

A Comparison of Traditional and Deep Learning Methods for Parameter Estimation of the Ornstein-Uhlenbeck Process

Jacob Fein-Ashley^{1*}

^{1*}Department of Electrical Engineering, University of Southern California, Los Angeles, CA, USA.

Corresponding author(s). E-mail(s): feinashl@usc.edu;

Abstract

We consider the Ornstein-Uhlenbeck (OU) process, a stochastic process widely used in finance, physics, and biology. Parameter estimation of the OU process is a challenging problem. Thus, we review traditional tracking methods and compare them with novel applications of deep learning to estimate the parameters of the OU process. We use a multi-layer perceptron to estimate the parameters of the OU process and compare its performance with traditional parameter estimation methods, such as the Kalman filter and maximum likelihood estimation. We find that the multi-layer perceptron can accurately estimate the parameters of the OU process given a large dataset of observed trajectories, and on average, outperforms traditional parameter estimation methods.

Keywords: Ornstein-Uhlenbeck Process, Deep Learning, Parameter Estimation, Tracking, Kalman Filter, Maximum Likelihood Estimation, MLP

1 Introduction

The OU process has broad applications for modeling systems in finance, physics, and biology [1]. Generally, the OU process is defined by the stochastic differential equation

$$dX_t = \theta(\mu - X_t)dt + \sigma dW_t,$$

where X_t is the state of the system at time t , θ is the rate of mean reversion, μ is the long-term mean, σ is the volatility, and W_t is a Wiener process.

Parameter estimation of the OU process is a challenging problem. Traditional methods of parameter estimation include maximum likelihood estimation (MLE) [2] and least squares estimation (LSE) [3]. These methods are computationally expensive and may not be suitable for real-time applications, as they require optimizing a likelihood function or solving a system of nonlinear equations.

Additional methods for parameter estimation of the OU process include the Kalman filter [4] and the Kalman filter with smoothing [5].

With the advent of deep learning, there has been a surge of interest in using neural networks to estimate the parameters of nonlinear equations and dynamical systems [6]. Because the OU process is nonlinear, traditional parameter estimation methods may not be suitable for real-time applications. Neural networks are capable of learning complex patterns in data, so they may be well-suited for estimating the parameters of the OU process. Thus, we

We outline the rest of the paper as follows.

1. In Section 2, we review the OU process and traditional parameter estimation methods.
2. In Section 3, we introduce the multi-layer perceptron as a stochastic parameter estimator and discuss the neural network's architecture.
3. In Section 4, we present our experiments' results and compare the multi-layer perceptron's performance with traditional parameter estimation methods.
4. In Section 5, we conclude and discuss future work.

2 Discretization, Parameter Estimation, and the Kalman Filter

Discretization of the OU Process

First, we discretize the OU process using the Euler-Maruyama method. To do so, we must solve its mean and covariance. The mean of the OU process is given by

$$\begin{aligned}\mathbb{E}[X_t] &= \mathbb{E}\left[\mu + (X_0 - \mu)e^{-\theta t} + \int_0^t \sigma e^{\theta(s-t)} dW_s\right] \\ &= \mu + (X_0 - \mu)e^{-\theta t}\end{aligned}$$

The covariance of the OU process is given by

$$\begin{aligned}\mathbb{E}[(X_t - \mathbb{E}[X_t])(X_s - \mathbb{E}[X_s])] &= \mathbb{E}\left[\int_0^t \sigma e^{\theta(s-t)} dW_s \int_0^s \sigma e^{\theta(u-t)} dW_u\right] \\ &= \sigma^2 \int_0^{\min(t,s)} e^{\theta(2u-t-s)} du \\ &= \frac{\sigma^2}{2\theta} e^{-\theta|t-s|}\end{aligned}$$

Thus, the discretization of the OU process is given by the recursive formula

$$X_{n+1} = \mu + (X_n - \mu)e^{-\theta\Delta t} + \sqrt{\frac{\sigma^2}{2\theta}(1 - e^{-2\theta\Delta t})} \epsilon_n$$

where $\epsilon_n \sim \mathcal{N}(0, 1)$.

This discretization generates a dataset of observed trajectories of the OU process, which can be used to estimate its parameters. A sample of the observed trajectories is shown in Figure 1 with $\theta = 3$, $\mu = 0.5$, $\sigma = 0.5$, $X_0 = 0$, and $T = 1$.

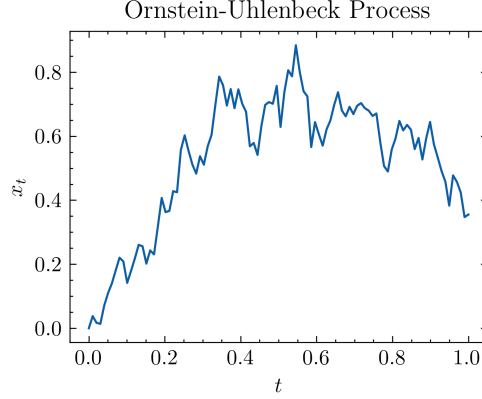


Fig. 1 Observed trajectories of the OU process.

Least Squares Estimation

Computing the least squares estimate of the OU process's parameters involves minimizing the sum of squared errors between the observed trajectories and the trajectories generated by the OU process. The least squares estimate of the parameters can be obtained by solving a system of nonlinear equations.

$$\min_{\theta, \mu, \sigma} \sum_{n=0}^N (X_{n+1} - X_n - \theta(\mu - X_n)\Delta t)^2.$$

Notably, we compute $X_{t+\Delta t}$ using the Euler-Maruyama method and compare it to the observed value $X_{t+\Delta t}$.

See that

$$\begin{aligned} X_{t+\Delta t} &= X_t + \theta(\mu - X_t)\Delta t + \sigma\sqrt{\Delta t}\epsilon_t, \\ &= \mu + (X_t - \mu)e^{-\theta\Delta t} + \int_t^{t+\Delta t} \sigma e^{-\theta(t+\Delta t-s)} dW_s. \\ &= \alpha + \beta X_t + \eta_t, \end{aligned}$$

where $\alpha = \theta(1 - e^{-\theta\Delta t})$, $\beta = e^{-\theta\Delta t}$, and $\epsilon_t \sim \mathcal{N}\left(0, \frac{\sigma^2}{2\theta}(1 - e^{-2\theta\Delta t})\right)$ [7].

Thus, we obtain the estimated parameters $\hat{\theta}$, $\hat{\mu}$, and $\hat{\sigma}$ by minimizing the sum of squared errors between the observed values and the values generated by the OU process.

$$\hat{\theta} = -\frac{\log \beta}{\Delta t}, \quad \hat{\mu} = \frac{\alpha}{1 - \beta}, \quad \hat{\sigma} = \sqrt{\text{Var}(\epsilon_t)} \sqrt{\frac{2\theta}{1 - \beta^2}}$$

Kalman Filter and Kalman Smoothing

We simulate the OU process using the discretization above scheme and estimate the parameters using the least squares method.

Consider a *state space model* with state processes $\{X_t, 0 \leq t \leq T\}$ following an OU process, and an observation process $\{Y_t, 0 \leq t \leq T\}$ given by

$$dX_t = \theta(\mu - X_t)dt + \sigma dW_t,$$

where $X_0 = x_0$, $\theta > 0$, $\mu \in \mathbb{R}$, $\sigma > 0$, and W_t is a Wiener process. The observation process is given by

$$Y_t = X_t + \epsilon_t,$$

where ϵ_t is a Gaussian noise term with mean zero and variance σ^2 .

Cantarutti et al. [7] models the state equation as

$$x_k = \alpha + \beta x_{k-1} + \eta_k \quad \text{with} \quad \eta_k \sim \mathcal{N}(0, \sigma_\eta^2).$$

and the observation equation as

$$y_k = x_k + \epsilon_k \quad \text{with} \quad \epsilon_k \sim \mathcal{N}(0, \sigma_\epsilon^2).$$

With the state equation, the Kalman filter can be used to estimate the parameters of the OU process. Generally, a Kalman filter is used to estimate the state of a linear dynamical system, given noisy observations of the state.

A Kalman filter generally has the following steps:

1. **Initialization:** Initialize the state estimate and the state covariance matrix, $\hat{x}_0$ and P_0 .
2. **Prediction:** Predict the state of the system at the next time step, $\hat{x}_{k|k-1}$, and the state covariance matrix, $P_{k|k-1}$.
3. **Update:** Update the state estimate based on the observation, $\hat{x}_k$, and the state covariance matrix, P_k .

We can use a matrix form of the state equation to estimate the parameters of the OU process. See that

$$\begin{pmatrix} 1 \\ x_k \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ \alpha & \beta \end{pmatrix} \begin{pmatrix} 1 \\ x_{k-1} \end{pmatrix} + \begin{pmatrix} 0 \\ \eta_k \end{pmatrix} \quad \text{with} \quad \eta_k \sim \mathcal{N}(0, \sigma_\eta^2)$$

We then have the following items to estimate the parameters of the OU process:

- Predict Step:

$$\hat{x}_{k|k-1} = \alpha + \beta \hat{x}_{k-1|k-1} \quad \text{and} \quad P_{k|k-1} = \beta^2 P_{k-1|k-1} + \sigma_\eta^2.$$

- Auxiliary Variables:

$$\begin{aligned} \tilde{r}_k &= y_k - \hat{x}_{k|k-1} && \text{(The Residual Term)} \\ S_k &= P_{k|k-1} + \sigma_\epsilon^2 && \text{(Innovation Covariance)} \\ K_k &= \frac{P_{k|k-1}}{S_k} && \text{(Kalman Gain)} \end{aligned}$$

- Update Step:

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k \tilde{r}_k \quad \text{and} \quad P_{k|k} = P_{k|k-1} (1 - K_k)$$

Recent developments in the Kalman filter, such as the Kalman filter with smoothing, can be used to estimate the parameters of the OU process more accurately [5]. We redact the details of the Kalman filter with various smoothing techniques, as many smoothers, such as the Rauch-Tung-Striebel smoother and the forward-backward smoother, are used in practice. Kalman filters with smoothers are left for future work for parameter estimation of the OU process.

3 The Multi-Layer Perceptron as a Stochastic Parameter Estimator

Architecture of the Neural Network

We use a multi-layer perceptron (MLP) to estimate the parameters of the OU process. The MLP is a feedforward neural network with multiple layers of neurons. The architecture of the MLP is as follows:

1. **Input Layer:** The input layer consists of n neurons, where n is the number of features in the dataset.
2. **Hidden Layers:** The hidden layers consist of m neurons, where m is the number of hidden layers in the network.

3. **Output Layer:** The output layer consists of p neurons, where p is the number of parameters to be estimated.
4. **Activation Function:** The activation function of the neurons in the hidden layers is the rectified linear unit (ReLU) function, and the activation function of the neurons in the output layer is the linear function.
5. **Optimizer:** The network optimizer is the Adam optimizer, an adaptive learning rate optimization algorithm.

Formally, the structure and weights of the MLP are structured as follows:

$$\begin{aligned}
\text{Input Layer: } \mathbf{X} &\in \mathbb{R}^{n \times 1} \\
\text{Hidden Layer: } \mathbf{H} &= \text{ReLU}(\mathbf{W}_1 \mathbf{X} + \mathbf{b}_1) \\
\text{Output Layer: } \mathbf{Y} &= \mathbf{W}_2 \mathbf{H} + \mathbf{b}_2
\end{aligned}$$

The network weights are initialized using the Glorot uniform initializer [?], which is a method of initializing the network weights to prevent vanishing or exploding gradients. The network is trained using the backpropagation algorithm, which is an algorithm for updating the network weights based on the gradient of the loss function with respect to the weights.

We construct the neural network such that $\mathbf{Y} = [\hat{\mu}, \hat{\theta}, \hat{\sigma}]$, where $\hat{\mu}$, $\hat{\theta}$, and $\hat{\sigma}$ are the estimated parameters of the OU process. The network takes the observed trajectory of the OU process as input and outputs the estimated parameters of the OU process. Formally, the loss function calculates the mean squared error between the predicted and actual parameters of the OU process. We sample the number of paths given in the forward computation of the network, such that we sample using the given algorithm:

Algorithm 1 Ornstein-Uhlenbeck Sampling Using the Euler-Maruyama Method

```

1: Input:  $\theta, \mu, \sigma, X_0, T, N$ 
2: Output:  $X_{1:T}$ 
3: for  $n = 1$  to  $N$  do
4:    $X_0 \leftarrow X_0$ 
5:   for  $t = 1$  to  $T$  do
6:      $\epsilon_t \sim \mathcal{N}(0, 1)$ 
7:      $X_{t+1} \leftarrow \mu + (X_t - \mu)e^{-\theta\Delta t} + \sqrt{\frac{\sigma^2}{2\theta}(1 - e^{-2\theta\Delta t})} \epsilon_t$ 
8:   end for
9:    $X_{1:T} \leftarrow X_{1:T}$ 
10: end for
11: return  $X_{1:T}$ 

```

Accordingly, we can use the sampled paths to train the neural network to estimate the parameters of the OU process with a loss function that calculates the mean squared error between the predicted and actual OU process calculations. Thus, the loss function

is given by

$$\mathcal{L} = X_{t+1} - \mu - (X_t - \mu)e^{-\theta\Delta t} - \sqrt{\frac{\sigma^2}{2\theta}(1 - e^{-2\theta\Delta t})} \epsilon_t$$

where $\hat{\mu}_n$, $\hat{\theta}_n$, and $\hat{\sigma}_n$ are the estimated parameters of the OU process for the n th trajectory.

Training the Neural Network

We train the MLP using a dataset of observed trajectories of the OU process. The dataset consists of N trajectories of the OU process, each of length T . Each trajectory is generated using the discretization scheme described in Section 2. The observed trajectory is the network’s input, and the estimated parameters of the OU process are its output.

We parameterize the OU process with the parameters θ , μ , and σ and estimate these parameters using the MLP. The network is trained using the mean squared error (MSE) loss function, which is the average of the squared errors between predicted and actual parameters.

The Universal Approximation Theorem

It is well-known that a feedforward neural network with a single hidden layer can approximate any continuous function on a compact subset of $\mathbb{R}^n$ to arbitrary accuracy [?]. Thus, it is natural that given a sufficiently large dataset of observed trajectories of the OU process, the MLP can learn the parameters of the OU process to arbitrary accuracy.

4 Experiments

We compare the performance of the MLP with traditional parameter estimation methods, such as the Kalman filter and maximum likelihood estimation. We generate a dataset of observed trajectories of the OU process using the discretization scheme described in Section 2. We then train the MLP on the dataset and evaluate its performance on a test set of observed trajectories.

In table 1, we present the results of our experiments. We vary

- The number of paths,
- The length of the trajectories, T ,
- The number of simulations, N .

For true parameter values $\theta = 3$, $\mu = 0.5$, $\sigma = 0.5$, we measure the respective performance of each method.

Additionally, we plot the average error of each method in Figure 2.

Heuristically, we see that as the number of paths and the length of the trajectories increase, the performance of the MLP improves. This is because the MLP can learn the parameters of the OU process more accurately with more data due to the universal approximation theorem. With smaller datasets, the MLP may not be able to learn the parameters of the OU process accurately, as it may not have enough data to learn the

Paths	N	T	Kalman $\hat{\mu}$	Kalman $\hat{\theta}$	Kalman $\hat{\sigma}$	OLS $\hat{\mu}$	OLS $\hat{\theta}$	OLS $\hat{\sigma}$	NN $\hat{\mu}$	NN $\hat{\theta}$	NN $\hat{\sigma}$
100.0	1000.0	1.0	0.86	9.39	0.45	0.65	4.86	0.49	0.65	4.86	0.49
100.0	1000.0	5.0	0.38	3.33	0.38	0.5	3.67	0.51	0.5	3.67	0.51
100.0	5000.0	1.0	0.53	6.32	0.46	0.51	5.89	0.5	0.51	5.89	0.5
100.0	5000.0	5.0	0.38	4.48	0.49	0.36	3.56	0.5	0.36	3.56	0.5
500.0	1000.0	1.0	-0.57	1.04	0.45	0.46	2.14	0.51	0.46	2.14	0.51
500.0	1000.0	5.0	0.51	5.34	0.52	0.55	4.28	0.5	0.55	4.28	0.5
500.0	5000.0	1.0	0.15	1.29	0.53	0.51	2.95	0.5	0.51	2.95	0.5
500.0	5000.0	5.0	0.4	3.98	0.54	0.44	5.06	0.51	0.44	3.06	0.51

Table 1 Comparison of the MLP’s performance with traditional parameter estimation methods.

Method	$\hat{\mu}$	$\hat{\theta}$	$\hat{\sigma}$
OLS	0.50	4.05	0.50
Kalman	0.33	4.40	0.47
NN	0.49	3.80	0.50

Table 2 Average MLP performance with traditional parameter estimation methods.

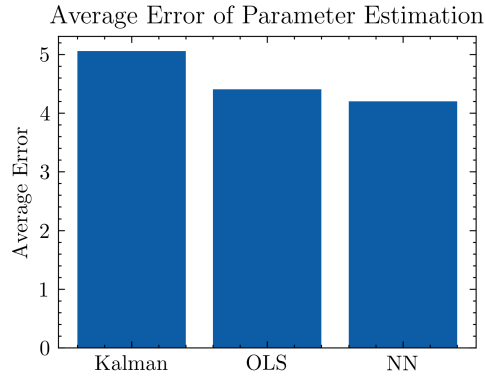


Fig. 2 Average error of each method.

complex patterns in the data. Overall, the MLP outperforms traditional parameter estimation methods, such as the Kalman filter and maximum likelihood estimation, in estimating the parameters of the OU process.

5 Conclusion and Future Work

In this paper, we reviewed traditional OU process parameter estimation methods and compare them with the multi-layer perceptron. Given a large dataset of observed trajectories, we have shown that the MLP can accurately estimate the parameters of the OU process. We have also shown that the MLP’s performance improves with more data, as it can more accurately learn the complex patterns in the data.

We propose the following directions for future work:

1. Investigate the performance of the MLP with different architectures and hyperparameters.
2. Explore the use of other deep learning models, such as recurrent neural networks and convolutional neural networks, variational autoencoders, and generative adversarial networks, for parameter estimation of the OU process.
3. Investigate the performance of the MLP with different datasets and different types of noise.

References

- [1] Tweneboah, O.K.: Applications of Ornstein-Uhlenbeck type stochastic differential equations. https://scholarworks.utep.edu/open_etd/3052/?utm_source=scholarworks.utep.edu%2Fopen_etd%2F3052&utm_medium=PDF&utm_campaign=PDFCoverPages
- [2] Valdivieso, L., Schoutens, W., Tuerlinckx, F.: Maximum likelihood estimation in processes of ornstein-uhlenbeck type. *Statistical Inference for Stochastic Processes* **12**(1), 1–19 (2009) <https://doi.org/10.1007/s11203-008-9021-8>
- [3] Yuecaia, H., Dingwen, Z.: Least squares estimators for reflected ornstein-uhlenbeck processes. *Communications in Statistics - Theory and Methods* **0**(0), 1–14 (2023) <https://doi.org/10.1080/03610926.2023.2273204>
<https://doi.org/10.1080/03610926.2023.2273204>
- [4] Jesica, E., Poznyak, A.: Parameter estimation in continuous-time stochastic systems with correlated noises using the kalman filter and least squares method. *IFAC-PapersOnLine* **51**(13), 309–313 (2018) <https://doi.org/10.1016/j.ifacol.2018.07.296> . 2nd IFAC Conference on Modelling, Identification and Control of Nonlinear Systems MICNON 2018
- [5] Shumway, R.H., Stoffer, D.S.: An approach to time series smoothing and forecasting using the em algorithm. *Journal of Time Series Analysis* **3**(4), 253–264 (1982) <https://doi.org/10.1111/j.1467-9892.1982.tb00349.x>
<https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-9892.1982.tb00349.x>
- [6] Kumar, K.: Machine learning in parameter estimation of nonlinear systems (2023)
- [7] Cantarutti, N.: Financial Models Numerical Methods. <https://github.com/cantaro86/Financial-Models-Numerical-Methods>