Efficient Valuation of SCR via a Neural Network Approach

Seyed Amir Hejazi[a], Kenneth R. Jackson[a]

[a]*Department of Computer Science, University of Toronto, Toronto, ON, M5S 3G4, Canada*

Abstract

As part of the new regulatory framework of Solvency II, introduced by the European Union, insurance companies are required to monitor their solvency by computing a key risk metric called the Solvency Capital Requirement (SCR). The official description of the SCR is not rigorous and has lead researchers to develop their own mathematical frameworks for calculation of the SCR. These frameworks are complex and are difficult to implement. Recently (Bauer et al., 2012) has suggested a nested Monte Carlo (MC) simulation approach to calculate the SCR. But the proposed MC approach is computationally expensive even for a simple insurance product. In this paper, we propose a neural network approach to compute the SCR that significantly reduces the computational complexity in the calculation. We study the performance of our neural network approach in estimating the SCR for a large portfolio of an important type of insurance products called Variable Annuities (VAs). Our experiments show that the proposed neural network framework is both efficient and accurate.

Keywords: Variable annuity, Spatial interpolation, Neural network, Portfolio valuation, Solvency Capital Requirement (SCR)

1. Introduction

The Solvency II Directive is the new insurance regulatory framework within the European Union. Solvency II enhances consumer protection by

Email addresses: amir@cs.toronto.edu (Seyed Amir Hejazi), krj@cs.toronto.edu (Kenneth R. Jackson)

requiring insurers to monitor the risks facing their organization. An integral part of Solvency II is the Solvency Capital Requirement (SCR) that reduces the risk of insurers' insolvency. SCR is the amount of reserves that an insurance company must hold to cover any losses within a one year period with a confidence level of 99.5%.

The calculation standards are described in the documents of the Committee of European Insurnace and Occupational Pensions Supervisors (CEIOP) (e.g., (CEIOP, 2011)). The regulation allows insurance companies to use either the standard formula or to develop an internal model based on a market-consistent valuation of assets and liabilities. Because of the imprecise language of the aforementioned standards, many insurance companies are struggling to implement the underlying model and to develop efficient techniques to do the necessary calculations. In (Christiansen and Niemeyer, 2014; Bauer et al., 2012), rigorous mathematical definitions of SCR are provided. Moreover, (Bauer et al., 2012) describes an implementation of a simplified, but approximately equivalent, notion of SCR using nested Monte Carlo (MC) simulations.

The results of the numerical experiments in (Bauer et al., 2012) to find the SCR for a simple insurance product show that the proposed nested MC simulations are too expensive, even for the simplified notion of SCR. Hence, insurance companies cannot directly use the proposed MC approach to find the SCR for their large portfolios of insurance products. In this paper, we propose a neural network approach to ameliorate the computational complexity of MC simulations which allows us to efficiently compute the SCR for large portfolios of insurance products. We provide insights into the efficiency of the proposed framework by studying its performance in computing the SCR for a large portfolio of Variable Annuities (VAs), a well-known and important type of insurance product.

A VA is tax-deferred retirement vehicle that allows a policyholder to invest in financial markets by making payment(s) into a predefined set of subaccounts set up by an insurance company. The investment of the policyholder should be payed back as a lump-sum payment or a series of contractually agreed upon payments over a period of time in the future. VA products provide embedded guarantees that protect the investment of a policyholder in a bear market and/or from mortality risk (TGA, 2013). For a detailed description of VA products and the different types of guarantees offered in these products, see our earlier paper (Hejazi et al., 2015) and the references therein.

Because of the innovative structure of embedded guarantees in VA products, insurance companies have been successful in selling large volumes of these products (IRI, 2011). As a result, VA products are a large portion of the investment market around the globe and big insurance companies have accumulated large portfolios of these products. The embedded guarantees of VA products expose insurers to a substantial amount of market risk, mortality risk, and behavioral risk. Hence, big insurance companies have developed risk management programs to hedge their exposures, especially after the market crash of 2008.

The rest of this paper is organized as follows. In Section 2, we describe the mathematical definition of SCR as well its simplified, almost equivalent, version described in (Bauer et al., 2012). In Section 3, we describe the nested simulation approach of (Bauer et al., 2012) that we use to approximate the SCR. Furthermore, we define a simple asset and liability structure that allows us to remove the assets from the required calculation of the SCR for the portfolio. In Section 4, we describe the neural network framework that we use to estimate the one-year probability distribution of liability for the input portfolio of VA products. In Section 5, we compare the efficiency and accuracy of our method to that of a simple nested MC approach. In Section 6, we conclude the paper.

2. Solvency Capital Requirement

A rigorous treatment of SCR[1] requires the definition of Available Capital (AC) which is a metric that determines the solvency of a life insurer at each point in time. The AC is the difference between the Market Value of Assets (MVA) and Market Value of Liabilities (MVL):

$$\mathrm{AC}_t = \mathrm{MVA}_t - \mathrm{MVL}_t \tag{1}$$

where the subscript t denotes the time, in years, at which each variable is calculated.

Assuming the definition (1) of AC, the SCR, under Solvency II, is defined as the smallest amount of AC that a company must currently hold to insure a positive AC in one year with a probability of 99.5%. In other words, the SCR is the smallest amount x that satisfies the following inequality.

[1]The material in this section is based largely on the discussion in (Bauer et al., 2012).

$$P(\mathrm{AC}_1 \geq 0 | \mathrm{AC}_0 = x) \geq 99.5\% \tag{2}$$

In practice, it is hard to find the SCR using definition (2). Hence, Bauer et al. use a simpler, approximately equivalent notion of the SCR which is based on the one-year loss function evaluated at time zero

$$\Delta = \mathrm{AC}_0 - \frac{\mathrm{AC}_1}{1+r} \tag{3}$$

where r is the one-year risk-free rate. The SCR is then defined as the one-year Value-at-Risk (VaR)

$$\mathrm{SCR} = \mathrm{argmin}_x\{P(\Delta > x) \leq 0.5\%\} \tag{4}$$

This is the definition of the SCR that we use in the rest of this paper.

3. Nested Simulation Approach

A key element in both definitions (2) and (4) of the SCR is the calculation of AC. From (1) we see that the calculation of AC requires a market consistent valuation of assets and liabilities. Insurance companies can follow a mark-to-market approach to value their assets in a straightforward way. However, the innovative and complex structure of insurance products does not allow for such straightforward calculation of liabilities. In practice, insurance companies often have to calculate the liabilities of insurance products by direct valuation of the cash flows associated with them (direct method (Girard, 2002)). Hence, the difficulty in calculation of SCR is primarily associated with the difficulty in calculation of liabilities.

In order to focus on the problem of calculating the liabilities and to make the analysis more tractable, we assume that the company has taken a passive approach (i.e., no hedging is involved) and the only asset of the company is a pool of shareholders' money M_0 that is invested in a money market account and hence accrues risk-free interest. The assets are adjusted yearly as required. This structure of assets allows us to eliminate the assets in the definition of Δ in (3) as follows.

$$\begin{aligned}\Delta &= \mathrm{AC}_0 - \frac{\mathrm{AC}_1}{1+r} \\ &= (M_0 - \mathrm{MVL}_0) - (\frac{(M_0(1+r) - \mathrm{MVL}_1)}{1+r}) \\ &= -\mathrm{MVL}_0 + \frac{\mathrm{MVL}_1}{1+r} \end{aligned} \tag{5}$$

Hence, the problem of calculating the SCR reduces to the problem of calculating the current liability and the distribution of the liability in one-year's time. The value of the liability at time zero can be determined according to the risk-neutral valuation formula as the expected sum of discounted liability cash flows under some risk-neutral measure. We can estimate the probability distribution of MVL in one year by the nested simulation approach of (Bauer et al., 2012). In the nested simulation approach, we first draw N independent identically distributed sample paths $O^{(i)}, 1 \leq i \leq N$, for the development of financial market. For each sample path $O^{(i)}$, we use the aforementioned risk-neutral valuation technique to find the liability $\mathrm{MVL}_1^{(i)}$ assuming the state of the financial market. The values $\mathrm{MVL}_1^{(i)}, 1 \leq i \leq N$, can be used to determine the empirical distribution of liability in one-year's time. In order to compute the 99.5%-quantile for AC_1 (or negative of liability as defined in (5)), as required by the definition of the SCR, we can sort the $-1 \times \mathrm{MVL}_1^{(i)}, 1 \leq i \leq N$, values and choose the $\lfloor N \times 0.995 + 0.5 \rfloor$ element amongst the sorted values. Call the negative of this value $\hat{\mathrm{MVL}}_1$. Using the estimated value of liability at time zero, $\hat{\mathrm{MVL}}_0$, and the value of $\hat{\mathrm{MVL}}_1$, we can then estimate the SCR as

$$\hat{\mathrm{SCR}} = -\hat{\mathrm{MVL}}_0 + \frac{\hat{\mathrm{MVL}}_1}{1+r} \tag{6}$$

This approach to estimating the liability, requires the calculation of the liability at various points in time and under different market conditions. In (Bauer et al., 2012), a MC simulation approach is suggested to calculate the liability at different points in the space. Moreover, through mathematical analysis, an optimal number of projections for the number of MC projection paths and the value of N is suggested. The proposed MC scheme, even for one insurance contract, requires a significant amount of computation and hence does not scale well to large portfolios of insurance products. As we discuss

in detail in (Hejazi et al., 2015), traditional portfolio valuation techniques such as replication portfolio (Dembo and Rosen, 1999; Oechslin et al., 2007; Daul and Vidal, 2009) and Least Squares Monte Carlo (LSMC) (Cathcart and Morrison, 2009; Longstaff and Schwartz, 2001; Carriere, 1996) are not effective in reducing the computational cost. The computational complexity of these methods for sophisticated insurance products, such as VAs, is comparable to, or more than, the computational complexity of MC schemes. Reducing the amount of computation in these methods often requires significant reduction in the accuracy of these methods.

Recently, a spatial interpolation scheme (Hejazi et al., 2015; Gan, 2013; Gan and Lin, 2015) has been proposed to reduce the required computation of the MC scheme by reducing the number of contracts that must be processed by the MC method. The spatial interpolation framework, first, generates a sample of contracts in the space in which the insurance products of the input portfolio are defined. The value of interest, i.e., liability in this paper, for the sample contracts are evaluated using MC simulations. The outputs of the MC scheme are then used to estimate the value of interest for other contracts in the input portfolio by a spatial interpolation scheme. In (Hejazi and Jackson, 2015), we describe how a neural network approach to the spatial interpolation can not only solve the problem associated with finding a good distance metric for the portfolio but also provide a better balance between efficiency, accuracy, and granularity of estimation. Although the numerical experiments of (Hejazi and Jackson, 2015) provided insights into the performance of our proposed neural network approach in estimation of Greeks for a portfolio of VAs, we show in this paper how the same type of network can be used to find the liabilities and subsequently the SCR for an input portfolio of VA products in an efficient and accurate manner.

4. Neural Network Framework

In this section, we provide a brief review of our proposed neural network framework. For a detailed treatment of this approach, in particular the reason behind our choice of network and training method, see our paper (Hejazi and Jackson, 2015).

4.1. The Neural Network

Our proposed estimation scheme is an extended version of the Nadaraya-Watson kernel regression model (Nadaraya, 1964; Watson, 1964). Assuming

$y(z_1), \cdots, y(z_n)$ are the observed values at known locations $z_1, \cdots, z_n$, our model estimates the value at an unknown location z as

$$\hat{y}(z) = \sum_{i=1}^{n} \frac{G_{h_i}(z - z_i) \times y(z_i)}{\sum_{j=1}^{n} G_{h_j}(z - z_j)} \tag{7}$$

where G is a nonlinear differentiable function and the subscript, h_i, denotes the range of influence of each $y(z_i)$ on the estimated value. The variable h_i is a location dependent vector that determines the range of influence of each pointwise estimator in each direction of feature space of the input data. In our application of interest in this paper, $y(\cdot)$ is the MC estimation of liability. The variables $z_i, 1 \leq i \leq n$, are n vectors in $\mathbb{R}^m$ representing the attributes of a sample set of n representative VA contracts in the space of the input portfolio.

We choose to implement our model (7) using a feed-forward neural network (Bishop, 2006) that allows us to fine-tune our model to find the optimum choices of the h_i values that minimize our estimation error.

As shown in Figure 1, our feed-forward neural network is a collection of interconnected processing units, called neurons, which are organized in three layers. The first and the last layers are, respectively, called the input layer and the output layer. The intermediate layer is called the hidden layers.

The neurons in the first layer provide the network with the feature vector (input values). Each neuron in the input layer represents a value in the set $\{F^c, F^-, F^+\}$. Each f in F^c has the form

$$f = \begin{cases} 0 & \text{if } x_c = x_{c_i} \\ 1 & \text{if } x_c \neq x_{c_i} \end{cases} \tag{8}$$

where x_c represents the category of categorical attribute c for input VA policy z, and x_{c_i} represents the category of categorical attribute c for representative VA policy z_i in the sample. Each f in F^- has the form $f = [t(\mathbf{x}_{n_i}) - t(\mathbf{x}_n)]^+ / R_{t_h}$, and each f in F^+ has the form $f = [t(\mathbf{x}_n) - t(\mathbf{x}_{n_i})]^+ / R_{t_h}$. In both of these formulas, $\mathbf{x}_n$ is the vector containing the numeric attributes of input VA policy z, $\mathbf{x}_{n_i}$ is the vector containing the numeric attributes of representative VA policy z_i in the sample, $t(\cdot)$ is a transformation (linear/nonlinear), determined by the expert user, that assumes a value in an interval of length R_t and $[\cdot]^+ = \max(\cdot, 0)$. In essence, our choice of input values allows different bandwidths (h_i values in (7)) to be used for different

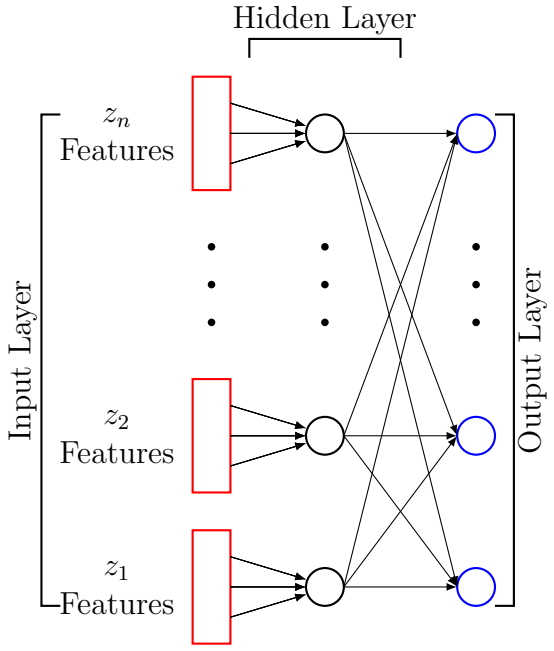


Figure 1: Diagram of the proposed neural network. Each circle represents a neuron. Each rectangle represent the set of neurons that contains input features corresponding to a representative contract.

attributes of VA policies and in different directions around a representative VA contract in the sample.

Since we are interested in calibrating the G functions of equation (7), the number of neurons in the output and hidden layer are equal to the number of representative contracts in the sample. The inputs of neuron i in the hidden layer are those values of f in the input layer that are related to the representative VA policy i. In other words, the input values of neuron i in the hidden layer determine the per attribute difference of the input VA contract z with the representative VA contract z_i using the features $f \in \{F^c, F^-, F^+\}$. Assuming $x_1, \cdots, x_n$ are the inputs of neuron j at the hidden level, first a linear combination of input variables is constructed

$$a_j = \sum_{i=1}^{n} w_{ij} x_i + b_j \tag{9}$$

where parameters w_{ij} are referred to as weights and parameter b_j is called the bias. The quantity a_j is known as the activation of neuron j. The activation

a_j is then transformed using an exponential function to form the output of neuron j.

The output of neuron i in the output layer is the normalized version of the output for neuron i in the hidden layer. Hence the outputs of the network, i.e., $o_i, i \in \{1, \cdots, n\}$, represent a softmax of activations in the hidden layer. These outputs can be used to estimate the value of the liability for input VA z as $\hat{y}(z) = \sum_{i=1}^{n} o_i \times y(z_i)$, in which $y(z_i)$ is the value of the liability for representative VA policy z_i. In summary, our proposed neural network allows us to rewrite equation (7) as

$$\hat{y}(z) = \sum_{i=1}^{n} \frac{\exp(\mathbf{w_i}^T \mathbf{f_i}(z) + b_i) \times y(z_i)}{\sum_{j=1}^{n} \exp(\mathbf{w_j}^T \mathbf{f_j}(z) + b_j)} \tag{10}$$

where vector $\mathbf{f_i}$ represents the features in the input layer that are related to the representative VA policy z_i, and vector $\mathbf{w_i}$ contains the weights associated to each feature in $\mathbf{f_i}$ at neuron i of the hidden layer.

4.2. Network Training Methodology

In order to calibrate (train) the network and find the optimal values of weights and bias parameters, we select a small set of VA policies, which we call the training portfolio, as the training data for the network. The objective of the calibration process is to find a set of weights and bias parameters that minimizes the Mean Squared Error (MSE) in estimation of liability values of the training portfolio. In other words, our objective function is

$$E(\mathbf{w}, \mathbf{b}) = \frac{1}{2n} \sum_{k=1}^{n} ||\hat{y}(z_k, \mathbf{w}, \mathbf{b}) - y(z_k)||^2 \tag{11}$$

We use an iterative gradient descent scheme (Boyd and Vandenberghe, 2004) to train the network. However, to speed up the training process, we do mini-batch training (Murphy, 2012) with Nestrov's Accelerated Gradient (NAG) method (Nesterov, 1983). In mini-batch training, in each iteration, we select a small number of training VA policies at random and compute the gradient of the following error function for this batch.

$$E(\mathbf{w^{(t)}}, \mathbf{b^{(t)}}) = \frac{1}{2|B^{(t)}|} \sum_{k \in B^{(t)}} ||\hat{y}(z_k, \mathbf{w^{(t)}}, \mathbf{b^{(t)}}) - y(z_k)||^2 \tag{12}$$

where $B^{(t)}$ is the set of indices for the selected VA policies and superscript t denotes the iteration number. Instead of updating the weights and biases

by the gradient of (12), in the NAG method, we use a velocity vector that increases in value in the direction of persistent reduction in the objective error function across iterations. We use a particular implementation of the NAG method described in (Sutskever et al., 2013). In this implementation of NAG, weights and biases are updated according to the rules

$$\begin{aligned} v_{t+1} &= \mu_t v_t - \epsilon \nabla E([\mathbf{w}^{(t)}, \mathbf{b}^{(t)}] + \mu_t v_t) \\ [\mathbf{w}^{(t+1)}, \mathbf{b}^{(t+1)}] &= [\mathbf{w}^{(t)}, \mathbf{b}^{(t)}] + v_{t+1} \end{aligned} \tag{13}$$

where v_t is the velocity vector, $\mu_t \in [0, 1]$ is known as the momentum coefficient and ϵ is the learning rate. The momentum coefficient is an adaptive parameter defined by

$$\mu_t = \min(1 - 2^{-1-\log_2(\lfloor \frac{t}{50} \rfloor + 1)}, \mu_{\max}) \tag{14}$$

where $\mu_{\max} \in [0, 1]$ is a user defined constant.

Because of the amount of investments and the structure of guarantees in VA products, the liability values can become large. Big liability values can result in big gradient values which produce big jumps in the updates of (13). Therefore, to avoid numerical instability, only in the training stage, we normalize the values of $y(z_i)$ in (12), by dividing each $y(\cdot)$ value by the range of guarantee values in the input portfolio.

If we allow the network to train for a long enough time, it will start to converge towards a local optimum. Depending on our choice of the representative contracts and the training data, further training of the network after a certain number of iterations might result in overfitting or might not result in significant change in the value of weights and biases, and the associated error. To avoid these pitfalls, we use a set of randomly selected VA policies from the input portfolio as our validation portfolio (Murphy, 2012) and stop the training using a two step verification process. First, we observe if the MSE of the training data drops dramatically or if there is an initial decrease in the MSE of the validation portfolio to a local minimum followed by an increase in the MSE of the validation portfolio. Once any of these events, called stopping events, happens, we train the network for a few more iterations until the mean of network's liability estimates for the validation portfolio is within a δ relative distance of the mean of MC estimated liability of the validation portfolio via MC simulations or a maximum number of training iterations is

reached. The relative distance between network estimated liablity L_{NN} and the MC estimated liability L_{MC} is calculated as

$$\text{dist} = |\frac{L_{NN} - L_{MC}}{L_{MC}}| \tag{15}$$

As shown in the graph of Figure 2, the actual graph of the MSE for the validation portfolio or the training portfolio as a function of iteration number might be volatile. However, a general trend exists in the data. To make the trend clearer, we use a simple moving average with a window of $\bar{W}$ to smooth the data and polynomial fitting of the smoothed data. We detect stopping events using a window of length W on the polynomial approximation of the MSE values. A stopping event occurs if the MSE of the validation set has increased in the past $W - 1$ recorded values after attaining a minimum. We evaluate the MSE values of the validation set every I^{th} iteration of the training, to avoid slowing down the training process. I, W and $\bar{W}$ are user defined parameters and are application dependent.

5. Numerical Experiments

In order to provide insights into the performance of the proposed neural network framework, we estimate the SCR for a synthetic portfolio of $100,000$ VA contracts. Each contract in the portfolio is assigned attribute values uniformly at random from the space defined in Table 1. The guarantee values (death benefit and withdrawal benefit) of GMWB riders are chosen to be equal[2], but they are different than the account value. The account values of the contracts follow a simple log-normal distribution model (Hull, 2006) with a risk free rate of return of $\mu = 3\%$, and volatility of $\sigma = 20\%$.

We use the framework of (Gan and Lin, 2015) to value each VA contract. Similar to (Hejazi et al., 2015), we use $10,000$ MC simulations to value each contract. In our experiments, we use the mortality rates of the 1996 I AM mortality tables provided by the Society of Actuaries.

We implement our experiments in Java and run them on a machine with a dual quad-core Intel X5355 CPUs. For each valuation of the input portfolio using the MC simulations, we divide the input portfolio into 10 sub-portfolios, each with an equal number of contracts and run each sub-portfolio on one

[2]This is typical of the beginning of the withdrawal phase.

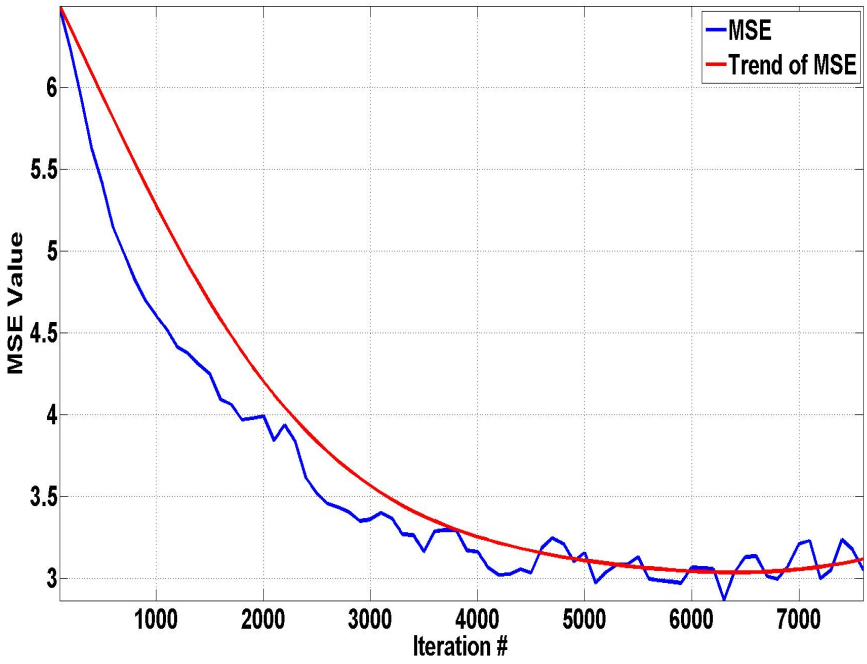


Figure 2: MSE of the validation set and the trend in the MSE as a function of the iteration number for a run of the training algorithm. The trend is found using a moving average with a window size of 10 and then polynomial fitting with a polynomial of degree 6.

Attribute	**Value**
Guarantee Type	{GMDB, GMDB + GMWB}
Gender	{Male, Female}
Age	$\{20, 21, \ldots, 60\}$
Account Value	$[1e4, 5e5]$
Guarantee Value	$[0.5e4, 6e5]$
Widthrawal Rate	$\{0.04, 0.05, 0.06, 0.07, 0.08\}$
Maturity	$\{10, 11, \ldots, 25\}$

Table 1: GMDB and GMWB attributes and their respective ranges of values.

thread, i.e., a total of 10 threads, to value these 10 sub-portfolios in parallel. Although we use the parallel processing capability of our machine for MC simulations, we do not use parallel processing to implement our code for our

	Experiment 1
Guarantee Type	{GMDB, GMDB + GMWB}
Gender	{Male, Female}
Age	$\{20, 30, 40, 50, 60\}$
Account Value	$\{1e4, 1e5, 2e5, 3e5, 4e5, 5e5\}$
Guarantee Value	$\{0.5e4, 1e5, 2e5, 3e5, 4e5, 5e5, 6e5\}$
Withdrawal Rate	$\{0.04, 0.08\}$
Maturity	$\{10, 15, 20, 25\}$

Table 2: Attribute values from which representative contracts are generated for experiments.

proposed neural network scheme: our neural network code is implemented to run sequentially on one core.

5.1. Network Setup

Although a sagacious sampling scheme can significantly improve the performance of the network, for the sake of simplicity, we use a simple uniform sampling method similar to that used in (Hejazi and Jackson, 2015). We postpone the discussion on the choice of a better sampling method to our future work. We construct a portfolio of all combinations of attribute values defined in Table 2. In each experiment, we randomly select 300 VA contracts from the aforementioned portfolio as the set of representative contracts.

As discussed in Section 4, in addition to the set of representative contracts, we need to introduce two more portfolios, the training portfolio and the validation portfolio, to train our neural network. For each experiment, we randomly select 250 VA contracts from the input portfolio as our validation portfolio. The training portfolio, in each experiment, consists of 200 contracts that are selected uniformly at random from the set of VA contracts of all combinations of attributes that are presented in Table 3. In order to avoid unnecessary overfitting of the data, the attributes of Table 3 are chosen to be different than the corresponding values in Table 2.

We train the network using a learning rate of 20, a batch size of 20 and we set $\mu_{\max}$ to 0.99. Moreover, we fix the seed of the pseudo-random number generator that we use to select mini batches to be zero. For a given set of the representative contracts, the training portfolio, and the validation portfolio, fixing the seed allows us to reproduce the trained network. We set the initial values of the weight and bias parameters to zero.

	Experiment 1
Guarantee Type	{GMDB, GMDB + GMWB}
Gender	{Male, Female}
Age	$\{23, 27, 33, 37, 43, 47, 53, 57\}$
Account Value	$\{0.2e5, 1.5e5, 2.5e5, 3.5e5, 4.5e5\}$
Guarantee Value	$\{0.5e5, 1.5e5, 2.5e5, 3.5e5, 4.5e5, 5.5e5\}$
Withdrawal Rate	$\{0.05, 0.06, 0.07\}$
Maturity	$\{12, 13, 17, 18, 22, 23\}$

Table 3: Attribute values from which training contracts are generated for experiments.

We estimate the liability of the training portfolio and the validation portfolio every 50 iterations and record the corresponding MSE values. We smooth the recorded MSE values using a moving average with a window size of 10. Moreover, we fit a polynomial of degree 6 to the smoothed MSE values and use a window size of length 4 to find the trend in the MSE graphs. In the final stage of the training, we use a δ of 0.005 as our threshold for maximum relative distance in estimation of the liabilities for the validation portfolio.

We use the rider type and the gender of the policyholder as the categorical features in F^c. The numeric features in F^+ are defined as follows.

$$f(z, z_i) = \frac{[t(x) - t(x_i)]^+}{R_t}$$
$$t \in \{\text{maturity, age, AV, GD, GW, withdrawal rate}\} \tag{16}$$

where AV is the account value, GD is the guaranteed death benefit, GW is the guaranteed withdrawal benefit, R_t is the range of values that t can assume, x and x_i are vectors denoting the numeric attributes of the input VA contract z and the representative contract z_i, respectively. We define the features of F^- in a similar fashion by swapping x and x_i on the right side of equation (16).

5.2. Performance

The experiments of this section are designed to allow us to compare the efficiency and the accuracy of the proposed neural network framework with the MC simulations. In each experiment, we define $N = 40,000$ realization

of the market by describing the relative change of account values. Assuming a price of A_0 as the current account value of a VA, each realization gives us a coefficient C_1, from the above-mentioned log-normal distribution, that allows us to determine the account value in one year's time as $A_1 = C_1 * A_0$. We determine a range (interval) based on the maximum value and the minimum value of realized coefficients and divide that range into 99 sub-intervals, using 100 end points. We value, using MC simulations or the proposed neural network framework, the one year projection of liability for the input portfolio assuming the realization of the market by each of these end points. Using the estimated values, we use piecewise-linears to approximate the value of each of the original $40,000$ realizations. We chose piecewise-linear approximation because of the smoothness, as we demonstrate shortly, of the curve describing the relation between liability values and the aforementioned coefficients. We use the liability values for these $40,000$ realizations to define the 99.5%-quantile of AC_1 as described in Section 3. We then use the estimated 99.5%-quantile and the estimation of current liability to determine the SCR of the input portfolio as in (6).

Given a selection of representative contracts, the training portfolio, and the validation portfolio, we train the network using the current liability of VAs to estimate the liability of the network at time 0 (current liability). We use the trained network to estimate the one year liability of the input portfolio for each end point. However, before each estimation, we perform the last stage of the training method to fine-tune the network. In other words, we train the network for a maximum of 200 iterations until the network estimated liability for the validation portfolio is within $\delta = 0.01$ relative distance of the MC estimated liability of the validation portfolio. Notice that in (10), we use the estimated one year liability of the representative contracts for the given realization of the market as the $y(z_i), 1 \leq i \leq n$, values. If the fine-tuning of the network was unable to estimate the liability of the validation portfolio within the defined δ relative distance, we define a new network using the set of representative contracts, the training portfolio and the validation portfolio and train the new network. We then use the new trained network in the subsequent liability estimation. With each realization of the market, the local minimum of the network for the validation portfolio may change. If the local minimum in the new realization is close to the previous local minimum, the fine-tuning stage allows us to reach that local minimum without going through our computationally expensive training stage that searches for the local minimum in the whole space. If the fine-tuning stage fails, then we

Value Of Interest	Relative Error (%)					
	S1	S2	S3	S4	S5	S6
SCR	−0.85	0.69	−0.81	−3.58	3.02	1.52
MVL_0	0.22	0.52	0.96	−0.36	−0.27	0.70
$\text{MVL}_1^{(99.5)}$	0.43	0.11	0.91	0.97	−1.20	−0.05

Table 4: Relative error in the estimation of the current liability value, one year liability value, and the SCR for the input portfolio.

can conclude that the local minimum has changed significantly and hence a re-training in the whole space is required.

We compare the performance of the interpolation schemes using 6 different realization, $S_i, 1 \leq i \leq 6$, of the representative contracts, the training portfolio, and the validation portfolio. Table 4 denotes the accuracy of our proposed scheme in estimating the MVL_0, the 99.5%-quantile of MVL_1, and the SCR value for each scenario. Accuracy is recorded as the relative error

$$\text{Err} = \frac{X - X_{MC}}{|X_{MC}|} \tag{17}$$

where X_{MC} is the value of interest (liability or the SCR) in the input portfolio computed by MC simulations and X is the estimation of the value of interest computed by the proposed neural network method.

The results of Table 4 provide strong evidence that our neural network method is accurate. Except in a few scenarios, S_4 and S_5, the proposed approach is very accurate in its estimation of the MVL_0, the $\text{MVL}_1^{(99.5)}$, and the SCR. Our numerical experiments in (Hejazi and Jackson, 2015) show that our proposed neural network framework has low sensitivity to the particular realization of the representative contracts, and the training/validation portfolio once the size of these portfolios are fixed. The results of Table 4 further corroborate our finding in (Hejazi and Jackson, 2015) as the realization of the representative contracts, and the training/validation portfolio is different in each scenario.

The accuracy of the proposed method can be further examined by considering Figure 3 in which the estimated liability values, for the end points of the interval, by the proposed method is compared with their respective MC estimations. The graphs of Figures 3b and 3c show that the liability estimated values by the proposed method are very close to those of the MC method, which demonstrates the projection capabilities of the proposed

Method	Running Time	
	Mean	STD
MC	49334	0
NN	8370	2465

Table 5: simulation time of each method to estimate the SCR. All times are in seconds.

framework. The smoothness of the MC liability curve allows us to use piecewise linear interpolation to estimate the value of the liability for points inside the sub-intervals. However, the estimation of this curve using the proposed framework does not result in a smooth curve. Therefore, we might be able to increase the accuracy of the proposed framework by using a non-linear curve fitting technique. In this work, we tried to avoid using different curve fitting techniques so that we can have a fair pointwise comparison between the proposed technique and the MC technique. Furthermore, the curve fitting stage is independent of the liability estimation, and our focus, in this paper, is on the performance of the proposed framework for estimating the liability.

Table 5 presents the statistics on running time of the proposed method (denoted as NN in this table and elsewhere in the paper) and the MC method. The results suggest a speed-up of $4 - 8$ times, depending on the scenario, and an average speed-up of 6 times. Considering that the implementation of the neural network was sequential and we compared the running time of the neural network with the implementation of MC simulations that uses parallel processing on 4 cores, we observe that even a simple implementation of the network can be highly efficient. Notice that we used a moderate number of MC simulation scenarios compared to the suggested values in (Bauer et al., 2012). An increase in the number of MC scenarios will not increase the running time of our neural network significantly because we only need MC simulations for representative contracts and for the validation/training portfolio; however, it increases the running time of the MC simulations significantly. In general, a similar argument shows that any more complex implementation of the per policy valuation process increases the efficiency of our proposed framework relative to MC simulation, as the number of policies that must go through the valuation process is limited.

6. Concluding Remarks

The new regulatory framework of Solvency II has been introduced by European Union to reduce the risk facing insurance/re-insurance companies. An important part of the new regulation is the calculation of the SCR. Because of the imprecise language used to describe the standards, many insurance companies struggle to understand and implement the framework.

In recent years, mathematical frameworks for calculation of the SCR have been proposed to address the former issue (Christiansen and Niemeyer, 2014; Bauer et al., 2012). Furthermore, (Bauer et al., 2012) has suggested a nested MC simulation approach to calculate the SCR to address the later issue. The suggested MC approach is computationally expensive, even for one simple insurance contract. Hence, it cannot be easily generalized to a large portfolio of insurance products.

In this paper, we propose a spatial interpolation approach to the computation that uses a neural network engine to interpolate the liability values of insurance products based on the known liability values of a small representative set of these product. We study the performance of the proposed approach in finding the SCR value for a portfolio of VA products. The results of our numerical experiments in Section 5 corroborate the superior accuracy and efficiency of the sequential implementation of our proposed neural network approach compared with an implementation of the MC approach that uses parallel processing.

Although our method requires us to train our neural network using three small ($< 1\%$ of the input portfolio) portfolios that are selected uniformly at random, but given the size of each of the small portfolios, the performance of the method has low sensitivity to the particular realization of these portfolios.

Despite the superior performance of the proposed approach that uses a simple uniform sampling method to select the small portfolios required to train the network, we believe our neural network approach can be further improved by incorporating a more sophisticated sampling method that takes into account the distribution of the input portfolio. We intend to address this issue in our future research.

In this paper, we chose to study the performance of our neural network approach on estimation of liabilities; however, the application of our proposed approach is much more general than that. In particular, one can change the type of the insurance product and the approach used to value individual insurance products and incorporate them with our framework to estimate

the value of a large portfolio of the aforementioned insurance product.

7. Acknowledgements

This research was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

Bauer, D., Reuss, A., Singer, D., 2012. On the Calculation of the Solvency Capital Requirement Based on Nested Simulations. ASTIN Bulletin 42, 453–499.

Bishop, C.M., 2006. Pattern Recognition and Machine Learning (Information Science and Statistics). Springer-Verlag, NJ, USA.

Boyd, S., Vandenberghe, L., 2004. Convex Optimization. Cambridge University Press, NY, USA.

Carriere, J., 1996. Valuation of the Early-Exercise Price for Options Using Simulations and Nonparametric Regression. Insurance: Mathematics and Economics 19, 19–30.

Cathcart, M., Morrison, S., 2009. Variable Annuity Economic Capital: the Least-Squares Monte Carlo Approach. Life & Pensions , 36–40.

CEIOP, 2011. EIOPA Report on the Fifth Quantitative Impact Study (QIS5) for Solvency II URL: https://eiopa.europa.eu/Publications/Reports/QIS5_Report_Final.pdf.

Christiansen, M.C., Niemeyer, A., 2014. Fundamental Definition of the Solvency Capital Requirement in Solvency II. ASTIN Bulletin 44, 501–533.

Daul, S., Vidal, E., 2009. Replication of Insurance Liabilities. RiskMetrics Journal 9, 79–96.

Dembo, R., Rosen, D., 1999. The Practice of Portfolio Replication: A Practical Overview of Forward and Inverse Problems. Annals of Operations Research 85, 267–284.

Gan, G., 2013. Application of Data Clustering and Machine Learning in Variable Annuity Valuation. Insurance: Mathematics and Economics 53, 795–801.

Gan, G., Lin, X.S., 2015. Valuation of Large Variable Annuity Portfolios Under Nested Simulation: A Functional Data Approach. Insurance: Mathematics and Economics 62, 138–150.

Girard, L., 2002. An Approach to Fair Valuation of Insurance Liabilities Using the Firm's Cost of Capital. North American Actuarial Journal 6, 18–41.

Hejazi, S.A., Jackson, K.R., 2015. A Neural Network Approch to Efficient Valuation of Large Portfolios of Variable Annuities. URL: http://www.cs.toronto.edu/pub/reports/na/IME-Paper2.pdf.

Hejazi, S.A., Jackson, K.R., Gan, G., 2015. A Spatial Interpolation Framework for Efficient Valuation of Large Portfolios of Variable Annuities. URL: http://www.cs.toronto.edu/pub/reports/na/IME-Paper1.pdf.

Hull, J.C., 2006. Options, Futures, and Other Derivatives. 6th ed., Pearson Prentice Hall, Upper Saddle River, NJ.

IRI, 2011. The 2011 IRI Fact Book. Insured Retirement Institute .

Longstaff, F., Schwartz, E., 2001. Valuing American Options by Simulation: A Simple Least-Squares Approach. The Review of Financial Studies 14, 113–147.

Murphy, K.P., 2012. Machine Learning: A Probabilistic Perspective. The MIT Press.

Nadaraya, E.A., 1964. On Estimating Regression. Theory of Probability and its Applications 9, 141–142.

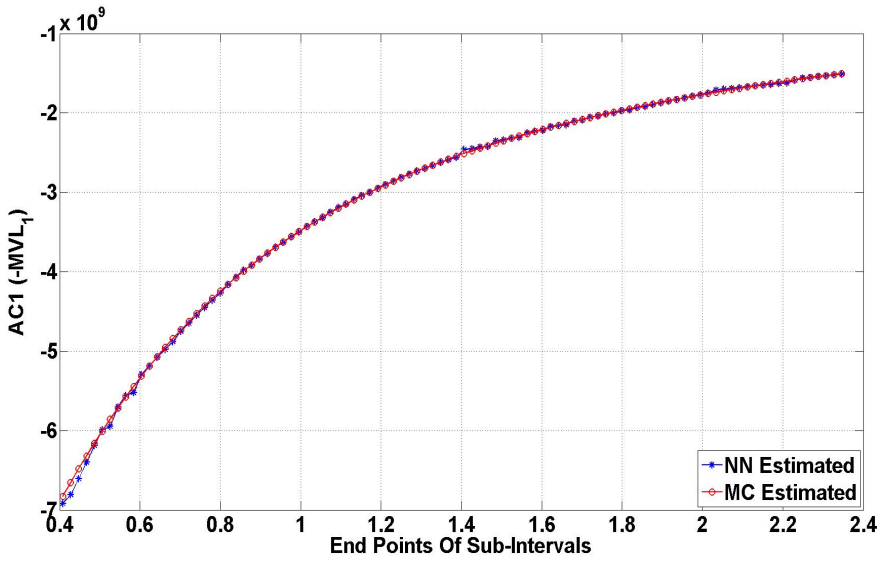
Nesterov, Y., 1983. A Method of Solving a Convex Programming Problem with Convergence Rate O(1/sqrt(k)). Soviet Mathematics Doklady 27, 372–376.

Oechslin, J., Aubry, O., Aellig, M., Kappeli, A., Bronnimann, D., Tandonnet, A., Valois, G., 2007. Replicating Embedded Options in Life Insurance Policies. Life & Pensions , 47–52.

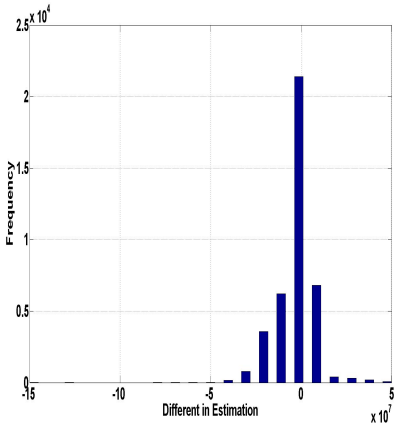
Sutskever, I., Martens, J., Dahl, G., Hinton, G., 2013. On the Importance of Initialization and Momentum in Deep Learning, in: Proceedings of the 30th International Conference on Machine Learning (ICML-13), JMLR Workshop and Conference Proceedings. pp. 1139–1147.

TGA, 2013. Variable Annuities—An Analysis of Financial Stability. The Geneva Association URL: https://www.genevaassociation.org/media/ga2013/99582-variable-annuities.pdf.

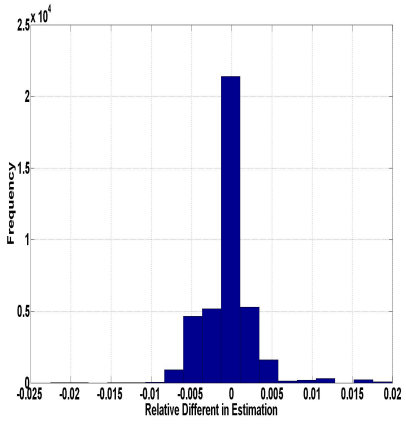
Watson, G.S., 1964. Smooth Regression Analysis. Sankhyā: Indian Journal of Statistics 26, 359–372.



(a) Estimated one year liability values by the neural network framework and the MC method.



(b) Historgram of difference at each end point of sub-intervals in estimation of the liability via the neural network approach and the MC simulations.



(c) Historgram of the relative difference (17) at each end point of sub-intervals in estimation of liability via the neural network approach and the MC simulations.

Figure 3: Comparing estimation of one year liability values of the input portfolio by the proposed neural network framework and the MC method.