Deep surrogate for Zero-coupon bond pricing

Ahmad Roukain

May 2024

1 Abstract

This paper presents a comprehensive methodology for modeling and predicting zero-coupon bond prices using the Vasicek model and neural network approaches.

The implementation leverages significant computational power, emphasizing the importance of high-performance computing in financial trading. Rapid and accurate pricing tools are crucial for traders to respond to market changes effectively and maintain a competitive edge. Our model demonstrates the potential for advanced machine learning techniques to enhance financial decision-making processes by providing quick bond pricing estimates. This capability is essential in the fast-paced trading environment, where timely and precise information is paramount.

2 Introduction

The realm of financial derivatives, particularly option pricing, presents significant computational challenges, often exacerbated by the curse of dimensionality. This issue arises as the number of variables in a model increases, leading to an exponential growth in computational complexity. Traditional methods, such as Monte Carlo simulations and finite difference methods, struggle to manage this complexity efficiently, especially in high-dimensional settings or when dealing with intricate models and path dependencies.

Deep surrogate models offer a promising solution to these challenges. These models leverage deep learning to approximate complex financial models, significantly reducing computational time while maintaining high accuracy. By learning the intricate relationships within the data, deep surrogates can provide rapid predictions for option prices, even in scenarios with high dimensionality and complex dependencies.

The application of deep surrogate models in option pricing not only mitigates the curse of dimensionality but also enhances the computational efficiency

crucial for real-time trading and risk management. This paper explores the integration of deep surrogate models in financial analytics, demonstrating their potential to revolutionize the approach to complex option pricing problems. Through the use of neural networks, we can achieve fast, reliable pricing tools that are indispensable for modern financial markets.

3 Relevant literature

The study by Chen, Didisheim, and Scheidegger (2023) introduces deep surrogates, which are high-precision approximations of structural models based on deep neural networks. They demonstrate that these surrogates significantly accelerate model evaluation and estimation, enabling compute-intensive applications that were previously infeasible.

Complementing this, Gerebrink et al. (2018) explore the Maximum Likelihood calibration of the Vasicek model to the Swedish interest rate market. Their work focuses on the practical application of the Vasicek model in capturing the dynamics of interest rates and highlights the computational challenges associated with model calibration.

4 Methodology

4.1 Vasicek model

In this study, we employ the Vasicek model to simulate interest rate paths and estimate zero-coupon bond prices. The methodology comprises two primary stages: parameter estimation using Maximum Likelihood Estimation (MLE) and Monte Carlo simulation for bond pricing.

First, we estimate the parameters of the Vasicek model, specifically the speed of reversion (a), the long-term mean rate (b), and the volatility (σ). We define the log-likelihood function for the Vasicek model as follows:

$$\mathcal{L}(\theta) = -\frac{1}{2}\sum_{i=1}^{n}\left[\log(2\pi\sigma_v^2) + \frac{(r_i - \mu_i)^2}{\sigma_v^2}\right] \tag{1}$$

where $\mu_i = b + (r_{i-1} - b)\exp(-a\Delta t)$ and $\sigma_v = \sqrt{\left(\frac{\sigma^2}{2a}\right)(1 - \exp(-2a\Delta t))}$. We use historical interest rate data and the `minimize` function from the `scipy.optimize` package to maximize the log-likelihood function and obtain the parameter estimates.

Next, we utilize the estimated parameters to simulate interest rate paths. We simulate $10,000$ paths with 100 steps each, assuming a daily time step $\Delta t = \frac{1}{252}$. The simulation follows the Vasicek model's stochastic differential equation:

$$dr_t = a(b - r_t)\Delta t + \sigma\sqrt{\Delta t}\epsilon_t \tag{2}$$

where ϵ_t is a standard normal random variable. For each simulated path, we calculate the zero-coupon bond price using the following closed-form solution:

$$P(t,T) = A(t,T)\exp(-B(t,T)r_t) \tag{3}$$

with

$$B(t,T) = \frac{1-\exp(-a(T-t))}{a} \tag{4}$$

$$A(t,T) = \exp\left((b-\frac{\sigma^2}{2a^2})(B(t,T)-(T-t)) - \frac{\sigma^2 B(t,T)^2}{4a}\right) \tag{5}$$

4.1.1 Estimation test

The parameters of the model are estimated using the method of Maximum Likelihood Estimation (MLE). Once the parameter estimates $\hat{\theta} = (\hat{a}, \hat{b}, \hat{\sigma})$ are obtained, it is crucial to assess their precision and reliability. This involves computing the Hessian matrix, which approximates the curvature of the log-likelihood function around the estimated parameters.

To numerically approximate the Hessian matrix $\mathbf{H}$, finite difference methods are employed. The Hessian matrix is defined as the matrix of second-order partial derivatives of the log-likelihood function $\mathcal{L}(\theta)$:

$$\mathbf{H}_{ij} = \frac{\partial^2 \mathcal{L}(\theta)}{\partial\theta_i \partial\theta_j}.$$

For numerical approximation, a small step size $\Delta\theta_i$ proportional to the parameter estimates is used. Specifically, each element of the Hessian matrix is computed as:

$$\mathbf{H}_{ij} \approx \frac{\mathcal{L}(\hat{\theta}+\Delta\theta_i+\Delta\theta_j) - \mathcal{L}(\hat{\theta}+\Delta\theta_i-\Delta\theta_j) - \mathcal{L}(\hat{\theta}-\Delta\theta_i+\Delta\theta_j) + \mathcal{L}(\hat{\theta}-\Delta\theta_i-\Delta\theta_j)}{4\Delta\theta_i\Delta\theta_j}.$$

Once the Hessian matrix is approximated, its inverse provides the variance-covariance matrix $\mathbf{V}$ of the parameter estimates:

$$\mathbf{V} = \mathbf{H}^{-1}.$$

The diagonal elements of $\mathbf{V}$ yield the variances of the parameter estimates, from which the standard errors are derived:

$$\text{Std. Err}(\hat{\theta}_i) = \sqrt{\mathbf{V}_{ii}}.$$

To assess the statistical significance of the parameter estimates, t-statistics are computed as the ratio of the parameter estimates to their standard errors:

$$t_i = \frac{\hat{\theta}_i}{\text{Std. Err}(\hat{\theta}_i)}.$$

These t-statistics are used to test the null hypothesis that each parameter is equal to zero against the alternative hypothesis that it is not. The results, including parameter estimates, standard errors, and t-statistics, are summarized to facilitate interpretation and hypothesis testing.

4.2 Neural Network Model for Bond Pricing

In the subsequent phase of the study, we develop a neural network model to predict zero-coupon bond prices based on simulated data. This section outlines the preprocessing steps, neural network architecture, and training procedure, focusing on the choice of activation and loss functions.

First, we split the dataset into features and target variables. The features (X) include the interest rate, time, and maturity, while the target variable (y) is the bond price. The data is then divided into training, validation, and test sets using an 70-15-15 split. We normalize the features to standardize the data, ensuring that each feature has a mean of 0 and a standard deviation of 1.

The model consists of a sequential stack of dense layers with ReLU (Rectified Linear Unit) activation functions. The ReLU activation function is defined as $f(x) = \max(0, x)$, which allows the network to capture non-linear relationships by introducing non-linearity into the model. ReLU is computationally efficient and helps mitigate the vanishing gradient problem, which can occur with other activation functions like the sigmoid or tanh.

The architecture of the model includes an input layer with 128 neurons, followed by layers with 64 and 32 neurons, respectively. Each of these layers uses the ReLU activation function. The final output layer has a single neuron without an activation function, appropriate for regression tasks where the output is a continuous variable, such as bond prices.

The model is compiled with the Adam optimizer, a popular choice for training deep learning models due to its adaptive learning rate and efficient computation. The loss function used is the mean squared error (MSE), defined as:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{6}$$

where y_i is the actual bond price, and $\hat{y}_i$ is the predicted bond price. The MSE loss function is appropriate for regression tasks as it penalizes larger errors more significantly, thereby providing a clear measure of prediction accuracy.

We train the model on the training set with a batch size of 32, employing early stopping to halt training if the validation loss does not improve for 5 consecutive epochs. This technique helps prevent overfitting, ensuring that the model generalizes well to unseen data.

4.2.1 Model Evaluation

To evaluate the performance of our neural network model in predicting zero-coupon bond prices, we utilize several statistical metrics and visualizations. After training the model, we assess its accuracy on the test dataset using both the model's built-in evaluation method and additional performance metrics.

First, we compute the test loss using the mean squared error (MSE) as the loss function, which provides a measure of the average squared differences between the predicted and actual bond prices. The test loss is calculated as follows:

$$\text{Test Loss} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{7}$$

where y_i represents the actual bond prices, and $\hat{y}_i$ represents the predicted bond prices from the model.

Next, we predict bond prices on the test set and calculate several evaluation metrics:

- Mean Absolute Error (MAE): The average of the absolute differences between predicted and actual values, providing an easily interpretable error measure.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{8}$$

- Mean Squared Error (MSE): As mentioned earlier, it quantifies the average squared difference between predicted and actual values, heavily penalizing larger errors.

- R-squared (R^2): A statistical measure indicating the proportion of the variance in the dependent variable that is predictable from the independent variables. It is calculated as:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{9}$$

  where $\bar{y}$ is the mean of the actual bond prices.

These metrics provide a comprehensive view of the model's prediction accuracy, with lower MAE and MSE values indicating better performance and an R^2 value closer to 1 suggesting a strong predictive capability.

To visually inspect the model's performance, we create a scatter plot of the actual versus predicted bond prices. This plot includes a reference line ($y = x$) indicating perfect predictions, helping to visually assess the alignment between actual and predicted values. Ideally, the points should lie close to this reference line, indicating high prediction accuracy.

The evaluation results, including the computed metrics and the scatter plot, demonstrate the effectiveness of our neural network model in capturing the relationship between the features and the bond prices.

5 Data

The dataset used in this analysis comprises US Treasury yields, sourced from the Federal Reserve Economic Data (FRED) via the St. Louis Fed. It spans from January 1, 2004, to the present date, providing a comprehensive view of the interest rate environment over an extended period. The data was retrieved using the pandas_datareader library in Python. No particular cleaning was necessary, as it was a simple timeserie.

6 Implementation

This code was implemented in python. The data science workflow common libraries such as NumPy and pandas we used and we retrieved financial data using the pandas data reader. For the estimation of the Vasicek model parameters, we used SciPy's optimization to maximize the log likelihood function.

Subsequently, the simulated data is used to train a neural network model. Data is split into training, validation, and test sets, and normalized using scikit-learn. The neural network is built and trained using TensorFlow, with model evaluation performed to assess prediction accuracy.

Finally, performance metrics such as MAE, MSE, and R-squared are calculated, and a plot comparing actual and predicted bond prices is generated using the same libraries decscribed above and Plotly form Matplotlib.

7 Results

7.0.1 Parameters estimation

Parameter	**Estimate**	**Std. Err.**	**T-stat**
a	0.071688	0.121638	0.58936
b	0.044704	0.058515	0.76397
σ	0.010643	0.000103	103.11862

Table 1: Estimation results for parameters a, b, and σ.

The estimation results for the parameters a, b, and σ reveal varying degrees of statistical significance. The parameter a has an estimate of 0.071688 with a standard error of 0.121638, resulting in a T-statistic of 0.58936, indicating it is not statistically significant. Similarly, the parameter b has an estimate of 0.044704 and a standard error of 0.058515, yielding a T-statistic of 0.76397, which also suggests it is not statistically significant. On the other hand, the parameter σ stands out with an estimate of 0.010643 and a remarkably low standard error of 0.000103, leading to a T-statistic of 103.11862, highlighting

its high statistical significance. These results suggest that while a and b are not significant predictors, σ is a crucial and precise parameter in the model.

7.0.2 Deep surrogate

The machine learning model's performance on the test data is highly impressive. The evaluation metrics are as follows:

- **Test Loss:** $2.090818362887603 \times 10^{-8}$
- **Mean Absolute Error (MAE):** 0.0001312809589413363
- **Mean Squared Error (MSE):** $2.090810846159205 \times 10^{-8}$
- **R-squared (R^2):** 0.9999204215137494

The plot of actual versus predicted zero-coupon bond prices further demonstrates the model's accuracy. The predicted prices align very closely with the actual prices, as evidenced by the nearly perfect diagonal line in the scatter plot. The extremely low test loss and MSE indicate that the model has minimal error in its predictions. The MAE value of approximately 0.000131 further supports this, showing that the average deviation of the predicted prices from the actual prices is very small. The R^2 value of 0.9999204215137494 indicates that the model explains nearly all the variance in the data, highlighting its excellent fit and predictive power.

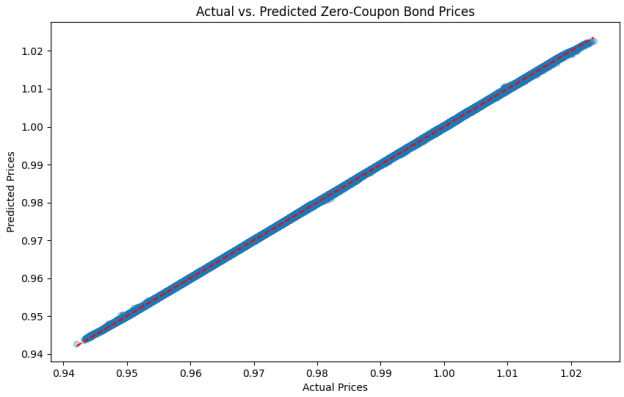

Figure 1: Actual vs. Predicted Zero-Coupon Bond Prices

7.0.3 Comparison with live data

Calculating the US treasury bills prices from the yields found online (worldgovernmentbonds.com, 26.05.2024), we calculated the current price using the following formula :

$$\mathrm{P} = \frac{1}{(1+y)^T} \tag{10}$$

we obtain the following results :

	Actual price	Predicted
US3M	0.986912	0.986948
US6M	0.974097	0.973989
US1Y	0.950471	0.948650

Table 2: Caption

We can observe that the prices found online and the predictions can be similar up to the fourth decimal place, although the accuracy diminishes for longer maturities. Nevertheless, it is important to remember that these types of instruments are typically traded in large volumes, with transactions reaching hundreds of millions for major institutional investors such as pension funds, companies, or asset managers. Thus, even the smallest decimal can make a significant difference, necessitating highly accurate pricing.

Despite this, it remains a very useful tool due to its rapid computation speed compared to simulating full paths, even for a simple model. For instance, in options pricing, some complex models require significantly more computation time. In our case, for simulating 10'000 paths of 252 steps took 53 seconds, while the surrogate took milliseconds.

8 References

Chen, X., Didisheim, T., & Scheidegger, B. (2023). [Deep Surrogates for Finance: With an Application to Option Pricing].
Retrieved from https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3782722

Gerebrink, R., Kopp, S., & Ringström, T. (2018). [Maximum Likelihood calibration of the Vasicek model to the Swedish interest rate market].
Retrieved from https://odr.chalmers.se/server/api/core/bitstreams/ac1c6e19-ecb6-4731-9ff7-ce496f447b0b/content

Kevin Sheppard. Score test.
Retrieved from https://www.kevinsheppard.com/code/matlab/mfe-toolbox/

Loss function. (n.d.). In Wikipedia.
Retrieved from https://en.wikipedia.org/wiki/Loss_function

Mean absolute error. (n.d.). In Wikipedia.
Retrieved from https://en.wikipedia.org/wiki/Mean_absolute_error

Mean squared error. (n.d.). In Wikipedia.
Retrieved from https://en.wikipedia.org/wiki/Mean_squared_error

R-squared. (n.d.). In Wikipedia.
Retrieved from https://en.wikipedia.org/wiki/Coefficient_of_determination

Yield data. (n.d.).
Retrieved from https://www.worldgovernmentbonds.com/country/united-states/: :text=The

Chat-GPT. (n.d.). Coding, commenting the code, phrasing, rephrasing.