Optimal Insurance Strategies: A Hybrid Deep Learning Markov Chain Approximation Approach

Xiang Cheng, * Zhuo Jin, † Hailiang Yang,‡

April 23, 2020

Abstract

This paper studies deep learning approaches to find optimal reinsurance and dividend strategies for insurance companies. Due to the randomness of the financial ruin time to terminate the control processes, a Markov chain approximation-based iterative deep learning algorithm is developed to study this type of infinite-horizon optimal control problems. The optimal controls are approximated as deep neural networks in both cases of regular and singular types of dividend strategies. The framework of Markov chain approximation plays a key role in building the iterative equations and initialization of the algorithm. We implement our method to classic dividend and reinsurance problems and compare the learning results with existing analytical solutions. The feasibility of our method for complicated problems has been demonstrated by applying to an optimal dividend, reinsurance and investment problem under a high dimensional diffusive model with jumps and regime-switching.

Key Words. Neural Networks, Deep Learning, Markov Chain Approximation, Reinsurance Strategies

*Centre for Actuarial Studies, Department of Economics, The University of Melbourne, VIC 3010, Australia, cheng.x@unimelb.edu.au.

†Centre for Actuarial Studies, Department of Economics, The University of Melbourne, VIC 3010, Australia, zjin@unimelb.edu.au.

‡Department of Statistics and Actuarial Science, The University of Hong Kong, Hong Kong, hlyang@hku.hk.

1 Introduction

Since the pioneering work by Borch (1960), Borch (1962) and Arrow (1963), there has been extensive research on optimal reinsurance. Reinsurance is a standard tool to reduce and eliminate risks born by primary insurance carriers. The primary insurance carrier pays the reinsurance company a certain part of premiums in return of protections from reinsurance companies against the adverse claim volatilities. The recent book on reinsurance Albrecher et al. (2017) provides an impressive list of references on the subject.

Dividend payment strategies are crucial to companies. It represents an important financial signal about a firm's future growth opportunities and may influence the wealth of the shareholders. Optimal dividend strategies for insurance companies was first studied by De Finetti (1957), who solved the problem in a discrete-time model by assuming that the surplus process follows a simple random walk and that decision-makers aim to maximize the expected discounted total dividends until financial ruin. Gerber (1972) provided solutions for optimal dividend problem under both discrete and continuous models. Asmussen and Taksar (1997) re-examined the problem using the theory of controlled diffusion processes. Types of dividend controls such as regular, singular or impulse controls together with reinsurance policies are investigated under various scenarios thereafter. See Høgaard and Taksar (1999), Wei et al. (2010) and references therein. In past decades, a huge amount of research has been devoted to finding optimal insurance strategies using analytical techniques under various mathematical models. Due to increasing complexity of stochastic systems such as considering regular, singular and impulse controls simultaneously, and multiple decision-makers in a stochastic game framework, etc, closed-form solutions are virtually impossible in many cases. Recently, there is emerging research on solving the optimization problems numerically using finite difference or similar type methods. See Jin et al. (2013), Van Staden et al. (2018) and Jin et al. (2018).

On the other hand, the fast developments of computer science, artificial intelligence (AI), big data analytics, and machine learning are changing our society and life in almost all aspects. More and more research has been conducted on what the impact of data science on the insurance industry is, how we can use the high tech and data science to the insurance industry, such as improve risk management by reducing losses, claim reserve estimation, policy design and increasing profit. See Wüthrich (2018a), Wüthrich (2018b) and Aleandri (2018). A comprehensive set of machine learning techniques in non-life insurance pricing and data science such as regression trees, neural networks and unsupervised learning are presented in Wüthrich and Buser (2019).

When managing an insurance portfolio, the decision-making process forms a stochastic control problem, which generally is categorized into two types: finite-time horizon and infinite-time horizon. There exists some literature on applying deep learning methods to solve finite-time horizon problems. Han and E (2016) and E et al. (2017) expressed the control family by parametric neural networks, and approximate the expectation of multiple time-step objective by its Monte Carlo mean. Thus, searching for the optimal control strategy is transferred to a simplified expectation maximization where a complicated function is maximized over neural network parameters. Bachouch et al. (2019) and Huré et al. (2019) integrated deep learning methods into Monte Carlo backward optimization algorithms. They

used parametric neural networks to approximate control strategies as well, but the optimization is conducted backwards in every time step. Recently, there are emerging application of deep learning methods to various stochastic models in finance and risk management, see Carmona and Laurière (2019); Fecamp et al. (2019); Pereira et al. (2019). Overall, these methods are dealing with optimization in finite time horizon.

For infinite-time horizon problems, since there is no fixed terminal time, we can hardly reformulate optimization problems as the maximization of a simple expectation of projections. There exist very few literature on applying deep learning methods to optimize infinite-time horizon stochastic controls. In this paper, the hybrid feature of the proposed algorithm lies in the integration of deep learning method and Markov chain approximation method to solve optimal dividend and reinsurance problem. Particularly, we approximate the controls with neural networks, apply the Markov chain approximation method to identify a neighborhood of optimal controls, fit the initial values of neural network's parameters, and learn the optimal parameters of neutral networks with gradient descent algorithm. We benchmark the existing theoretical results to justify the correctness of our method. Particularly, we work on an optimal dividend and reinsurance problem that was studied in Høgaard and Taksar (1999). Besides, we apply the proposed method for a problem with more factors being controlled in a much more complicated jump-diffusion with the regime-switching stochastic environment. The numerical results demonstrate the effectiveness of our method in solving complex stochastic control problems.

Comparing with the existing numerical methods on infinite-time stochastic control problems, our proposed algorithm has two main advantages. First, as we all know, it is inevitable to face the problem of "curse of dimensionality" when dealing with optimization problems with multiple control variables and in high dimensional stochastic environment. The introduction of machine learning framework enables us to replace the optimization over the piecewise control grid for every state value by the simultaneous maximization among parametric neural networks for all state values. Now the computational complexity mainly comes from the evaluation of gradients for every state value, and thus increases linearly with respect to the number of points in the state lattice. Hence the computation efficiency can be improved. Second, the accuracy of numerical results can be improved by the proposed method as well. Traditional methods generally use piecewise constant controls to approximate the optimal control. Then the accuracy of the control strategy is subject to the denseness of the control grid. On the contrary, neural networks allow the control strategy to take values in the continuous range and will conquer the difficulty of effectively selecting the scales of control grid to meet the requirement of accuracy.

The rest of the paper is organized as follows. A general formulation of surplus, dividend and reinsurance strategies and assumptions are presented in Section 2. Section 3 deals with the case of restricted dividend payment rate. The deep learning Markov chain approximation method is presented. The neural networks are constructed accordingly. The case of unrestricted dividend payment rate is investigated in Section 4. A complicated optimal dividend, reinsurance, and investment problem under multi-dimensional regime-switching jump-diffusion model is presented in Section 5. Numerical examples are provided in Section 6 to illustrate the performance of the algorithms in all cases. Some concluding remarks are provided in Section 7.

2 Formulation

Let $(\Omega, \mathcal{F}, \mathbb{F}, \mathbb{P})$ be a complete filtered probability space where $\mathbb{F} := \{\mathcal{F}_t\}$ is a right-continuous, $\mathbb{P}$-complete filtration. We consider an insurance company that collects the premiums and purchases reinsurance policies to share the risks with reinsurance companies. Similarly to Høgaard and Taksar (1999), we denote the claim process as $Z(t)$ and formulate $Z(t)$ by the diffusion-approximation of the classical Cramér-Lundberg model:

$$dZ(t) = adt - \sigma dW(t), \tag{2.1}$$

where constants $a > 0$ and $\sigma > 0$ represent the mean and the volatility of the claim process respectively, and $W(t)$ is a standard $\mathbb{P}$-Brownian motion. The premium is paid continuously at a constant rate $c = (1+\eta)a$, where $\eta > 0$ is the safety loading. Then the dynamics of the insurer's surplus process is given by

$$\begin{aligned} d\widetilde{R}(t) &= cdt - dZ(t) \\ &= a\eta dt + \sigma dW(t). \end{aligned}$$

We assume that the insurer can purchase proportional reinsurance to manage insurance business risks. Denote by $\phi(t) \in [0,1]$ the reinsurance strategy of the insurer at time t. It means the reinsurance company will compensate the insurer for $1-\phi(t)$ of claims at time t, therefore the net liability for the insurance company will be $\phi(t)$ of original claims. Suppose that the reinsurance premium is also determined by the expected value premium principle. Under the proportional reinsurance contract $\phi(t)$, the reinsurance premium rate is $\kappa = (1+\rho)(1-\phi(t))a$ with safety loading $\rho \geq \eta$. When $\rho = \eta$, we call it cheap reinsurance. Therefore, the surplus process with such a proportional reinsurance treaty is governed by

$$\begin{aligned} dR(t) &= cdt - \phi(t)dZ(t) - \kappa dt \\ &= [(\eta - \rho) + \rho\phi(t)]adt + \sigma\phi(t)dW(t), \end{aligned} \tag{2.2}$$

where $(\eta - \rho) \leq 0$.

A dividend strategy $D(\cdot)$ is an $\mathcal{F}_t$-adapted process $\{D(t) : t \geq 0\}$ corresponding to the accumulated amount of dividends paid up to time t such that $D(t)$ is a nonnegative and nondecreasing stochastic process that is right continuous and has left limits with $D(0^-) = 0$. Two types of dividend controls will be considered: regular and singular control. The regular dividend control problem corresponds to the cases where the dividend payments are paid continuously. In the restricted dividend payment rate case, we consider dividend strategies in the form of $dD(t) = u(t)dt$, where the control variable $u(\cdot)$ represent a restricted dividend rate. The singular dividend control corresponds to the situation where the dividend process is not continuous, and the surplus level changes drastically on a dividend payday. Hence the dividend payment rate is unrestricted in this case. We will consider the dividend policies in restricted and unrestricted cases, respectively. Denote $X(t)$ as the surplus process in the presence of dividend payments. In both cases, $X(t)$ can be written as

$$dX(t) = dR(t) - dD(t), \tag{2.3}$$

where $X(0) = x$.

By choosing the optimal reinsurance and dividend payment strategies, we aim to maximize the present value of cumulative discounted dividend payments until financial ruin. Let γ be the constant discount factor. We assume that $\gamma > 0$. For an arbitrary pair of controls $\pi(\cdot) = (\phi(\cdot), D(\cdot))$, the objective function is defined as

$$J(x, \pi) = \mathbb{E}_x \left(\int_0^\tau e^{-\gamma t} dD(t) \right), \tag{2.4}$$

where $\tau = \inf\{t \geq 0 : X(t) < 0\}$ represents the time of ruin, and $\mathbb{E}_x$ denotes the expectation conditioned on $X(0) = x$. Hence, the value function $V(x) = \sup_\pi J(x, \pi)$.

3 Restricted Dividend Payment Rate

In this section, we consider how to optimize reinsurance and dividend strategy under the case where there is a bound M on the dividend rate. Since the optimal dividends policy is either a barrier or a band strategy, $D(t)$ is an absolutely continuous process. We write $D(t)$ as

$$dD(t) = u(t)dt, \quad 0 \leq u(t) \leq M, \tag{3.1}$$

where $u(t)$ is an $\mathcal{F}_t$-adapted process and $0 < M < \infty$. Then the surplus process $X(t)$ in the presence of dividend payments is given by

$$dX(t) = dR(t) - u(t)dt, \; X(0) = x \geq 0 \tag{3.2}$$

for all $t < \tau$ and we impose $X(t) = 0$ for all $t > \tau$. Suppose the dividend is paid at a rate $u(t)$, where $u(t)$ is an $\mathcal{F}_t$-adapted process, and the optimal payout strategy is applied subsequently. Then the expected discounted dividend until ruin is given by

$$J(x, \pi(\cdot)) = \mathbb{E}_x \left[\int_0^\tau e^{-\gamma t} u(t) dt \right], \tag{3.3}$$

where $\mathbb{E}_x$ denotes the expectation conditioned on $X(0) = x$. The value function of maximizing expected dividend payoff is defined by the following optimization problem:

$$V(x) = \sup_{\phi \in [0,1], u \in [0,M]} J(x, \pi(\cdot)). \tag{3.4}$$

3.1 Markov Chain Approximation Method (MCAM)

In this subsection, a brief introduction of Markov chain approximation method (MCAM) is presented. We will formulate the transition probabilities of MCAM to construct an iterative computational scheme. A comprehensive introduction of MCAM can be referred to Kushner and Dupuis (2001). In Section 3.2, deep learning neural networks will be introduced to approximate control functions and will be used along with the recursive equation constructed here to locate the optimal control strategy.

In what follows, we first define some notations:

$$\begin{aligned}
\pi &\triangleq (\phi, u)\,; \\
\Pi &\triangleq [0,1] \times [0, M]\,; \\
\alpha(x,\pi) &\triangleq \frac{1}{2}\sigma^2\phi^2; \\
\beta(x,\pi) &\triangleq (\eta - \rho + \rho\phi)a - u; \\
\delta(x,\pi) &\triangleq u; \\
\omega(x,\pi) &\triangleq |\beta(x,\pi)|\,\Delta x + 2\alpha(x,\pi)\,; \\
\Delta t(x,\pi) &\triangleq \frac{\Delta x^2}{\omega(x,\pi) + \gamma\Delta x^2};
\end{aligned}$$

where Δx is the step size of states. Then for a given control strategy π, the change of objective value can be approximated by the following Markov chain:

$$\begin{aligned}
S(x,V,\pi) \approx & e^{-\gamma\Delta t(x,\pi)}\left[p(x, x+\Delta x|\pi)\,V(x+\Delta x) + p(x, x-\Delta x|\pi)\,V(x-\Delta x)\right] \\
& + \delta(x,\pi)\,\Delta t,
\end{aligned} \tag{3.5}$$

where transition probabilities are defined as follows:

$$\begin{aligned}
p(x, x+\Delta x|\pi) &\triangleq \frac{\alpha(x,\pi) + \max\{\beta(x,\pi), 0\}\,\Delta x}{|\beta(x,\pi)|\,\Delta x + 2\alpha(x,\pi)}, \\
p(x, x-\Delta x|\pi) &\triangleq \frac{\alpha(x,\pi) + \max\{-\beta(x,\pi), 0\}\,\Delta x}{|\beta(x,\pi)|\,\Delta x + 2\alpha(x,\pi)}.
\end{aligned} \tag{3.6}$$

The optimal iterative control strategy and value function are given as

$$V(x) = \sup_{\pi\in\Pi} S(x,V,\pi),$$

$$\pi^* = \arg\max_{\pi\in\Pi} S(x,V,\pi).$$

3.2 Deep Learning Markov Chain Approximation Method

In this part, we will present a new method as an integration of Markov chain approximation method and deep learning method, which can preserve the stable convergence of MCAM, the strong generalizability of neural network, and the high computation efficiency of machine learning at the same time. Considering the fact that the neural network owns outstanding capability of fitting non-linear functions, we adopt neural networks to model the relationship between control strategy $\pi = (\phi, u)$ and state value x as in Figure 1. Without loss of generality, we assume the neural network only contains two hidden layers of three nodes.

Remark 3.1. Generally speaking, neural networks with more layers are equipped with stronger ability to learn more complicated control strategies. However, if the neural network is far more complicated than it is required by the optimization problem, excessive parameters

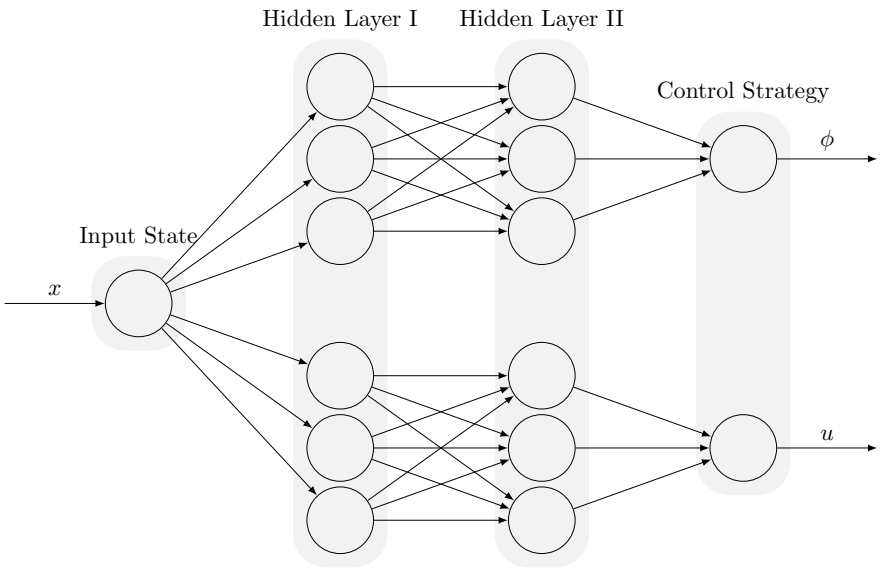

Figure 1: Control Network

will more probably lead to issues of gradient vanishing or local minimum. We find that the two-layer neural networks perform well in both the simple and complex numerical examples in Section 6. The rule we choose the layers of neural networks is as follows: For complex problems which our method is targeting at, the architecture of neural networks can be chosen by referring to the results of similar simple problems.

The computational structure of every neural node follows the pattern in Figure 2. Besides, given the different admissible ranges of reinsurance and dividend strategies, we use independent neural networks for the dividend strategy u and the insurance strategy ϕ.

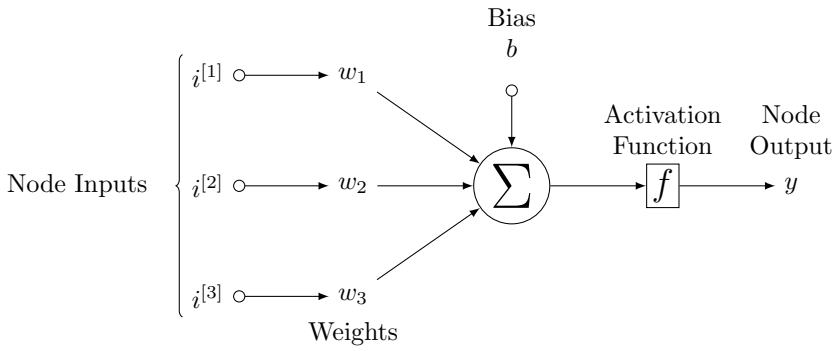

Figure 2: Neural Node Computation

The output range of neural networks should match the admissible range of the corresponding control, where the output of ϕ should be in $[0, 1]$, and u should be in $[0, M]$. This is achieved by assigning a specific output function to the neural network. Moreover, the architecture of the neural network, e.g. the number of nodes in every layer, the number of

hidden layers, and the intermediate activation functions, should be appropriately determined according to the property and complexity of the problem. Excessively complicated neural networks may overestimate the sophistication of the relationship between the state value and the optimal control values, which will potentially cause the disappearance of gradients, and then result in the failure of the learning algorithm. However, a relatively simple structure will suffer from the insufficient ability of generalization, which makes the family of parametric neural network control strategies not big enough to effectively approximate the admissible control family.

Define Θ as the collection of all weights and bias terms in two neural networks, then denote the neural network control strategy by:

$$\hat{\pi}(x) \triangleq \mathrm{N}(x|\Theta) = (\hat{\phi}, \hat{u}).$$

Given a state lattice, $\{x_i\}_{i=1}^n$, define the global improvement function G as:

$$G \triangleq G(V(x_1), V(x_2), \cdots, V(x_n)).$$

The global improvement function G reflects, how much the iterative value function will improve globally through the iterative Markov chain. The choice G should serve the goal that the value function will be improved on most states rather than on every state of the state lattice. This global improvement is achieved by iteratively adopting the neural network control strategy $\hat{\pi}$ which is optimized in every iteration. This is equivalent to searching for Θ maximizing G with a given iterative value function. Usually, G can be chosen as the average value of the value function:

$$G(V(x_1), V(x_2), \cdots, V(x_n)) = \frac{1}{n}\sum_{i=1}^{n} V(x_i).$$

Remark 3.2. We can choose other general global improvement functions such as the weighted average of $V(x_n)$. The impact to efficiency depends on the formulation of the infinite horizon optimization problem. For the present formulation, it does not show much difference, so we choose the average value for simplicity.

Assuming we are currently in the k-th iteration with the iterative value function V^{k-1} obtained from the previous iteration, the best parameters in the current iteration Θ^k is determined by:

$$\Theta^k = \arg\max_{\Theta} G\left(S^k(x_1), S^k(x_2), \cdots, S^k(x_n)\right), \tag{3.7}$$

$$S^k(x) = S(x, V^{k-1}, \mathrm{N}(x|\Theta)).$$

We will show how to search for Θ^k by gradient descent algorithm in section 3.3. With Θ^k, the iterative control strategy is expressed as:

$$\hat{\pi}^k(x) = \mathrm{N}\left(x|\Theta^k\right),$$

which can be adopted in (3.5) to obtain the k-th iterated value function:

$$V^k(x_i) = S(x_i, V^{k-1}, \hat{\pi}^k(x_i)),\ 1 \le i \le n.$$

The above iteration will repeat until the termination condition is met:

$$\sum_{i=1}^{n} \left(V^k(x_i) - V^{k-1}(x_i)\right)^2 < \epsilon_I,$$

where ϵ_I is a predefined small positive number.

Overall, the iteration constructed in this section can be summarized as a computational flow in Figure 3.

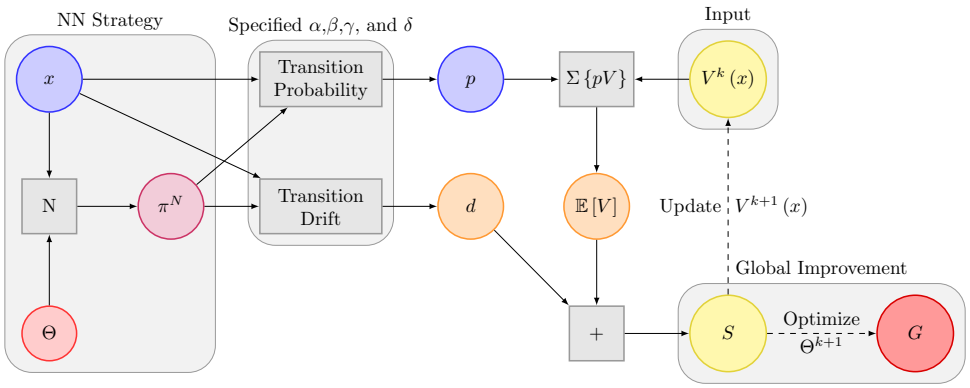

Figure 3: Iterative Learning Cycle

3.3 Determining The Iterative Neural Network Control Strategy

In every iteration step, the fundamental job is to obtain the iterative control strategy, where Θ^{k+1} is located through maximizing G by the gradient descent algorithm. The implementation of this algorithm will be briefly described with respect to the following optimization:

$$\Theta^k = \arg\max_{\Theta} G\left(S^k(x_1), S^k(x_2), \cdots, S^k(x_n)\middle|\Theta\right). \tag{3.8}$$

Given an initial value Θ_0^k, we construct the following iterative sequence:

$$\Theta_{l+1}^k = \Theta_l^k + h\left[\frac{\partial G(\cdot,\Theta)}{\partial\Theta}\middle|_{\Theta=\Theta_l^k}\right], \tag{3.9}$$

where h is called learning rate. It can be proved that:

$$\lim_{l\to\infty} \Theta_l^k = \Theta^k. \tag{3.10}$$

There exists rich literature on improvements of the standard gradient descent algorithm, where an outstanding one is called Adam algorithm, which will adjust the learning rate adaptively along with the learning process (see Kingma and Ba (2014) for details). Further, the gradient descent algorithm will terminate when

$$\left|G\left(\cdot,\Theta_l^k\right) - G\left(\cdot,\Theta_{l+1}^k\right)\right| < \epsilon_D, \tag{3.11}$$

where ϵ_D is a pre-defined small value for the acceptable precision.

The accuracy of gradient computation plays an essential role in the effective implementation of the gradient descent algorithm. Here we adopt automatic differentiation method, which follows the idea that most functions are just sequence of simple operations, and every simple operation can be easily differentiated. Thus, the gradient can be evaluated recursively using gradients of this sequence of simple operations. Precisely, the computational process of $\frac{\partial G}{\partial \Theta}$ in the k-th iteration follows

$$\begin{aligned}
&\frac{\partial G\left(S^k\left(x_1\right), S^k\left(x_2\right), \cdots, S^k\left(x_n\right)|\Theta\right)}{\partial \Theta} \\
&= \sum_{i=1}^{n}\left\{\frac{\partial G\left(S^k\left(x_1\right), S^k\left(x_2\right), \cdots, S^k\left(x_n\right)|\Theta\right)}{\partial S^k\left(x_i\right)} \frac{\partial S^k\left(x_i\right)}{\partial \hat{\pi}\left(x_i\right)} \frac{\partial \hat{\pi}\left(x_i\right)}{\partial \Theta}\right\}.
\end{aligned} \tag{3.12}$$

Here $\frac{\partial G}{\partial S^k(x_i)}$ relies on the form of global improvement function G, and $\frac{\partial S^k(x_i)}{\partial \hat{\pi}(x_i)}$ is derived from Markov chain computational rule (3.5) as

$$\begin{aligned}
&\frac{\partial S^k\left(x_i\right)}{\partial \hat{\pi}\left(x_i\right)} \\
&= \frac{\partial S(x_i, V^{k-1}, \hat{\pi}\left(x_i\right))}{\partial \hat{\pi}\left(x_i\right)} \\
&= -\gamma e^{-\gamma \Delta t\left(x_i, \hat{\pi}\left(x_i\right)\right)}\left[p\left(x_i, x_{i+1}|\hat{\pi}\left(x_i\right)\right) V^{k-1}\left(x_{i+1}\right)+p\left(x_i, x_{i-1}|\hat{\pi}\left(x_i\right)\right) V^{k-1}\left(x_{i-1}\right)\right] \frac{\partial \Delta t\left(x_i, \hat{\pi}\left(x_i\right)\right)}{\partial \hat{\pi}\left(x_i\right)} \\
&\quad+e^{-\gamma \Delta t\left(x_i, \hat{\pi}\left(x_i\right)\right)}\left[\frac{\partial p\left(x_i, x_{i+1}|\hat{\pi}\left(x_i\right)\right)}{\partial \hat{\pi}\left(x_i\right)} V^{k-1}\left(x_{i+1}\right)+\frac{\partial p\left(x_i, x_{i-1}|\hat{\pi}\left(x_i\right)\right)}{\partial \hat{\pi}\left(x_i\right)} V^{k-1}\left(x_{i-1}\right)\right] \\
&\quad+\frac{\partial \delta\left(x_i, \hat{\pi}\left(x_i\right)\right)}{\partial \hat{\pi}\left(x_i\right)} \Delta t\left(x_i, \hat{\pi}\left(x_i\right)\right)+\delta\left(x_i, \hat{\pi}\left(x_i\right)\right) \frac{\Delta t\left(x_i, \hat{\pi}\left(x_i\right)\right)}{\partial \hat{\pi}\left(x_i\right)},
\end{aligned}$$

where

$$\frac{\partial \Delta t\left(x_i, \hat{\pi}\left(x_i\right)\right)}{\partial \hat{\pi}\left(x_i\right)}=\frac{\partial \Delta t\left(x_i, \hat{\pi}\left(x_i\right)\right)}{\partial \alpha\left(x_i, \hat{\pi}\left(x_i\right)\right)} \frac{\partial \alpha\left(x_i, \hat{\pi}\left(x_i\right)\right)}{\partial \hat{\pi}\left(x_i\right)}+\frac{\partial \Delta t\left(x_i, \hat{\pi}\left(x_i\right)\right)}{\partial \beta\left(x_i, \hat{\pi}\left(x_i\right)\right)} \frac{\partial \beta\left(x_i, \hat{\pi}\left(x_i\right)\right)}{\partial \hat{\pi}\left(x_i\right)},$$

$$\frac{\partial p\left(x_i, x_{i \pm 1}, \hat{\pi}\left(x_i\right)\right)}{\partial \hat{\pi}\left(x_i\right)}=\frac{\partial p\left(x_i, x_{i \pm 1}, \hat{\pi}\left(x_i\right)\right)}{\partial \alpha\left(x_i, \hat{\pi}\left(x_i\right)\right)} \frac{\partial \alpha\left(x_i, \hat{\pi}\left(x_i\right)\right)}{\partial \hat{\pi}\left(x_i\right)}+\frac{\partial p\left(x_i, x_{i \pm 1}, \hat{\pi}\left(x_i\right)\right)}{\partial \beta\left(x_i, \hat{\pi}\left(x_i\right)\right)} \frac{\partial \beta\left(x_i, \hat{\pi}\left(x_i\right)\right)}{\partial \hat{\pi}\left(x_i\right)}.$$

Note that $\frac{\partial p(x_i, x_{i\pm1}, \hat{\pi}(x_i))}{\partial \alpha(x_i, \hat{\pi}(x_i))}$, $\frac{\partial p(x_i, x_{i\pm1}, \hat{\pi}(x_i))}{\partial \beta(x_i, \hat{\pi}(x_i))}$, $\frac{\partial \Delta t(x_i, \hat{\pi}(x_i))}{\partial \alpha(x_i, \hat{\pi}(x_i))}$, and $\frac{\partial \Delta t(x_i, \hat{\pi}(x_i))}{\partial \beta(x_i, \hat{\pi}(x_i))}$ can be easily derived from functions in (3.6). And $\frac{\partial \alpha(x_i, \hat{\pi}(x_i))}{\partial \hat{\pi}(x_i)}$, $\frac{\partial \beta(x_i, \hat{\pi}(x_i))}{\partial \hat{\pi}(x_i)}$, and $\frac{\partial \delta(x_i, \hat{\pi}(x_i))}{\partial \hat{\pi}(x_i)}$ are characterized by the stochastic optimization problem itself, thus are straight-forward computations. The last required element making (3.12) complete is $\frac{\partial \hat{\pi}(x_i)}{\partial \Theta}$, which follows the backward propagation differentiation method for neural networks.

In addition to the accurate computation of gradients, an appropriate choice of initial Θ is of great importance to successfully run the gradient descent algorithm as well. Due to the complexity of stochastic optimization problem, the learning objective is a sophisticated nested function of parameters Θ, which will lead to the issue that the learning objective has

some local maximal points. Although we can randomly generate several initial combinations of parameters, it cannot guarantee that we will reach the global maximum. Finding a way to determine the starting values of parameters is always a key step. Due to the curse of dimensionality, MCAM will mostly give inaccurate result, which however will preserve a rough shape of the control strategy. Based on this idea, we propose a method to locate the initial value of Θ using the information collected from running MCAM on a small state lattice. Specifically, in every iteration, we run standard MCAM on a small state lattice and a control grid to obtain the piecewise control values. These values will keep the rough shape of control strategy, and then we can improve the neural network control strategy from this general shape to avoid searching for the best Θ in a very big space.

Denote the small state lattice by $\{y_j\}_{j=1}^m$ with $m < n$. Focusing iteration step k, suppose we have obtained the piecewise control strategy on the small lattice denoted as $\left\{\tilde{\pi}^k(y_j)\right\}_{j=1}^m$, then the initial value of Θ_0^k is obtained from fitting neural network control strategy $\mathrm{N}(x|\Theta)$ against $\tilde{\pi}^k(y)$:

$$\Theta_0^k = \arg\min_{\Theta} \sum_{j=1}^{m} \left[\tilde{\pi}^k(y_j) - \mathrm{N}(y_j|\Theta)\right]^2.$$

Note Θ_0^k is obtained by gradient descent algorithm as well. Now the initial neural network control strategy $\hat{\pi}(\cdot|\Theta_0^k)$ will preserve the almost same information as the resulted control from ordinary MCAM. This is where we will start from to further improve the strategy through maximizing objective G.

Remark 3.3. Since only a rough shape of control strategy is needed, the step size of the piecewise control grid should be relatively large to accelerate the computation.

Remark 3.4. It is generally known that the local minimal issue does not matter much when fitting smooth functions by neural networks. Thus the initial value of Θ for running fitting step can be generated randomly.

3.4 Description and Discussion of The Algorithm

To summarize all above construction, a complete description of the algorithm will be given here. The algorithm starts from following initialization steps.

Initialization 1: Construct the state lattice for deep learning algorithm denoted as $\{x_i\}_{i=1}^n$, and the state lattice for obtaining initial value of Θ denoted as $\{y_j\}_{j=1}^m$. These two state lattices satisfy following conditions:

$$x_0 = y_0,\ x_n = y_m,\ m \leq n.$$

Initialization 2: Choose three sets of computation precision ϵ and a maximal number of learning times. They are used to obtain initial values of Θ_0, to determine iterative control strategy $\mathrm{N}(x, \Theta^k)$, and to stop the MCAM iteration respectively.

Initialization 3: Pick up an appropriate function $f(x)$ to compute initial value for iteration. The choice is subject to properties of the problem. Compute U^0 as:

$$U^0(y_j) = f(y_j),\ j = 1, \cdots, m.$$

Initialization 4: Use the same function $f(x)$ as in *Initialization 3* to compute V^0 as:

$$V^0(x_i) = f(x_i),\ i = 1, \cdots, n,$$

After initialization, the algorithm will repeat below iterative steps. The repetition will stop until the algorithm achieves the desired precision, which is set up in *Initialization 2*.

Step 1: For $j = 1, \cdots, m$, $\tilde{\pi}^k(y_j)$ is obtained from standard MCAM. The input values U^{k-1} are from *Initialization 3* or *Step 4* in the last round.

Step 2: Fit against $\tilde{\pi}^k(y)$ by the gradient descent algorithm to obtain parameter starting values Θ_0^k:

$$\Theta_0^k = \arg\min_{\Theta} \sum_{j=1}^{m} \left(\tilde{\pi}^k(y_j) - \mathrm{N}(y_j|\Theta)\right)^2.$$

The fitting process will stop if the desired precision is achieved or if the maximal number of fitting iteration is reached, whichever comes first.

Step 3: Maximize $G\left(S^k(x_1), S^k(x_2), \cdots, S^k(x_n)\right)$ by the gradient descent algorithm to obtain the iterative control strategy. The input values V^{k-1} are from *Initialization 4* or *Step 5* in the last round. The learning process will stop if the desired precision is achieved or if the maximal number of learning iteration is reached, whichever comes first. Now, we have Θ^k, which yields $\hat{\pi}^k(x) = \mathrm{N}\left(x|\Theta^k\right)$.

Step 4: For $j = 1, \cdots, m$, iterate to $U^k(y_j)$ in the following way:

$$U^k(y_j) = S(y_j, U^{k-1}, \hat{\pi}^k(y_j)).$$

Step 5: For $i = 1, \cdots, n$, iterate to $V^k(x_i)$ in the following way:

$$V^k(x_i) = S(x_i, V^{k-1}, \hat{\pi}^k(x_i)).$$

Step 6: Compute $\sum_{i=1}^{n} \left(V^k(x_i) - V^{k-1}(x_i)\right)^2$, and then check the termination condition:

- If $\sum_{i=1}^{n} \left(V^k(x_i) - V^{k-1}(x_i)\right)^2 < \epsilon$, stop.
- If $\sum_{i=1}^{n} \left(V^{k+1}(x_i) - V^k(x_i)\right)^2 > \epsilon$,
  - If the maximal number of iterations is reached, stop.
  - Otherwise, go to *Step 1*.

Remark 3.5. The parameter m is the size of lattice used to obtain the rough shape of the control strategy. Since the algorithm requires to conduct a traversal on the control grid for each state of the lattice, the smaller m will improve the time-efficiency. However, smaller m will lead to higher numerical error introduced by the longer distance of the lattice, which will make the rough shape not suitable for later learning. In practice, we will make m as small as possible as long as it will generate a stable guess of optimal controls.

Remark 3.6. The overall algorithm involves two layers of loop. The outer loop is the iteration of the value function, which follows the computation rule of Markov chain approximation. For every sub-step of the outer loop, there are two different iterations for different

purposes. One is to locate the rough shape of neural network control strategy by fitting against piecewise controls, while the other one is to further improve the neural network control strategy through maximizing the objective G by the deep learning algorithm.

Remark 3.7. In step 4 and step 5, the same control strategy $\hat{\pi}^k = \mathrm{N}\left(\cdot|\Theta^k\right)$ is used to update both U^k and V^k, since we want to keep U^k owning a comparably similar shape as V^k to make the $\tilde{\pi}^{k+1}$ meaningful to use in *Step 2* of the next round.

Remark 3.8. Sometimes, we can let the small state lattice $\{y_j\}_{j=1}^m$ be a subset of $\{x_i\}_{i=1}^n$ with $y_j = x_{i_j}$, $1 \leq j \leq m$, and then let $U^k(y_j) = V^k\left(x_{i_j}\right)$, $1 \leq j \leq m$. Thus, we can avoid iteration of values on two lattices simultaneously.

After describing our hybrid method in detail, we will discuss its pros and cons in two main aspects: the curse of dimensionality and the accuracy of control value.

Bellman (1961) raises the problem of the curse of dimensionality in numerically solving controls that the computation nodes of optimal controls grow exponentially when state variables increase. A combination of all states should be considered for finding optimal controls on the lattice. The classical MCAM suffers from the curse of dimensionality, since, for every point in the state lattice, we need to conduct a traversal over the control value grid. As a result, the computational complexity of MCAM is of the same order of the product of the number of points in the state lattice and the number of nodes in the control value grid. In order to enhance the accuracy of the result, we need to narrow the difference distance of the state lattice and introduce a more precise piecewise control value grid. However, this exponential increase of computational time-consumption makes this method become an unaffordable choice for obtaining accurate results. This phenomenon is especially outstanding when the stochastic dynamic is driven by high dimensional randomness, or there are several factors being controlled.

With the help of the deep learning method, the optimization over the piecewise control grid can be avoided by calibrating neural network parameters for all states of the lattice simultaneously. The algorithmic complexity is mainly determined by the operation of evaluations of gradient, which is linearly with the size of the state lattice. The enhancement of algorithm efficiency brings more practical feasibility than existing lattice traversal methods. For simple problems, a finer difference can be applied to the construction of the state lattice in order to lower the error in numerical results. Many complicated problems, whose computational cost is beyond acceptance under the implementation of the ordinary MCAM, are now numerically solvable by our method.

The accuracy of the control strategy is another aspect to be improved by our method. In the implementation of the ordinary MCAM, the control strategy is optimized among a control value grid which only contains a finite number of nodes. Thus, the accuracy of the control strategy is always subject to the denseness of the control value grid. By contrast, neural networks allow the control strategy to take values in a given continuous range and will easily conquer the difficulty of effectively selecting the preciseness of the control grid. Moreover, when considering to approximate admissible control strategy by piecewise controls, we have no idea which range should be more precise to account for the rapid change of control. However, in our method, by the gradient descent algorithm, the neural network control strategy will be adjusted appropriately according to the sensitivities

of value function towards the control strategy. Furthermore, for some problems, admissible controls can take values from infinite real intervals, but we can only fix a large finite range to draw control values. This raises a general problem that how can we effectively select such large enough intervals. With the flexibility of neural networks, this can be easily solved by mapping the output of parametric control neural networks to the required infinite real interval.

Besides the above two main aspects, our method also preserves the main virtue of MCAM that the numerical result will converge to the optimal result in a very stable way. This stability of convergence is kept by using a small number of state nodes to obtain a rough shape of the control. As the source of stability, this shape is incorporated into the neural network control strategy through the information captured by the initial value of Θ. Maximizing G will improve the value on most states on top of MCAM result by enabling controls to take values in continuous ranges. Overall, with the integration of MCAM and deep learning, we can achieve the stable convergence, the high computational efficiency, and the good accuracy of control strategy simultaneously.

4 Unrestricted Dividend Payment Rate

In this section, we consider dividend strategy as a singular control and use the convention that $D(0^-) = 0$. The jump size of D at time $t \geq 0$ is denoted by $\Delta D(t) := D(t) - D(t^-)$, and $D^c(t) := D(t) - \sum_{0 \leq s \leq t} \Delta D(s)$ denotes the continuous part of $D(t)$.

Considering the dividend payment, the surplus process in the presence of dividend payments is written as

$$dX(t) = dR(t) - dD(t),\ X(0) = x \geq 0, \tag{4.1}$$

where x is the insurer's initial surplus. The performance function is the expected value of discounted future dividend payments

$$J(x, \pi(\cdot)) = \mathbb{E}_x \left(\int_0^\tau e^{-\gamma t} dD(t) \right), \tag{4.2}$$

where τ is the time of ruin, and $\mathbb{E}_x$ denotes the expectation conditioned on $X(0) = x$. The value function of maximizing expected dividend payoff is defined by the following optimization problem:

$$V(x) = \sup_{\pi} J(x, \pi(\cdot)), \tag{4.3}$$

where $\pi(\cdot) = (\phi(\cdot), D(\cdot))$. By applying MCAM to (4.3) again, we have the following iterative solution:

$$\begin{aligned} B(x, V, \phi) \triangleq\ & e^{-\gamma \Delta t(x,\phi)} \left[p\left(x, x + \Delta x | \phi\right) V\left(x + \Delta x\right) + p\left(x, x - \Delta x | \phi\right) V\left(x - \Delta x\right) \right] \\ & + \delta\left(x, \phi\right) \Delta t\left(x, \phi\right), \\ L\left(x\right) =\ & \sup_{\phi \in [0,1]} B(x, V, \phi). \end{aligned}$$

Here we adopt same notations p and Δt as in (3.6), where the corresponding terms are

defined as follows

$$\begin{aligned}\alpha(x,\phi) &\triangleq \frac{1}{2}\sigma^2\phi^2;\\ \beta(x,\phi) &\triangleq (\eta-\rho+\rho\phi)a;\\ \delta(x,\phi) &\triangleq 0.\end{aligned}$$

Let

$$K(x) = V(x-\Delta x)+\Delta x.$$

By combining above two parts, we now can construct the new iterative rules. Precisely, the reinsurance strategy is obtained as:

$$\phi^{(k+1)} = \arg\max_{\phi\in[0,1]} B(x, V^k, \phi). \tag{4.4}$$

Then we have the value by inputting the reinsurance strategy (4.4) as

$$L^{k+1}(x) = \sup_{\phi\in[0,1]} B(x, V^k, \phi). \tag{4.5}$$

When dividend is paid out, the value changes as

$$K^{k+1}(x) = V^k(x-\Delta x)+\Delta x.$$

Then the new value function is obtained as

$$V^{k+1}(x) = \max\left\{L^{k+1}(x), K^{k+1}(x)\right\}. \tag{4.6}$$

Meanwhile, the cutting point for dividend strategy can be approximated as:

$$\tilde{u}^{k+1}(x) = \max\left\{x : L^{k+1}(x) - K^{k+1}(x) < 0\right\} - \frac{1}{2}\Delta x. \tag{4.7}$$

Since it is impossible to guarantee that there always exists a point in state lattice making $L^{k+1}(x) - K^{k+1}(x) = 0$, this approximation will bound the error by the half of step size.

To apply our deep learning method, we only model the relation between the state x and the reinsurance strategy by a neural network as

$$\phi^N \triangleq N(x|\Theta).$$

Observing from (4.5), the optimization over admissible control values are conducted only for evaluation of L, thus we define the global improvement function G on L instead of V as:

$$G \triangleq G\left(L(x_1), L(x_2), \cdots, L(x_n)\right).$$

By inputing previous value function V^k, we can determine the iterative reinsurance strategy and the iterative function Q as:

$$\Theta^{k+1} = \arg\max_{\Theta} G\left(L(x_1), L(x_2), \cdots, L(x_n)\right),$$

$$L^{k+1}(x_i) = B(x_i, V^k, \mathrm{N}\left(x_i|\Theta^{k+1}\right))\ 1 \le i \le n.$$

Here the gradient descent algorithm can be run again to search for the best Θ. After obtaining Q^{k+1}, we can use same functions as (4.7) and (4.6) in MCAM to compute the iterative dividend strategy u^{k+1} and to iterate the value function to V^{k+1}. As well, this iteration procedure will be repeated until:

$$\sum_{i=1}^{n} \left(V^{k+1}(x_i) - V^k(x_i)\right)^2 < \epsilon.$$

Remark 4.1. This unrestricted case is used to offer basic illustration about how to extend our method for more general stochastic optimization problems in actuarial field. For more complex controlled dynamics or more complicated objectives, the optimization may involve selecting from maximums of different parts such as possible capital injections to avoid financial ruin. It does not add much difficulty to learn such types of impulse controls. Since the crucial logic of our method relies on improving the function which directly reflects the optimization over control values, we just need to apply deep learning method for every function we need to independently optimize.

5 Optimal Dividend Reinsurance and Investment under Regime-Switching Jump Diffusion Model

In this section, we will use the proposed algorithm to solve the optimal dividend reinsurance and investment problem under a complicated regime-switching jump-diffusion model where no analytical results are available. A numerical example will be provided in Section 6 to illustrate the performance of the algorithm.

To delineate the random environment and other random factors, we use a continuous-time Markov chain $\chi(t)$ whose generator is $Q = (q_{rs}) \in \mathbb{R}^{h\times h}$ and state space is $\mathcal{M} = \{1, \ldots, h\}$. Let ζ_j be the arrival time of the j-th claim. Corresponding to each $r \in \mathcal{M}$, $\Lambda_r(t) = \max\{j \in \mathbb{N} : \zeta_j \le t\}$ is the number of claims up to time t, which is a Poisson counting process.

The surplus process under consideration is a regime-switching jump-diffusion. Let $H_r(t) := \sum_{j=1}^{\Lambda_r(t)} Y_k$, $j = 1, 2, \ldots,$ the accumulated claims in regime i up to time t. For $r \in \mathcal{M}$, the claim arrival process $\Lambda_r(t)$ is a Poisson process with intensity $\lambda_r > 0$, and claim sizes $Y_j, j = 1, 2, \ldots,$ are i.i.d random variables with density function $g(x)$ that is independent of $\Lambda_r(t)$. Suppose that the claim size has finite first and second moments, respectively. Further, we assume for each $r \in \mathcal{M}$, the premium rate $c(r)$ follows the expectation premium principle:

$$c(r) = (1 + \iota)\lambda_r \mathbb{E}[Y].$$

Let ϕ be an exogenous retention level, which is a control chosen by the insurance company representing the proportional reinsurance policy. We allow the insurance companies to continuously reinsure a fraction of its claim with the retention level $\phi \in [0, 1]$. By using the variance premium principle, the reinsurance premium rate at time t is

$$\psi(\phi) = (1 - \phi)\mathbb{E}[Y] + \nu(1 - \phi)^2 \mathrm{Var}[Y]. \tag{5.1}$$

where $\nu > 0$ is the safety loading of the reinsurer. We use different premium principles for insurers and reinsurers since we aim to show the feasibility of our numerical method for a general model. Other types of premium principles are also accepted.

Following the work in Yang and Zhang (2005), we assume the surplus process is invested in a financial market under the regime-switching progress with a risk free asset whose price follows

$$dS_0(t) = S_0(t)r_0(\chi(t))dt$$

and K risky assets whose prices are given by

$$dS_i(t) = S_i(t)(\mu_i(\chi(t))dt + \sum_{j=1}^{K} \sigma_{ij}(\chi(t))dW_j(t)),$$

where $W(t) = (W_1(t), \ldots, W_k(t))'$ is an K-dimensional standard Brownian motion. In the above and hereafter, A' denotes the transpose of A with A being either a vector or a matrix with appropriate dimensions, and $|A|$ denotes the Euclidean norm of A. Set

$$B(\chi(t)) = (\mu_1(\chi(t)) - r_0(\chi(t)), \ldots, \mu_K(\chi(t)) - r_0(\chi(t)))', \quad \text{and} \quad \sigma(\chi(t)) = (\sigma_{ij}(\chi(t)))_{K\times K}. \tag{5.2}$$

We use the proportional portfolio $\varpi(t) = (\varpi_1(t), \ldots, \varpi_K(t))'$ to represent an investment strategy, where $\varpi_i(t)$ is the percentage in the total wealth of the capital invested in the i-th risky asset. To better reflect the reality in certain markets where short selling is not allowed, we further set a borrowing constraint on the investment strategy, which means $\sum_{i=1}^{K} \varpi_i(t) \leq 1$ at any time. Denote

$$[0,1]^K = [0,1] \times [0,1] \times \cdots \times [0,1],$$

and denote the constraint set of the controls as

$$\Gamma := \left\{ \varpi \in [0,1]^K : \sum_{i=1}^{K} \varpi_i \leq 1 \right\}. \tag{5.3}$$

The surplus process with dividend payment is given by

$$\begin{aligned} dX(t) = &\{X(t)[r_0(\chi(t)) + \varpi'(t)B(\chi(t))] + c(\chi(t))dt - \lambda_{\chi(t)}\psi(\phi)\}dt \\ &+X(t)\varpi'(t)\sigma(\chi(t))dW(t) - \phi dH_{\chi(t)}(t) - dD(t). \end{aligned} \tag{5.4}$$

We are working on a filtered probability space $(\Omega, \mathcal{F}, \{\mathcal{F}_t\}, P)$, where $\mathcal{F}_t$ is the σ-algebra generated by $\{\chi(s), W(s), \Lambda_r(s) : 0 \leq s \leq t,\ r \in \mathcal{M}\}$.

Here the dividend rate is bounded by M as well, thus $dD(t) = u(t)dt$. For $r \in \mathcal{M}$, and $V(\cdot, r) \in C^2(\mathbb{R})$, define an operator $\mathcal{L}$ by

$$\begin{aligned} \mathcal{L}V(x,r) = &V_x(x,r)[x(r_0(r) + \varpi' B(r)) + c(r) - \lambda_r\psi(\phi) - u] + \frac{1}{2}x^2 V_{xx}|\varpi'\sigma(r)|^2 \\ &+\lambda_r \int_0^x [V(x - \phi z, r) - V(x,r)]g(z)dz + QV(x,\cdot)(r), \end{aligned} \tag{5.5}$$

where V_x and V_{xx} denote the first and second derivatives with respect to x, and

$$QV(x,\cdot)(r) = \sum_{s\neq r} q_{rs}(V(x,s) - V(x,r)),$$

where q_{rs} is the transition rate from regime r to regime s. In this case, denote the control as $\pi := (\phi, u, \varpi) \in [0,1] \times [0,M] \times \left\{ [0,1]^L \cap \left\{ \sum_{l=1}^K \varpi_i \leq 1 \right\} \right\}$.

As before, we define following terms:

$$\begin{aligned}
\alpha(x,r,\pi) &\triangleq \frac{1}{2}x^2 \sum_{j=1}^K \left(\sum_{i=1}^K \varpi_i \sigma_{ij}(r) \right)^2; \\
\beta(x,r,\pi) &\triangleq (r_0(r) + \varpi' B(r))x + c(r) - \lambda_r \psi(\phi) - u; \\
\delta(x,r,\pi) &\triangleq u; \\
\omega(x,r,\pi) &\triangleq |\beta(x,r,\pi)| \Delta x + 2\alpha(x,r,\pi) - q_{rr}\Delta x^2; \\
\Delta t(x,i,\pi) &\triangleq \frac{\Delta x^2}{\omega(x,r,\pi) + \gamma \Delta x^2}.
\end{aligned}$$

To simplify the derivation, the Markov chain update rule will be given directly for the state lattice $\{x_i\}_{i=0}^n$ as:

$$V^k(x_i,r) = \sup_{\pi\in\Pi} \left\{ \begin{array}{l}
(1-\lambda_r \Delta t(x_i,r,\pi)) e^{-\Delta t(x_i,r,\pi)\gamma} \left[\begin{array}{l} p(x_i,r,x_{i+1},r|\pi) V^{k-1}(x_{i+1},r) \\ + p(x_i,r,x_{n-1},r|\pi) V^{k-1}(x_{n-1},r) \end{array} \right] \\
+ (1-\lambda_r \Delta t(x_i,r,\pi)) e^{-\Delta t(x_i,r,\pi)\gamma} \sum_{s\neq r} p(x_i,r,x_i,s|\pi) V^{k-1}(x_i,s) \\
+ \lambda_r \Delta t(x_i,r,\pi) e^{-\Delta t(x_i,r,\pi)\gamma} \sum_{j=0}^{i-1} m(x_i,r,x_j,r|\pi) V^{k-1}(x_j,r) \\
+ \delta(x_i,r,\pi) \Delta t(x_i,r,\pi)
\end{array} \right\},$$

where transition probabilities are defined as follows:

$$\begin{aligned}
p(x_i,r,x_{i+1},r|\pi) &\triangleq \frac{\alpha(x_i,r,\pi) + \max\{\beta(x_i,r,\pi),0\}\Delta x}{\omega(x_i,r,\pi)}; \\
p(x_i,r,x_{n-1},r|\pi) &\triangleq \frac{\alpha(x_i,r,\pi) - \min\{\beta(x_i,r,\pi),0\}\Delta x}{\omega(x_i,r,\pi)}; \\
p(x_i,r,x_i,s|\pi) &\triangleq q_{rs}\Delta t(x_i,r,\pi); \\
m(x_i,r,x_j,r|\pi) &\triangleq \int_{\frac{x_i-x_{j-1}}{\phi}}^{\frac{x_i-x_j}{\phi}} g(z)dz, \ 0<j<i; \\
m(x_i,r,x_0,r|\pi) &\triangleq \int_{\frac{x_i-x_1}{\phi}}^{\infty} g(z)dz.
\end{aligned}$$

6 Numerical Examples

Computation precision ϵ and maximal numbers of learning times for locating the initial value of Θ, for determining iterative control strategy Θ^k, and for iteratively updating value function V^k are as follows:

	Triggering Error	Max # of steps
Control Fit	10^{-3}	10 000
Gradient Descent	10^{-6}	5 000
Global Iteration	10^{-7}	50 000

Due to the fact the value function is concave, root function $\sqrt{x}$ is used as the initial guess of the value function. All methods are coded by Python with TensorFlow package and run on x64 platform of Intel Xeon E-2186 2.90GHz CPU with 64GB RAM and NVIDIA Quadro P5200 GPU with 16GB RAM.

6.1 Restricted Dividend Payment Rates

In this case, we exactly follow the framework of Høgaard and Taksar (1999) and assume the bound for dividend rate M is 1.0. The effectiveness of the proposed method will be examined numerically against the setup as follows:

η	ρ	σ	γ
0.1	0.1	0.2	0.125

According to the analytical results in Høgaard and Taksar (1999), the closed-form solutions for optimal restricted reinsurance strategy, optimal dividend strategy are obtained with feedback control type. The optimal controls and value function with above parameters are as follows

$$\begin{aligned} u(x) &= 1_{\{x \geq 0.4283\}}, \\ \phi(x) &= \min\{1, 5x\}, \\ V(x) &= \begin{cases} 0.5267\,(5x)^{0.5}, & x < 0.2, \\ -0.3648e^{-6.0355x} + 0.5169e^{1.0355x}, & 0.2 \leq x < 0.4283, \\ 8 - 7.6633e^{-0.1385x}, & x \geq 0.4283. \end{cases} \end{aligned} \tag{6.1}$$

For the deep learning Markov chain approximation method, the value function and the control strategies after different numbers of iterations are plotted in Figure 4. To better demonstrate the change of control strategy and the value function during the course of iterations, we will also show the convergence for the first 1000 iterations as shown in Figure 5. To examine the effectiveness of our method, the theoretical results computed from explicit solutions are provided as benchmarks.

It can be observed from Figures 4 and 5 that the control strategy converges much faster than the value function. After the first 1000 iterations, the control strategy almost converges

to the optimal, but the value function does not rise much. This is due to the fact the iterative control strategy is updated using the information of the shape of the value function instead of the absolute location of the value function. Our method is of more help to achieve faster convergence of the control by introducing deep learning to improve the shape of the value function globally.

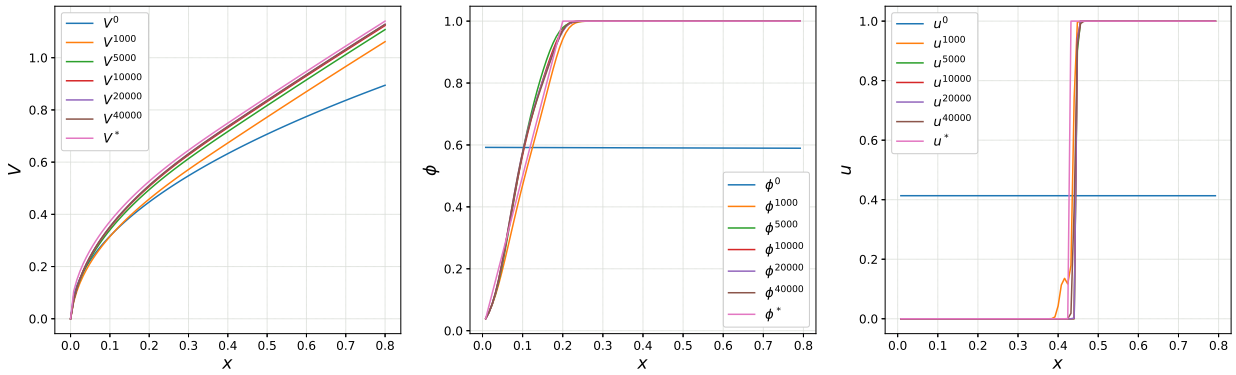

Figure 4: Convergences of Value Function and Control Strategies

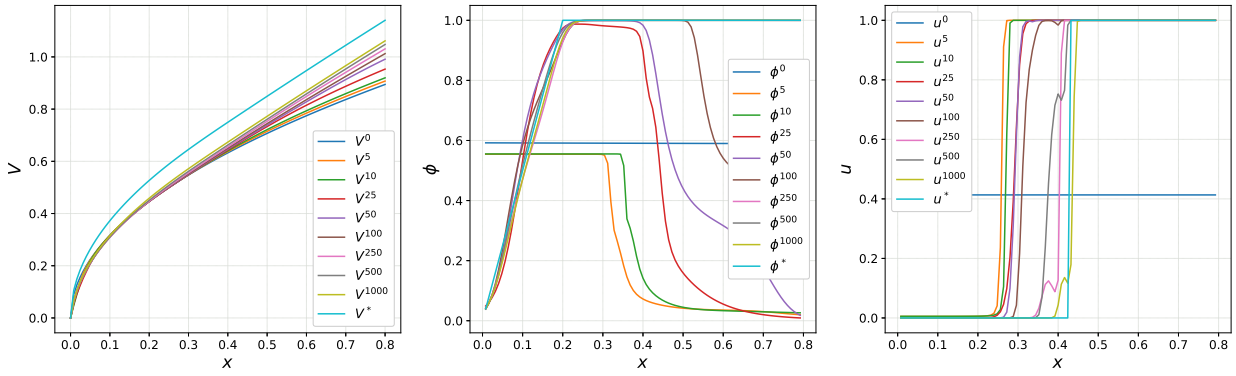

Figure 5: Convergence of Value Function and Control Strategies in First 1000 Iterations

6.2 Unrestricted Dividend Payment Rates

All setups and parameters of the numerical test remain the same in this section. Similar to the case of restricted dividend payment rates, the closed-form solutions of optimal feedback-type controls and value functions are obtained in Høgaard and Taksar (1999) as follows:

$$
\begin{aligned}
&D(x) = (x - 0.4493)^{+}, \\
&\phi(x) = \min\{1, 5x\}, \\
&V(x) = \begin{cases} 0.5275\,(5x)^{0.5} & x < 0.2, \\ -0.3653e^{-6.0355x} + 0.5176e^{1.0355x} & 0.2 \le x < 0.4493, \\ x + 0.3507 & x \ge 0.4493. \end{cases}
\end{aligned} \tag{6.2}
$$

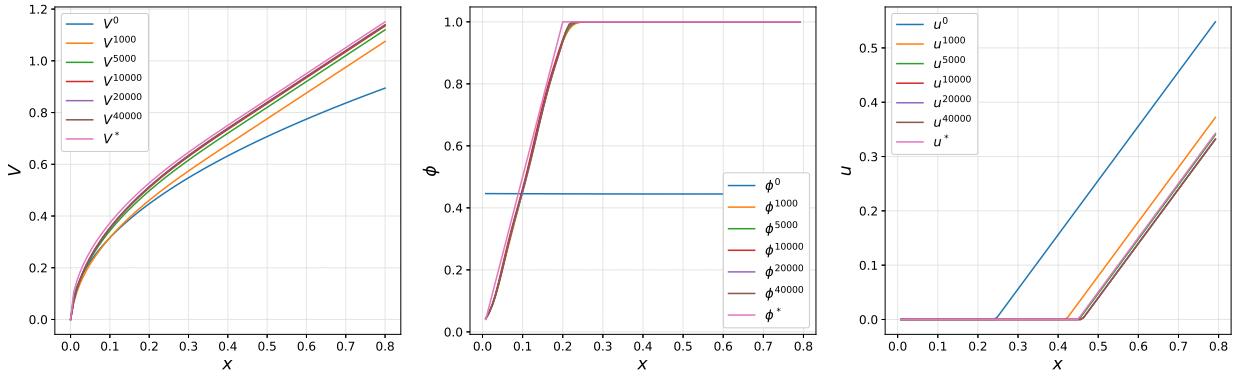

Figure 6: Convergences of Value Function and Control Strategies

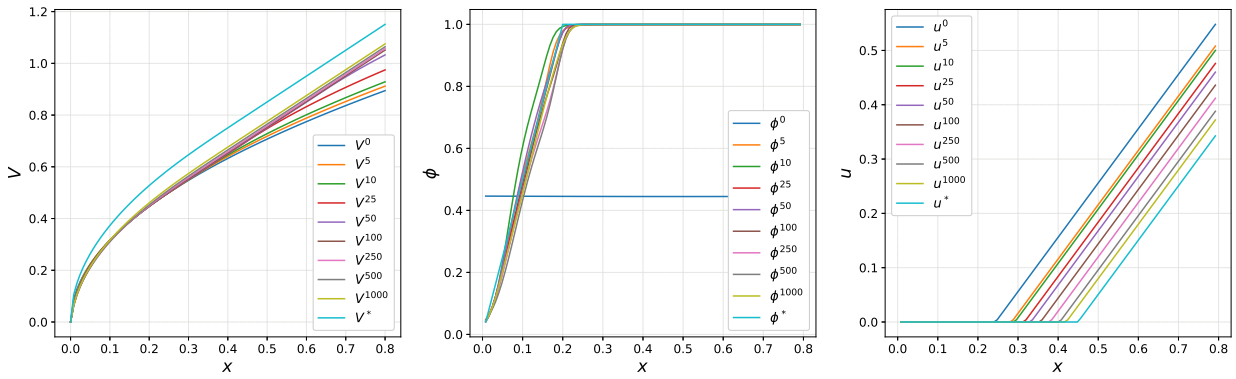

Figure 7: Convergence of Value Function and Control Strategies in First 1000 Iterations

The value function, reinsurance strategies, and accumulated dividend amount are plotted in Figure 6, and the convergence for first 1000 iterations is in Figure 7.

Similarly as in Section 6.1, the value function and optimal controls are learned well in 50 000 iterations. Due to the singularity of the dividend payment, the total amount of

dividend payment is approximated directly. It is shown that total accumulated dividend payment converges to a ramp function, where the turning point represents the barrier to trigger the dividend payment. This method does not rely on the technique of quasi-variational inequality which is generally referred to in the optimization problems with singular controls. Instead, it deals with the objective functions directly to approximate the dividend payment amount. In addition, we should bear in mind that it does not add extra difficulties to extend this algorithm to optimization problems with more types of controls, such as investment and capital injections.

6.3 Optimal Dividend Reinsurance and Investment

In this part, we will examine the effectiveness when our method is implemented under a complicated random environment. Suppose there are two regimes with the following generator matrix:

$$\begin{bmatrix} -0.05 & 0.05 \\ 0.1 & -0.1 \end{bmatrix}.$$

Further we assume the claim follows exponential distribution with density $g(x) = \zeta e^{-\zeta x}$, and there are two risky assets available in the financial market. The parameters are setup as follows:

ζ	ν	ι	γ	M
1	0.02	0.1	0.125	1.5

Regime	λ	r_0	μ_1	μ_2	σ_{11}	σ_{12}	σ_{21}	σ_{22}
1	0.05	0.05	0.18	0.2	0.15	0.5	0.3	0.6
2	0.1	0.12	0.2	0.25	0.15	0.4	0.25	0.6

The convergence results are plots as Figure 8 for regime 1 and Figure 9 for regime 2. The comparison of final results between regime 1 and regime 2 are plots as Figure 10.

It can be seen from figures that the value functions hardly change from 10 000 iterations to 20 000 iterations, which indicates the convergence of numerical results is achieved by our method. From the comparison of results, our method effectively learn the difference between controls of different regimes, which demonstrates the effectiveness in searching for the optimal strategy under complicated stochastic dynamics.

We also implement the original MCAM algorithm for this problem with the same size of lattice on the same computational platform as a benchmark of time-efficiency. It turns out that our method can approximately reduce 80% of the time consumed by the classical MCAM method for this example. The reason is that, as the neural network control strategy is approaching the optimal control strategy, the number of deep learning iterations decrease dramatically. In other words, since the computation work is reduced to a small number of deep learning iterations plus the valuation of the control value for every state using the trained neural network, the time spent on conducting traversal over the control grid for every state in the lattice can be saved.

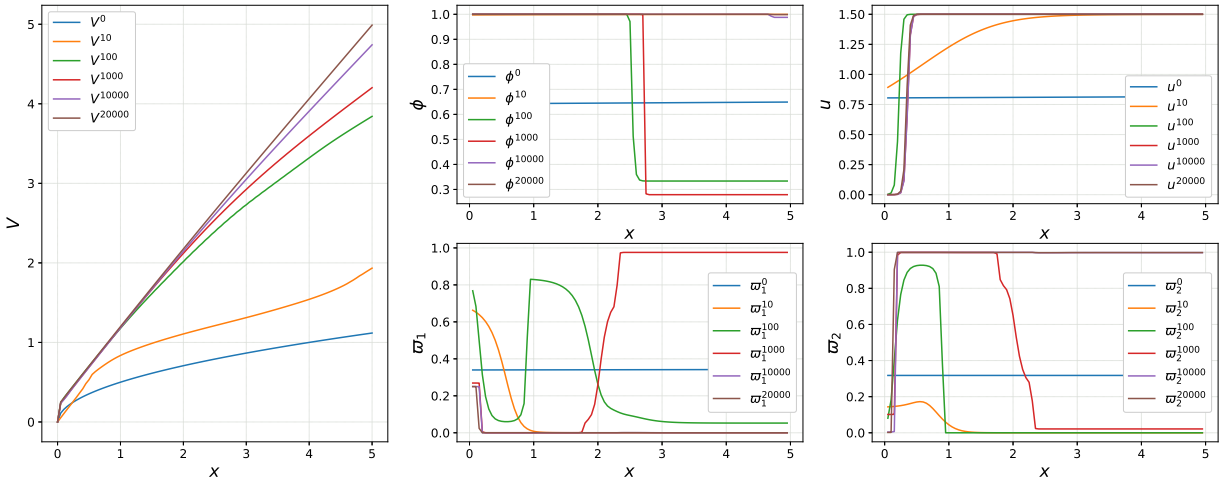

Figure 8: Convergences of Value Function and Control Strategies for Regime 1

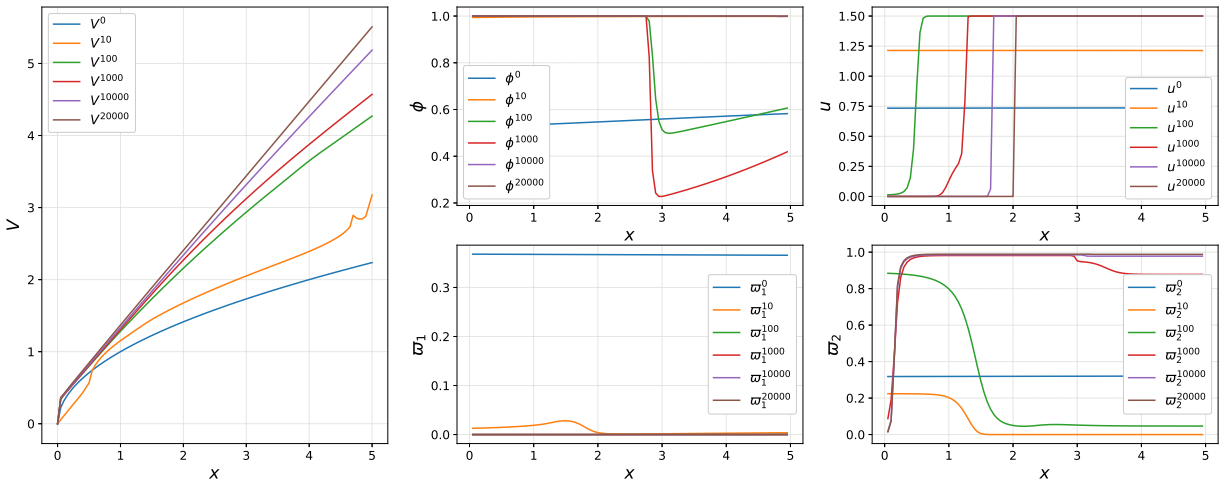

Figure 9: Convergences of Value Function and Control Strategies for Regime 2

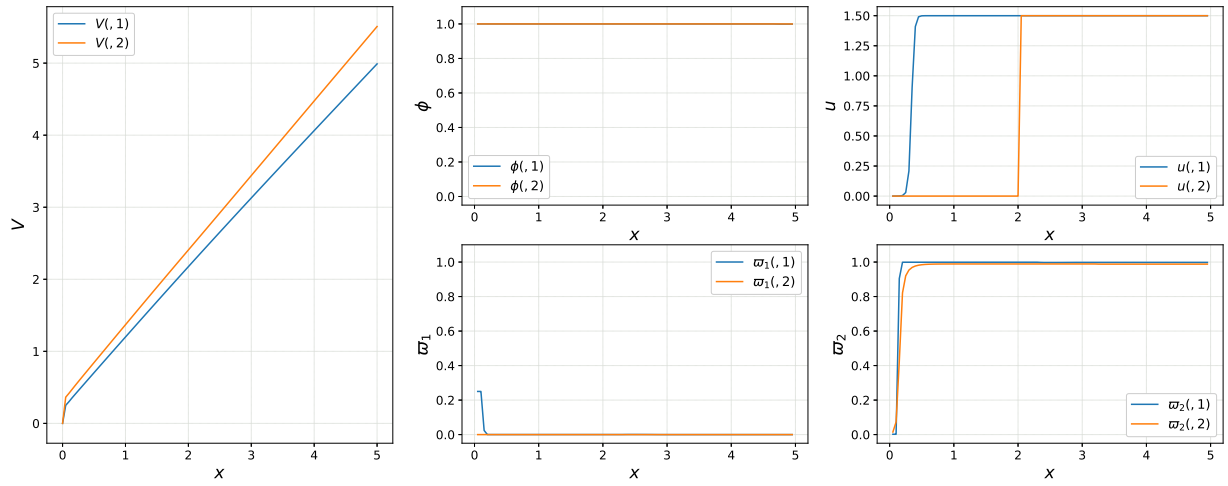

Figure 10: Comparison of Final Results between Regime 1 and Regime 2

In the following, we will provide an example to show how the efficiency and accuracy can be improved. Consider a lattice of classical MCAM with $N \times M$ computation nodes, where N is the number of nodes of state and M is the number of nodes for controls. Now we implement the propose algorithm in two steps: First, we deal with a lattice with $N/5$ computation nodes for rough guess. By using the MCAM, we need M nodes to determine the initial guess of controls. Then the total step is $N \times M/5$. Second, we use gradient descent method to determine the parameters of the neural networks. Assume that the number of gradient calculation nodes to obtain initial guess of parameter at each state is K_1, and the gradient calculation nodes for the parameter converge at each state is K_2, the computation for gradient descent is $N \times (K_1 + K_2)$. Thus, the total computation for our method is $N \times (M/5 + K_1 + K_2)$.

Practically, the proposed deep learning algorithm's total computation load $N \times (M/5 + K_1 + K_2)$ is significantly smaller than the transversal grid in the classical MCAM with size $N \times M$ when M is sufficiently large. Moreover, with larger M the control accuracy is improved as well. Thus, if we compare results with the same accuracy level state lattice, the proposed algorithm's computation efficiency is significantly improved in finding optimal controls for a given M. In addition, if the algorithm runs the same number of computation nodes as MCAM, M can be set much larger in our method, marking approximating controls more accurate.

7 Concluding Remarks

This paper develops a hybrid Markov chain approximation-based deep learning method to approximate the optimal reinsurance and dividend strategies. The optimal controls are subject to a random termination stopping time, thus leading to an infinite-horizon optimization

problem. The developed deep learning algorithm directly approximate the value function and controls by deep neural networks.

The accuracy of approximating piecewise controls in MCAM much depends on the density of the control grid. However, subject to the computation capability, the computation grids cannot be arbitrarily dense. The proposed method implements the gradient descent method to find the optimal control strategy. Hence we can obtain more accurate control values from the continuous output of the neural network. Meanwhile, our proposed method can also handle the curse of dimensionality from which existing numerical methods are suffering. Given that the time spent on searching among the control grid has been saved, we can build a finer state lattice with limited computing power.

In future studies, we will develop a deep learning algorithm to solve for optimization problem with finite horizon. To find the optimal controls as neural networks of a class of implicit parameters, we will simulate a set of Monte Carlo sample paths, and gradually improve the strategies by training the neural network for each control. Also, similar to the infinite-horizon case, the curse of dimensionality becomes even worse because of the additional time intervals. With the Markov chain approximation-based deep learning method, we work on the optimization criteria directly, and only need more parameters and sample paths when more states or control variables are included. Then the amounts of computation work only increase linearly. Hence, the computation cost is largely reduced.

Further, we can also use stochastic approximation methods instead of gradient descent method to train neural networks. Stochastic approximation method has its advantage to approximate gradients in complex stochastic systems. A comprehensive introduction of stochastic approximation can be referred to Kushner and Yin (2003). Then, the forward-backwards propagation method will be applied to calibrate the neural networks to learn the optimal strategies along the timeline.

Acknowledgments

We are grateful to the editors and anonymous referees for their insightful comments and suggestions. These comments/suggestions greatly improved the quality and readability of the paper. This research was supported in part by the Research Grants Council of the Hong Kong Special Administrative Region (project No. 17330816) and by a Faculty Research Grant from The University of Melbourne.

References

Albrecher, H., Beirlant, J., and Teugels, J.L. (2017). *Reinsurance: Actuarial and Statistical Aspects,* Wiley, West Sussex.

Aleandri, M. (2018) Modeling dynamic policyholder behaviour through machine learning techniques. Working paper.

Arrow, K. (1963). Uncertainty and the welfare economics of medical care. *American Economic Review*, 53: 941–973.

Asmussen, S. and Taksar, M. (1997). Controlled diffusion models for optimal dividend pay-out. *Insurance: Mathematics and Economics*, 20: 1–15.

Bachouch, A., Huré, C., Langrené, N., and Pham, H. (2018). Deep neural networks algorithms for stochastic control problems on finite horizon, Part 2: numerical applications. *arXiv preprint arXiv:1812.05916.*

Bellman, R. E. (1961). *Adaptive control processes: a guided tour.* Princeton university press, Princeton, N.J.

Borch, K. (1960). Reciprocal reinsurance treaties. *ASTIN Bulletin*, 1(4): 170–191.

Borch, K. (1962). Equilibrium in a reinsurance market. *Econometrica*, 30: 424–444.

Carmona, R., and Laurière, M. (2019). Convergence analysis of machine learning algorithms for the numerical solution of mean field control and games: II–The finite horizon case. *arXiv preprint arXiv:1908.01613.*

De Finetti, B. (1957). Su unimpostazione alternativa della teoria collettiva del rischio. *Transactions of the XVth International Congress of Actuaries* 2: 433–443.

E, W., Han, J., and Jentzen, A. (2017). Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. *Communications in Mathematics and Statistics,* 5(4), 349-380.

Fahrenwaldt, M. A., Weber, S., and Weske, K. (2018). Pricing of cyber insurance contracts in a network model. *ASTIN Bulletin*, 48(3), 1175–1218.

Fecamp, S., Mikael, J., and Warin, X. (2019). Risk management with machine-learning-based algorithms. *arXiv preprint arXiv:1902.05287.*

Gerber, H. U. (1972). Games of economic survival with discrete and continuous income processes. *Operations Research*, 20(1): 37–45.

Han, J. and E, W. (2016). Deep learning approximation for stochastic control problems. *arXiv preprint arXiv:1611.07422.*

Højgaard, B. H., and Taksar, M. (1999). Controlling risk exposure and dividends payout schemes: insurance company example. *Mathematical Finance,* 9(2), 153–182

Huré, C., Pham, H., Bachouch, A., and Langrené, N. (2018). Deep neural networks algorithms for stochastic control problems on finite horizon, part I: convergence analysis. *arXiv preprint arXiv:1812.04300.*

Jin, Z., Yang, H., and Yin, G. (2013). Numerical methods for optimal dividend payment and investment strategies of regime-switching jump diffusion models with capital injections. *Automatica*, 49(8): 2317–2329.

Jin, Z., Yang, H., and Yin, G. (2018). Approximation of optimal ergodic dividend and reinsurance strategies using controlled Markov chains, *IET Control Theory & Applications.*, 12(16), 2194–2204.

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

Kushner, H. and Dupuis, P. (2001). *Numerical Methods for Stochstic Control Problems in Continuous Time*, volume 24 of *Stochastic Modelling and Applied Probability*. Springer, New York, second edition.

Kushner, H. and Yin, G. (2003). *Stochastic Approximation and Recursive Algorithms and Applications*, volume 35 of *Stochastic Modelling and Applied Probability*. Springer, New York, second edition.

Pereira, M., Wang, Z., and Theodorou, E. A. (2019). Neural network architectures for stochastic control using the nonlinear Feynman-Kac lemma. *arXiv preprint arXiv:1902.03986.*

Van Staden, P. M., Dang, D. M., and Forsyth, P. A. (2018). Time-consistent mean-variance portfolio optimization: A numerical impulse control approach, *Insurance: Mathematics and Economics*, 83, 9–28.

Wei, J., Yang, H., and Wang, R. (2010). Classical and impulse control for the optimization of dividend and proportional reinsurance policies with regime switching. *Journal of Optimization Theory and Applications*, 1:358–377.

Wüthrich, M. V., (2018a). Machine learning in individual claims reserving. *Scandinavian Actuarial Journal,* 6:465–480.

Wüthrich, M. V., (2018b). Neural networks applied to chain-ladder reserving. *European Actuarial Journal,* 8(2), 407–436.

Wüthrich, M. V., and Buser, C. (2019). Data analytics for non-life insurance pricing. *Swiss Finance Institute Research Paper,* 16–68.

Yang, H., and Zhang, L. (2005). Optimal investment for insurer with jump-diffusion risk process. *Insurance: Mathematics and Economics*, 37(3), 615–634.