The School of Mathematics

THE UNIVERSITY
of EDINBURGH

Derive Optimal Stopping Time with Neural Network

by

Zhongtian Zheng

Dissertation Presented for the Degree of
MSc in Financial Mathematics

August 2022

Supervised by
Dr Stefan Engelhardt

Abstract

An optimal stopping problem in a discrete setting can be theoretically solved by the well-known Snell envelope, while its numerical approximation especially when the dimension of the underlying assets is large can be difficult to achieve. This dissertation aims at explaining a deep learning approach proposed in Becker et al. [2018], illustrating the advantages of the method by some representative examples, and discussing the limitation of the method in terms of its generalizability.

Additional works focus on discussing the impact of sampling method on the robustness of the model and comparing it with a similar algorithm used by BECKER et al. [2021]and Reppen et al. [2022], which abandoned the backwards induction scheme. In addition, we also investigate the possibility of faster calculation of upper bounds by neural regression method. The model compression by knowledge distillation is also discussed.

Acknowledgments

I would like to express my deepest appreciation for my dissertation supervisor, Dr. Stefan Engelhardt for his continual feedback and valuable discussion, from which I have been able to continuously refine my work. I am also thankful to every lecturer of mine for their patience and guidance. Last but not least, thanks to my families for all their support and assistance.

Own Work Declaration

I confirm that all this work is my own except where indicated and that I have clearly referenced all sources as appropriate. Every figure, and reported numerical result in the two illustrated examples is generated from my own codes, using Python programming. The codes distribute in three Jupyter notebooks which can run on Google Colab Pro with 25 GB CPU RAM, 16 GB VRAM (Tesla P100) and are made open source. The three notebooks contains different content for the sake of management - `Uniform.ipynb`,`Interception.ipynb`,`Girsanov` display different sampling methods as described in 4.2.1. Estimation for lower, and upper bound are realized in `Uniform.ipynb`.

Contents

List of Tables

List of Figures

1 Introduction

Among the financial derivatives, the options that allow early exercise can be difficult to price since these options essentially involve optimal stopping problems. In general, they can be categorized into either the American type which allows early exercise any time until maturity, or the Bermudan type which only allows early exercise on specified dates. While the practitioners in the industries often resort to sample-based methods for pricing these derivatives, the size of required samples explodes as the dimension of underlying assets increases, thus invalidating the traditional method that takes all data to train a model, calling for an algorithm compatible with large data set. With the advance in computational hardware, neural networks become a powerful solution for large data sets. In this dissertation, we explain and replicate two representative algorithm proposed by Becker et al. [2018], BECKER et al. [2021] respectively (see also Reppen et al. [2022]), which take good advantages of neural networks' large data handling ability and flexibility. The former algorithm is called "backward induction algorithm" and the latter is called "one-shot algorithm" for short. We will reveal in the sequel that the one-shot algorithm is more suitable for derivatives with large number of exercise times and is significantly faster than the backward induction algorithm. While the two algorithms can yield similarly accurate estimates, our additional work aims at making the two algorithm more generalizable in the sense that the trained models can be used to price a range of initial states. With this objective, we investigate several sampling methods including an importance sampling technique by the Girsanov theorem.

In this section, we start with a formulation of our discussed problem, devoted particularly to the Bermudan type since the algorithm to be introduced relies on effectively sampling from the target stochastic process of the underlying assets while it is only feasible to simulate countably many time points, we have to approach an optimal stopping problem from its discretised version. Although the problem is formulated in a discrete-time setting, it is proved by Becker et al. that an American type option can be accurately approximated by a Bermudan type with any precision if only we set a sufficiently large number of exercise dates. Therefore, we devote the problem formulation to the Bermudan type here.

Problem Formulation:
Denote the terminal time $T \in (0, \infty)$ (e.g. Maturity date of options), the number of underlying assets by $d \in \mathbb{N}^+$ [1]. For $N \in \mathbb{N}^+$, the specified exercise date[2] given by a sequence of real value time points $0 =: t_0 < t_1 < ... < t_n < ... < t_N := T$. Denote the process of the underlying assets by $(S_t)_{t\in[0,T]}$, which is an $\mathbb{R}^d$-valued continuous Markov process. Denote the observed values of the underlying assets at the exercise dates by $X = (X_n)_{n=0}^N := (S_{t_n})_{n=0}^N$ which is an $\mathbb{R}^d$-valued discrete-time Markov process. Let $\mathcal{F}_n := \sigma(X_0, X_1, ..., X_n)$ for all $n \in \{0, 1, ..., N\}$ which forms a and define a random variable $\tau : \Omega \to \{0, 1, ..., N\}$ such that $\{\tau = n\} \in \mathcal{F}_n$ for all $n \in \{0, 1, ..., N\}$. By definition, τ is also called X-stopping time.

This dissertation is devoted mainly to explaining the algorithm proposed by Becker et al. [2018] where they use the deep learning method to derive an approximated optimal policy for stopping problems of the form

$$\sup_{\tau\in\mathcal{T}} \mathbb{E}g(\tau, X_\tau), \tag{1.1}$$

where $g : \{0, 1, ..., N\} \times \mathbb{R}^d \to R$ is a measurable payoff function and $\mathcal{T}$ denotes the set of all X-stopping time. Particularly in the financial context, n always enters the expression of g because of discounting. The problem admits an optimal solution under the integrability assumption

$$g(n, X_n) \in L^1(\Omega, \mathcal{F}_n, \mathbb{P}) \text{ for all } n \in \{0, 1, 2, ...N\}, \tag{1.2}$$

[1] In the sequel of this dissertation, d refers to the number of assets if without specification. In practice, one could add additional $\mathcal{F}_n$-measurable features if they help with neural networks' training, such as the current payoff, discounting value etc.

[2] For $N = 1$, the problem is of European type. That is, one either does not buy the derivative or waits till the

while the stronger assumption will be needed to derive the upper bound of our estimation for the optimal value in (1.1)

$$g(n, X_n) \in L^2(\Omega, \mathcal{F}_n, \mathbb{P}) \text{ for all } n \in \{0, 1, 2, ...N\}, \tag{1.3}$$

which can be satisfied for all real-world financial derivatives. It follows to define the auxiliary stopping problems

$$V_n := \sup_{\tau \in \mathcal{T}_n} \mathbb{E}g(\tau, X_\tau) \quad n \in \{0, 1, ..., N\}, \tag{1.4}$$

where $\mathcal{T}_n$ is a subset of $\mathcal{T}$ given by

$$\mathcal{T}_n := \{\tau \in \mathcal{T} : n \leq \tau \leq N\}.$$

In other words, $\mathcal{T}_n$ is the set of all X-stopping times taking values in $\{n, n+1, ..., N\}$
Remark: It is common to simulate S_{t_n} with approximation, for example by the Euler-Maruyama scheme with equidistant time-step. Note the partition for simulation and the partition by the exercise dates are essentially different. However, when the discretisation is made to approximate an American type derivative, the two partition can be the same. In addition, The Markov assumption can be general in the sense that any non-Markovian process can be transformed into Markovian one if only we include all the relevant past information into the current state X (enlarging the state space).

Analytically, the optimal stopping time can be explicitly given by constructing a Snell envelope of the process $(g(n, X_n))_{n\geq 0}^N$ and then determine the time point when the process $(g(n, X_n))_{n\geq 0}^N$ hits the Snell envelop. As the Snell envelop theory is the backbone guiding us to solve the problem numerically, the theorem and definitions involved are included in the Appendix A.

The remainder of this dissertation is organized as follows. Section 2 reviews the algorithms proposed in the representative past papers approaching similar problems, explains the connection between the past algorithms and the deep learning algorithm proposed in the main reference Becker et al. [2018], which solved the same problem using the deep learning algorithm. Section 3 introduce the backward induction algorithm in details. In addition, an alternative design of the loss function and architecture is introduced, which provides the one-shot algorithm that is more appropriate for pricing American type derivatives with significantly less computational costs and a necessary continuity constraint on the approximated Snell envelope. Section 4 applies the two algorithms on both a non-Markovian case (one-dimensional fractional Brownian motion) and a Markovian case (Bermudan Max-Call option) with a focus on models' interpretation and comparison.

The fractional Brownian motion is a continuous process case will be made Markovian by enlarging the state space as many as time step since the evolution is correlated with the history. This example is typically high-dimensional when the time step is large and is beyond the capability of traditional method, thus emphasizes the advantage of using neural network. The latter case is Markovian and based on risk-neutral Black-Schole assumption. In order to interpret the black-box neural network, we will focus more on the 2-D case for the sake of visualization of the prediction of the neural network. The sampling methods to improve generalizability of the models are discussed in subsection 4.2.1.

Section 5 illustrates the calculation of the lower and upper bound. The challenges of this task lies in how to estimate the conditional expectation of the optimal continuation payoff with less computational resource. Our attempt uses one single neural network for regression, including time as a variable, which achieves reliable results for the studied case. The final section 6 outlines the possible future work regarding simplification of the working architecture by knowledge

maturity T.

distillation, which was first motivated to simplify a cumbersome ensemble of models (Hinton et al. [2015]) and widely used in industry to achieve faster inference without losing generalization ability.

2 Literature Review

Theorem A.2 of the Snell Envelop indicates theoretically the optimal stopping time, which is given by the first time our current payoff meets the optimal continuation payoff. Much literature about this topic including the work by Becker et al. [2018] where they use a backward induction scheme that follows the construction of the Snell envelope and involves calculating a series of conditional expectations backward from N (follows the construction of the Snell envelope).

One may consult oksendal [2014] for the properties of conditional expectations under Markov assumption and also the Doob-Dynkin's Lemma (see e.g. Taraldsen [2018]) for the characterization

$$\mathbb{E}(g(\tau_n, X_{\tau_n})|\sigma(X_n)) = \mathbb{E}(g(\tau_n, X_{\tau_n})|X_n) = h_n(X_n),$$

where $\tau_n \in \mathcal{T}_n$ and a measurable function $h_n : \mathbb{R}^d \to \mathbb{R}$, exists with (a.e.) uniqueness. If one follows Kolmogorov's definition of conditional expectation in Brookes [1951], then Doob-Dynkin's lemma is not used. Note the function h_n does not always have a closed-form expression for a wide range of discounted payoff function g based on a variety of underlying assets, which motivates lots of academics to approximate h using Monte Carlo simulation.

It was once a prevailing belief that the American style financial derivatives can not be efficiently priced by simulation because during the later 1980s in both the industrial and academic world, a conditional expectation was usually estimated by simulating abundant continuation trajectories from the same starting point $X_n = x$ at a certain time point n. One would probably need to repeat this process on many starting points which forms a sufficiently fine grid to approximate the target function $h_n(x)$. Using this method to solve an optimal stopping problem or pricing American-Style derivatives would require such a grid for every time point, which was too computationally expensive and thus considered infeasible.

A classic literature (Tilley [1993]) dispelled this belief and proposed an algorithm that gave a reliable estimate with 5,040 samples simulated from one initial point for a one-dimension case. The algorithm points out that the continuation payoff for X_n in a small region around x should be alike, therefore, their average payoff of continuation could be an estimate of the function value $h_n(x)$. The shortcoming of this algorithm is that the samples are grouped according to predefined rules, and do not account for the dissimilarity between the samples in the same group (a sample closer to x may be more important). A modification to this crude average was proposed by Carriere [1996], where two estimators using local polynomial regression and q-splines were used to estimate the target $h_n(x)$. The objective function to minimize is given by the definition of conditional expectation that for $g(\tau_n, X_{\tau_n}) \in \mathbf{L}^2(\Omega, \mathcal{F}_{\tau_n}, \mathbf{P})$, $h_n(X_n)$ is a orthogonal projection of $\mathbf{L}^2(\Omega, \mathcal{F}_{\tau_n})$ onto its subspace $\mathbf{L}^2(\Omega, \mathcal{F}_n)$, which can be found by 'least-square' best predictor. The proof of this is omitted and can be found by Williams [1991]. The quadratic spline algorithm features 'knots' which are the representative data points chosen in the state space $\mathbb{R}^d$, and in the context of estimating $h_n(x)$, the knots are chosen from simulated samples at time point n. The number and positions of knots may directly affect the accuracy of the approximation. In contrast, local polynomial regression does not require such a decision instead uses every sample point in the training and predicts $h_n(x_n)$ as the intercept of the fitted regression, which assigns different sample weights to the data point x_i around x depending on the Euclidean distance between x_i, x. This algorithm essentially gives a weighted sum of the data points nearby and allows different levels of smoothness depending on the kernel function that determines the weights. A more detailed description of the local polynomial regression can

be referred to Cleveland and Devlin [1988]. These regression-based approaches show satisfying goodness of fit in low dimensions, but when the state space is enlarged because of either increasing underlying assets or incorporating historical data, the number of samples required for fitting a regression might increase exponentially to the extent beyond the computational capacity of the algorithm. The demand for solving higher dimension problems motivates the use of deep learning algorithms together with mini-batch and gradient descent optimization.

Besides the estimates for the Snell envelop, it is also of academic and practical interest to give a valid confidence interval. The methodology follows noisy estimation, consult e.g. Kogan and Haugh [2001];Rogers [2002];Andersen and Broadie [2004].

3 Algorithm Explanation

This section begins with justifying that an optimal stopping time can be formulated as a product of decision sequence, then introduce basic notions in neural networks. After that, two possible designs of the loss function are explained. Finally, we summarize the implementation procedures of the two algorithms, which are called backward induction algorithm and one-shot algorithm respectively.

3.1 Optimal Stopping Rules

In this subsection, we first explain the idea of formulating a stopping time from the perspective of the Snell envelope theory, then we introduce a proof provided by Becker et al. [2018] which does not use the notion of Snell envelope instead use Mathematical induction. As the Snell envelope theory basically requires one to check if the current payoff meets the optimal payoff of continuation recursively at each time point n, if satisfied, stops immediately. Following the notation as done by Becker et al., for given $n \in \{0, 1, ..., N\}$ and a sequence of measurable functions $f_n : \mathbb{R}^d \to \{0, 1\}$ to indicate the decision of such a checking procedure, and a random variable $h_n(X_n) : \Omega \to \mathbb{R}$ to denote the optimal payoff of continuation $\mathbb{E}(g(\tau_{n+1}, X_{\tau_{n+1}})|X_n), \tau_{n+1} \in \mathcal{T}_{n+1}$, as justified by Doob-Dynkin 's theorem

$$f_n(x) = \begin{cases} 1 & g(n, x) \geq h_n(x) \\ 0 & g(n, x) < h_n(x) \end{cases}.$$

As formulated in Section 1, N is the terminal time when the Bermudan type derivative can be exercised. That is the decision function $f_N \equiv 1$. Therefore, the optimal stopping time can be expressed as a product of such successive decisions. Becker et al. [2018] rigorously proves this by mathematical induction without the construction of the Snell envelope, and starts their proof generally assuming τ_{n+1} to be an arbitrary stopping time of continuation in $\mathcal{T}_{n+1}$, instead of the optimal stopping time of continuation described here but will be revealed to be consistent at the end of the proof. For completeness, it is included in details with minor modification.

Theorem 3.1. *(Theorem 1 from Becker et al. [2018]) For a given $n \in \{0, 1, ..., N-1\}$, suppose an arbitrary stopping time $\tau_{n+1} \in \mathcal{T}_{n+1}$ of the form*

$$\tau_{n+1} = \sum_{m=n+1}^{N} m f_m(X_m) \prod_{j=n+1}^{m-1} (1 - f_j(X_j))$$

for a series of measurable functions $f_{n+1}, ..., f_N : \mathbb{R}^d \to \{0, 1\}$ with $f_N \equiv 1$. Then there exists a measurable function $f_n : \mathbb{R} \to \{0, 1\}$ such that the stopping time $\tau_n \in \mathcal{T}_n$ given by [3]

$$\tau_n = \sum_{m=n}^{N} m f_m(X_m) \prod_{j=n}^{m-1} (1 - f_j(X_j)) \tag{3.1}$$

satisfies

$$\mathbb{E}g(\tau_n, X_{\tau_n}) \geq V_n - (V_{n+1} - \mathbb{E}g(\tau_{n+1}, X_{\tau_{n+1}})),$$

where we denote the difference between the optimal payoff of continuation V_{n+1} and the expected payoff $\mathbb{E}g(\tau_{n+1}, X_{\tau_{n+1}})$ by

$$\varepsilon_{n+1} := V_{n+1} - \mathbb{E}g(\tau_{n+1}, X_{\tau_{n+1}}).$$

Clearly, ε_{n+1} is non-negative by definition, the target inequality is equivalent to $\varepsilon_{n+1} \geq \varepsilon_n$.

Proof. First of all, the form of τ_{n+1} indeed formulate a stopping time as $\{\tau_{n+1} = m\} = \{f_m(X_m) \prod_{j=n+1}^{m-1}(1 - f_j(X_j)) = 1\}$ is a measurable set in $\mathcal{F}_m$, since f_j is $\mathcal{F}_m$-measurable for

[3] To make this form well-defined, we understand the empty product $\prod_{j=n}^{m-1}(1 - f_j(X_j))$ as 1

$j = n+1, ..., m$, so is their product. $g(\tau_{n+1}, X_{\tau_{n+1}})$ is therefore $\mathcal{F}_m$-measurable, where m is given by $\inf\{j \geq n+1 : f_j(X_j) = 1\}$. By the Markov property of X, and the Doob-Dynkin lemma, there exist a measurable function $h : \mathbb{R}^d \to \mathbb{R}$ such that $\mathcal{F}_n$-measurable random variable $h_n(X_n)$ satisfies

$$\mathbb{E}(g(\tau_{n+1}, X_{\tau_{n+1}})|\mathcal{F}_n) = \mathbb{E}(g(\tau_{n+1}, X_{\tau_{n+1}})|X_n) = h_n(X_n).$$

It follows that the event $D = \{g(n, X_n) \geq h_n(X_n)\} \in \mathcal{F}_n$, and for an arbitrary stopping time $\tau \in \mathcal{T}_n$, $E = \{\tau = n\} \in \mathcal{F}_n$ by definition of stopping time. We can construct

$$\tau_n = n\mathbf{1}_D + \tau_{n+1}\mathbf{1}_{D^c} \in \mathcal{T}_n,$$
$$\tilde{\tau} = \tau_{n+1}\mathbf{1}_E + \tau\mathbf{1}_{E^c} \in \mathcal{T}_{n+1} .$$

It follows that

$$\begin{aligned}
&\mathbb{E}[g(\tau_{n+1}, X_{\tau_{n+1}})(\mathbf{1}_E + \mathbf{1}_{E^c})] \\
&= \mathbb{E}(g(\tau_{n+1}, X_{\tau_{n+1}})) \\
&= V_{n+1} - \varepsilon_{n+1} && \text{(By definition of } \varepsilon_{n+1}) \\
&\geq \mathbb{E}(g(\tilde{\tau}, X_{\tilde{\tau}})) - \varepsilon_{n+1} && \text{(By definition of } V_{n+1}) \\
&= \mathbb{E}[g(\tilde{\tau}, X_{\tilde{\tau}})(\mathbf{1}_E + \mathbf{1}_{E^c})] - \varepsilon_{n+1}.
\end{aligned}$$

Since $\mathbb{E}(g(\tau_{n+1}, X_{\tau_{n+1}})\mathbf{1}_E) = \mathbb{E}(g(\tilde{\tau}, X_{\tilde{\tau}})\mathbf{1}_E)$, we have

$$\mathbb{E}[g(\tau_{n+1}, X_{\tau_{n+1}})\mathbf{1}_{E^c}] \geq \mathbb{E}[g(\tilde{\tau}, X_{\tilde{\tau}})\mathbf{1}_{E^c}] - \varepsilon_{n+1} = \mathbb{E}[g(\tau, X_\tau)\mathbf{1}_{E^c}] - \varepsilon_{n+1}.$$

It follows that

$$\begin{aligned}
&\mathbb{E}g(\tau_n, X_{\tau_n}) \\
&= \mathbb{E}[g(n, X_n)\mathbf{1}_D + g(\tau_{n+1}, X_{\tau_{n+1}})\mathbf{1}_{D^c}] \\
&= \mathbb{E}[g(n, X_n)\mathbf{1}_D + h_n(X_n)\mathbf{1}_{D^c}] && \text{(Since } D^c \in \mathcal{F}_n \text{ by Partial Averaging property)} \\
&= \mathbb{E}[max(g(n, X_n) + h_n(X_n))] && \text{(By definition of the set D)} \\
&= \mathbb{E}[max(g(n, X_n) + h_n(X_n))(\mathbf{1}_E + \mathbf{1}_{E^c})] \\
&\geq \mathbb{E}[g(n, X_n)\mathbf{1}_E + h_n(X_n)\mathbf{1}_{E^c}] && \text{(Since } E^c \in \mathcal{F}_n \text{ by Partial Averaging property)} \\
&= \mathbb{E}[g(n, X_n)\mathbf{1}_E + g(\tau_{n+1}, X_{\tau_{n+1}})\mathbf{1}_{E^c}] \\
&\geq \mathbb{E}[g(n, X_n)\mathbf{1}_E + g(\tau, X_\tau)\mathbf{1}_{E^c}] - \varepsilon_{n+1}. && \text{(By the inequality above)} \\
&= \mathbb{E}g(\tau, X_\tau) - \varepsilon_{n+1}. && (\tau = n\mathbf{1}_E + \tau\mathbf{1}_{E^c})
\end{aligned}$$

Since $\tau \in \mathcal{T}_n$ is arbitrary, we have $\mathbb{E}g(\tau_n, X_{\tau_n}) \geq V_n - \varepsilon_{n+1}$, that is $\varepsilon_{n+1} \geq \varepsilon_n$, if and only if we define $f_n(X_n) = \mathbf{1}_D$, that is

$$f_n(x) = \begin{cases} 1 & g(n, x) \geq h_n(x) \\ 0 & g(n, x) < h_n(x) \end{cases}.$$

□

Proposition 1. *(Remark 3 in Becker et al. [2018]) The formulation of τ_n given by*

$$\tau_n = \sum_{m=n}^{N} m f_m(X_m) \prod_{j=n}^{m-1} (1 - f_j(X_j))$$

is an optimal stopping time in $\mathcal{T}_n$.

Proof. Since $f_N \equiv 1$, the stopping time formulated as above $\tau_N = N \cdot f_N(X_N) = N$, we have

$\varepsilon_N = V_N - g(\tau_N, X_{\tau_N}) = 0$. By Theorem 3.1, $0 = \varepsilon_N \geq \varepsilon_{N-1} \geq ...\varepsilon_0 \geq 0$, we conclude $\varepsilon_n = V_n - \mathbb{E}g(\tau_n, X_{\tau_n}) = 0$, that is

$$\mathbb{E}g(\tau_n, X_{\tau_n}) = \sup_{\tau \in \mathcal{T}_n} \mathbb{E}g(\tau, X_\tau),$$

which concludes the proof. □

Remark: In many financial examples (e.g. pricing), the Markov process starts from a deterministic initial value x_0, the corresponding decision $f_0(x_0)$ would be a deterministic constant in $\{0, 1\}$. Therefore, at time 0, it does not require a model for estimation. If so, however, one has to calculate the initial payoff manually and compare it to the estimated continuation payoff. In this dissertation, we also fit a model for time 0 with a designed sampling method because we aim at increasing the models' automaticity and generalizability.

3.2 Introduction of Neural network

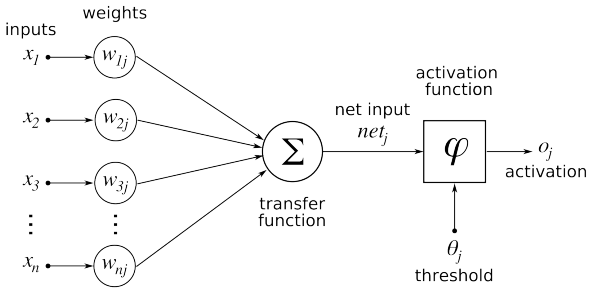

Figure 1: A sketch of a basic unit in neural network - Neuron. Sourced from internet

3.2.1 Neuron and Layer

Without presuming a reader to be familiar with the notion of the neural network, a brief introduction is included here. Figure 1 shows the sketch of a neuron - the most fundamental unit of a neural network, which is named because of its similar structure as a biological neuron. A neuron[4] j can be seen as a composite of its affine functions $a_j : \mathbb{R}^d \to \mathbb{R}$ and activation $\varphi : \mathbb{R} \to \mathbb{R}$. Some activation functions have trainable hence neuron-specific parameters, while in our implementation none of the activation functions have such parameters, therefore, no subscript j is used here.

An affine function a_j takes input data[5] $x \in \mathbb{R}^d$, and calculates the weighted sum, which is often added to a trainable offset w_{0j} before passing to the activation function φ. The common choices of activation function are sigmoid function and ReLu function.
Sigmoid function is given by

$$S(x) = \frac{1}{1 + e^{-x}},$$

which has a range $(0, 1)$, this feature makes it favorable when the prediction is imitating a probability or something that has the same range. In the implementation of the algorithm, a sigmoid function is chosen to be the activation function in the combined layer for the same reason.
ReLu (Rectified Linear Unit, consult Agarap [2018]) is given by,

$$R(x) = max(x, 0).$$

While one such neuron might be too simple to fit the data, numerous neurons working together with the features of truncation at 0 allows the neural network to obtain non-linearity in the state space $\mathcal{X}$, which could make a non-linear[6] transformation on the original data. One perspective to understand this transformation is that the neurons are learning to construct more informative features. Another favorable advantage is the fact that the product of such neuron does not vanish because the gradient is either 0 or 1.

A layer of neurons may contain as many neurons as one designs it, with each neuron as a composite function $\varphi \circ a_j$. When several layers work together, the neural network is able to make complex transformations on input data x, and for instance, when the objective is a binary classification or clustering case, the transformed state space could be (almost) linearly separable. For a simpler notation, we can denote the mathematical operations of all neurons in a layer I by an operator $\varphi_{q_I} \circ a_I$, where q_I is the number of neurons in the layer I, $a_I : \mathbb{R}^{q_{I-1}} \to \mathbb{R}^{q_I}$ is all the affine operations performed individually by each neuron, note the dimension q_{I-1} here

[4] j in the description of Neuron means j th neuron in a layer.
[5] To match the notation, we use d-dimension input for description, instead of n in the sketch.
[6] When the activation function is linear, the operation will be linear

implies the layer I takes as many inputs as the number of neurons in the last layer[7].

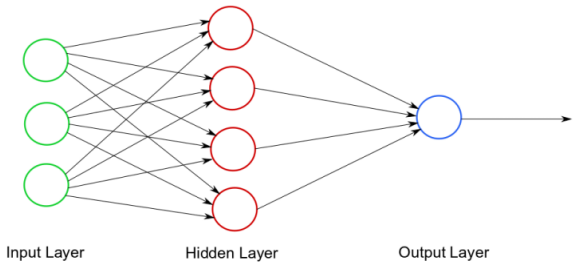

Figure 2: A sketch of Neural network with one hidden layer, and one Output layer. The input layer shall match the dimension of the state space, and the neurons in each hidden layer take all the outputs from the last layer independently (The bias of each neuron is omitted in the sketch).

With a brief description of these key notions, we can now describe how to approximate the measurable function f_n defined in 3.1 that essentially checks the Snell envelop condition. As specified in the next subsection, a neural network updates its parameter by gradient descent, therefore, introducing an indicator function that is not everywhere-differentiable to directly imitate the output of f_n - $\{0, 1\}$ is infeasible. The workaround is to use the Sigmoid function $S(x)$ as it has a similar gradient compared to an indicator function while being differentiable everywhere. The sequential neural network of the same architecture $F^\theta \to (0, 1)$ is used to approximate the measurable function f_n, and can be formulated as,

$$F^\theta = S \circ a_I^\theta \circ R \circ a_{I-1}^\theta \circ ... \circ R \circ a_1^\theta,$$

where

• I refers to the depth of the neural network, that is the number of layers including both hidden layer and output layer. $q_1, q_2, ..., q_{I-1}, q_I$ refers to the number of neurons in each layer. Particularly, in the output layer, we have only one neuron, so $q_I \equiv 1$. For consistency of expression, One may regard the input layer as many void neurons as the dimension of state space $\mathcal{X}$ which does nothing but pass the input variate to the hidden layer, then we can define $q_0 \equiv D$. $\theta \in \mathbb{R}^q$ refers to the set of parameters of all layers with $q = \sum_{j=1}^{I} q_j + \sum_{j=0}^{I-1} q_j q_{j+1}$. The first summation is for the bias parameter b, while the second is for the weights in A_j.

• $a_j^\theta : \mathbb{R}^{q_{j-1}} \to \mathbb{R}^{q_j}, j = 1, 2, ..., I$ refers to the affine function in the j th hidden layer. To be specific, $a_j^\theta(z_{j-1}) = A_j z_{j-1} + b_j$, where $b_j \in \mathbb{R}^{q_j}$ is the bias of jth layer, $A_j \in \mathbb{R}^{q_j \times q_{j-1}}$ is a matrix, where each row has the weights of shape $(1, q_{j-1})$ of a neuron, given to the q_{j-1} outputs z_{j-1} from the last layer $j - 1$. Particularly, the output from the input layer is the input itself, i.e. $z_0 = x \in \mathbb{R}^d$

• As specified above, $R(\cdot), S(\cdot)$ refers to ReLu and Sigmoid activation. The Sigmoid has value range $(0, 1)$ and mimics an indicator function. It is worth mentioning that a ReLu neuron can be sensitive to the covariate shift. Covariate shift refers to the changes of the distribution of the training data, which is very common when Mini-batch is used, therefore ReLu is often used with batch-normalization in practice.

3.2.2 Neural Network Training

As mentioned before, the neural network learns to construct features by a series of non-linear transformations on the original input, which is achieved by multiple layers combined and nu-

[7]Particularly, the input layer can be seen to have only void neurons, which takes data input and pass them to the hidden layer without any operation. And in this dissertation, the output layer has only one sigmoid neuron

merous neurons in a layer. It has been infeasible to design such a complex network until the advent of back-propagation in Rumelhart et al. [1986]. Before understanding this notion, we need to understand 'Forward-propagation' and 'Loss function'.

In general, loss function $\mathcal{L}$ is a metric to measure how well a model fits the training data. For supervised learning where we know the true labels[8], loss function may take the form $\mathcal{L} : \mathbb{R}\times\mathbb{R} \to \mathbb{R}$, which measures the difference between our prediction and the corresponding true label (e.g. $(y_i - \hat{y}_i)^2$ for regression). For unsupervised learning where we have no such label in training data, we can custom a metric to optimize, for example, the distance between clusters in a clustering task. The total loss would be a weighted average of the loss of each observation. When every observation has an equal sample weight, then it is an arithmetic average. The 'Forward-propagation' is the process of taking each data input and evaluating each layer sequentially.

Back-propagation is the essence of the training, by which the neural network derives the gradients with respect to the parameters in each layer and updates the parameter along the gradient. Mathematically, it is the result of the chain rule. This efficient gradient calculation allows the wide use of the gradient descent method, which updates the parameter set along the gradient. As stochastic gradient descent can be sensitive to the choice of the learning rate and takes possibly a long time to find minima. An alternative optimizer is employed in programming named Adam optimizer Kingma and Ba [2014], which indirectly updates the parameter set, based on the first moment and second-moment estimate. This approach is more robust against noisy data and in general, achieves a convergence faster. A brief description is given below for completeness.

Algorithm 1 Adam Optimization Algorithm:
The method adapts learning rates for each parameter individually based on the estimated first and second moments of the gradients. Let g_t denote the gradient of the loss function w.r.t the parameters, $g_t^2 := g_t \odot g_t$ denote the element-wise square of the gradient.

Require: α: Stepsize, Default value as 0.001, larger stepsize may lead to faster convergence when appropriate.
Require: $\beta_1, \beta 2 \in [0, 1)$, Exponential decay rates for the moment estimates. Default value as suggested by a wide range of tested problems: $beta_1 = 0.9, \beta_2 = 0.999$
Require: $\mathcal{L}(\theta; x)$, the loss function of the parameters θ, evaluated on the given data x.
Require: $\theta^{(0)} \in \mathbb{R}^q$, an initial parameter vector. The index (t) in $\theta^{(t)}$ keeps track of how many times the parameters have been updated.
$m_0 \leftarrow 0$ (Initialize the first moment vector of the gradient)
$v_0 \leftarrow 0$ (Initialize the second moment vector of the gradient)
$t \leftarrow 0$ (Initialize a time point)
while $\theta^{(t)}$ not converged **do**
$t \leftarrow t + 1$
$g_t \leftarrow \nabla_\theta L(\theta^{(t-1)}; x_t)$ (By backpropagation, derive the gradients w.r.t the loss function)
$m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
$v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
$\hat{m}_t \leftarrow m_t/(1 - \beta_1^t)$ (Correct the bias of the estimate; β^t refers to β to the power of t)
$\hat{v}_t \leftarrow v_t/(1 - \beta_2^t)$
$\theta^{(t)} \leftarrow \theta^{(t-1)} - \alpha \cdot \hat{m}_t/(\sqrt{\hat{v}_t} + \varepsilon)$ (Update the parameter; The ε term defaults to 10^{-8}, to prevent the denominator being too close to zero.)
end while

---

to give a prediction.
[8]True labels can be a class in the classification task or a value in the regression task.

3.3 Design of the loss function

The design of the loss function is not unique, depending on how one characterizes the condition of optimal stopping. The two designs of the loss functions to be introduced in this subsection could be seen as the reward-oriented, which uses the logic that if the decision functions approximated by neural networks are ideal, then the associated payoff or reward should approximate the true maximal value assuming the sample size is sufficiently large as required by the law of large numbers. Specifically, the rate of convergence is studied by Belomestny [2009].

3.3.1 Backwards Induction Scheme

To approximate an optimal stopping time formulated as equation 3.1, we construct in total N decision functions approximated by the neural networks $F^{\theta_n}, n = 0, 1, ..., N-1$ with F^{θ_n} corresponding to the optimal decision function f_n. As the output of F^{θ_n} is within $(0, 1)$, after the training of F^{θ_n}, we can modify it to $\{0, 1\}$ to eliminate the difference.

We define the modified version of the F^{θ_n} as $f^{\theta_n} : \mathbb{R}^d \to \{0, 1\}, n = 0, 1, ..., N-1$, given by,

$$f^{\theta_n}(x) = \begin{cases} 1 & F^{\theta_n} \geq 0.5 \\ 0 & F^{\theta_n} < 0.5 \end{cases}.$$

As we use sigmoid function $S(\cdot)$ as the final activation which strictly increases on $\mathbb{R}$ and $S(0) = 0.5$, the definition of f_n is equivalent to replacing the sigmoid activation of the neuron in the final layer with an indicator function, that is

$$f^{\theta_n} := \mathbf{1}_{[0,\infty)} \circ a_I^{\theta_n} \circ R \circ a_{I-1}^{\theta_n} \circ ... \circ R \circ a_1^{\theta_n}$$

while

$$F^{\theta_n} = S \circ a_I^{\theta_n} \circ R \circ a_{I-1}^{\theta_n} \circ ... \circ R \circ a_1^{\theta_n}.$$

Therefore, the modified $f^{\theta_n}, n = 0, 1, ..., N-1$ can still be regarded as a neural network, even though it is not trainable. We will notice in the examples, that this change will not impact the performance of a well-trained neural network, as it already gives predictions extremely close to either 1 or 0.
The associated expected payoff at each time point $n \leq N-1$ is given by

$$\mathbb{E}g(\tau_\Theta, X_{\tau_\Theta}) = \mathbb{E}[g(n, X_n)f^{\theta_n}(X_n) + g(\tau_{n+1}^\Theta, X_{n+1})(1 - f^{\theta_n}(X_n))], \tag{3.2}$$

where $\Theta := (\theta_0, \theta_1, \theta_2, ..., \theta_{N-1}) \in \mathbb{R}^{(N-1)q}$ is a vector of all the parameters of in total N neural network[9]. τ_{n+1}^Θ is the optimal stopping time of continuation indicated by the neural networks formulated as

$$\tau_{n+1}^\Theta = \sum_{m=n+1}^{N} m f^{\theta_m}(X_m) \prod_{j=n+1}^{m-1} (1 - f^{\theta_j}(X_j)).$$

The theorem of the Snell envelope in Appendix A reveals that these formulations aim to mimic the expected payoff of the Snell envelope. However, it remains to show that the neural network indeed has the flexibility to achieve a similar payoff. This is justified by the following proposition.

Proposition 2. *(Proposition 4 by Becker et al. [2018]) For $n \leq N-1$ and an arbitrary stopping time of continuation $\tau_{n+1} \in \mathcal{T}_{n+1}$. Then for every depth $I \geq 2$ and constant $\varepsilon > 0$, there exist*

[9] Each neural network has the same architecture with q parameters; Only N neural networks are fitted, as the terminal decision is deterministic.

positive integers $q_1, q_2, ...q_{I-1}$ *such that,*

$$\sup_{\theta \in \mathbb{R}^q} \mathbb{E}[g(n, X_n) f^\theta(X_n) + g(\tau_{n+1}, X_{n+1})(1 - f^\theta(X_n))]$$
$$\geq \sup_{f \in \mathcal{I}} \mathbb{E}[g(n, X_n) f + g(\tau_{n+1}, X_{n+1})(1 - f(X_n))] - \varepsilon,$$

where $\mathcal{I}$ *is the set of all measurable functions that acts as an indicator function* $f : \mathbb{R}^d \to \{0, 1\}$

Proof. $\forall \varepsilon \geq 0$, under the assumed integrability condition $g(n, X_n) \in L^1(\Omega, \mathcal{F}_n, \mathbb{P})$, there must exist a measurable function $\tilde{f} \in \mathcal{I} : \mathbb{R} \to \{0, 1\}$ such that,

$$\mathbb{E}[g(n, X_n) f(\tilde{X}_n) + g(\tau_{n+1}, X_{n+1})(1 - \tilde{f}(X_n))]$$
$$\geq \sup_{f \in \mathcal{I}} \mathbb{E}[g(n, X_n) f + g(\tau_{n+1}, X_{n+1})(1 - f(X_n))] - \varepsilon/4.$$

Again, by the integrability condition, we can define two Borel measures $u, v : \mathcal{B}(\mathbb{R}^d) \to \mathbb{R}$,

$$u(B) = \mathbb{E}[|g(n, X_n)| \mathbf{1}_B(X_n)]$$
$$v(B) = \mathbb{E}[|g(\tau_{n+1}, X_{\tau_{n+1}})| \mathbf{1}_B(X_n)],$$

for which the countably additive condition can be easily verified, and $u(\phi), v(\phi) = 0$. u, v being finite Borel measures implies they are tight measures, it follows that there exists a compact subset $K \subseteq \tilde{f}^{-1}(1)$ such that,

$$\mathbb{E}[g(n, X_n) \mathbf{1}_K(X_n) + g(\tau_{n+1}, X_{\tau_{n+1}})(1 - \mathbf{1}_K(X_n))]$$
$$\geq \mathbb{E}[g(n, X_n) \tilde{f}(X_n) + g(\tau_{n+1}, X_{n+1})(1 - \tilde{f}(X_n))] - \varepsilon/4.$$

Define the distance between a point x and the set K by $\rho_K = \inf_{y \in K} \|x - y\|_2$. Note if $x \in K$, the distance is zero, otherwise non-negative. Construct a sequence of continuous functions $k_j : \mathbb{R}^d \to [-1, 1] \quad j \in \mathbb{N}$ given by,

$$k_j(x) = max\{1 - j\rho_K(x), -1\}, \quad j \in \mathbb{N}.$$

Note as $j \to \infty$, if $x \in K$, $k_j(x) = \mathbf{1}_{x \in K} = 1$, if $x \notin K$, $k_j(x) = -\mathbf{1}_{x \in K^c} = -1$. That is, k_j converges pointwise to $\mathbf{1}_{x \in K} - \mathbf{1}_{x \in K^c}$. By the dominated convergence theorem, $\forall \varepsilon, \exists j \in \mathbb{N}$ such that,

$$\mathbb{E}[g(n, X_n) \mathbf{1}_{k_j(X_n) \geq 0} + g(\tau_{n+1}, X_{\tau_{n+1}})(1 - \mathbf{1}_{k_j(X_n) \geq 0})]$$
$$\geq \mathbb{E}[g(n, X_n) \mathbf{1}_K(X_n) + g(\tau_{n+1}, X_{\tau_{n+1}})(1 - \mathbf{1}_K(X_n))] - \varepsilon/4.$$

As k_j is uniformly bounded, $k_j \in L^1(u), L^1(v)$ following $g(n, X_n) \in L^1(\Omega, \mathcal{F}_n, \mathbb{P})$, we can apply the Theorem 1 in Leshno et al. [1993], which rigorously stated k_j can be uniformly approximated on compact sets by functions of the form

$$\sum_{i=1}^{r} (v_i^T x + c_i)^+ - \sum_{i=1}^{s} (w_i^T x + d_i)^+,$$

for $r, s \in \mathbb{N}, v_1, ..., v_r, w_i, ..., w_s \in \mathbb{R}^d$. Note this form is expressible by a composite of affine functions and ReLu activations. That is k_j can be uniformly approximated by a neural network of the form f^θ as defined above. Therefore, we have

$$\mathbb{E}[g(n, X_n) f^\theta(X_n) + g(\tau_{n+1}, X_{\tau_{n+1}})(1 - f^\theta(X_n)]$$
$$\geq \mathbb{E}[g(n, X_n) \mathbf{1}_{k_j(X_n) \geq 0} + g(\tau_{n+1}, X_{\tau_{n+1}})(1 - \mathbf{1}_{k_j(X_n) \geq 0})] + \varepsilon/4,$$

where f^θ as defined is a depth I neural network, including at least one hidden layer and one

output layer ($I \geq 2$). As f^θ itself is not directly trainable due to the indicator function, in practice, however, we often need more layers and large enough neurons. Summarize the above inequality, we get

$$\begin{aligned}&\mathbb{E}[g(n, X_n)f^\theta(X_n) + g(\tau_{n+1}, X_{\tau_{n+1}})(1 - f^\theta(X_n)] \\ &\geq \mathbb{E}[g(n, X_n)\mathbf{1}_{f(X_n)\geq 0} + g(\tau_{n+1}, X_{\tau_{n+1}})(1 - \mathbf{1}_{f(X_n)\geq 0})] + \varepsilon.\end{aligned}$$

□

By Theorem 3.1 and Proposition 2, we can conclude that a stopping time τ_Θ indicated by N neural networks $f^{\theta_n}, n = 0, 1, ..., N-1$ can arbitrarily approximate the optimal stopping payoff, which concludeds the following corollary.

Corollary 1. *For a stopping problem described in (1.1), provided depth $I \geq 2$, enough neurons $q_0, q_1, ..., q_{I-1}$, there exists a parameter vector Θ of N neural networks such that the stopping time τ_Θ controlled by the neural network satisfies*

$$\mathbb{E}g(\tau_n^\Theta, X_{\tau_n^\Theta}) \geq \sup_{\tau_n} \mathbb{E}g(\tau_n, X_{\tau_n}) - \varepsilon, \quad \forall \varepsilon \geq 0,$$

where

$$\tau_n^\Theta = \sum_{m=n}^{N} m f^{\theta_m}(X_m) \prod_{j=n}^{m-1} (1 - f^{\theta_j}(X_j)),$$

with $f^{\theta_n}, n = 0, ..., N-1$ given by neural networks and $f^{\theta_N} \equiv f^N \equiv 1$ by problem constraint.

Remark: At the time N, $\tau_N \equiv N$, so it does not require modeling and $f^{\theta_N} \equiv f^N$ are set to 1.

The algorithm could be called 'backward-fitting' as it requires knowing the optimal stopping time of continuation τ_{n+1}^Θ just like the other algorithm, which requires us to fit $f^{\theta_{n+1}}, f^{\theta_{n+2}}, ..., f^{\theta_N}$ first just like the construction of the Snell envelope. In some sense, it is equivalent to approximating the Snell envelope condition, i.e. checking if the current payoff is larger than the optimal continuation payoff. However, the algorithm deviates from the historical approaches under the backward induction scheme as it does not estimate the optimal continuation payoff but directly approximates the decision.

There are several motivations to deviate from this scheme. One is that when we try to approximate American type derivatives with a large N, the N neural networks can cost extremely long time for training as we train every neural network with sufficiently large epochs to ensure convergence of parameters. Another motivation is under this scheme, the N neural networks are trained independently by stochastic optimizers, which tends to overfitts the pattern of simulated (noisy) data[10]. In a 2-D case, we could effectively monitor the symptoms of overfitting by visualization of the decision boundary. At time point n, the decision boundary is given by

$$b_n := \{x \in \mathbb{R}^d : g(n, x) = \mathbb{E}[g(\tau_{n+1}, x_{\tau_{n+1}})|X_n = x]\}, \quad n \in \{0, 1, ..., N\},$$

where $\tau_{n+1} \in \mathcal{T}_{n+1}$ is the optimal stopping of continuation (consult Peskir and Shiryaev [2006]), which in a wide range of examples is a hypersurface that partitions the state space into strictly stopping region and strictly continuing region, while the points on the boundary see equal current payoff and equal continuation payoff. The evolution of the decision boundary has been studied by Peskir [2018], De Angelis and Stabile [2017], who point out that the decision boundary evolves continuously with respect to time in America style derivatives. If that is the case, then it implies

[10]The noise may come from actually two sources. One is the randomness of stochastic optimizers themselves, and the other is the noise of data simulation. That is to say, even we train the models on the identical data, there would be no guarantee to yield identical sets of model parameters. And the later implies even though we achieve a global minimum of the loss function for training data, the trained model might overfit the noise of simulation.

that the partial decisions made at neighboring time point t_n, t_{n+1} should be almost identical. Note $F^{\theta_n}, \quad n \in \{0, 1, ..., N\}$ that approximate f_n would behave similarly to

$$S \circ (g(n, x) - h_n(x)), \quad n \in \{0, 1, ..., N-1\},$$

where $S : \mathbb{R} \to (0, 1)$ is the sigmoid function, $h_n : \mathbb{R}^d \to \mathbb{R}$ is a measurable function such that $h_n(X_n)$ is a version of $\mathbb{E}[g(\tau_{n+1}, X_{\tau_{n+1}})|X_n]$ and $\tau_{n+1} \in \mathcal{T}_{n+1}$ refers to the optimal stopping time of continuation. We make an observation here to justify that as $N \to \infty$ the increment $h_{n+1}(x) - h_n(x) \to 0$. Assume that the continuous Markov asset process $(S_t)_{[0,T]}$ is uniquely determined by the initial states and given evolution rules (e.g. a diffusion process uniquely given by initial states, a fBm discussed in the first example), it follows that for almost every $\omega \in \Omega$, there exists $\omega' \in \Omega$ such that the trajectory $(X_m(\omega))_{m \geq n}^{N-1}$ starting with $X_n(\omega) = x$ matches the trajectory $(X_m(\omega'))_{m \geq n+1}^{N}$ starting with $X_{n+1}(\omega') = x$, that is $X_m(\omega) = X_{m+1}(\omega')$ for $m \geq n$ and $n = 0, 1, ..., N-1$.

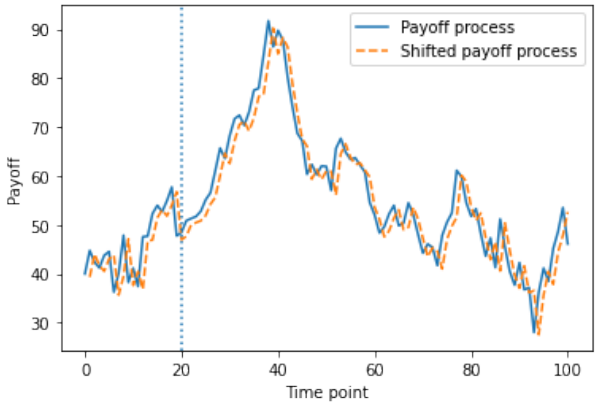

Figure 3: The shifted payoff process that stops at the same time when the original process optimally stops, have similar payoff when the equidistant time step $\frac{T}{N}$ tends to 0.

As shown in Figure 3, the trajectories of the associated payoff process $(g(m, X_m))_{m \geq n+1}^{N}$ corresponding to $(X_m(\omega))_{m \geq n+1}^{N}$ and $(X_m(\omega'))_{m \geq n+1}^{N}$ respectively are almost identical since we assume $(S_t)_{t \in [0,T]}$ is uniquely given by evolution rule and initial states. We can then define a random process $(\mathcal{Z}_n)_{n=1}^{N}$ such that $\mathcal{Z}_{n+1}(\omega) = X_n(\omega), \quad \forall \omega \in \Omega$ for $n = 0, 1, ..., N-1$. By this definition, $(\mathcal{Z}_m)_{m \geq n}^{N} | \mathcal{Z}_n = x$ and $(X_m)_{m \geq n}^{N} | X_n = x$ are equivalent. As we increase N and set equidistant time step such that $t_{n+1} - t_n = \frac{T}{N} \to 0$ for all $n \in \{0, 1, ..., N-1\}$ in order to approximate an American type derivative (with precision given by BECKER et al. [2021]), we have

$$\begin{aligned}
&h_n(x) - h_{n+1}(x) \\
&= \mathbb{E}[g(\tau_{n+1}, X_{\tau_{n+1}})|X_n = x] - \mathbb{E}[g(\tau_{n+2}, X_{\tau_{n+2}})|X_{n+1} = x] \\
&= \mathbb{E}[g(\tau_{n+1}, X_{\tau_{n+1}})|X_n = x] - \mathbb{E}[g(\tau_{n+2}, \mathcal{Z}_{\tau_{n+2}})|\mathcal{Z}_{n+1} = x] \\
&= \mathbb{E}[g(\tau_{n+1}, X_{\tau_{n+1}}) - g(\tau_{n+2}, \mathcal{Z}_{\tau_{n+2}})|X_n = x].
\end{aligned}$$

It follows that

$$RHS \geq 0 \quad \text{since } \mathcal{T}_{n+2} \subseteq \mathcal{T}_{n+1},$$

and

$$\begin{aligned}
&RHS \\
&\leq \mathbb{E}[g(\tau_{n+1}, X_{\tau_{n+1}}) - g(\tau_{n+1}, \mathcal{Z}_{\tau_{n+1}})|X_n = x] \quad (\text{since } \tau_{n+2} \in \mathcal{T}_{n+2} \text{ is optimal}) \\
&= \mathbb{E}[g(\tau_{n+1}, X_{\tau_{n+1}}) - g(\tau_{n+1}, X_{\tau_{n+1}-1})|X_n = x].
\end{aligned}$$

Since sample-path continuity of $(S_t)_{t\in[0,T]}$ implies

$$\lim_{N\to\infty} g(\tau_{n+1}, X_{\tau_{n+1}}) - g(\tau_{n+1}, X_{\tau_{n+1}-1}) = 0 \quad a.s., \quad n \in 0, 1, ..., N-1,$$

and absolute payoff $g(n, X_n)$ is bounded by undiscounted absolute payoff which is always bounded in the financial modelling, we can apply dominated convergence theorem which yields

$$\lim_{N\to\infty} h_n(x) - h_{n+1}(x) = 0 \quad a.s., \quad n \in 0, 1, ..., N-1.$$

It follows that there exist a continuous function $F : [0, T] \times \mathbb{R}^d \to [0, 1]$ such that $F(t_n, x) = S\circ(g(n, x)-h_n(x)), \quad n \in \{0, 1, ..., N-1\}$. The observation is not enough to justify modeling the partial decision function as a function $F(t, x)$ and further approximation by the neural network, however, it can at least serve as a hypothesis and test tool, by which one can construct such a decision boundary, validate the properties, etc. For this purpose, we follow the design of the neural network as done by BECKER et al. [2021] and Reppen et al. [2022] in the next subsection.

3.3.2 One-shot Scheme

As we approximate an American type option with its discretized version by increasing the number of exercise times, i.e., $N \longrightarrow \infty$, for given assets value $x \in \mathbb{R}^d$, the neighbouring time points t_n, t_{n+1} see infinitesimal change in continuation optimal payoff, so do the corresponding partial stopping signals $F^{\theta_n}(t_n, x)$, $F^{\theta_{n+1}}(t_{n+1}, x)$. In order to fulfill this constraint, instead of fitting N independent neural network $F^{\theta_n}, \quad n = 0, 1, ..., N-1$ to give N partial stopping decision, we fit only one neural network, but include real value $t \in [0, T]$ as an additional time variate into the modeling, i.e., $F^{\tilde{\theta}} : [0, T] \times \mathbb{R}^d \to (0, 1)$. As pointed out in Becker et al. [2018], we can include any additional input that by $\mathcal{F}_n$ measurable feature (e.g. the discounted current payoff) to efficiently train a neural network.

Besides the difference of the inputs, the loss function will directly mimic the optimal payoff at time $n = 0$,

$$\begin{aligned}
&\sup_{\tau\in\mathcal{T}_0} \mathbb{E}g(\tau, X_\tau) \\
&= \sup_{\tau\in\mathcal{T}_0} \mathbb{E}[g(\tau, X_\tau)\mathbf{1}_{\{\tau=0\}} + g(\tau, X_\tau)(1 - \mathbf{1}_{\{\tau=0\}})] \\
&= \sup_{\tau\in\mathcal{T}_0} \mathbb{E}[g(\tau, X_\tau)\mathbf{1}_{\{\tau=0\}} + g(\tau, X_\tau)\mathbf{1}_{\{\tau=1\}}(1 - \mathbf{1}_{\{\tau=0\}}) + g(\tau, X_\tau)\prod_{j=0}^{1}(1 - \mathbf{1}_{\{\tau=j\}})] \\
&= \sup_{\tau\in\mathcal{T}_0} \mathbb{E}[\sum_{n=0}^{N} g(n, X_n)\mathbf{1}_{\{\tau=n\}}\prod_{j=0}^{n-1}(1 - \mathbf{1}_{\{\tau=j\}})],
\end{aligned}$$

where we define the empty product $\prod_{j=0}^{-1}(1 - \mathbf{1}_{\{\tau=j\}}) := 1$, and $\mathbf{1}_{\{\tau=n\}}$ is approximated by f^{θ_n} in the previous scheme, that is the partial stopping decision in the range $(0, 1)$. Moreover, in the previous scheme we prove a neural network f^{θ_n} approximate f_n with activated value in the output layer of the form,

$$\sum_{i=1}^{r}(v_i^T x + c_i)^+ - \sum_{i=1}^{s}(w_i^T x + d_i)^+.$$

Let $G : [0, T] \times \mathbb{R}^d \mapsto \mathbb{R}$ denote the continuous version of the payoff function g such that $G(t_n, x) \equiv g(n, x)$ for all $n \in \{0, 1, ..., N\}$. Add time variate as an additional input and use only one neural network i.e., $F^{\tilde{\theta}}(t_n, x)$ to approximate N decisions given by $f^{\theta_n}(x), n = 0, 1, ..., N-1$. One can also achieve uniform approximation of $G(t, x) - h(t, x)$ on compact set $X \subseteq \mathbb{R}^{d+1}$ by

adding sufficiently more neurons r', s' according to Leshno et al. [1993],

$$\sum_{i=1}^{r+r'}(v_i^T x + v_i' t + c_i)^+ - \sum_{i=1}^{s+s'}(w_i^T x + w_i' t + d_i)^+,$$

where v_i', w_i' are trainable parameters for variate t. The neural network's optimal payoff is then given by

$$\sup_{\tau\in\mathcal{T}_0}\mathbb{E}[\sum_{n=0}^{N} g(n, X_n)F^{\tilde{\theta}}(t_n, X_n)\prod_{j=0}^{n-1}(1 - F^{\tilde{\theta}}(t_j, X_j))],$$

where t_n correspond to the real time at time point n, and $F^{\tilde{\theta}}(t_N, X_N) \equiv 1$ by problem constraint without training. Note

$$\sum_{n=0}^{N} F^{\tilde{\theta}}(t_n, X_n)\prod_{j=0}^{n-1}(1 - F^{\tilde{\theta}}(t_j, X_j)) = 1.$$

Hence, one can regard $F^{\tilde{\theta}}(t_n, X_n)\prod_{j=0}^{n-1}(1-F^{\tilde{\theta}}(t_j, X_j))$ as the approximated probability of stopping at time n, $F^{\tilde{\theta}}(n, X_n)$ as approximated conditional probability of stopping n conditioned on the event that the process is not stopped prior to n. Similarly to the previous scheme, after training, we replace the sigmoid activation with a indicator function, which is equivalent to rounding a partial decision with threshold 0.5. Let $\psi^{\tilde{\theta}} : [0, T]\times\mathbb{R}^D \to \{0, 1\}$ denote the modified function, note that the only difference between the two schemes is that we integrate N functions $f^{\theta_n}, n = 0, 1, ..., N-1$ into a single function $\psi^{\tilde{\theta}}$. The formulation of the stopping time τ_n under the one-shot scheme is given by[11]

$$\tau_n^{\tilde{\theta}} = \sum_{m=n}^{N} m\cdot\psi^{\tilde{\theta}}(t_m, X_m)\prod_{j=n}^{m-1}(1 - \psi^{\tilde{\theta}}(t_j, X_j)).$$

3.4 Parameter Optimization

In this subsection, we describe the implementation details of both the 'Backwards Induction' algorithm and 'One-shot' algorithm.

We begin with specifying some common notation for the algorithm.
$X_{(M,N+1,d)}$ denote the simulated training data of dimension $(M, N+1, d)$ where M stands for M independent samples, d is for the dimension of inputs. Note d is not necessarily the number of relevant assets, as one can add any $\mathcal{F}_n$-measurable random variables into the state process $(X_n)_{n=0}^N$ besides the assets, as per the definition of stopping time. $(x_n^m)_{n=0}^N$ denotes the mth simulated trajectory, of shape $(N+1, d)$.

$\tau_n^m \in \mathcal{T}_n$ is the optimal stopping time of the mth trajectory at time point n, which is indicated by the neural networks $f^{\theta_n}, f^{\theta_{n+1}}, ..., f^{\theta_N}$ following the formulation in 1. (For Backwards induction scheme particularly)

$\mathcal{R}(\theta)$ denotes the estimated reward[12] function based on the stopping signals given by neural network with parameter θ.

[11]Note this is identical to the previous formulation of τ^{Θ}, and we understand the empty product $\prod_{j=n}^{n-1}(1 - \psi^{\tilde{\theta}}(t_j, X_j))$ as 1.

[12]In programming, an optimizer in common tools such as Tensorflow, Pytorch defaults to minimize a loss

3.4.1 Backwards Induction Scheme

1. Simulate M trajectories of length $N+1$ corresponding to $N+1$ exercise times $n = 0, 1, ..., N$.

2. Initialize an array `decision_seq` of shape $(M, N+1)$ to record decision made in $\{0, 1\}$ for M trajectories at each time point, where 0 denotes continue, 1 denotes stop. As each column records decision made for a particular time point, we set the terminal decision at time point[13] N to 1 (i.e. $\tau_N^m \equiv N$, $m = 1, ..., M$), while the decisions at time point n for M trajectories are initialized to be zeros, pending decision made by neural network f^{θ_n}.

```
decision_seq = numpy.zeros(M,N+1)
decision_seq[:,-1] = 1
```

   As per the formulation τ_n^m, the optimal stopping time of each trajectory is essentially given by the index of the first non-zero element, starting from the index n of each row.

3. For time point n, according to the decision sequence, we have τ_n^m for the mth trajectory, calculate optimal continuation payoff $g(\tau_n^m, x_{\tau_n^m}^m)$, also calculate the current payoff $g(n, x_n^m)$ for each trajectory $m = 1, 2, ..., M$. Note the Backwards induction scheme starts from $n = N-1$, all trajectories have $\tau_{n+1}^m = N$ as optimal stopping time of continuation by the initialization in step 2.

4. Training F^{θ_n}:
   • Initialize parameters $\theta_n^{(0)}$ for neural network F^{θ_n} by Xavier[14] method.
   • Slice the data $X_{(M,N+1,d)}$ for time point n, use only $X[:, n, :]$ by the Markov presumption, then divide it into $[\frac{M}{B}]$ batches[15] of size B, with each batch $X^i, i = 1, 2, ..., [\frac{M}{B}]$ denoted as $\{x_n^{i1}, x_n^{i2}, ..., x_n^{iB}\}$, where i refers to the batch index, n referes to time step, $1, ..., B$ specify the indices of the samples. Note x_n^{ij} is of shape $(1, d)$.
   • For a batch X^i, evaluate the loss function by forward-propagation

$$\mathcal{L}(\theta_n^{(i-1)}) = -\mathcal{R}(\theta_n^{(i-1)}; X^i) = -\frac{1}{B}\sum_{j=1}^{B}[(g(n, x_n^{ij})F^{\theta_n^{(i-1)}}(x_n^{ij}) + g(\tau_n^{ij}, x_{\tau_n^{ij}}^{ij})(1 - F^{\theta_n^{(i-1)}}(x_n^{ij}))].$$

   Update the parameter θ_n according to the Adam optimizer described in 1. Repeat this procedure for all batches $X^i, i = 1, 2, 3, ..., [\frac{M}{B}]$. Note $i = 1, 2, ..., \frac{M}{B}$ refers to the training iterations[16].
   • Shuffling the data $X_{(M,N+1,d)}$, re-batch, and repeat the above procedure. Training for large enough epochs, until $\theta_n^{(i)}$ converges to the possibly optimal parameters θ_n^*.

5. Derive f^{θ_n}: As F^{θ_n} makes only partial decision, derive f^{θ_n} by

$$f^{\theta_n}(x) = \begin{cases} 1 & F^{\theta_n} \geq 0.5 \\ 0 & F^{\theta_n} < 0.5 \end{cases}.$$

6. Assign the decisions of the time step n to the n th column of the `decision_seq`.

function $\mathcal{L}(\theta)$, thus $\mathcal{L}(\theta)$= - $\mathcal{R}(\theta)$ since in the problem we aims to maximize the payoff.

[13] In python, specifically Numpy package, indices starts from 0. Thus each index match our stopping time.

[14] Parameter generator with randomness, either from a truncated normal distribution or from a uniform distribution. Xavier initialization helps neurons to be more different from each other which possibly perform feature transformation more usefully, also helps neuron generate weights within a more sensible range so that the gradient is not too small.

[15] To ensure the current and continuation payoff are correctly batched, to match the X^i, we concatenate $g(n, x_n^m), g(\tau_n^m, x_{\tau_n^m}^m)$ as an array of shape $(N, 2)$. In the implementation, the payoff array can be passed into `Model.fit()` as true label `y_true` for most of Deep Learning package.

[16] We call it an epoch if all batches have been trained once.

7. $n \leftarrow n-1$, repeat step 3,4,5,6 for $n = N-2, N-3, ..., 0$

3.4.2 One-shot Scheme

1. Simulate M trajectories of length $N+1$ corresponding to $N+1$ exercise times $n = 0, 1, ..., N$.

2. Training $F^{\tilde{\theta}}$:
   • Initialize parameters $\tilde{\theta}^{(0)}$ for neural network $F^{\tilde{\theta}}$ by Xavier method.
   • Divide $X_{(M,N+1,d+1)}$ into batches of size B, with each batch $X^i, i = 1, 2, ..., [\frac{M}{B}]$ denoted as $\{x_n^{i1}, x_n^{i2}, ..., x_n^{iB}\}$. Note its difference from previous scheme is that, x_n^{ij} is of shape $(1, N+1, d)$, since the Neural network takes time variate t as additional input. To emphasize the additional input, $d+1$ is used, but actually we can add more possibly useful feature, as long as they are adapted process.
   • For each trajectory $(x_n^{ij})_{n=0}^N$ in a batch X^i, input[17] (t_n, x_n^{ij}) for $n = 0, 1, ..., N$ sequentially into the neural network, where $(t_n)_{n=0}^N$ is the real time arithmetic sequence in the time horizon $[0, T]$. Evaluate the loss function function by forward-propagation

$$\mathcal{L}(\tilde{\theta}^{(i-1)}) = -\mathcal{R}(\tilde{\theta}^{(i-1)}; X^i) = -\frac{1}{B}\sum_{i=1}^{B}\sum_{n=0}^{N}[g(n, x_n^{ij}) \cdot F^{\tilde{\theta}^{(i-1)}}(t_n, x_n^{ij}) \prod_{k=0}^{n-1}(1 - F^{\tilde{\theta}^{(i-1)}}(t_k, x_k^{ij}))],$$

   where the partial stopping decision at $t_N = T$ is always overridden to 1 by terminal constraint, i.e. $F^{\tilde{\theta}^{(i-1)}}(T, x_N^{ij}) \equiv 1$. Update the parameter $\theta^{(i-1)}$ according to the Adam optimizer described in 1. Repeat this procedure for all batches $X^i, i = 1, 2, 3, ..., [\frac{M}{B}]$.
   • Shuffling the data $X_{(M,N+1,d)}$, re-batch, and repeat the above procedure. Training for large enough epochs, until $\tilde{\theta}^{(i)}$ converges to the possibly optimal parameters $\tilde{\theta}^*$

3. Derive $\psi^{\tilde{\theta}}$: As $F^{\tilde{\theta}}$ makes only partial decision, derive $\psi^{\tilde{\theta}}$ by

$$\psi^{\tilde{\theta}}(t, x) = \begin{cases} 1 & F^{\tilde{\theta}}(t, x) \geq 0.5 \\ 0 & F^{\tilde{\theta}}(t, x) < 0.5 \end{cases}.$$

Remark: Comparing the two schemes, we shall notice two differences: 1. As we use only one neural network $\psi^{\tilde{\theta}}$ to make decisions $\psi^{\tilde{\theta}}(t_n, X_n), \quad n = 0, 1, ..., N-1$, the decision made for neighbouring time point can be similar while this is not ensured for backwards induction scheme, as there are N neural networks with independent architecture and parameters. 2. The One-shot scheme indeed solves a stopping time problem, as t_n is deterministic associated with n and X_n is $\mathcal{F}_n$-measurable while the formulation of the stopping time is identical to that of the backward induction scheme. In programming, the sequential inputs of (t_n, X_n), $n = 0, 1, ..., N$ into the neural network is realized by the `TimeDistributed` in Python package Keras.

4 Examples

In this section, we apply the two algorithms on two representative examples - pricing fBm (fractional Brownian motion) and Bermudan Max call option. The fBm will be first discretized (N=100) and then made a Markovian process by enlarging state space because of the long-range dependence property. This example is devoted to demonstrate the algorithms' capability of tackling high-dimension problems. The second example focuses on two underlying assets because the model interpretation and data pattern can be effectively visualized in two-dimensional problem, which allows us to monitor how the sampling method can impact the training of the two types of model.

[17] Note (T, x_N^{ij}) is fed into neural network only for the convenience of calculating the loss function, which involves the terminal payoff $g(N, X_N)$. Since the prediction of $F^{\tilde{\theta}}(T, X_N)$ is always overridden to 1, the neural network

4.1 Fractional Brownian Motion

The fractional Brownian motion (fBm) (see e.g. Dieker [2004]) can be regarded as a generalization of Brownian motion. The major difference is that its increment may not be independent and can have long range dependence. A fBm $\{B_H(t) : 0 \leq t < \infty\}$ with Hurst parameter $H \in (0, 1)$ is uniquely characterised by the following properties,

$$B_H(t) \text{ has stationary increments;} \tag{4.1}$$
$$B_H(0) = 0 \text{ and } \mathbb{E}B_H(t) = 0, \quad \forall t \geq 0; \tag{4.2}$$
$$\mathbb{E}B_H^2(t) = t^{2H}, \quad \forall t \geq 0; \tag{4.3}$$
$$B_H(t) \text{ has a Gaussian distribution}, \quad \forall t \geq 0. \tag{4.4}$$

From the first three properties, we have the covariance function given by

$$\rho(s, t) = \mathbb{E}B_H(s)B_H(t) = \frac{1}{2}(t^{2H} + s^{2H} - |t - s|^{2H}).$$

In addition, as it is a Gaussian process, the mean and covariance structure uniquely determine the finite-dimension distribution. From which we get the self-similarity of B_H,

$$B_H(at) \sim a^H B_H(t) \quad a > 0,$$

which allows us to always simulate $B_H(t)$ on time interval $[0, N]$, then multiplied by $(\frac{T}{N})^H$ to scale it to time interval $[0, T]$. For equidistant time increments, we first simulate the incremental process $(X_n)_{n\geq 0}^N$ also known as fractional Gaussian noise, given by

$$X_n = B_H(n + 1) - B_H(n) \sim B_H(1), n = 0, 1, ..., N - 1,$$

by the stationary property (4.1). It implies X_n is a standard normal variable for every n. Following the covariance structure of fBm, we can derive the autocovariance function given by

$$\text{Cov}(X_n, X_{n+k}) = \gamma(k) = \frac{1}{2}[|k - 1|^{2H} + |k + 1|^{2H} - 2|k|^{2H}.$$

As $k \to \infty$, we have

$$\gamma(k) \to H(2H - 1)k^{2H-2},$$

which implies long-range dependence. Therefore, the Hurst parameter H determines the correlation increments process $(X_n)_{n\geq 0}$ of the fractional Brownian motion,

1. If $H = \frac{1}{2}$, B_H is a standard Brownian motion.
2. If $0 < H < \frac{1}{2}$, the increment process is negatively correlated.
3. If $1/2 < H < 1$, the increment process is positively correlated.

The properties can be clearly seen in Figure 4. When H is close to 1, the trajectories in the third subplot are almost straight lines because of positive correlation, in contrast, the trajectories in the first subplot are extremely wiggly.

Suppose we simulate M trajectories with $N+1$ equidistant time points $t_n = n\frac{T}{N}, \quad n = 0, 1, ..., N$ from a fBm. Denote the covariance matrix of fractional Gaussian noise $(X_n)_{n\geq 0}^{N-1}$ by $\Sigma_{N\times N}$. According to the Cholesky decomposition, $\Sigma_{N\times N} = QQ^T$. It follows that $X = QZ$, where Z is a vector of standard normal variables. By cumulative summation and $B_H(0) = 0$, we have a trajectory of $B_H(n)_{n=0}^N$. Then we re-scale it to the time interval $[0, T]$ by multiplying $(\frac{T}{N})^H$.

$F^{\tilde{\theta}}(T, X_N)$ is not trained for the terminal time.

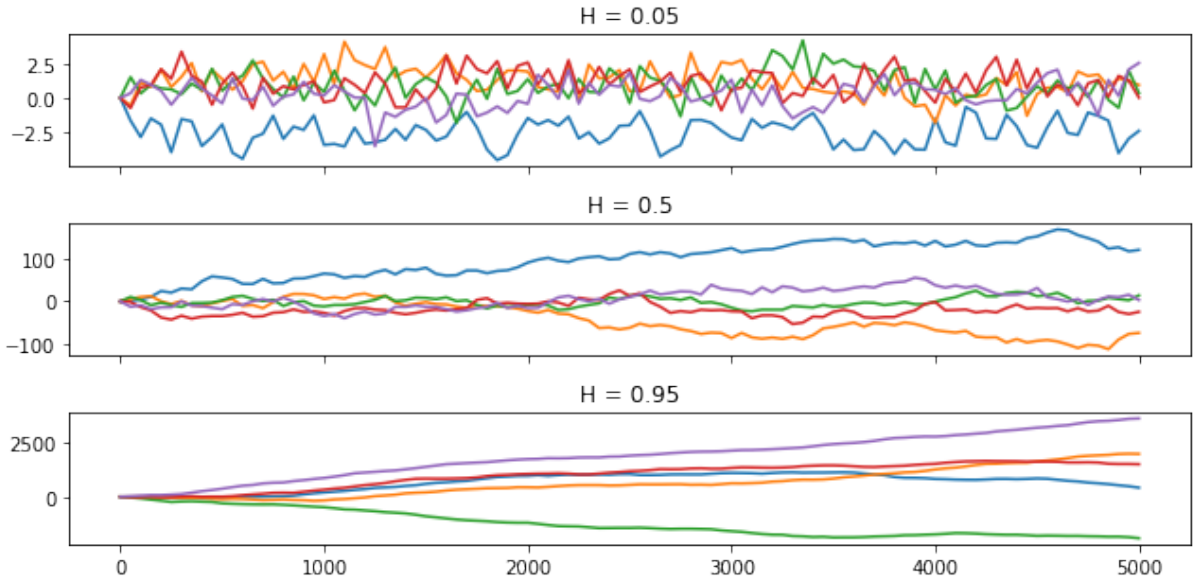

Figure 4: fBm trajectories in time interval [0,5000], simulated for Hurst = 0.05, 0.5, 0.95 respectively.

To approximate the supremum

$$\sup_{0\leq\tau\leq 1} B_H(\tau),$$

over all B_H-stopping times taking values in $[0,1]$. We set $N = 100$ which is relatively large compared to the length of time horizon $[0,1]$, in order to approximate the continuous payoff process. We denote $t_n = n/100$, n=0,1,...,100. As $B_H(t_n)$ has long-range dependence, one can transform it into an equivalent Markovian problem by enlarging the state space:

$$\begin{aligned}
Y_0^H &= (0,0,...,0)\\
Y_1^H &= (B_H(t_1),0,...,0)\\
Y_2^H &= (B_H(t_2),B_H(t_1),...,0)\\
&\vdots\\
Y_{100}^H &= (B_H(t_{100}),B_H(t_{99}),...,B_H(t_1)).
\end{aligned}$$

The original problem becomes

$$\sup_{\tau\in\mathcal{T}} \mathbb{E}g(Y_\tau),$$

where $\mathcal{T}$ is the set of all Y-stopping times taking values in $\{0,1,...,100\}$ and the payoff function $g:\mathbb{R}^{100}\to\mathbb{R}$ is given by the projection $(y^1,y^2,...,y^{100})\mapsto y^1$. Also note that $(Y_n)_{n\geq 0}^H$ is a Markov process with H inidicating the Hurst parameter of the fBm. We estimate the expected optimal payoff for several representative Hurst parameters using $M = 100000$ trajectories of fBm. Figure 5 below shows the estimation by the Backwards induction scheme on training data, and blue points are approximations of $\sup_{\tau\in T_n}\mathbb{E}B_H(\tau)$, $n = 0,1,...,N-1$ with a decreasing trend since every Snell envelope is a supermartingale. The orange lines show the approximation of $\mathbb{E}B_H(t_n), n = 0,1,...,N$, which are all close to zero as expected by the definition. Moreover, we note that as the fBm changes from negatively correlated increments ($H = 0.01$) to the standard Brownian motion ($H = 0.5$), the corresponding optimal payoff process see a decrease, while H increases from 0.5 to 1, the corresponding optimal payoff process see a increase.

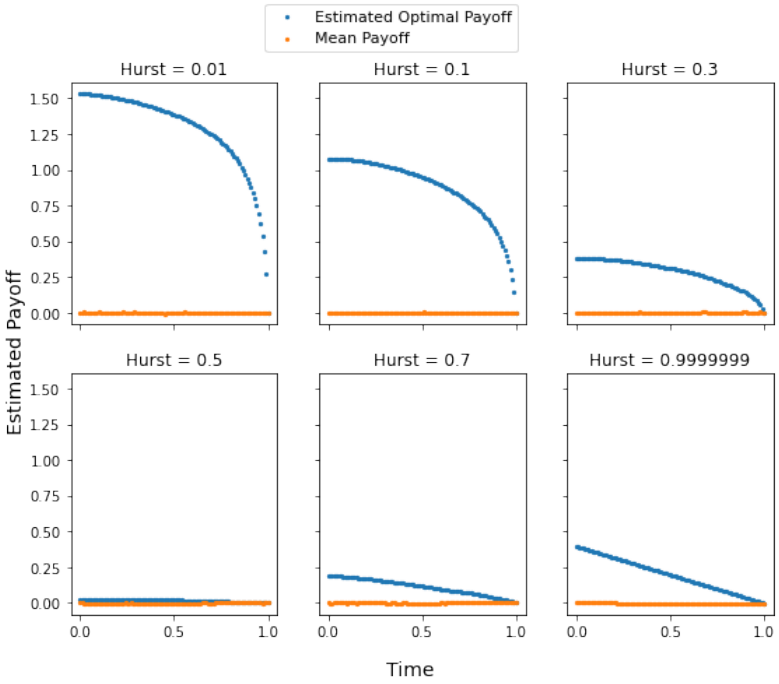

Figure 5: Backwards Induction Scheme - The approximation of the optimal payoff of $(B_H(t_n))_{n\geq 0}^N$ on time interval $[0,1]$, for Hurst parameter $H \in \{0.01, 0.1, 0.3, 0.5, 0.7, 0.9999999\}$.

For N time steps, one needs to fit N[18] neural networks under the Backwards induction scheme. For the above result, each neural network is trained by setting batch size at 1000 samples and training iterations as 50 epochs (about 1500 seconds for each Hurst parameter). In contrast, we need only one model $\psi^{\tilde{\theta}}(t,y)$ under the one-shot scheme, where $t \in \{t_0, t_1, ..., t_{N-1}\}$ takes the time input, and $y \in \mathbb{R}^N$ in this example. We report the estimations of the optimal payoff $V_H(0,y)$ under the two schemes in Figure 6, where $V_H(n,y) : \{0,1,...N\} \times \mathbb{R}^N \to \mathbb{R}$ denotes the optimal payoff at time t_n, given state y of the process $(Y_n)_{n\geq 0}$.

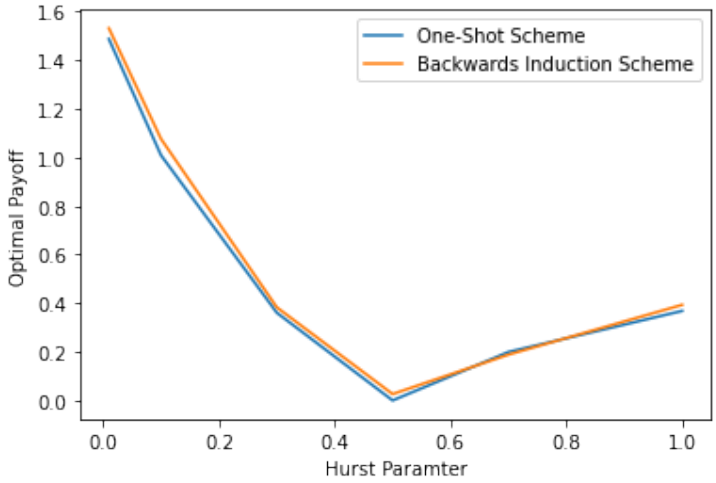

Figure 6: The estimated optimal payoff starting from the initial time point by two different schemes. The experiments are not particularly designed for benchmark in terms of speed and accuracy, given that the data preparation before training are different.

[18]If the initial value is deterministic, one normally need not fit for $n = 0$ for such a process. But in the second example, we will show that in order to make the model generalizable for a range of initial prices, we have to fit N neural networks.

In Figure 6, we can see that the estimation by the backwards induction algorithm seems to be slightly higher than those of the one-shot scheme. This should be expected as the one-shot scheme takes an additional continuity constraint for the sake of generalizability. However, this example focus on demonstration of the capability of the two scheme in high-dimensional example, and we can see that they are comparable in terms of accuracy since the estimates are only different in the second decimal places (also because of the limited size of samples). In the next example, we will focus on the discussion of the generalizability of the two scheme, which means the neural networks approximate the decision function f_n closely in a large subset of the state space.

4.2 Bermudan Max-Call Option

A Bermudan style option is different from an American style option in the sense that it can be only exercised on the specified exercise dates. In this example, we study the Bermudan max-call option whose payoff depends on the maximum of the underlying assets. Assume the underlying assets are d stocks given by the Black-Scholes stochastic differential equations under the risk-neutral measure, the price of asset i at time t is given by

$$S_t^i = s_0^i exp\left([r - \delta_i - \sigma_i^2/2]t + \sigma_i W_t^i\right), i = 1, 2, ..., d,$$

where $s_0^i \in (0, +\infty)$ is the initial price, $r \in [0, +\infty)$ is the risk-free interest rate, dividend yield $\delta_i \in [0, +\infty)$, and d-dimensional Brownian motion $(W_t)_{T \geq t \geq 0}$ (T is the terminal time) with constant instantaneous correlation $\rho_{ij} \in [-1, 1]$ between different components W^i, W^j. Suppose the specified exercise time of a Bermudan max-call option is given by $t_n, n = 0, 1, ..., N$ and $0 = t_0 < t_1 < ... < t_N = T$, if exercised at t_n, the payoff would be $(\max_{1 \leq i \leq d} S_{t_n}^i - K)^+$. Denote $X_n = S_{t_n} \in \mathbb{R}$ and denote the discounted payoff at t_n with assets value $x \in \mathbb{R}^d$ by function $g(n, x)$,

$$g(n, x) = e^{-rt_n} \left(\max_{1 \leq i \leq d} x^i - K \right)^+.$$

Then the price of a Bermudan max-call option at t_0 with initial asset value s_0 is given by

$$\sup_{\tau} \mathbb{E}[g(\tau, X_\tau) | X_0 = s_0].$$

For a 2-D case, one can effectively visualize the diffusion process by a 2-D histogram. The number of samples in each bin is proportional to the density in that area, which will shed light on the evolution of the process. We take a symmetric case here as an example, where $s_0^i = s = 90$, $\delta_i = \delta = 0.1$, $\sigma_i = \sigma = 0.2$ for all $i = 1, ..., d$, and terminal time $t_N = T = 3$, $N = 9$ and equidistant time grid. The evolution of the process is visualized in Figure 7 below.

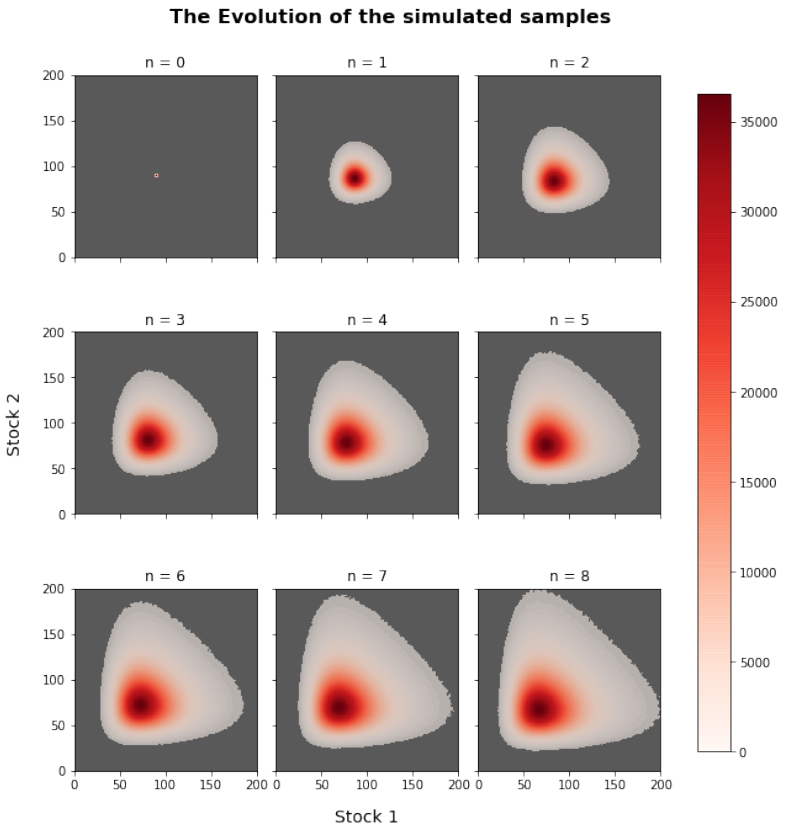

Figure 7: The evolution of the distribution of the simulated 30 million samples with $s_0 = [90, 90]$.

Different from the main reference Becker et al. [2018] where 3,000 + d batches of size 8,192 are simulated and trained only once, the implementation in this dissertation as described in Section 3.4 allows each sample to participate the training as many times as the number of epochs. We train the f^{θ_n} on 30 millions with batchsize 1 million[19], stepsize $\alpha = 0.01$, 50 epochs.

The trained models f^{θ_n} are accurate approximations of f_n in the sense that it can approximate an accurate Snell envelope even on an unseen data set[20] independently simulated from the same process. However, the f^{θ_n} is only a good approximation in a subset of state space where the simulated samples are dense, while $f_n : \mathbb{R}^d \to \{0, 1\}$ is defined to make decision for any current state $X_n = x > 0$ with the same continuation dynamics following the Markov property. Thanks to the 2-D setting, we can effectively monitor the decision boundary of the neural networks F^{θ_n} (partial decision in (0,1)), as shown in Figure 8. One can observe that when the state value X_n only appear in a small subset of the state space, the boundary does not make sense. This symptom motivates us to derive more generalizable models such that f^{θ_n} approximates f_n accurately in a larger subset of the state space and allows pricing a wide range of initial states X_0, instead of training the models for every initial states. Achieving this goal boils down to simulating the samples by design, which is the topic in the next subsection.

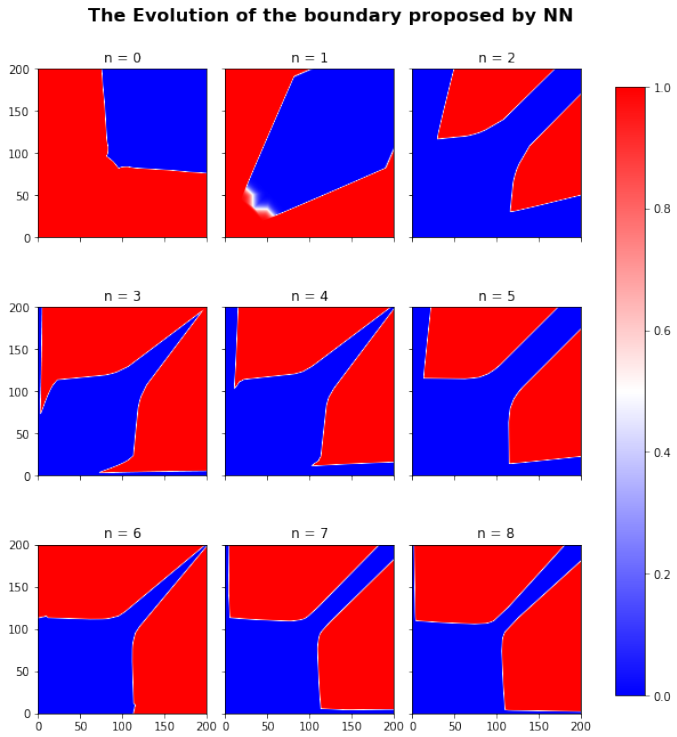

Figure 8: The partial decisions (1 is for stopping, 0 is for continue) given by the Backwards induction models F^{θ_n} visualized in the map justifies our further modification on the partial decision to f^{θ_n} which gives clear decision in $\{0, 1\}$, since the partial decisions are already close to 0 and 1. It can be observed that as the samples spread over the state space, the boundary become more sensible and smoother.

[19]The batchsize and stepsize have not been particularly tuned. According to our informal experiment with size set at 1000,10,000, 500,000, 1,000,000, 2,000,000 respectively, and with a range of stepsize (0.1, 0.02, 0.01, 0.001), the result are almost the same despite the speed of convergence. We conclude that the batchsize and stepsize in general can influence the speed of convergence in the parameter optimization but does not impact the result of the optimization. And it is encouraged to carry out more rigorous experiments regarding the speed and accuracy of the optimization of relevant parameter including batchsize, number of epochs, learning rate etc.

[20]In the same way we simulate the training data, an unseen data set is simulated independently from the symmetric diffusion process starting from $[90, 90]$, in order to test the robustness of the approximated f^{θ_n}. In later section, the pricing on the unseen data set is defined as the approximated lower bound of the true value.

4.2.1 Sampling Methods

For Markov processes, the continue or stop decision made by the decision function f_n is irrelevant to the initial point of the diffusion processes. Specifically in the Bermudan max-call example, as long as the dynamics of the underlying assets given by the SDE are unchanged, the decision function f_n is identical for any initial state X_0.

Considering the deep learning algorithms are computational expensive, it is more economic and desirable to fit a set of generalizable models $f^{\theta_n}, n = 0, 1, ..., N-1$ that is applicable for pricing a range of initial prices s_0. Geometrically, the objective is equivalent to making f^{θ_n} an accurate approximation in a larger subset of the state space, which relies on how to increase the density of the samples around the decision boundary. It should be underlined that due to the mechanic of the Deep Empirical Risk Minimization, only the trajectories crossing or possibly crossing the approximated decision boundary make a difference on the reward function, and then influence the movement of the boundary in the training process. In contrast, the trajectories that are far away from the current approximated boundary proposed by the neural network may not efficiently participate in the continuing training process, however, might have contributed previously to the formation of the boundary. This qualitative assumption guide us in the design of sampling, by which we carry out several sampling method applicable for both backward induction scheme and the one-shot scheme, including "uniformly spreading the initial states", "simulating and intercepting several diffusion processes". On top of that, the importance sampling technique is introduced , which is particularly compatible the one-shot scheme. The former sampling methods only organize the initial states of the trajectories without changing the drift, diffusion parameters, while the importance sampling by Girsanov theorem allows manipulation of the drift parameters by changing the probability measure and is worth possible further study.

Spreading the initial states
Spreading the initial prices is probably the most direct and easiest way to make the the samples denser around the latent true decision boundary [21]. Two specific approaches are experimented - one is uniformly spreading the initial states in several triangular area, another is simulating first from several initial states where the boundary may exist, followed by intercepting from a suitable time point. The effect of these two sampling methods can be seen in in Figure 12, 13 in the xix. Here, we compare the result of the two different schemes (one-shot and backward induction) fitted on the same training data, so that we understand the differences of the two algorithms in terms of the boundary approximation, which essentially decides the bias in pricing.

The backward induction scheme fits boundaries independently, which has no guarantee of continuity, as shown in the time steps $n = 1$ in Figure 9. One-shot scheme, however, can fit continuous boundaries that even allow extrapolation of boundaries for the intermediate time point that are not simulated for the trajectories, as reported in Figure 10. Despite this minor difference, the two sets of approximated boundaries have great generalizability[22] such that can price a range of initial states given the same drift, diffusion parameters. Moreover, one may also notice that the decision boundary proposed by the one-shot scheme tend to shrink, which is possibly caused by the design of loss function. Recall that we use $F^{\tilde{\theta}}(n, X_n)\prod_{j=0}^{n-1}(1 - F^{\tilde{\theta}}(j, X_j))$ to represent the probability stopping at time point n. This expression implies that a trajectory almost stopping earlier cannot efficiently participate the later training as the product term $(1 - F^{\tilde{\theta}}(j, X_j))$ will be extremely small. This concern motivates us to distribute the trajectories more evenly to the training of each time point, which is feasible using importance sampling technique.

[21] also known as free-boundary in specialized study De Angelis and Stabile [2017]

[22] The initial states within the area $[x_1, y_1, x_2, y_2] = [80, 120, 80, 120]$ can be accurately priced by experiments, the other area are not tested but might see the same level of accuracy.

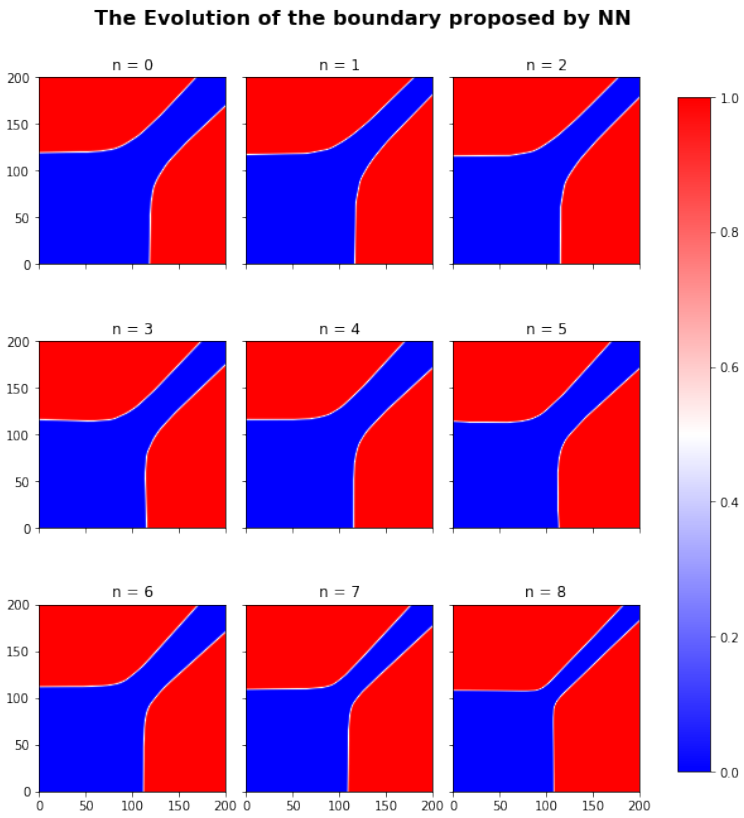

Figure 9: Sampling method - Uniformly distributed initial states as described in Figure 12; Algorithm: Backward induction scheme; The changes of the boundary from $n = 0$ to $n = 1$ seems suspicious.

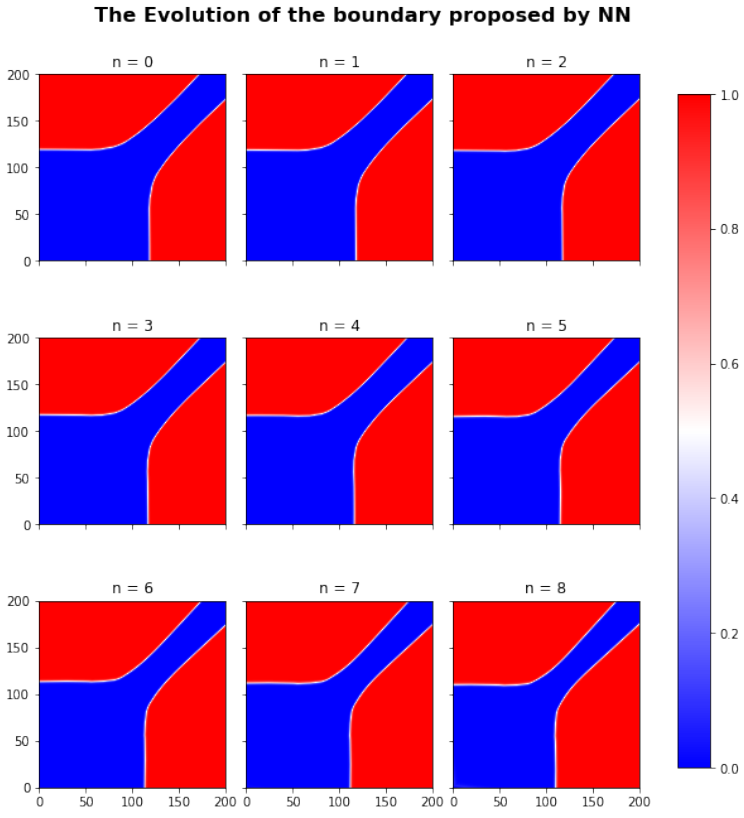

Figure 10: Sampling method - Uniformly distributed initial states as described in Figure 12; Algorithm: One-shot scheme; As the approximated boundaries are fitted under continuity constraint, there is no significant change between neighbouring time steps.

Importance Sampling by Girsanov's Theorem

Importance sampling is a well-known technique to increase the density of the samples in the region of interest (ROI). Since the probability density function involving stopping time is not available, one cannot directly change the source of sampling. Instead, we may achieve that by change of measure following Girsanov's Theorem, as done in risk-neutral pricing, where we replace the original drift with risk-free interest rate. While the Girsanov's theorem can be applied to many stochastic processes, we restrict ourselves to the special case - wiener process for simplicity.

Theorem 4.1. *Girsanov's Theorem for Wiener process*
Let $(\mathcal{F}_t)_{t\in[0,T]}$ be a filtration. Let $W = (W_t)_{t\in[0,T]}$ be a d-dimensional Wiener martingale with respect to $(\mathcal{F}_t)_{t\in[0,T]}$. Let $\varphi = (\varphi(t))_{t\in[0,T]}$ be a d-dimensional process adapted to $(\mathcal{F}_t)_{t\in[0,T]}$ such that

$$\mathbb{E}\int_0^T |\varphi(s)|^2 ds < \infty.$$

Let

$$L(t) := exp\left(-\int_0^t \varphi(s)^T dW_s - \frac{1}{2}\int_0^t |\varphi(s)|^2 ds\right),$$

and assume that $\mathbb{E}L(T) = 1$. Let $\mathbb{Q}$ be a new measure on $\mathcal{F}_T$ given by the Radon-Nikodym derivative $d\mathbb{Q} = L(T)d\mathbb{P}$. Then

$$W_t^{\mathbb{Q}} := W_t + \int_0^t \varphi(s)ds$$

is a $\mathbb{Q}$-Wiener martingale.

Remark 1: The condition $\mathbb{E}L(t) = 1 \quad \forall t \in [0,T]$ holds iff $L\varphi \in \mathcal{H}^d$. From which, we conclude L is a true martingale. Alternatively, check the Novikov condition:

$$\mathbb{E}[e^{\frac{1}{2}\int_0^T |\varphi(t)|^2 dt}] < \infty.$$

By symmetry, L^{-1} is a true martingale on $\mathbb{Q}$. It follows that for a $\mathcal{F}_t$-measurable r.v X

$$\mathbb{E}_{\mathbb{P}}X = \mathbb{E}_{\mathbb{Q}}[XL^{-1}(T)] = \mathbb{E}_{\mathbb{Q}}[\mathbb{E}_{\mathbb{Q}}[XL^{-1}(T)|\mathcal{F}_t]] = \mathbb{E}_{\mathbb{Q}}[XL^{-1}(t)],$$

since L^{-1} is a $\mathbb{Q}$ martingale and X is $\mathcal{F}_t$-measurable.

Remark 2: As shown in the above remark and the Girsanov's theorem, changing probability measure allows us to manipulate drift of the diffusion process. And to offset this change, we use $L^{-1}(t_\tau)$ as sample weights for each trajectory stopping at τ, where t_τ is the real time point correspond to τ.

In the context of deriving a more accurate decision boundary, the importance sampling helps the backward induction algorithm adjust the density of the trajectories around the underlying decision boundary at each time point. It could also help the one-shot algorithm to control the speed of the trajectories crossing the boundaries so that the proportions of trajectories stopping at each time point are comparable. It is worth noting that the importance sampling can be compatible with the previous method - spreading the initial states.

In the programming implementation, however, we only test the importance sampling with constant $\varphi = \lambda$ because of limitation of time and the complexity of finding the optimal drift. We refers interested readers to Virrion [2020], where the author use deep learning to find the optimal drift for variance reduction of Monte Carlo estimator. It is possible to derive a drift such that the trajectories intelligently cross the underlying boundary more frequently. One may argue that the approximated boundary is still too complex to detect or study, as it is expressed by cumbersome neural networks. This issue could actually be tackled as discussed in section 6.

5 Estimation of Lower, Upper bounds

In this section, we introduce the procedure of deriving the lower bound and upper bound of the estimate of $V_0 = \sup_{\tau \in \mathcal{T}_0} g(\tau, X_\tau)$. Our proof exactly follows Becker et al. [2018], but with a major difference in deriving upper bound, which lies in the way we estimate the optimal continuation payoff approximated by neural networks $\mathbb{E}[g(\tau^{\Theta}_{n+1}, X_{\tau^{\Theta}_{n+1}})|\mathcal{F}_n]$ (backward induction scheme)/$\mathbb{E}[g(\tau^{\tilde{\theta}}_{n+1}, X_{\tau^{\tilde{\theta}}_{n+1}})|\mathcal{F}_n]$. The authors estimate this value by simulating J continuation paths and taking crude average, while our solution is using regression instead. What motivate us is still the pursuit of generalizability. That is, we would like to derive upper bounds of V_0 for a range of similar initial states using one model, in order to save computational resource and time.

5.1 Lower Bound

After training, both the backward induction scheme and one-shot scheme are able to indicate an approximated stopping time for an arbitrary trajectory $(y_n)_{n=0}^N$ of the target Markov process $(X_n)_{n=0}^N$[23]. Following the previous notation, denote the stopping time given under backward induction scheme by τ^{Θ}, the counterpart from the one-shot scheme as $\tau^{\tilde{\theta}}$. As the two schemes are only different in the construction of neural network, we can describe their procedure in prediction altogether.

τ^{Θ}, $\tau^{\tilde{\theta}}$ actually give two lower bounds $L^{\Theta} = \mathbb{E}g(\tau^{\Theta}, X_{\tau^{\Theta}})$, $L^{\tilde{\theta}} = \mathbb{E}g(\tau^{\tilde{\theta}}, X_{\tau^{\tilde{\theta}}})$ respectively for the true value $V_0 = \sup_{\tau \in \mathcal{T}_0} \mathbb{E}g(\tau, X_\tau)$. For the example of pricing Bermudan max-call option, we can estimate the lower bounds for a range of initial states X_0 by simulating new sets of independent realizations $(y_n)_{n=0}^N$ from the process $(X_n)_{n=0}^N$. The specific procedure for the 2-D case follows:

1. Simulate $K_L = 10,000,000$ independent trajectories $(y_n^k)_{n=0}^N, k = 1, 2, .., K_L$ from the diffusion process with initial value s_0. [24]

2. Initialize an array `decision_seq_test` of shape $(K_L, N+1)$ to record decision made in $\{0, 1\}$ for K_L trajectories at each time point, where 0 denotes continue, 1 denotes stop. As each column records decision made for a particular time point, we set the terminal decision at time point N to 1 (i.e. $\tau_N^k \equiv N$, $k = 1, ..., K_L$), while the decisions at time point n for K_L trajectories are initialized to be zeros, pending decision made by neural network f^{θ_n} under backward induction scheme or $\psi^{\tilde{\theta}}(t_n, X_n)$ under the one-shot scheme.

3. Let $f^{\theta_n}(X_n)$ / $\psi^{\tilde{\theta}}(t_n, X_n)$ make stopping decisions in $\{0, 1\}$ for each time point respectively. According to the filled decision sequence `decision_seq_test`, we derive approximated optimal stopping time $\tau_k^{\Theta}/\tau_k^{\tilde{\theta}}, k = 1, ..., K_L$ for each trajectory.

4. Calculate the estimated lower bound by Monte Carlo approximation:

$$\text{Backward Induction: } L^{\Theta} = \frac{1}{K_L} \sum_{k=1}^{K_L} g(\tau_k^{\Theta}, X_{\tau^{\Theta}})$$

$$\text{One-Shot: } L^{\Theta} = \frac{1}{K_L} \sum_{k=1}^{K_L} g(\tau^{\tilde{\theta}}, X_{\tau^{\tilde{\theta}}})$$

[23] Recall that the models can price a range of initial states X_0

[24] As the derived models f^{θ_n} and $\psi^{\tilde{\theta}}(t_n, \cdot)$ can be accurate approximation of f_n in a large area. For example, the accurate range of the 2-D case is $[x_1, y_1, x_2, y_2] = [0, 250, 0, 250]$, we can accurately price the option with a range of initial states s_0, if only the corresponding process does not exceed the accurate range significantly.

5.2 Upper Bound

The following proof for an estimate of an upper bound for the optimal value V_0 is based on Andersen and Broadie [2004]. The only difference is we estimate conditional expectation by Least square regression based on neural network, so as to derive upper bounds for a range of initial states to reduce computational cost.
As proved in the Appendix A, the Snell envelope of the reward process $(g(n, X_n))_{n=0}^N$ is the smallest supermartingale with respect to $(\mathcal{F}_n)_{n=0}^N$ that dominates $(g(n, X_n))_{n=0}^N$. It is given by

$$Z_n = \sup_{\tau \in \mathcal{T}_n} \mathbb{E}(g(\tau, X_\tau)|\mathcal{F}_n), \quad n = 0, 1, ..., N.$$

By Doob's decomposition, Z can be *uniquely* written as

$$Z_n = Z_0 + M_n - A_n,$$

where M is the martingale given by

$$M_0 = 0 \text{ and } M_n - M_{n-1} = Z_n - \mathbb{E}[Z_n|\mathcal{F}_{n-1}], \quad n = 1, ..., N,$$

and A is the non-decreasing predictable process given by

$$A_0 = 0 \text{ and } A_n - A_{n-1} = Z_{n-1} - \mathbb{E}[Z_n|\mathcal{F}_{n-1}] \geq 0 \quad n = 1, ..., N,$$

since $(Z_n)_{n\geq 0}^N$ is a supermartingale. The uniqueness holds in the sense that if $Z_n = Z_0 + M'_n - A'_n$ for some other martingale M' and predictable process A' with $M'_0 = A'_0 = 0$, then $\mathbb{P}(M'_n = M_n, A'_n = A_n, \forall n) = 1$.

Proposition 3. *Proposition 7 in Becker et al. [2018] Let $(\epsilon_n)_{n=0}^N$ be a sequence of random variable with $\epsilon_n \in L^2(\Omega, \mathcal{F}_n, \mathbb{P})$. Then*

$$V_0 \geq \mathbb{E}\left[\max_{0\leq n\leq N}(g(n, X_n) - M_n - \epsilon_n)\right] + \mathbb{E}\left[\min_{0\leq n\leq N}(A_n + \epsilon_n)\right].$$

Moreover, if $\mathbb{E}[\epsilon|\mathcal{F}_n] = 0$ for all $n \in \{0, 1, ..., N\}$, one has

$$V_0 \leq \mathbb{E}\left[\max_{0\leq n\leq N}(g(n, X_n) - M'_n - \epsilon_n)\right]$$

for every martingale M' with $M'_0 = 0$.

Proof.

$$\begin{aligned}
&\mathbb{E}\left[\max_{0\leq n\leq N}(g(n, X_n) - M_n - \epsilon_n)\right] \\
&\leq \mathbb{E}\left[\max_{0\leq n\leq N}(Z_n - M_n - \epsilon_n)\right] && ((Z_n)_{n\geq 0}^N \text{ dominates } (g(n, X_n))_{n\geq 0}^N) \\
&= \mathbb{E}\left[\max_{0\leq n\leq N}(Z_0 - A_n - \epsilon_n)\right] && \text{(By Doob's decomposition)} \\
&= \mathbb{E}\left[Z_0 + \max_{0\leq n\leq N}(-A_n - \epsilon_n)\right] \\
&= V_0 - \mathbb{E}\left[\min_{0\leq n\leq N}(A_n + \epsilon_n)\right]. && \text{(max -x = - min x)}
\end{aligned}$$

That is,

$$V_0 \geq \mathbb{E}\left[\max_{0\leq n\leq N}(g(n, X_n) - M_n - \epsilon_n)\right] + \mathbb{E}\left[\min_{0\leq n\leq N}(A_n + \epsilon_n)\right].$$

Assume we have $\mathbb{E}[\epsilon_n|\mathcal{F}_n] = 0, \quad \forall n \in \{0, 1, ..., N\}$, and let τ be an X-stopping time[25]. By Tower property and Pull-out property, we have

$$\mathbb{E}\epsilon_\tau = \mathbb{E}\left[\sum_{n=0}^{N} \mathbf{1}_{\{\tau=n\}}\epsilon_n\right] = \mathbb{E}\left[\sum_{n=0}^{N} \mathbf{1}_{\{\tau=n\}}\mathbb{E}[\epsilon_n|\mathcal{F}_n]\right] = 0.$$

By the optional stopping theorem given $\tau \leq N$, for every martingale M' with $M'_0 = 0$, we have

$$\mathbb{E}M'_\tau = \mathbb{E}M'_0 = 0.$$

As $\mathbb{E}M'_\tau = \mathbb{E}\epsilon_\tau = 0, \quad \forall \tau \in \{0, 1, ..., N\}$, it follows that

$$\mathbb{E}g(\tau, X_\tau) = \mathbb{E}\left[g(\tau, X_\tau) - M'_\tau - \epsilon_\tau\right] \leq \mathbb{E}\left[\max_{0\leq n\leq N}(g(n, X_n) - M'_n - \epsilon_n)\right],$$

which implies $V_0 = \sup_{\tau\in\mathcal{T}_0} \mathbb{E}g(\tau, X_\tau) \leq \mathbb{E}\left[\max_{0\leq n\leq N}(g(n, X_n) - M'_n - \epsilon_n)\right]$. □

Remark 1: The second inequality of the proposition gives an upper bound for V_0 . In addition, if $M' = M$ and $\epsilon \equiv 0$, the upper bound would be tight, as justified by the first inequality of the proposition.

Remark 2: Because the backward induction algorithm and one-shot scheme are only different in the inputs of the neural networks and in the training procedure, and their prediction of the optimal payoff for each trajectory both rely on `decision_seq_test`. Thus for simplicity, in the following description of the procedure, we only use notation τ^Θ for approximated optimal stopping time and decision function f^{θ_n}. For the one-shot algorithm, simply replace every approximated decision function $f^{\theta_n}(n, X_n)$ with $\psi(t_n, X_n)$ - the counterpart of the one-shot algorithm.

Since the martingale part of the Snell envelope M is not readily available, we hope to estimate M using our approximated optimal stopping time τ^Θ. The closer the predicted optimal stopping time τ_n^Θ is to an real optimal stopping time with

$$\tau_n^\Theta = \sum_{m=n}^{N} m f^{\theta_m}(X_m) \prod_{j=n}^{m-1}(1 - f^{\theta_j}(X_j)), \quad n = 0, 1, ..., N,$$

the better the value process

$$Z_n^\Theta = \mathbb{E}\left[g(\tau_n^\Theta, X_{\tau_n^\Theta})|\mathcal{F}_n\right], \quad n = 0, 1, ..., N$$

approximates the Snell envelope $(Z_n)_{n\geq 0}^N$. The martingale part of Z^Θ (following Doob's decomposition) is denoted by M^Θ, which is given by

$$\begin{aligned}
&M_0^\Theta = 0 \\
&\text{For } n \geq 1 \\
&\Delta M_n^\Theta := M_n^\Theta - M_{n-1}^\Theta \\
&= Z_n^\Theta - \mathbb{E}[Z_n^\Theta|\mathcal{F}_{n-1}] \\
&= f^{\theta_n}(X_n)g(n, X_n) + (1 - f^{\theta_n}(X_n))\mathbb{E}[Z_{n+1}^\Theta|\mathcal{F}_n] - \mathbb{E}[Z_n^\Theta|\mathcal{F}_{n-1}] \\
&= f^{\theta_n}(X_n)g(n, X_n) + (1 - f^{\theta_n}(X_n))C_n^\Theta - C_{n-1}^\Theta,
\end{aligned}$$

where we denote the approximated optimal payoff of continuation $\mathbb{E}[Z_{n+1}^\Theta|\mathcal{F}_n]$ by C_n^Θ for sim-

[25] $\{\tau = n\}$ is $\sigma(X_0, X_1, ..., X_n)$-measurable.

plicity, that is

$$C_n^\Theta := \mathbb{E}[Z_{n+1}^\Theta|\mathcal{F}_n] = \mathbb{E}[Z_{n+1}^\Theta|X_n], \quad n = 0, 1, ..., N-1,$$

by Markov assumption. Hence, the estimation of C_n^Θ is the main problem to solve in the estimation of upper bound. The solution proposed in Becker et al. [2018] follows, first generate K_U independent trajectories $(x_n^k)_{n=0}^N, k = 1, 2, ..., K_U$ from the target process. Along a particular k th trajectory, for every $x_n^k, n = 0, 1, ..., N-1$ simulate J continuation paths. For each continuation paths, we calculate the corresponding optimal continuation payoff $C_n^{k,j}, j = 1, 2, ..., J$ with their stopping time decided by neural networks. It follows to estimate C_n^k by taking the crude average of $C_n^{k,j}$. This method is verified to give tight upper bound as shown in Becker et al. [2018]. However, this procedure has to be performed for every interested initial states X_0 even for a minor change, and might be difficult to employ in real-world industry. For example, one may want to price a Bermudan max-call option in real-time when the initial prices of the underlying deviate within a range, assuming the evolution rules are unchanged in a short time period. This concern motivates us to estimate C_n^Θ by Least square method since by definition of conditional expectation, C_n^Θ is the orthogonal projection of Z_{n+1}^Θ in the Hilbert space $L^2(\Omega, \mathcal{F}_{n+1})$ onto its subspace $L^2(\Omega, \mathcal{F}_n)$.

5.2.1 Regression by neural network

By the Doob's decomposition as shown in the last section,

$$\begin{aligned} Z_n^\Theta &= \mathbb{E}[Z_n^\Theta|X_{n-1}] + \Delta M_n^\Theta \\ &= C_{n-1}^\Theta + \Delta M_n^\Theta, \end{aligned}$$

where $\mathbb{E}[\Delta M_n^\Theta|X_n] = 0$ and $M_0^\Theta = 0$. To estimate C_{n-1}^Θ, we use trajectory-wise (noisy data) optimal continuation payoff indicated by neural network as response, denoted by $Z_n^k = g(\tau_n^k, X_{\tau_n^k})$ [26], and construct an estimator $\hat{h}_{n-1} : \mathbb{R}^d \to \mathbb{R}$ given by the neural network[27] to approximate C_{n-1}^Θ. The formulation is given by

$$Z_n^k = \hat{h}_{n-1}(X_{n-1}^k) + error^k.$$

One can follow either backward induction scheme where the modelling procedures begin with $\hat{h}_{N-1}$, and end with $\hat{h}_0$, or one-shot scheme using $\hat{h} : [0, T] \times \mathbb{R}^d \to \mathbb{R}$, i.e., $\hat{h}(t_n, X_n)$ corresponding to $\hat{h}_n(X_n)$ for $n = 0, 1, ..., N-1$. By experiments, we find that the one-shot scheme provides significantly better approximation thanks to the continuity constraint. Compared to the task of approximating the decision boundary, fitting a regression requires samples to be dense everywhere the target process has non-negligible density, while the former task only requires the samples to be dense around the subset of the decision boundary where the trajectories may cross. That is to say, a regression task has a significantly higher demand for the size of simulated data. The continuity of $\hat{h}$ is in fact a constraint on the neural network's behavior at different time point as justified in 3.3.1.

To estimate the upper bound for a range of initial states for the Bermudan max-call pricing example, we use the "spreading initial states" techniques to increase the density of the trajectories within the region of interest. The procedure of data preparation and model training follows:

1. Simulate $K_{train} = 30,000,000$ trajectories of length $N + 2$, $(x_n^k)_{n=0}^{N+1}, k = 1, 2, ..., K_{train}$ with three initial states $[90, 90], [100, 100], [110, 110]$[28], then intercept each trajectory from

[26] τ_n^k indicated by approximated decision boundaries

[27] We use neural network here since it can handle large set of simulated samples by mini-batch and can be flexible enough to capture the underlying pattern.

[28] This design of sampling allows generalization for initial states at least within the range $90 \leq s_0 \leq 110$. The sampling design mentioned above can also be used here, but we decrease the size initial area on purpose since it increase the density of trajectories in the ROI

$n = 1$. That is $(x_n^k)_{n=0}^N \leftarrow (x_n^k)_{n=1}^{N+1}$. Derive the optimal payoff indicated by the neural network $(z_n^k)_{n=0}^N$, following the prediction procedure as described in the "Lower Bound" section. Feed the training data (t_{n-1}, x_{n-1}) - explanatory variables, z_n - response into the neural network,

$$z_n^k = \hat{h}(t_{n-1}^k, x_{n-1}^k) + error^k, n = 1, 2, ..., N.$$

2. Using "mean squared error" as loss function, and update parameters using Mini-batch together with Adam optimizer as done previously.

3. Simulate $k_U = 10,000,000$ trajectories $(y_n^k)_{n=0}^N$ directly from a target process (Initial states s_0 can take any value within an appropriate range thanks to the generalizability.)
Predict $C_n^\Theta, n = 0, 1, ..., N-1$ for each trajectory:

$$\hat{C}_n^{\Theta,k} = \hat{h}(t_n^k, y_n^k), \quad k = 1, 2, ..., K_U.$$

Predict $Z_n^\Theta, n = 1, 2, ..., N$ for each trajectory:

$$\hat{Z}_n^{\Theta,k} = f^{\theta_n}(x_n^k)g(n, y_n^k) + (1 - f^{\theta_n}(y_n^k))\hat{C}_n^{\Theta,k}, \quad k = 1, 2, ..., K_U.$$

4. Estimate M_n^Θ for each trajectory:
Calculate the increments $\Delta M_n^{\Theta,k} = \hat{Z}_n^{\Theta,k} - \hat{C}_n^{\Theta,k}, \quad n = 1, 2, ..., N.$

$$M_n^{\Theta,k} = \begin{cases} 0 & \text{if } n = 0 \\ \sum_{m=1}^n \Delta M_m^{\Theta,k} & \text{if } n \geq 1 \end{cases}.$$

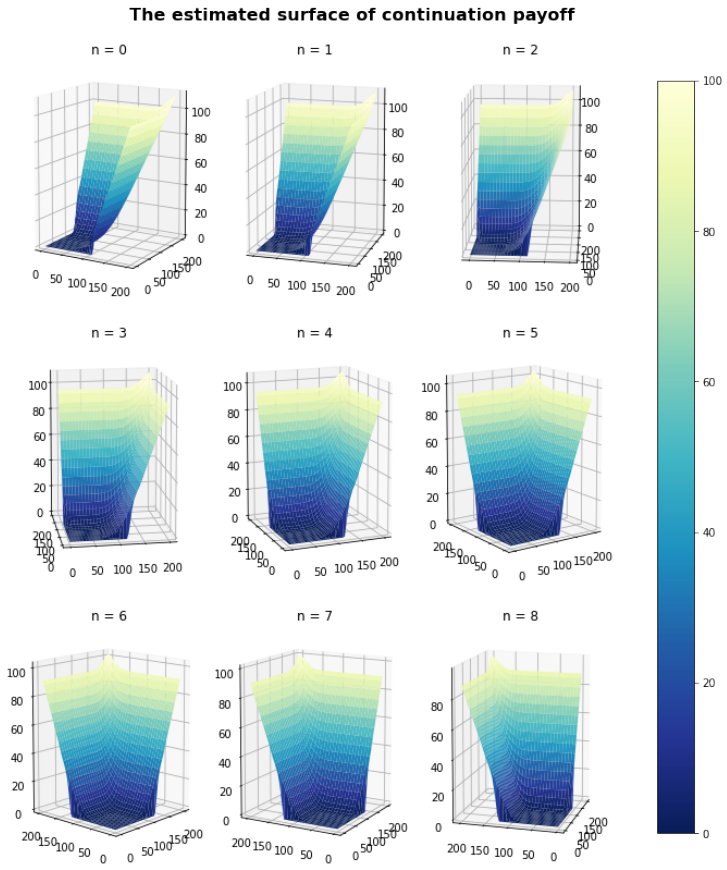

Figure 11: The estimated optimal continuation payoff $\hat{C}_n^\Theta$ given by $\hat{h}(t_n, x)$ for $n = 0, 1, ..., N-1$. forms a smooth surface in 2-D case; The fitness at the corner is suspicious possibly because of limitation of sample size.

The trained regression surfaces $\hat{h}(t_n, x), n = 0, 1, ..., N-1$ are visualized with rotation in Figure 11. The estimations of the lower bounds and the upper bounds for the Bermudan max-call option with initial states $s_0 = [90, 90], [100, 100], [110, 110]$, parameters $\delta_i = \delta = 0.1$, $\sigma_i = \sigma = 0.2$ for all $i = 1, ..., d$, and terminal time $t_N = T = 3$, $N = 9$ share the trained generablizable decision boundaries and the trained generablizable continuation payoff surface since the dynamics are identical.

d	s_0	$\hat{L}$	$\hat{U}$	Scheme	Reference $[\hat{L}, \hat{U}]$
2	[90,90]	8.0626	8.1077	Backward	[8.072, 8.075]
2	[100,100]	13.8753	14.0854	Backward	[13.895, 13.903]
2	[110,110]	21.3298	21.9034	Backward	[21.353, 21.346]
2	[90,90]	8.0671	8.1875	One-shot	
2	[100,100]	13.8973	14.4461	One-shot	
2	[110,110]	21.3355	22.3181	One-shot	

Table 1: Compared to the result in Becker et al. [2018],the estimated upper bounds are significantly looser due to limitation of sample size (25 GB CPU RAM); We use K_{train} = 30,000,000 trajectories to fit the regression, which is used to derive upper bounds for the 3 cases. The estimation for upper bounds in Becker et al. [2018] essentially use about 3 times the amount of samples for each case ($K_U = 1024$, continuation paths $J = 16,384$ for in total $N = 9$ time steps), in total, that is roughly 9 times amount of samples used for the three cases here. The lower bounds are much closer to their result, with difference only in the second decimal places.

6 Knowledge Distillation

Knowledge distillation was first proposed in Hinton et al. [2015] to compress cumbersome ensemble models into a simpler model that preserves most of the former's knowledge. In the recent study regarding deep learning, the knowledge distillation is often used to reduce computational costs associated with model evaluation so that even mobile devices gets to run a compact version of the large model without compromise on accuracy (e.g. mobile application of OCR, NLP).

In the context of deriving optimal stopping time by deep neural networks, knowledge distillation (Response-based knowledge) would make it possible to represent the approximated decision boundary using a much concise representation, which can be any sufficiently flexible models such as a two-layer neural network, SVM (support vector machine), Bayesian classifier, K-nearest neighbors. There are several benefits for doing this:

1. Reduce evaluation time, which could be possibly usable in real-time pricing application.

2. Increase interpretability of the models. For example, a neural network performs non-linear transformation on the data inputs, but the operations are too complex to interprete for business professionals, and therefore often regarded as 'black-box' models.

3. A simpler representation of the decision boundaries benefits further study of the evolution of the decision boundaries and make it possible to embed the approximated boundary in the other relevant study.

Here, we include the procedure of response-based knowledge distillation.

1. Uniformly simulate training data $X_n, n = 0, 1, ..., N$ from state space $\mathcal{X}$

2. Use the well-trained deep neural network, for example F^{θ_n}, to predict on the training data, record the partial decision as true label

3. Create a simpler model F'_n that takes the same data inputs as the deep neural network

4. Train F'_n by minimising the differences of the responses between $F'_n(X_n)$ and $F^{\theta_n}(X_n)$ (with possible regularization)

5. Modify F'_n to give decision in $\{0, 1\}$.

Particularly, the one-shot model can be regarded as a simpler model with continuity constraint added to the N independent neural networks. Therefore, one could distill the knowledge from the backward induction models to the one-shot model as a matter of course. In the way around, the one-shot model respect the continuity (when it holds) and does not suffer from over-fitting. Therefore, it is feasible to design a knowledge exchange structure. The procedure follows, simulate data to optimize the backward induction models $f^{\theta_n}, n = 0, 1, ..., N-1$, then distill its knowledge (the boundaries do not respect continuity) to the one-shot model ψ by minimizing their difference of partial decisions (the boundaries will be regularized with continuity constraint). After that, distill the knowledge back to f^{θ_n}. This procedure could be repeated and could possibly yield a more accurate boundary.

However, a more direct modification would be using backward induction scheme with architecture ψ so as to overcome the shortcoming that the previously stopped trajectories cannot effectively participate training for later time steps, by which one may derive better decision boundaries.

7 Summary

This dissertation explains and investigate two algorithms that derives underlying decision boundaries corresponding to the Snell envelope conditions. While the backward induction algorithm exactly follows the main reference Becker et al. [2018] where the algorithm is executed for every initial states of interest, we deviate from replicating their work in order to explore the possibility of fitting generalizable models, such that a well-trained model(one-shot) or a set of models (backward induction) can estimate the optimal payoff for a range of initial states X_0 of the Markov process X. As shown by visualization in 2-D case, we conclude that with suitable sampling method, it is feasible to derive accurate decision boundaries in a larger subset of state space $\mathcal{X}$. We also point it out that designing such a sampling method often requires knowledge and observation of the underlying decision, and therefore, it is useful to simplify representation of a trained deep neural networks by knowledge distillation in section 6.

In the estimation of upper bounds, we use regression by neural network in order to preserve generalizability. However, the upper bounds are significantly looser than those in the reference, which is due to limitation of sample size, which could be possibly overcome with better management of CPU RAM and more powerful computational platform.

Beyond what have been programmed, there are possible future work that may improve the precision of boundary approximation, but not implemented because of limitation of time. The first is intelligently manipulating the drift of trajectories by change of measure following the Girsanov's theorem as mentioned in section 4.2.1 while we test the method with a constant change on the wiener process. If successful, this method can largely decrease demand of sample size. The second is possible modification on the training scheme, considering the drawbacks of the two explained scheme - backwards induction scheme ignores the continuity constraint and overfits the training data while the one-shot scheme cannot efficiently use previously stopped trajectories. It would of interest to let the two methods complement each other, for which one could try train the two models together with a knowledge-exchange structure or directly use the architecture of ψ in backward induction as described in section 6.

Appendices

A Snell Envelope

The following theorems and proof was given by Peskir and Shiryaev [2006] and taught by Dr. Yvain Bruned.

Definition A.1. *As X is an adapted, integrable process, with the assumption that g is integrable. Process $g(X)$ is also adapted and integrable. The process Z defined by*

$$Z_n = \begin{cases} g(N, X_N) & n = N \\ max\{g(n, X_n), \mathbb{E}(Z_{n+1}|\mathcal{F}_n)\} & n \leq N-1 \end{cases},$$

is an Snell envelope of $(g(n, X_n))_{n\geq 0}^N$.

Theorem A.1. *The Snell envelope Z of $g(X)$ is a supermartingale, and is the smallest supermartingale that dominates X, i.e. $Z_n \geq g(X_n), \forall n$.*

Proof. By definition,

$$\begin{aligned} &Z_n \geq g(n, X_n), && \textbf{(domination)} \\ &Z_n \geq \mathbb{E}(Z_{n+1}|\mathcal{F}_n). && \textbf{(supermartingale)} \end{aligned}$$

It follows to prove that this process is the smallest among all the supermartingales that dominates $g(X)$. Let $Y = (Y_n)$ be another supermartingale dominating $g(X)$. We can complete the remained proof by mathematical induction - From $Y_N \geq g(X_N) = Z_N$. Assume $Y_n \geq Z_n$, then

$$\begin{aligned} &Y_{n-1} \geq \mathbb{E}(Y_n|\mathcal{F}_{n-1}) && (Y\, supermartingale) \\ &\geq \mathbb{E}(Z_n|\mathcal{F}_{n-1}) && \text{(by assumption)} \\ &\geq max\{g(n-1, X_{n-1}), \mathbb{E}(Z_n|\mathcal{F}_{n-1})\} && (Y_{n-1} \geq g(n-1, X_{n-1})) \\ &= Z_{n-1}. \end{aligned}$$

That is $Y_n \geq Z_n$, $\forall n$. $(Z_n)_{n\geq 0}$ is indeed the smallest supermartingale that dominates $(g(n, X_n))_{n\geq 0}$.

□

Lemma A.1. *The random variable $\tau^* := \inf\{n \geq 0 : Z_n = g(X_n)\}$ is a stopping time, and the stopped process $Z^{\tau^*} = \{Z_{n\wedge\tau^*}\}_n$ is a martingale.*

Proof. Since $Z_N = X_N$, τ^* is well-defined (Unique and bounded). To verify it is a stopping time, we first show $\{\tau^* = k\} \in \mathcal{F}_k$, $\forall k \geq 0$.

$$\begin{aligned} &\text{For k} = 0, \{\tau^* = 0\} = \{Z_0 = g(X_0)\} \in \mathcal{F}_0 \\ &\text{For k} \geq 1, \{\tau^* = k\} = \{z_0 > g(X_0)\} \cap ... \cap \{z_{k-1} > g(X_{k-1})\} \cap \{z_k = g(X_k)\} \in \mathcal{F}_k, \end{aligned}$$

since every event in the above equation is $\mathcal{F}_k$-measurable. On top of that,

$$\begin{aligned} &\mathbb{E}(Z_{n+1}^{\tau^*} - Z_n^{\tau^*}|\mathcal{F}_n) \\ &= \mathbb{E}(Z_{(n+1)\wedge\tau^*} - Z_{n\wedge\tau^*}|\mathcal{F}_n) \\ &= \mathbb{E}(\mathbf{1}_{\{n+1\leq\tau^*\}}(Z_{n+1} - Z_n)|\mathcal{F}_n) \quad (\text{Since } Z_n^{\tau^*} = Z_0 + \Sigma_{i=1}^n \mathbf{1}_{\{i\leq\tau^*\}}(Z_i - Z_{i-1})) \\ &= \mathbb{E}(\mathbf{1}_{\{n+1\leq\tau^*\}}(Z_{n+1} - \mathbb{E}(Z_{n+1}|\mathcal{F}_n))|\mathcal{F}_n) \quad (\text{Since on the event})\{n+1 \leq \tau^*\}, Z_n = \mathbb{E}(Z_{n+1}|\mathcal{F}_n) > g(X_n) \\ &= \mathbf{1}_{\{n+1\leq\tau^*\}}\mathbb{E}(Z_{n+1} - \mathbb{E}(Z_{n+1}|\mathcal{F}_n)|\mathcal{F}_n) = 0. \quad (\text{By Tower Property and } 1_{\{n+1\leq\tau^*\}} \text{ is predictable}) \end{aligned}$$

□

Theorem A.2. *For all $n \leq N$, let $\mathcal{T}_n$ be the set of stopping times taking values in $\{n, n+1, ..., N\}$. Let $\tau_n^* := \inf\{j \geq n : Z_j = g(j, X_j)\}$. we have that*

$$Z_n = \mathbb{E}(g(\tau_n^*, X_{\tau_n^*})|\mathcal{F}_n) = \sup_{\tau \in \mathcal{T}_n} \mathbb{E}(g(\tau, X_\tau)|\mathcal{F}_n)$$

Proof.

$$Z_n \geq \mathbb{E}(Z_\tau|\mathcal{F}_n) \geq \mathbb{E}(g(\tau, X_\tau)|\mathcal{F}_n)$$

Since (Z_n) is a supermartingale, and by the construction of Snell Envelope.
That is

$$\begin{aligned} Z_n &\geq \sup_{\tau \in \mathcal{T}_n} \mathbb{E}(g(\tau, X_\tau)|\mathcal{F}_n) \\ Z_n &= Z_n^{\tau_n^*} = \mathbb{E}(Z_{\tau_n^*}|\mathcal{F}_n) = \mathbb{E}(g(\tau_n^*, X_{\tau_n^*})|\mathcal{F}_n)) \quad \text{By Lemma 1} \end{aligned}$$

Therefore, we have

$$Z_n = \mathbb{E}(g(\tau_n^*, X_{\tau_n^*})|\mathcal{F}_n) = \sup_{\tau \in \mathcal{T}_n} \mathbb{E}(g(\tau, X_\tau)|\mathcal{F}_n) \quad \text{since } \sup_{\tau \in \mathcal{T}_n} \mathbb{E}(g(\tau, X_\tau)|\mathcal{F}_n) \geq \mathbb{E}(g(\tau_n^*, X_{\tau_n^*})|\mathcal{F}_n).$$

□

Remark: The snell envelop Z_n indicates the optimal continuation payoff. That is to say, we achieve optimal stopping when the current payoff meets the optimal continuation payoff.

Corollary 2. *Particularly, $\mathcal{F}_0 = \{\emptyset, \Omega\}$,*

$$Z_0 = \mathbb{E}g(\tau^*, X_{\tau^*}) = \sup_{\tau \in \mathcal{T}} \mathbb{E}g(\tau, X_\tau)$$

$\tau_0^* := \inf\{j \geq 0 : Z_j = g(j, X_j)\}$

Remark. If we write $Z_n = g(n, X_n)\mathbf{1}_{n=\tau_n^*} + \mathbb{E}(g(\tau_{n+1}^*, X_{\tau_{n+1}^*})|\mathcal{F}_n)(1 - \mathbf{1}_{n=\tau_n^*})$, by partial averaging property and $\mathbf{1}_{n=\tau_n^*}$ is $\mathcal{F}_n$-measurable, we have,

$$\mathbb{E}Z_n = \mathbb{E}[g(n, X_n)\mathbf{1}_{n=\tau_n^*} + \mathbb{E}(g(\tau_{n+1}^*, X_{\tau_{n+1}^*})(1 - \mathbf{1}_{n=\tau_n^*})]$$

this expression reveals the construction of the loss function.

B Figures

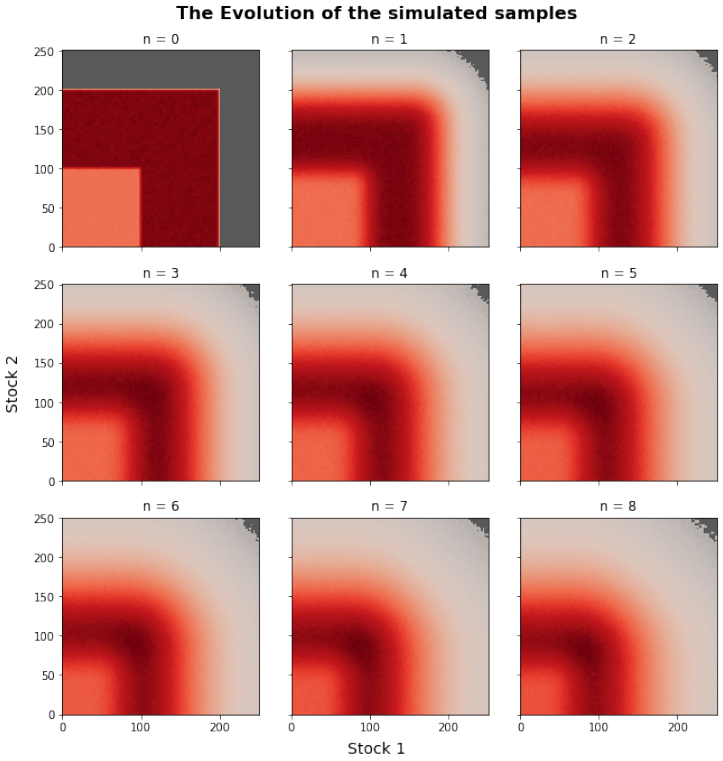

Figure 12: The initial states X_0 uniformly distributed in each square block. On purpose, we decrease the density in the [0,100,0,100] area where the redness are lighter as they are below the strike price and are considered less important.

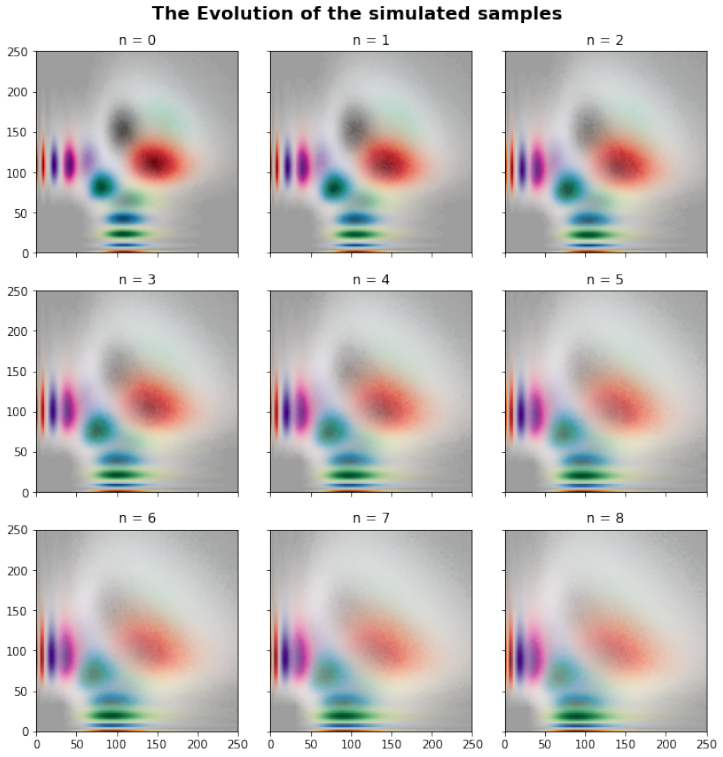

Figure 13: Sampling from several diffusion process with different initial states $[s_0^1, s_0^2]$. Then intercept the processes from n = 3 so that the samples spread over the state space since $n = 0$.

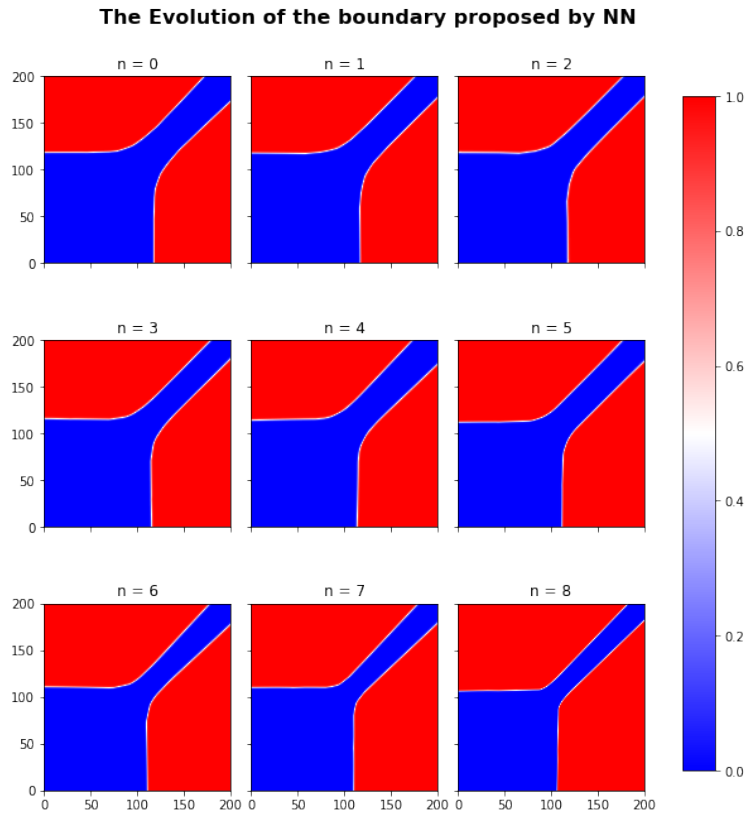

Figure 14: Sampling method - Several diffusion processes intercepted from time step 3 as described in 13; Algorithm: backward induction scheme; The approximated boundaries are now reliable in a larger state space for all time point.

References

Abien Fred Agarap. Deep learning using rectified linear units (relu), 2018. URL https://arxiv.org/abs/1803.08375.

Leif Andersen and Mark Broadie. Primal-dual simulation algorithm for pricing multidimensional american options. *Manag. Sci.*, 50:1222–1234, 2004.

Sebastian Becker, Patrick Cheridito, and Arnulf Jentzen. Deep optimal stopping. 2018. doi: 10.48550/ARXIV.1804.05394. URL https://arxiv.org/abs/1804.05394.

SEBASTIAN BECKER, PATRICK CHERIDITO, ARNULF JENTZEN, and TIMO WELTI. Solving high-dimensional optimal stopping problems using deep learning. *European Journal of Applied Mathematics*, 32(3):470–514, apr 2021. doi: 10.1017/s0956792521000073. URL https://doi.org/10.1017%2Fs0956792521000073.

Denis Belomestny. On the rates of convergence of simulation based optimization algorithms for optimal stopping problems, 2009. URL https://arxiv.org/abs/0909.3570.

B. C. Brookes. Foundations of the theory of probability. by a. n. kolmogorov. pp. 71, viii. 2.50. 1950. (chelsea publishing co., new york). *Mathematical gazette*, 35(314):292–293, 1951. ISSN 0025-5572.

Jacques F. Carriere. Valuation of the early-exercise price for options using simulations and nonparametric regression. *Insurance: Mathematics and Economics*, 19(1):19–30, 1996. ISSN 0167-6687. doi: https://doi.org/10.1016/S0167-6687(96)00004-2. URL https://www.sciencedirect.com/science/article/pii/S0167668796000042.

William S. Cleveland and Susan J. Devlin. Locally weighted regression: An approach to regression analysis by local fitting. *Journal of the American Statistical Association*, 83(403): 596–610, 1988. ISSN 0162-1459.

Tiziano De Angelis and Gabriele Stabile. On lipschitz continuous optimal stopping boundaries, 2017. URL https://arxiv.org/abs/1701.07491.

Ton Dieker. *Simulation of fractional Brownian motion*. PhD thesis, Masters Thesis, Department of Mathematical Sciences, University of Twente . . . , 2004.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015. URL https://arxiv.org/abs/1503.02531.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014. URL https://arxiv.org/abs/1412.6980.

Leonid Kogan and Martin Haugh. Pricing american options: A duality approach. *Operations Research*, 52, 12 2001. doi: 10.2139/ssrn.294821.

Moshe Leshno, Vladimir Ya. Lin, Allan Pinkus, and Shimon Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks*, 6(6):861–867, 1993. ISSN 0893-6080. doi: https://doi.org/10.1016/S0893-6080(05)80131-5. URL https://www.sciencedirect.com/science/article/pii/S0893608005801315.

Bernt oksendal. *Stochastic Differential Equations: An Introduction with Applications*. Universitext. Springer Berlin / Heidelberg, Berlin, Heidelberg, 2014. ISBN 9783540047582.

G. Peskir and A. Shiryaev. *Optimal Stopping and Free-Boundary Problems*. Lectures in Mathematics. ETH Zürich. Birkhäuser Basel, 2006. ISBN 9783764373900. URL https://books.google.co.uk/books?id=UinZbLqpUDEC.

Goran Peskir. Continuity of the optimal stopping boundary for two-dimensional diffusions. *Annals of Applied Probability*, 29(1):505–530, 2018. ISSN 1050-5164. doi: 10.1214/18-AAP1426.

A. Max Reppen, H. Mete Soner, and Valentin Tissot-Daguette. Neural optimal stopping boundary, 2022. URL `https://arxiv.org/abs/2205.04595`.

L. C. G. Rogers. Monte carlo valuation of american options. *Mathematical finance*, 12(3): 271–286, 2002. ISSN 0960-1627.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature (London)*, 323(6088):533–536, 1986. ISSN 0028-0836.

Gunnar Taraldsen. Optimal learning from the doob-dynkin lemma. 2018. doi: 10.48550/ARXIV. 1801.00974. URL `https://arxiv.org/abs/1801.00974`.

James Tilley. 111 valuing american options in a path simulation model. *Transactions of the Society of Actuaries*, 45, 01 1993.

Benjamin Virrion. Deep importance sampling, 2020. URL `https://arxiv.org/abs/2007.02692`.

D. (David) Williams. *Probability with martingales / David Williams.* Cambridge mathematical textbooks. Cambridge University Press, Cambridge, 1991. ISBN 9780521406055.