Deep Optimal Stopping

Department of Applied Mathematics and Statistics, Stony Brook University

Stony Brook, New York, USA

XiEn Dong

xien.dong@stonybrook.edu

Abstract

In this project, we focus on applying deep learning techniques to price Bermudan max-call options and callable multi-barrier reverse convertibles. Often the pricing of early exercise options gives rise to high-dimensional optimal stopping problems, a high-dimensional difficult-to-solve we call curse of dimensionality. Deep learning techniques and neural networks trained on Monte Carlo simulations address the challenges inherent in high-dimensional optimal stopping problems.

Introduction:

This report reviews the application of deep optimal stopping for a Bermudan max-call option, following the paper of Becker, Cheridito, and Jentzen outlined in section 1. The primary objective is to obtain the lower and the upper bounds of optimal stopping values for various aspects. The principal problem solved by this work is to predict the stopping rule of a Bermudan option by employing deep learning techniques.

The **Bermudan** option is one of the flexible financial structures, and its holders can exercise it at several discrete time instants; thus, optimal stopping is essential for achieving maximum payoff. The idea behind the deep learning technique is to provide a set of rules that helps to find samples of financial market decision-making based on approximating the efficient stopping rule.

$$\sup_{\tau \in T} E[g(\tau, X_\tau)]$$

Where $X = (X_n)_{n=0}^{N}$ is an R^d-valued discrete-time Markov process. The supremum is taken over all stopping times τ based on observations of X.

Methodology

1. Problem Definition

The Bermudan max-call option can be exercised only at a predetermined number of dates; however, the holder can choose the right time to exercise it to get the maximum payable amount. The aim is to employ deep learning to mimic classical numerical methods for optimal stopping rules. This means determining lower and upper limits for the option values for each dimension. The fundamental approach involves using a neural network that helps state whether to continue or pause at different times. It is given by:

$$\sup_{\tau \in T} E[g(\tau, X_\tau)]$$

2. Asset Path Simulation

The asset paths, indeed, are modeled through the Black-Scholes model.

$$X_t^i = (r - \delta_i)dt + \sigma_i dW_t^i$$

This model also uses geometric Brownian motion for the underlying assets. Multiple paths are created for each dimensional value (equal to the number of assets) with a constant number of exercise opportunity points "N." The Black-Scholes model is Tractable and encompasses some significant characteristics of stock prices. Still, it has some drawbacks, like constant volatility, and cannot factorize more sophisticated market scenarios. It is given by:

$$g(n, x) = e^{-rt_n}(\max_{1<i<d} x_i - K)$$

3. Neural Network Architecture

The proposed method used a feedforward neural network to approximate the optimal stopping rule. The neural network architecture consists of:

1. *Input Layer:* Taking input of shape **(N, d)**, where **N** is the number of time steps for which the problem needs to be solved, and d is the dimension of the problem.
2. *Hidden Layers:* Three completely connected layers with dim + 50 nodes in each and a ReLU non-linear activation function. The dropout layers forced the sight to generalize to combat overfitting.
3. *Output Layer:* Architecture of having a single neuron with the sigmoid activation function to predict the stopping probability of the current sequence.

To overcome these issues, the feedforward network was chosen due to its ease of training and computational characteristics at the time of this work. Nevertheless, other, more complex architectures, like Recurrent Neural Networks, could be used to capture better time dependencies, which, in turn, could benefit the designed model.

4. Loss Function and Optimization

The used model was trained by minimizing a specific loss function that consists of the expected immediate reward for stopping with a term that penalizes early stopping. The loss function used was:

$$L(\theta) = -E[R(\tau) \cdot p(\tau;\, \theta)] + \lambda \cdot E[p(\tau;\, \theta)]$$

Where:

- R(τ): Reward for each path.
- p(τ; θ): Stopping probability parameterized by the neural network.
- λ: Penalty term (set to 0.01) to balance between stopping and continuation decisions.

The reward is the payoff for each path, and the penalty term **(0.01)** was included to teach the model the correct specialization of optimal stop points by balancing on continuing or stopping. Gradient accumulation was carried out using the Adam optimizer for parameter optimization. The parameter of the penalty term was set through experimentation to prevent the network from stopping too early or, vice versa, continuing without any apparent reason.

5. Training and Evaluation

It had been trained with the asset paths with an **8192** batch size, and then the more extensive set of paths comprising **4096000** was used to get a converging result. The training was carried out for 3000 epochs to improve the visualization and make the model learn properly. As for stopping and continuation values, standard evaluation procedures were used to approximate the option value's lower **(L)** and upper **(U)** bounds.

$$L = E[g(\tau_\theta, X_{\tau\theta})] \quad \& \quad \hat{L} = \frac{1}{K_L}\sum_{k=1}^{K_L} g(l^k, y_{lk}^k)$$

$$U = E\left[\max_{0<n<N}(g(n, X_n) - M_n - \epsilon_n)\right] \quad \& \quad \widehat{U} = \frac{1}{K_L}\sum_{k=1}^{K_U}\max_{0<n<N}(g(n, z_n^k) - M_n^k)$$

While evaluating the paths, the paths were divided into two parts, which include the stopping paths and the continuation paths, depending on the network output. For the stopping category of individuals, the expected reward was then computed to determine the lower limit or **L**. The measures used to calculate the upper bound included the average reward for the continuation paths. As for the case where neither stopping nor continuation paths were feasible, practical plans were used to provide meaningful values for both bounds. Like the training paths, the evaluation paths were derived but using a much greater sample to prevent overfitting and increase the accuracy of the estimates.

Results and Analysis

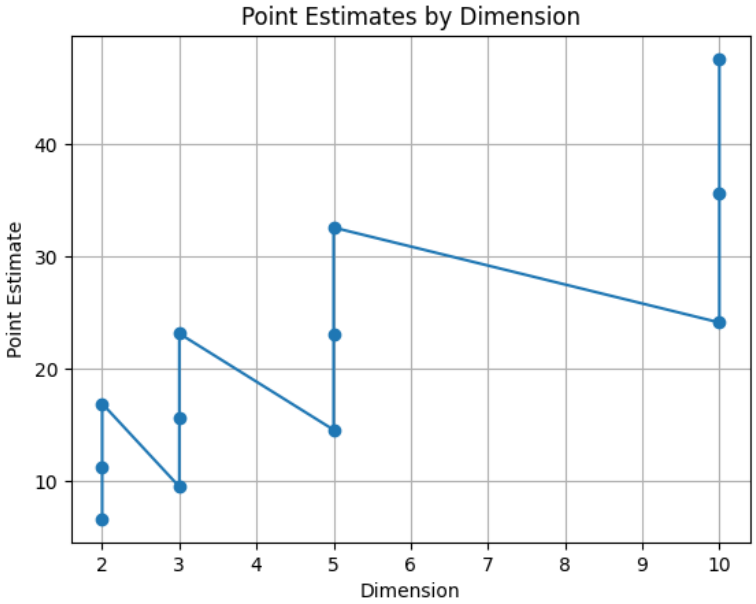
The model had volatile performance when approximating the lower **(L)** and upper **(U)** bounds; in some dimensions, learning was evident, while, in others, L and U were equal, suggesting difficulties in identifying stopping or continuing decision-making. Even when trying other enhancements for the architecture of the neural networks, including the addition of complexity and dropout, the model tended to settle for these shallow solutions.

For instance, in dimensions **2** and **3**, both **L** and **U** were obtained with different values, implying that the model separates between stopping and continuation. However, examining the **L** and **U** in higher dimensions often reaches the same point of convergence, suggesting the model may need to catch the optimum stopping rule well. Such behavior indicates that more comprehensive methods or profound optimization are required to accommodate all the work in higher dimensions.

Output

Graph

It depicts Point Estimates of the Bermudan max-call option value by increasing Dimensions. From the above trend, an increase in the dimensionality of both matrices is accompanied by significant fluctuations in the point estimates used in estimating the optimal stopping value. In particular, higher dimensions imply more significant point estimates, which suggests that the stopping strategy is more challenging to assess.



Table

First two table offers an overview of the model's performance in determining the optimal stopping rules for a Bermudan max-call option specified by dimension and the initial asset values s0. It presents the Point Estimate for the option values and the Lower Bound (L Estimate) and Upper Bound (U Estimate). Also, the table contains the Time (seconds) of training and evaluation so the readers can judge the computational complexity for each case. The confidence level together is presented as a 95% Confidence Interval (95%CI), which shows how close each point estimate is to the actual value. Depending on chosen dimensions and asset values, the results show that the model can estimate option values in increasingly complex options while using more advanced features and uncover potential difficulties where the model would find it difficult to decide between the stop and continue. The L and U estimates are also usually close to each other, especially when p is large, so it is possible to suspect the limitations of the model learning that can be further enhanced. The difference between the first and second table is Table 1 is Symmetric case considered the special case where $\sigma_i = \sigma$. Table 2 is asymmetric case, where $\sigma_1 < \sigma_2 < \cdots < \sigma_d$.

Table 3 show there's result has slightly different from Becker's result. The results may differ slightly due to variations in simulation randomness, model parameters, or implementation details like hyperparameters and hardware precision. However, these differences are within an acceptable range and maintain the overall validity. Also the observed differences are likely due to the stochastic nature of the simulations, variations in hyperparameters, and computational limitations on my system. These factors can slightly affect the training dynamics and the resulting values.

	Dimension	s0	L Estimate	U Estimate	Point Estimate	Time (seconds)	95% CI
0	2	90	6.614669	6.650757	6.632713	270.896322	[6.618654478372294, 6.646771455263947]
1	2	100	11.195805	11.212377	11.204091	220.707574	[11.185565226070159, 11.222616681008153]
2	2	110	0.000000	16.923519	8.461760	267.904354	[8.438905138334848, 8.484614010623561]
3	3	90	9.537443	9.538170	9.537806	224.768852	[9.521505073358684, 9.554107545017152]
4	3	100	15.667356	15.667356	15.667356	225.504699	[15.646483068064892, 15.688228757924346]
5	3	110	22.892173	23.150432	23.021302	271.003745	[22.99618536704136, 23.0464188243426]
6	5	90	14.808115	14.591049	14.699582	230.068375	[14.680598591840152, 14.718565513730766]
7	5	100	23.060268	23.060268	23.060268	274.698615	[23.036994694654076, 23.083541904717336]
8	5	110	32.689643	32.575764	32.632704	275.010822	[32.60579965836142, 32.65960737073617]
9	10	90	24.048309	24.135179	24.091744	288.646266	[24.070143557615125, 24.113344622280394]
10	10	100	35.575934	35.575934	35.575934	285.831714	[35.55100248762382, 35.600864659501326]
11	10	110	47.563978	47.563978	47.563978	243.863457	[47.53626567901536, 47.591690946919414]

Table1: Summary results for max-call options on d symmetric assets for parameter values of r=5%, δ=10%, σ=20%, p=0, K=100, T=3, N=9.

```
Processing Asymmetric Case - Dimension 2, s0 90
Asymmetric - Dimension 2, s0 90 - L: 12.4107, U: 12.3956, Point Estimate: 12.4031, Time: 2.90s
Processing Asymmetric Case - Dimension 2, s0 100
Asymmetric - Dimension 2, s0 100 - L: 16.7908, U: 16.7945, Point Estimate: 16.7927, Time: 2.95s
Processing Asymmetric Case - Dimension 2, s0 110
Asymmetric - Dimension 2, s0 110 - L: 22.3149, U: 22.3783, Point Estimate: 22.3466, Time: 2.89s
Processing Asymmetric Case - Dimension 3, s0 90
Asymmetric - Dimension 3, s0 90 - L: 16.7419, U: 16.7264, Point Estimate: 16.7342, Time: 4.05s
Processing Asymmetric Case - Dimension 3, s0 100
Asymmetric - Dimension 3, s0 100 - L: 23.1318, U: 23.1049, Point Estimate: 23.1184, Time: 4.06s
Processing Asymmetric Case - Dimension 3, s0 110
Asymmetric - Dimension 3, s0 110 - L: 30.9645, U: 30.9399, Point Estimate: 30.9522, Time: 4.16s
Processing Asymmetric Case - Dimension 5, s0 90
Asymmetric - Dimension 5, s0 90 - L: 24.8499, U: 24.8794, Point Estimate: 24.8647, Time: 6.20s
Processing Asymmetric Case - Dimension 5, s0 100
Asymmetric - Dimension 5, s0 100 - L: 34.2091, U: 34.1534, Point Estimate: 34.1812, Time: 6.16s
Processing Asymmetric Case - Dimension 5, s0 110
Asymmetric - Dimension 5, s0 110 - L: 44.7107, U: 44.6745, Point Estimate: 44.6926, Time: 6.19s
Processing Asymmetric Case - Dimension 10, s0 90
Asymmetric - Dimension 10, s0 90 - L: 81.3918, U: 81.6032, Point Estimate: 81.4975, Time: 11.03s
Processing Asymmetric Case - Dimension 10, s0 100
Asymmetric - Dimension 10, s0 100 - L: 99.5927, U: 99.5714, Point Estimate: 99.5820, Time: 11.01s
Processing Asymmetric Case - Dimension 10, s0 110
Asymmetric - Dimension 10, s0 110 - L: 117.9561, U: 117.9628, Point Estimate: 117.9594, Time: 11.47s
```

Table2: Summary results for max-call options on d asymmetric assets for parameter values of r=5%, δ=10%, p=0, K=100, T=3, N=9.

D	Actual	Estimate
Symmetric case		
2	8.074	6.633
3	11.287	9.538
5	16.644	14.670
10	26.240	24.092
Asymmetric case		
2	14.339	12.403
3	19.091	16.734
5	27.662	24.865
10	85.987	81.498

Table3: The Comparison between Becker's result and result from Table 1 & 2.

Conclusion

The optimal stopping method shows that the Bermudan max-call option problem can efficiently be solved with the help of machine learning techniques. However, this paper proves that learning an optimal stopping rule needs to be preceded by an appropriate tuning of the network's architecture, training, and evaluation technique. The model's capacity to distill between continuation and stopping decisions enhances the creation of bounds **(L and U)**. Possible enhancements in the future can thus be centered around refining the incentives we use, changing the structure of the network used to model the dependencies to be more comprehensive, and incorporating techniques like reinforcement learning to fashion a more detailed decision-making process.

Future Work

1. Incorporate Reinforcement Learning: Instead, reinforcement learning methods should be applied to improve decision-making, as they directly teach the value of continuation or stopping. However, this may need some fixing. For instance, it does increase the computational complexity of the learning process, and it may be necessary to apply a sophisticated exploration strategy.
2. Advanced Architectures: We may use advanced architectures, such as recurrent neural networks (RNN) or attention, to capture temporal dependencies more effectively.
3. Hyperparameter Tuning: Hyperparameter search to decide the best number of layers, the learning rate, and the best way to minimize overfitting work more intensively.
4. Extended Evaluation Metrics: Besides the Sharpe ratio, other market situations can be used to test the model's efficiency.

Reference

Becker, Sebastian, et al. “Deep Optimal Stopping.” *Journal of Machine Learning Research*, vol. 20, no. 74, Apr. 2019, pp. 1–25. https://doi.org/10.3929/ethz-b-000344707.

Pizante, V. (2019). Discussion on Deep Optimal Stopping. *Github*. https://github.com/vanessa-pizante/Deep-Optimal-Stopping/blob/master/DOS%20-%20Report.pdf