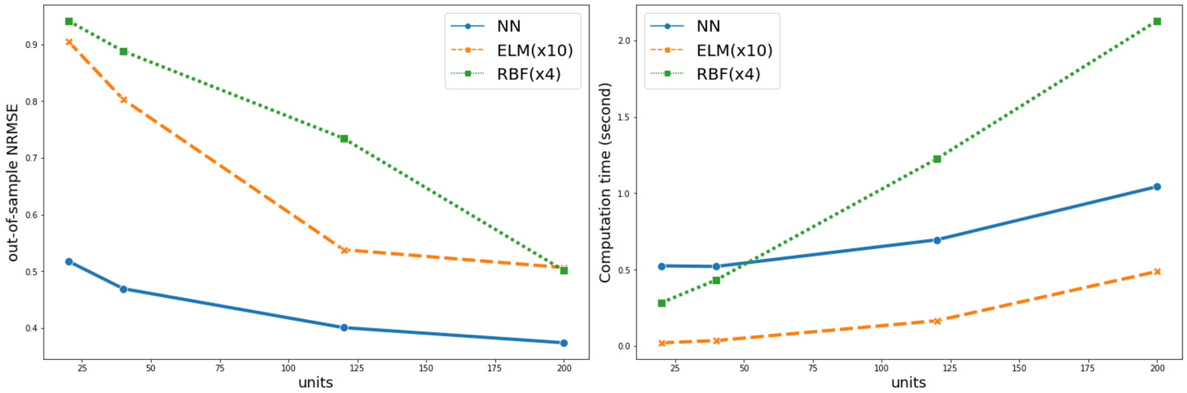

Figure 3 – (*Left*) RMSE of learned ES by NN_Huber VaR (cf. Figure 2) versus true ES and (*Right*) Computation time in second. NN uses Adam optimizer of 0.01 learning rate, 0.5 regularization weight and 300 epochs. ELM networks use Achlioptas projection and matrix inversion solution with 0.1 regularization weight. RBF networks use Thin plate spline activation with shape parameter = 10 and matrix inversion solution with regularization weight = 0.5. In comparison with standard networks, ELM networks have ten times more hidden units, while RBF networks have four times more hidden units.

Taking the prediction of NN_Huber for conditional VaR, Figure 3 analyzes the estimation of ES based on two-step approach the. One is thus back to a quadratic loss and ELM networks seem to work, providing a good trade-off between an acceptable accuracy and a fast training process. However, more units does not automatically mean more accuracy in this ES estimation (cf. density plot until 400 units in Figure 6). Moreover, even though also based on the quadratic loss function, ELM networks perform less efficiently than the pricing networks of Section 7.1 and they are in fact less accurate than standard NN. This last point can be explained by referring to the convergence analysis of ES (cf. Theorem 3.7). When learning from the same estimate of VaR (cf. by NN_Huber), the second and third factors in inequality (32) stay the same for ELM and standard NN, but the first factor for standard NN is better (smaller) than for ELM.

B. Joint approach [9]

Regarding Figure 4, the prediction of conditional VaR is similar in the joint approach implemented by standard NN and in the two-step approaches. When comparing the prediction of ES by standard networks, the two-step approach achieves a better accuracy but takes a significantly longer time (cf. Figure 4 (*Left*)). By using ELM networks for ES estimation, the two-step approach becomes computationally more efficient, but this comes at the cost of a loss in accuracy (cf. Figure 4 (*Right*)).

Intuitively, due to the complexity of the joint loss, the randomized networks perform very poorly on the joint approach and their results therefore are not reported here.

[9] The density plot of learned vs true metrics in joint approach can be found in Appendix Figure 7.

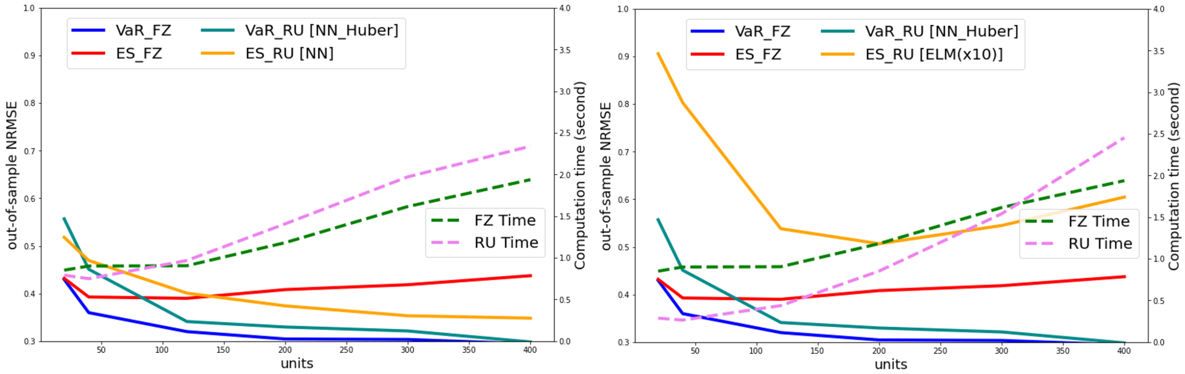

Figure 4 – RMSE of learned versus true metrics on the left axis and Computation time on the right axis, comparing joint (FZ) and two-step (RU) approach. For two-step approach, standard networks minimizing Huber loss are selected to learn VaR, while standard networks (*Left*) and ELM networks with ten times more hidden units (*Right*) are respectively selected to learn ES. The joint approach is learned by 1-hidden-layer two-outputs neural networks with 0.1 learning rate over 300 epochs and no regularization.

8 Conclusion

In this work, we studied the functional estimation of conditional Value-at-Risk and Expected Shortfall based on two-steps (Rockafellar & Uryasev 2000) and joint (Fissler & Ziegel 2016) approach. We found out an upper bound for the approximation error of ES estimates. Moreover, with the purpose of fast computation in order to prepare for our future work on XVA Analysis, we investigated the class of randomized neural networks (RVFL networks in particular). Our numerical experiment on simulation case studies reveals the efficiency of randomized networks for learning the quadratic loss (i.e. conditional expectation and conditional ES on two-steps). However, randomized networks do not perform accurately for the quantile loss nor for the joint loss. The efficient way to learn conditional VaR is to use a standard NN architecture in combination with the Huber loss function. To conclude on the conditional VaR and ES estimation, we rank three workable strategies in order of increasing accuracy and also increasing computation time:

- Two-steps approach: Learning VaR via Huber loss by standard network and then learning ES by Extreme learning machine network
- Joint approach: Applying directly two-outputs standard neural network
- Two-steps approach: Learning VaR via Huber loss and ES by two separate standard networks.

9 Appendix

9.1 Risk Measures

Definition 9.1 (Coherent risk measure)**.** *A measurable function which maps a linear space $\mathcal{L}$ of measurable functions to real positive number $\varrho : \mathcal{L} \to \mathbb{R}^+ \cup \{+\infty\}$ is said to be coherent risk* *measure for $\mathcal{L}$ if it satisfies the following properties:*

- ***Monotonicity*** *If $Z_1, Z_2 \in \mathcal{L}$ and $Z_1 \leq Z_2$ a.s., then $\varrho(Z_1) \geq \varrho(Z_2)$*
  If portfolio Z_2 is (almost surely) worthier than portfolio Z_1, then the risk of Z_2 could be less than the risk of Z_1.
- ***Positive homogeneity*** *If $\alpha \geq 0$ and $Z \in \mathcal{L}$, then $\varrho(\alpha Z) = \alpha\varrho(Z)$*
  If we change our investment capital on a portfolio Z, our risk will change with the same factor.
- ***Translation invariance*** *If A is a deterministic with guaranteed return a and $Z \in \mathcal{L}$ then*
  $$\varrho(Z + A) = \varrho(Z) - a$$
  An adjustment of a risk free asset A should reduce the risk of the portfolio.
- ***Sub-additivity*** *If $Z_1, Z_2 \in \mathcal{L}$, then $\varrho(Z_1 + Z_2) \leq \varrho(Z_1) + \varrho(Z_2)$*
  That implies the effect of diversification. A portfolio combining two assets $Z1$ and $Z2$ must be less risky than the sum of holding two assets separately.

9.2 Some Computational Learning Theories

Theorem 9.2 (Rademacher Complexity Bound [10])**.** *Given a space Z with a fixed distribution P_Z, let $\mathcal{S}_n = \{z_1, \ldots, z_n\}$ be i.i.d samples drawn from P_Z. Consider a class of functions $\mathcal{H} = \{h : Z \to [a,b] \subset \mathbb{R}\}$. For every $h \in \mathcal{H}$ and every $\delta \in (0,1)$, with probability at least $(1-\delta)$,*

$$\mathrm{E}[h(Z)] \leq \frac{1}{n}\sum_{i=1}^{n} h(z_i) + 2\mathcal{R}_n(\mathcal{H}) + (b-a)\sqrt{\frac{\log(1/\delta)}{2n}}$$

In addition, over the draw of S_n, with probability at least $(1-\delta)$,

$$\mathrm{E}[h(Z)] \leq \frac{1}{n}\sum_{i=1}^{n} h(z_i) + 2\hat{\mathcal{R}}(\mathcal{H}, \mathcal{S}_n) + 3(b-a)\sqrt{\frac{\log(2/\delta)}{2n}}$$

Theorem 9.3 (Rademacher complexity regression bounds (Theorem 10.3 in Mohri et al. (2018)))**.** *Let $p \geq 1$ and assume a target function $f(Z)$ such that $\|h - f\|_{\mathrm{P}_Z,\infty} \leq M$. Then, for any $\delta > 0$, with probability at least $1-\delta$ over a sample $\mathcal{S}_n$, each of the following inequalities holds for all $h \in \mathcal{H}$:*

$$\mathrm{E}\left[|h(Z) - f(Z)|^p\right] \leq \frac{1}{n}\sum_{i=1}^{n}\left[|h(z_i) - f(z_i)|^p\right] + 2pM^{p-1}\mathcal{R}_n(\mathcal{H}) + M^p\sqrt{\frac{\log(1/\delta)}{2n}} \tag{59}$$

$$\mathrm{E}\left[|h(Z) - f(Z)|^p\right] \leq \frac{1}{n}\sum_{i=1}^{n}\left[|h(z_i) - f(z_i)|^p\right] + 2pM^{p-1}\hat{\mathcal{R}}(\mathcal{H}, \mathcal{S}_n) + 3M^p\sqrt{\frac{\log(2/\delta)}{2n}} \tag{60}$$

[10] The proof of Rademacher Complexity can be found in Wolf (2018).

Proof. Following directly Rademacher complexity bound, we have

$$\mathrm{E}\left[|h(Z)-f(Z)|^p\right] \leq \frac{1}{n}\sum_{i=1}^{n}\left[|h(z_i)-f(z_i)|^p\right] + 2\hat{\mathcal{R}}(\mathcal{H}_p,\mathcal{S}_n) + 3M^p\sqrt{\frac{\log(n/\delta)}{2n}} \tag{61}$$

with $\mathcal{H}_p = \{Z \mapsto |h(Z)-f(Z)|^p : h \in \mathcal{H}\}$.

Let $\mathcal{H}' = \{Z \mapsto h(Z)-f(Z) : h \in \mathcal{H}\}$. Then, $\mathcal{H}_p = \xi \circ \mathcal{H}'$, with $\xi : x \mapsto |x|^p$. We observe that when ξ is bounded, the gradient of ξ is also bounded as following

$$\partial\xi = \frac{p|x|^p}{x} \leq pM^{p-1} \quad \text{for } |x| \leq M$$

Thus, ξ is said to be (pM^{p-1})-Lipschitz over $[-M, M]$. As a property of Rademacher complexity (see Wolf (2018, 5. Theorem 1.14)), also known as Talagrand's lemma, we have

$$\hat{\mathcal{R}}(\mathcal{H}_p,\mathcal{S}_n) = \hat{\mathcal{R}}(\xi \circ \mathcal{H}',\mathcal{S}_n) \leq pM^{p-1}\hat{\mathcal{R}}(\mathcal{H}',\mathcal{S}_n) \tag{62}$$

Furthermore,

$$\begin{aligned}
\hat{\mathcal{R}}(\mathcal{H}',\mathcal{S}_n) &= \mathrm{E}\left[\sup_{h-f\in\mathcal{H}'}\left(\frac{1}{n}\sum_{j=1}^{n}U_i(h(z_i)-f(z_i))\right)\right] \\
&= \mathrm{E}\left[\sup_{h\in\mathcal{H}}\left(\frac{1}{n}\sum_{j=1}^{n}U_ih(z_i)\right) - \frac{1}{n}\sum_{i=1}^{n}U_if(z_i)\right] \\
&= \mathrm{E}\left[\sup_{h\in\mathcal{H}}\left(\frac{1}{n}\sum_{j=1}^{n}U_ih(z_i)\right)\right] = \hat{\mathcal{R}}(\mathcal{H},\mathcal{S}_n) \qquad (63)
\end{aligned}$$

The third line is obtained by $U_i \perp z_i$ and $\mathrm{E}[U_i] = 0$ for $i = 1,\dots,n$. Combining (61), (62) and (63), we deduce inequality (60). Inequality (59) is then followed. □

Theorem 9.4 (Scaling of Rademacher complexity). *For a constant c,*

$$\hat{\mathcal{R}}(c\mathcal{H},\mathcal{S}_n) = |c|\hat{\mathcal{R}}(\mathcal{H},\mathcal{S}_n) \tag{64}$$

$$\mathcal{R}_n(c\mathcal{H}) = |c|\mathcal{R}_n(\mathcal{H}) \tag{65}$$

Proof. This is trivial from the symmetric definition of Rademacher variables σ_i (in Definition 3.3). □

Theorem 9.5 (Hoeffding's inequality). *Let $Z_1,\dots,Z_n$ be independent random variables such that $a_i \leq Z_i \leq b_i$ almost surely for each $i \in \{1,\dots,n\}$. Then, for any $\varepsilon > 0$*

$$\mathrm{P}\left[\frac{1}{n}\sum_{i=1}^{n}Z_i - \mathrm{E}\left(\frac{1}{n}\sum_{i=1}^{n}Z_i\right) \geq \varepsilon\right] \leq \exp\left(-\frac{2n^2\varepsilon^2}{\sum_{i=1}^{n}(b_i-a_i)^2}\right) \tag{66}$$

$$\mathrm{P}\left[\left|\frac{1}{n}\sum_{i=1}^{n}Z_i - \mathrm{E}\left(\frac{1}{n}\sum_{i=1}^{n}Z_i\right)\right| \geq \varepsilon\right] \leq 2\exp\left(-\frac{2n^2\varepsilon^2}{\sum_{i=1}^{n}(b_i-a_i)^2}\right) \tag{67}$$

9.3 Adam Optimization Algorithm

Algorithm 2 (Algorithm 1 in Kingma & Ba (2014))**.** Adam optimisation for minimizing a loss function $\mathcal{L}(\theta)$ w.r.t. parameter θ

```
require: learning rate $lr$,
         exponential decay rate for the first and second moment estimates $\beta_1, \beta_2$,
         small scalar $\epsilon$ (e.g. $10^{-8}$),
         maximum number of iterations $K$;
Initialize parameter $\theta_0 (= \hat{\theta})$ ;            // $\hat{\theta}$ for storing the best parameter
Initialize first and second moment vectors $m_0 = v_0 = 0$;
Initialize optimal loss $\hat{\mathcal{L}} = \infty$ and count index $k = 0$;
while $k < K$ and $\hat{\mathcal{L}} < \epsilon$ do      // Do until convergence or maximum iteration
 reached
    $g_k \leftarrow \nabla_\theta \mathcal{L}(\theta_k)$ ;                          // Get gradient of parameter
    $m_{k+1} \leftarrow \beta_1 m_k + (1-\beta_1) g_k$ ;          // Update biased first moment estimate
    $v_{k+1} \leftarrow \beta_2 v_k + (1-\beta_2)(g_k)^2$ ;      // Update biased second moment estimate
    $m_\theta = \frac{m_{k+1}}{1-(\beta_1)^k}$ ;           // Compute bias-corrected first moment estimate
    $v_\theta = \frac{v_{k+1}}{1-(\beta_2)^k}$ ;          // Compute bias-corrected second moment estimate
    $\theta_{k+1} \leftarrow \theta_k - \frac{lr}{\sqrt{v_\theta}+\epsilon} m_\theta$ ;                         // Update parameter
    if $\hat{\mathcal{L}} < \mathcal{L}(\theta_{k+1})$ then                    // Keep track of best parameter
        $\hat{\mathcal{L}} \leftarrow \mathcal{L}(\theta_{k+1})$;
        $\hat{\theta} \leftarrow \theta_{k+1}$;
    end
    $k \leftarrow k + 1$;
end
output: Best calibrated parameter $\hat{\theta}$.
```

9.4 Backpropagation

Backpropagation, short for backward propagation of errors, is a numerical scheme that calculates the gradient of the nested function (e.g. the feedforward networks) in an efficient and less memory consuming way. However, this term is loosely used to indicate the whole learning process of neural networks in machine learning community.

At each optimisation step, the gradients of errors w.r.t. the weights and biases of the entire networks structure are required. The backpropagation operates based on the chain rule [11], compute the gradient one layer at a time, iterate backward from the last layer in order to take of advantage the sequential calculation results.

By introducing an auxiliary δ^l for the partial derivative of loss $\mathcal{L}$ (short for $\mathcal{L}(\theta, x_{1:n}, y_{1:n})$ in equation (48)) w.r.t. to z^l pre-activated output of layer l, the backpropagation scheme

[11] The chain rule is a formula to compute the derivative of a composite function, e.g. the simplest case $\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x}$.

corresponding to structure (44) can be summarized as below

1. Compute directly gradient of last layer:
$$\delta^{h+1} = \frac{\partial \mathcal{L}}{\partial z^{h+1}}$$
2. For $l = h, \dots, 1$ compute:
$$\delta^l = \frac{\partial \mathcal{L}}{\partial z^l} = \frac{\partial \mathcal{L}}{\partial a^l} \cdot \frac{\partial a^l}{\partial z^l} = \left(\left(w^{l+1} \right)^T \delta^{l+1} \right) \odot \sigma(z^l)$$
3. The gradients of the cost function of networks parameters are easily obtained as:
$$\frac{\partial \mathcal{L}}{\partial w^l} = \delta^l \left(a^{l-1} \right)^T \quad \text{and} \quad \frac{\partial \mathcal{L}}{\partial b^l} = \delta^l \qquad \text{for } l = h+1, \dots, 1$$

with $\odot$ is the element-wise product and $(\cdot)^T$ is the notation for transposed matrix (vector).

9.5 Quantile Regression via Iterative Reweighted Least Squaresnor

The iteratively reweighted least squares (IRLS) is used to solve certain optimization problems with objective functions of the form of a p-norm. Hence, we can use this method to deal with the quantile regression. Suppose a linear model for problem (13), we aim to find out the best empirical estimator $\hat{\beta}$ such that

$$\begin{aligned} &\hat{\beta} \in \arg\min_{\beta} \frac{1}{n} \sum_{i=1}^{n} \left[\alpha(y_i - \beta^T x_i)^+ + (1-\alpha)(y_i - \beta^T x_i)^- \right] + \lambda \|\beta\|_2^2 \\ \Leftrightarrow\ &\hat{\beta} \in \arg\min_{\beta} \frac{1}{n} \sum_{i=1}^{n} w'_i |y_i - \beta^T x_i| + \lambda \|\beta\|_2^2 \end{aligned} \tag{68}$$

where
$$w'_i = \begin{cases} \alpha & \text{if } y_i - \beta^T x_i \geq 0 \\ (\alpha - 1) & \text{otherwise} \end{cases} \tag{69}$$

IRLS proposes replacing the l_1-norm and the weight w'_i in (68) by a l_2 norm weighted by w_i (Ren & Yang 2019, Wang & Ding 2019). The problem turns to the form:

$$\hat{\beta} \in \arg\min_{\beta} \frac{1}{n} \sum_{i=1}^{n} w_i (y_i - \beta^T x_i)^2 + \lambda \|\beta\|_2^2 \tag{70}$$

with $e_i = y_i - \beta^T x_i$, the weight w_i in (70) takes one of two following values conditioned by the sign of error e_i
$$w_i = \begin{cases} \alpha / e_i & \text{if } e_i \geq 0 \\ (\alpha - 1)/e_i & \text{otherwise} \end{cases} \tag{71}$$

Considering the data set $\mathcal{D}_n = \{(\mathbf{X}, \mathbf{y})\}$, with the boldface notations for row concentrated matrices and $\mathbf{W}$ is a $n \times n$ diagonal matrix of weight. The problem (70) can be then solved by matrix inversion algorithm as below:

$$\hat{\beta} = (\mathbf{X}^T \mathbf{W} \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^T \mathbf{W} \mathbf{y} \tag{72}$$

with $\mathbf{I}$ is a $n \times n$ identity matrix.

IRLS iteratively solves $\hat{\beta}$ by (72) and then updates the new weight matrix by (71) at each step until convergence. In numerical applications, the factor e_i in (71) is replace by $\max(e_i, \psi)$, with ψ a small number (e.g. $1\text{E} - 4$ in our experiment), to avoid infinity of w_i.

9.6 Supplementary Figures

Table 2 – Best hyperparameters of each model for Pricing problem

	NN	ELM	RBF
hidden layer	1		
hidden unit	400, 100	600, 200	400, 50
activation function	Softplus	Elu	Multiquadratic
batch size	Full batch		
learning rate	0.5, 0.1	Matrix inversion	Matrix inversion
regularization weight	0.1, 0	0, 0.1	0.5, 0.1
shape parameter			0.1
projection distribution		Gausian, Achlioptas	

Notes: In each cell, the first and second element corresponds the choice of hyperparameter for the data of shape $20,000 \times 40$ and of shape $50,000 \times 10$, respectively. The one-element cell means the same values for both two data shapes.

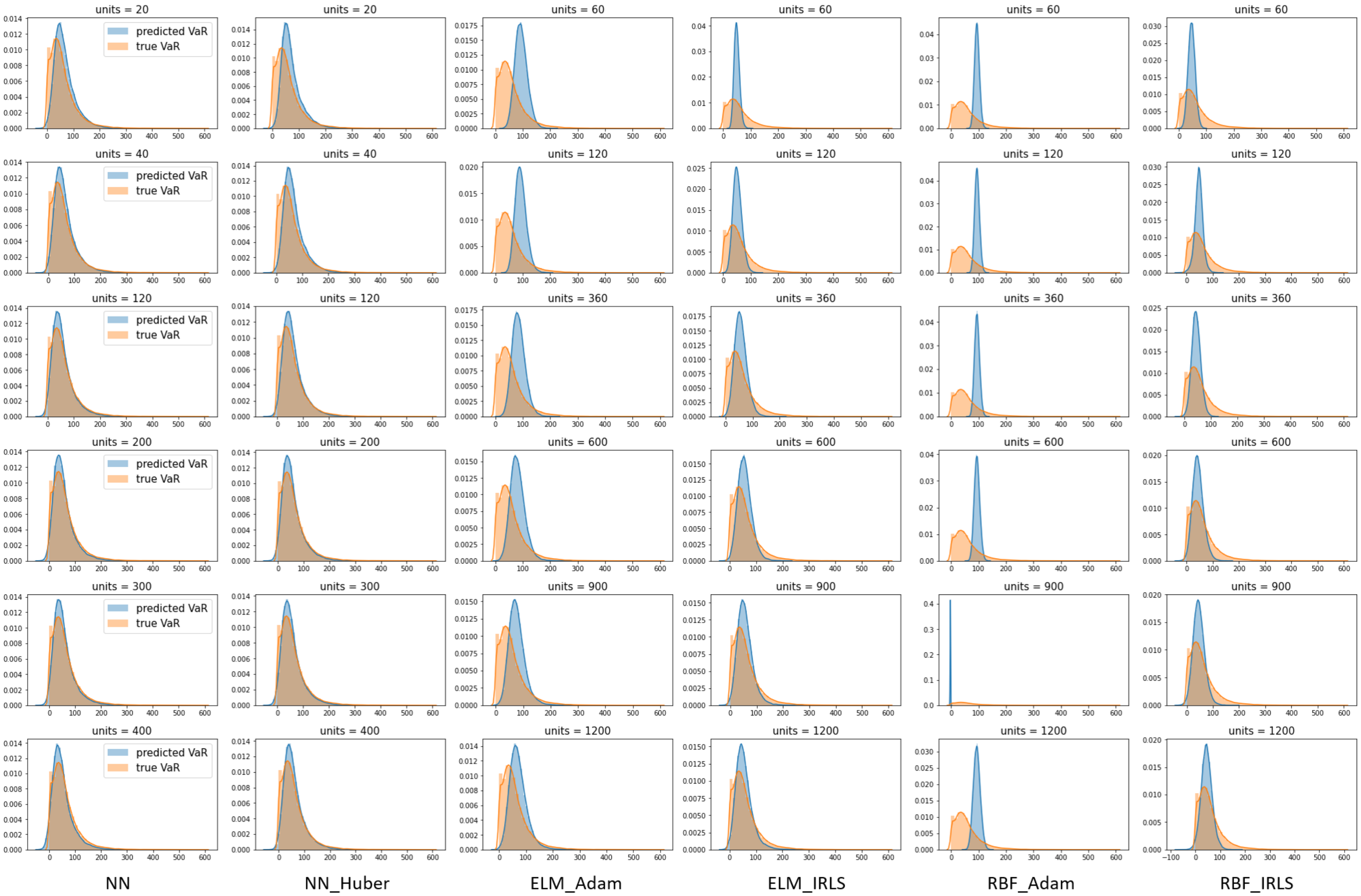

Figure 5 – Density plot of (out-of-sample) two-step approach VaR versus true VaR

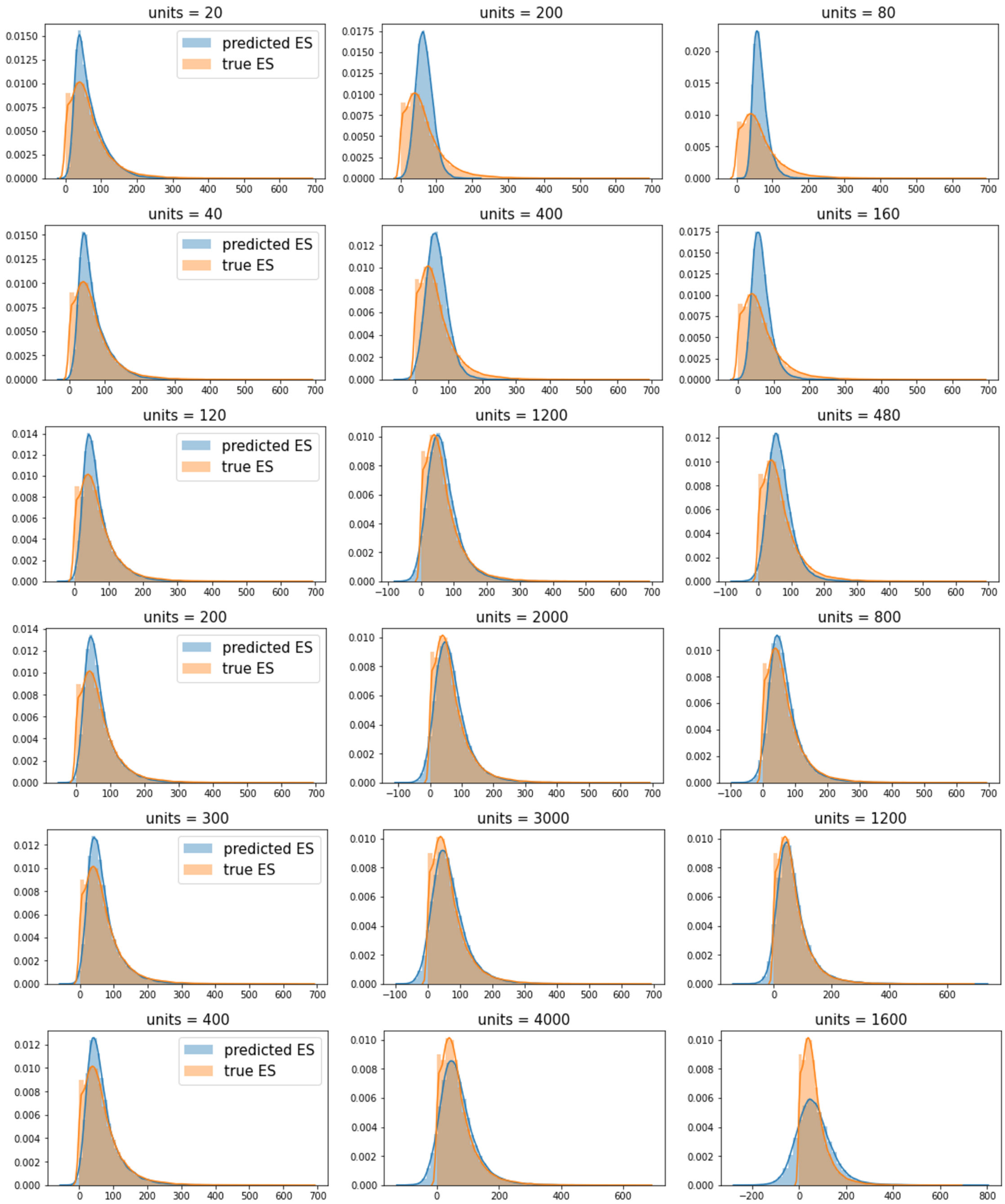

Figure 6 – Density plot of true ES versus (out-of-sample) two-step approach ES learned from Standard neural networks (*Left*), ELM networks (*Center*) and RBF networks (*Right*).

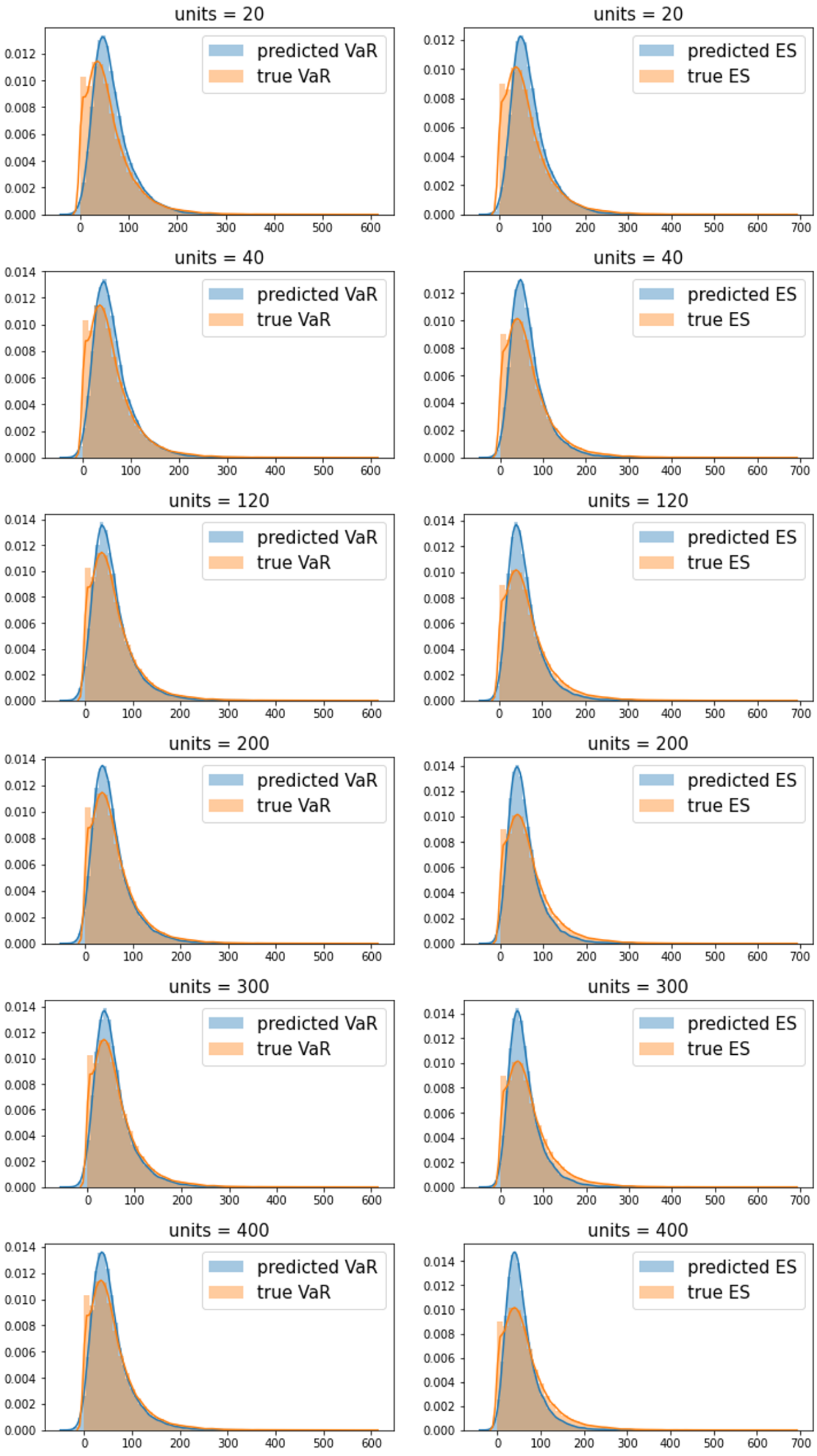

Figure 7 – Density plot of (out-of-sample) joint approach VaR (*left*) and ES (*right*) versus true metrics.

References

Achlioptas, D. (2003), 'Database-friendly random projections: Johnson-lindenstrauss with binary coins', *Journal of computer and System Sciences* **66**(4), 671–687.

Albanese, C., Crépey, S., Hoskinson, R. & Saadeddine, B. (2020), 'XVA analysis from the balance sheet', *Quantitative Finance* . Forthcoming (DOI 10.1080/14697688.2020.1817533).

Barrera, D., Crépey, S., Gobet, E. & Saadeddine, B. (2020), Non-parametric regression of conditional value-at-risk and expected shortfall. In preparation.

Bergstra, J. & Bengio, Y. (2012), 'Random search for hyper-parameter optimization', *The Journal of Machine Learning Research* **13**(1), 281–305.

Bhattacharya, P. K. & Gangopadhyay, A. K. (1990), 'Kernel and nearest-neighbor estimation of a conditional quantile', *The Annals of Statistics* pp. 1400–1415.

Broomhead, D. S. & Lowe, D. (1988), Radial basis functions, multi-variable functional interpolation and adaptive networks, Technical report, Royal Signals and Radar Establishment Malvern (United Kingdom).

Cannon, A. J. (2011), 'Quantile regression neural networks: Implementation in r and application to precipitation downscaling', *Computers & geosciences* **37**(9), 1277–1284.

Chakravarti, P. (2019), Convex optimization methods for quantile regression. In preparation.

Dimitriadis, T., Bayer, S. et al. (2019), 'A joint quantile and expected shortfall regression framework', *Electronic Journal of Statistics* **13**(1), 1823–1871.

Fissler, T. & Ziegel, J. F. (2016), 'Higher order elicitability and osband's principle', *The Annals of Statistics* **44**(4), 1680–1707.

Fissler, T., Ziegel, J. F. & Gneiting, T. (2015), 'Expected shortfall is jointly elicitable with value at risk-implications for backtesting', *arXiv preprint arXiv:1507.00244* .

Fujita, O. (1998), 'Statistical estimation of the number of hidden units for feedforward neural networks', *Neural networks* **11**(5), 851–859.

Gallicchio, C. & Scardapane, S. (2020), Deep randomized neural networks, *in* 'Recent Trends in Learning From Data', Springer, pp. 43–68.

Huang, G.-B., Zhu, Q.-Y. & Siew, C.-K. (2006), 'Extreme learning machine: theory and applications', *Neurocomputing* **70**(1-3), 489–501.

Igelnik, B. & Pao, Y.-H. (1995), 'Stochastic choice of basis functions in adaptive function approximation and the functional-link net', *IEEE transactions on Neural Networks* **6**(6), 1320–1329.

Kingma, D. P. & Ba, J. (2014), 'Adam: A method for stochastic optimization', *arXiv preprint arXiv:1412.6980* .

Koenker, R., Chernozhukov, V., He, X. & Peng, L. (2017), *Handbook of quantile regression*, CRC press.

Koenker, R. & Ng, P. (2005), ‘A frisch-newton algorithm for sparse quantile regression’, *Acta Mathematicae Applicatae Sinica* **21**(2), 225–236.

Koenker, R., Ng, P. & Portnoy, S. (1994), ‘Quantile smoothing splines’, *Biometrika* **81**(4), 673–680.

Leshno, M., Lin, V. Y., Pinkus, A. & Schocken, S. (1993), ‘Multilayer feedforward networks with a nonpolynomial activation function can approximate any function’, *Neural networks* **6**(6), 861–867.

Maillard, O.-A. & Munos, R. (2012), ‘Linear regression with random projections’, *The Journal of Machine Learning Research* **13**(1), 2735–2772.

Meinshausen, N. (2006), ‘Quantile regression forests’, *Journal of Machine Learning Research* **7**(Jun), 983–999.

Mohri, M., Rostamizadeh, A. & Talwalkar, A. (2018), *Foundations of machine learning*, MIT press.

Park, J. & Sandberg, I. W. (1991), ‘Universal approximation using radial-basis-function networks’, *Neural computation* **3**(2), 246–257.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L. & Lerer, A. (2017), ‘Automatic differentiation in pytorch’.

Pinkus, A. (1999), ‘Approximation theory of the mlp model in neural networks’, *Acta numerica* **8**(1), 143–195.

Ren, Z. & Yang, L. (2019), ‘Robust extreme learning machines with different loss functions’, *Neural Processing Letters* **49**(3), 1543–1565.

Rockafellar, R. T. & Uryasev, S. (2000), ‘Optimization of conditional value-at-risk’, *Journal of risk* **2**, 21–42.

Sontag, E. D. (1991), Feedback stabilization using two-hidden-layer nets, *in* ‘1991 American Control Conference’, IEEE, pp. 815–820.

Takeuchi, I., Le, Q. V., Sears, T. D. & Smola, A. J. (2006), ‘Nonparametric quantile estimation’, *Journal of machine learning research* **7**(Jul), 1231–1264.

Torossian, L., Picheny, V., Faivre, R. & Garivier, A. (2020), ‘A review on quantile regression for stochastic computer experiments’, *Reliability Engineering & System Safety* **201**, 106858.

Wang, D. & Li, M. (2017), ‘Stochastic configuration networks: Fundamentals and algorithms’, *IEEE transactions on cybernetics* **47**(10), 3466–3479.

Wang, K. & Ding, X. (2019), Pinball loss based extreme learning machines, *in* ‘IOP Conference Series: Materials Science and Engineering’, Vol. 569, IOP Publishing, p. 052061.

White, H. (1992), ‘Nonparametric estimation of conditional quantiles using neural networks’, pp. 190–199.

Wolf, M. M. (2018), ‘Mathematical foundations of supervised learning’.