Available online at www.sciencedirect.com

ScienceDirect

The Journal of Finance and Data Science 4 (2018) 172–182

http://www.keaipublishing.com/en/journals/jfds/

Improved parameter estimation of Time Dependent Kernel Density by using Artificial Neural Networks

Xing Wang [a], Chris P. Tsokos [b,*], Abolfazl Saghafi [c]

[a] *University of Wisconsin-Madison, Madison, WI, USA*
[b] *Department of Mathematics and Statistics, University of South Florida, Tampa, FL, USA*
[c] *Department of Mathematics, Physics, and Statistics, University of the Sciences, Philadelphia, PA, USA*

Received 7 December 2017; revised 12 March 2018; accepted 10 April 2018
Available online 17 April 2018

Abstract

Time Dependent Kernel Density Estimation (TDKDE) used in modelling time-varying phenomenon requires two input parameters known as bandwidth and discount to perform. A Maximum Likelihood Estimation (MLE) procedure is commonly used to estimate these parameters in a set of data but this method has a weakness; it may not produce stable kernel estimates. In this article, a novel estimation procedure is developed using Artificial Neural Networks which eliminates this inherent issue. Moreover, evaluating the performance of the kernel estimation in terms of the uniformity of Probability Integral Transform (PIT) shows a significant improvement using the proposed method. A real-life application of TDKDE parameter estimation on NASDQ stock returns validates the flawless performance of the new technique.

2010 MSC: 62G07; 62M45; 91G50

Keywords: Time Dependent Kernel Density Estimation; Artificial Neural Networks; Probability Integral Transform; Finance; Machine learning

1. Introduction

Kernel Density Estimation (KDE) method is a popular nonparametric procedure to estimate probability density function of a random phenomenon requiring only mild assumptions.[1] The KDE is a well-established tool in finance for volatility density estimation,[2] risk analysis of portfolio,[3] and loss distribution modeling,[4] among others. Time Dependent Kernel Density Estimation (TDKDE) developed by Harvey and Oryshchenko (2012) generalizes KDE to estimate time dependent probability density functions or corresponding cumulative distribution functions.[5]

Many real-life phenomenon change in time and TDKDE assists in capturing such temporal progress. To stress on the various applications of TDKDE, note that it has been utilized to estimate probability distribution of wind speed and

* Corresponding author.
E-mail address: ctsokos@usf.edu (C.P. Tsokos).
Peer review under responsibility of China Science Publishing & Media Ltd.

https://doi.org/10.1016/j.jfds.2018.04.002

to assess wind energy fluctuations,[6] to develop automated time-dependent strategies in stock trading market,[7,8] to detect and track moving objects in maritime,[9] and to model economic data.[10]

The TDKDE requires two input parameters to perform, known as bandwidth and discount. A Maximum Likelihood Estimation procedure to estimate these two parameters is proposed in the literature.[5] However, there is a critical issue with this method since the estimations vary with a preset algorithm parameter.[8] In this article, a supervised learning method using Artificial Neural Networks is developed that eliminates this issue and provides more reliable estimations leading to a significant improvement in overall performance of the TDKDE method.

2. Background and related methodologies

2.1. Kernel Density Estimation

Kernel Density Estimation (KDE) is a non-parametric method to estimate the Probability Density Function (PDF) of a random phenomenon. For a random sample $y_1, y_2, \ldots, y_T$, the traditional kernel estimator of the PDF f at point y can be expressed as:

$$\widehat{f}_T(y) = \frac{1}{Th}\sum_{i=1}^{T} K\left(\frac{y - y_i}{h}\right), \tag{1}$$

where T is the number of observations, h is the bandwidth that determines the smoothness of the estimate, and $K(\cdot)$ is the kernel.[11] In general, any positive function that integrates to one, has a mean of zero, and finite variance can be used as kernel.[12] Some of the classical kernel functions are listed in the Appendix.

2.2. Time Dependent Kernel Density Estimation

When the density estimation is applied to time series data, a weighting scheme is usually used to assign more weight to recent observations discounting older data.[13] One of the widely used weighting schemes is the Exponentially Weighted Moving Average (EWMA) filter which performs by discounting older observations in an exponentially decaying manner. The Time Dependent Kernel Density Estimation is such an estimation adjusted by the EWMA scheme given by:

$$\widehat{f}_T(y) = \frac{1}{h}\sum_{i=1}^{t} w_{t,i} K\left(\frac{y - y_i}{h}\right), \quad t = 1, 2, \ldots, T. \tag{2}$$

In general, the weights $w_{t,i}$ are assigned such that $\sum_{i=1}^{t} w_{t,i} = 1$. In case of the EWMA filter, the weights are assigned by:

$$w_{t,i} = \frac{1 - \omega}{1 - \omega^t}\omega^{t-i}, \quad i = 1, 2, \ldots, t, \tag{3}$$

where $0 \leq \omega < 1$ is the discount parameter. Consequently, the time adaptive kernel density estimator for the corresponding Cumulative Density Function (CDF) is expressed as:

$$\widehat{F}_t(y) = \sum_{i=1}^{t} w_{t,i} H\left(\frac{y - y_i}{h}\right), \tag{4}$$

where $H(\cdot)$ is the cumulative form of the corresponding kernel $K(\cdot)$. In order to obtain the time dependent PDF or CDF, feasible estimates of the bandwidth h and the discount ω parameters is required. Harvey and Oryshchenko (2012) proposed a Maximum Likelihood Estimation approach with the following log-likelihood function subject to $\omega \in (0, 1]$ and $h > 0$

$$L(\omega,h)=\frac{1}{T-m}\sum_{t=m}^{T-1}\ln\left(\widehat{f}_{t+1|t}(y_{t+1})\right),$$
$$=\frac{1}{T-m}\sum_{t=m}^{T-1}\ln\left(\frac{1}{h}\sum_{i=1}^{t}K\left(\frac{y_{t+1}-y_i}{h}\right)w_{t,i}\right), \tag{5}$$

where m is a preset parameter to initialize the estimation procedure. A guideline to select an appropriate m was not provided and a general suggestion was setting $m = 50$ or 100 if the sample size is large. However, estimations of the bandwidth h and the discount parameter ω vary with different selections of m.[8] Moreover, for bounded kernel functions including the popular Epanechnikov and Uniform, all estimations for outliers are zero. A possible solution provided by Harvey and Oryshchenko (2012) was setting $\widehat{f}_{t+1|t}(\cdot)$ equal to a very small positive number in such cases, but this may severely affect the estimations when large numbers of data points are adjusted this way. We propose an Artificial Neural Networks approach to estimate the bandwidth h and the discount ω parameters which eliminates the above issues and significantly improves the kernel estimates.

2.3. Artificial Neural Networks

Artificial Neural Networks (ANNs) are nonlinear models motivated by the physiological architecture of the nervous system which can be trained to learn approximate functions of complex non-linear systems that usually depend on a large number of inputs.[14] ANNs are typically organized in layers: the input layer, hidden layer and output layer, which consist of several neurons. Each neuron in a layer is connected to adjacent layers by weights. Data are presented to the network via the input layer, which are multiplied by the weights, and then go through the activation function of the neuron in one or more hidden layers. The function in the output layer computes the output of the artificial neuron.[15] The structure of a typical ANN and an artificial neuron are shown in Fig. 1.

Back-propagation (BP) algorithm is the most frequently used, effective, and easy way to learn model for multi-layered networks. It is a supervised learning technique which is based on the gradient descent method that attempts to minimize the error of the network by moving down the gradient of the error curve. Levenberg-Marquardt algorithm is one of the fastest back-propagation algorithm which works well for training small and medium sized networks.[16] This algorithm is applied in our study, because while providing a numerical solution to the problem of minimizing a nonlinear function, it has a speed advantage in the computations.

2.4. Probability Integral Transforms

An important tool for evaluating the adequacy of a density estimate is the Probability Integral Transform (PIT), which is the cumulative probability evaluated at the realized value of the target variable. If a sequence of density

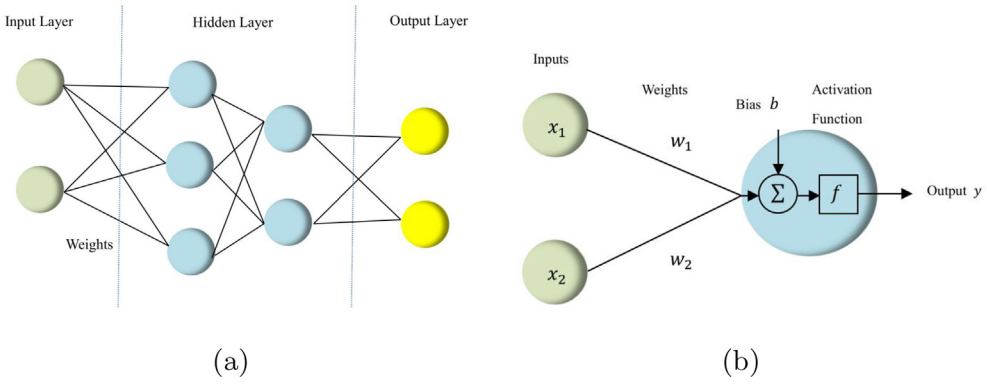

Fig. 1. (a) A typical structure of an ANN (b) A typical artificial neuron in ANNs.

estimates is correctly specified, then the corresponding PITs should be independently, uniformly distributed over the interval (0,1).[17] The PIT of y_{t+1}, denoted by u_{t+1}, is expressed as the following

$$u_{t+1} = \sum_{i=1}^{t} w_{t,i} H\left(\frac{y_{t+1} - y_i}{h}\right), \tag{6}$$

where $H(\cdot)$ is the CDF form of the corresponding kernel $K(\cdot)$ and $w_{t,i}$ are weights given in (3).[5]

3. New estimation method using Artificial Neural Networks

3.1. Dataset

Various synthetic and real-life data is used to assess the performance of the proposed method. To make the results comparable to those in the research of Harvey and Oryshchenko (2012), a series of $T = 1000$ observations from a MA(1)-GARCH(1,1) model is generated in which the errors follow a t-distribution with 7 degrees of freedom. To be specific, the MA(1)-GARCH(1,1) model[18] is expressed as

$$y_t = \mu + \theta\varepsilon_{t-1} + \varepsilon_t, \quad \varepsilon_t = z_t\sigma_t, \tag{7}$$

where

$$\sqrt{\frac{\nu}{\nu - 2}} z_t \sim t_{(\nu)}, \quad \sigma_t = \sqrt{\alpha_0 + \alpha_1\varepsilon_{t-1}^2 + \beta_1\sigma_{t-1}^2}. \tag{8}$$

The model parameters were chosen to be $\theta = 0.2102$, $\alpha_1 = 0.0979$, $\beta_1 = 0.9010$, $\nu = 7$, to be consistent with the model of the empirical data.

A sample of 3459 NASDAQ stock returns presenting daily prices from June 15th, 2004, to March 11th, 2018, is used to illustrate the excellent performance of the new proposed method in the results section.

3.2. Parameter estimation of TDKD using ANNs

To eliminate the existing issues with estimations of the bandwidth (h) and the discount (ω) parameters using MLE method and to improve performance of the kernel estimate, a novel supervised learning method is developed. The logic behind the estimation procedure is the fact that the CDF of a random variable Y should follow the uniform distribution on $(0, 1)$. Thus, when the estimated CDF $(\widehat{F}_t(y))$ is as close as possible to the empirical CDF of the standard uniform distribution, the corresponding estimates of the parameters (ω, h) should be optimal. Consequently, the estimation procedure could be turned into a supervised learning task which can be conducted using Artificial Neural Networks.

Therefore, a neural network is trained to map the estimated CDF using 4 to the CDF of the standard uniform distribution using 6. Ideally, we would like to see $\widehat{F}_{kt}(y) = U_t$ $(t = 1,\ldots,T)$, where $\widehat{F}_{kt}(y)$ is the time dependent kernel estimator for the CDF evaluated at y_t corresponding to the pair (ω_k,h_k), and U_t is the t^{th} observation of the CDF for the discrete standard uniform distribution. Measuring the deviation of $\widehat{F}_{kt}(y)$ from U_t using the following squared difference

$$E_k = \sum_{t=1}^{T} \left(\widehat{F}_{kt}(y) - U_t\right)^2, \tag{9}$$

the optimal pair of parameters, (ω^*,h^*), is the pair corresponding to the minimum E_k with constraints $0 < \omega < 1$ and $h > 0$. The key steps of the developed algorithm for estimating h and ω are presented in Algorithm 1.

4. Results

4.1. Step 1: selection of parameters' interval (h_1, h_2) and (ω_1, ω_2)

Proper selection of the parameters' interval, i.e. (h_1, h_2) and (ω_1, ω_2), for the proposed ANN procedure reduces the runtime of the estimation process. To that end, one can compute the squared differences using (9) for 341 possible combinations of (ω, h) where $\omega \in \{0.001, 0.1, 0.2, \cdots, 0.9, 0.99\}$ and $h \in \{0.001, 0.1, 0.2, \cdots, 3.0\}$. Then, select an interval (ω_1, ω_2) and (h_1, h_2) where E is at its smallest values. This step could be skipped by setting $h \in (0, 3)$ and $\omega \in (0, 1)$ at a computational cost.

Target parameters' interval selection is carried on for the simulated data where $\widehat{F}_{kt}(y)$ is calculated using Gaussian and Silverman Kernel. The intervals set to be $(\omega_1, \omega_2) = (0.8, 0.99)$ and $(h_1, h_2) = (0.001, 0.6)$. A wider interval for bandwidth is appropriate as its parameter space is unbounded. However, bandwidth rarely takes values beyond 3 as high bandwidth values reduce smoothness of the kernel estimate.

Algorithm 1 Parameter Estimation Algorithm.

1. Assign target parameters' interval for bandwidth (h_1, h_2) and discount (ω_1, ω_2), see Section 4.1.
2. Generate K random pairs of (h,ω): (h_k, ω_k), $k = 1, 2, \ldots, K$, where $h \in (h_1, h_2)$, and $\omega \in (\omega_1, \omega_2)$.
3. Generate T points: $U_t = \frac{t}{T}$, $t = 1, 2, \ldots, T$, to represent the CDF of the discrete standard uniform distribution.
4. Calculate $\widehat{F}_{kt}(y)$ according to each pair of (h_k, ω_k) where $\widehat{F}_{kt}(y) = \sum_{i=1}^{t} H\left(\frac{y_t - y_i}{h_k}\right) \frac{1-\omega_k}{1-\omega_k^t} \omega_k^{t-i}$ and $k = 1, 2, \ldots, K$. Sort $\widehat{F}_{kt}(y)$ such that: $\widehat{F}_{k1}(y) \leq \widehat{F}_{k2}(y) \leq \ldots \leq \widehat{F}_{kT}(y)$.
5. Calculate the squared difference between $\widehat{F}_{kt}(y)$ and U_t, that is $E_k = \sum_{t=1}^{T} [\widehat{F}_{kt}(y) - U_t]^2$, where $k = 1, 2, \ldots, K$, $t = 1, \ldots, T$.
6. Train an Artificial Neural Network with (h_k, ω_k) as the inputs, and E_k as the outputs to substitute step 3 to step 5.
7. Use the trained Neural Network to select the pair of (h_k, ω_k) corresponding to the minimal E_k with constraint $h > 0$ and $0 < \omega < 1$, that is (h^*, ω^*).

4.2. Steps 2–5: computations

To obtain relevant stable estimates, K is set to be 500. Therefore, 500 random pairs of (h,ω) are generated uniformly from $(h_1, h_2) \times (\omega_1, \omega_2)$ using statistical softwares. If one wishes to skip step 1 of the algorithm, a larger number for K should be selected to properly cover the parameter space. It is noteworthy to mention that the proposed algorithm with $K = 500$ took about 72 min to run on a PC with core i7 2.60 GHz processor and 8 Gb RAM.

After computations of steps 3 to 5, pairs of (h_k, ω_k, E_k) for $k = 1, 2, \ldots, 500$ should have been computed. Fig. 2 shows these values and illustrates a clear trend for the relationship between parameters h, ω, and the squared error E. In general, E decreases as ω increases and h decreases. The trend is more obvious in a 2-D view shown in Fig. 2(b) and (c). The visualization is based on the Gaussian Kernel for illustration.

4.3. Analysis of the network response

The 500 pairs of (h_k, ω_k) simulated following the guidelines in the previous section are the inputs of the neural network and the outputs are 500 squared errors E_k. The Levenberg-Marquardt optimization algorithm that works well for training small and medium sized networks is utilized for the Artificial Neural Networks designed with two hidden layers in this study.[16] The ANN finds the optimal pair of estimates for h and ω as 0.0604 and 0.9871 respectively, with a squared error $E = 0.0269$, which is marked by a black dot in Fig. 2.

Fig. 3 illustrates the network performance by regression plots where the network outputs are plotted versus the targets in each plot. The 45-degree dashed line represents the perfect fit indicating that outputs are equal to targets. The best linear fit is shown by a solid line. In this case, the best linear fit practically overlaps with the perfect fit line in each plot which indicates a very good fit. The R-value, which is a correlation coefficient between the outputs and targets, is close to 1 for the training, validation, and test sets, affirming a prime fit.

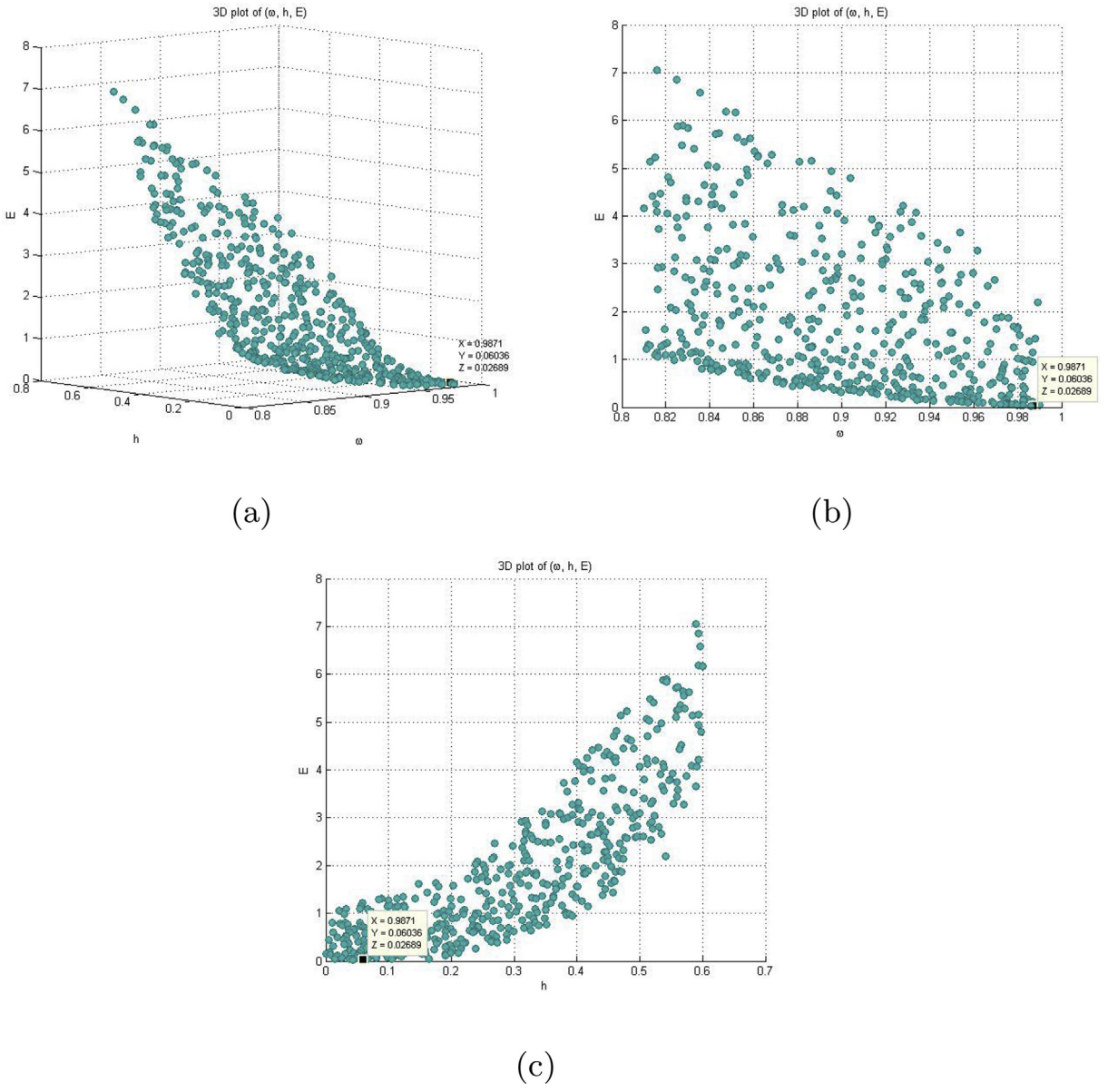

Fig. 2. (a) 3-D Plot of (h, ω, E) showing the relation between these parameters (b) E decreases with increasing ω and reaches its minimum in $\omega \in (0.8, 0.99)$ (c) E decreases with decreasing h and reaches its minimum in $h \in (0.001, 0.6)$.

4.4. Performance comparison

To initialize the ML estimation procedure, a preset parameter m must be selected where a general suggestion is setting $m = 50$ or 100 if the sample size is large. Here, a range of values from 1 to 100 is selected for m to see how the MLE algorithm responds to these selections. The Gaussian kernel is used in all cases for comparison. Table 1 shows how ML estimates for h and ω vary with different values of m while the ANN estimation remains fixed.

Furthermore, the Kolmogorov-Smirnov test (KS test) is employed to test whether the PITs follow the standard uniform distribution. The p-values for KS test reported in Table 1 indicate that the distribution of PITs generated from the new proposed technique is much closer to the standard uniform distribution compared to that of the MLE method. Notice how the uniformness of the distribution is rejected at 10% level of significance for all values of m.

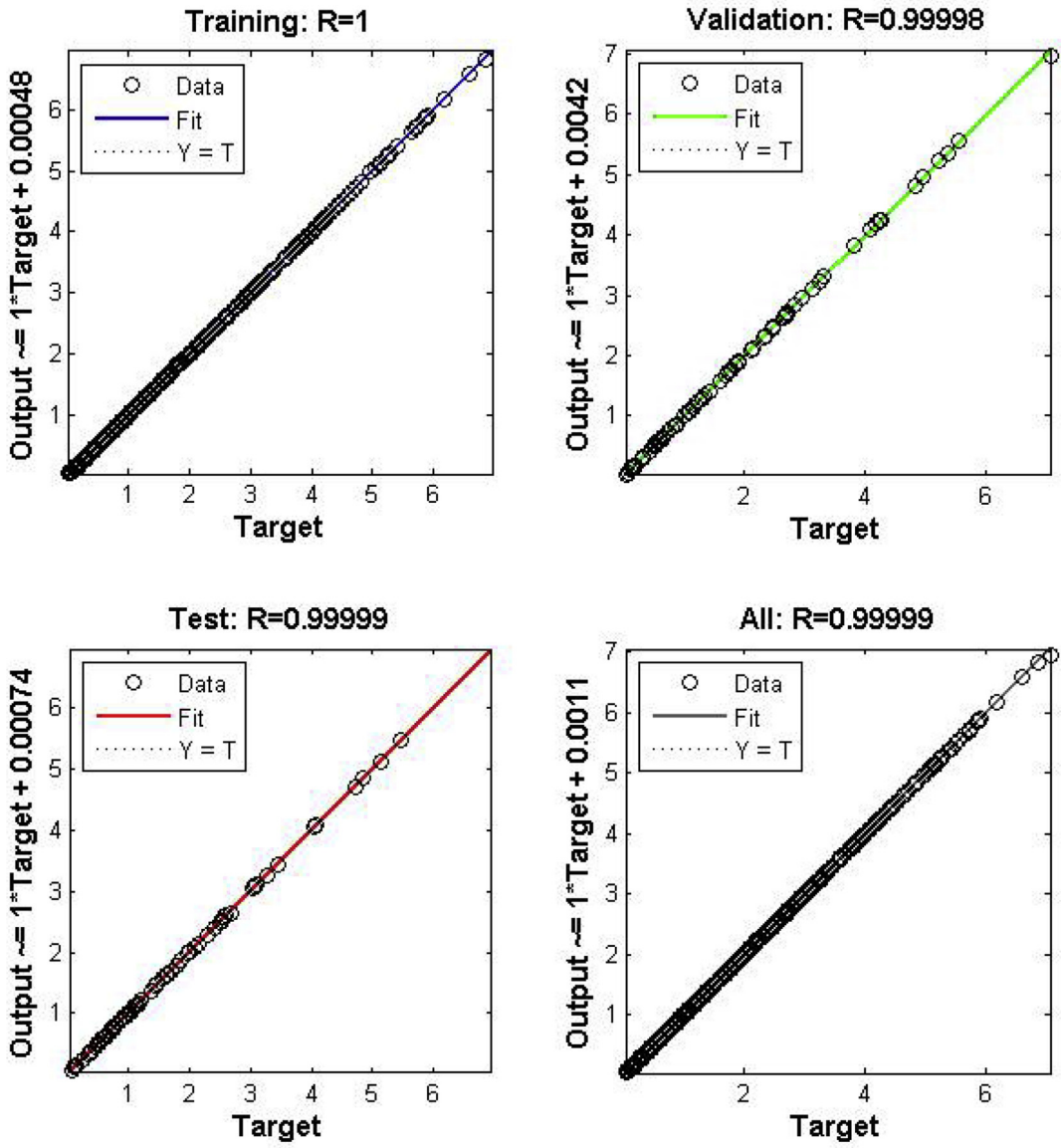

Fig. 3. Regression Plots of the network outputs with respect to targets for train, validation, and test sets.

Fig. 4 shows how empirical CDF of PITs using MLE method deviates from empirical CDF of uniform distribution and how perfect the proposed method is performing. For illustration, only the PITs in the MLE method with $m = 50$ and the PIT in the new proposed method are visualized. If PITs are uniformly distributed, their empirical CDF should be close to a 45-degree line. Deviation between the empirical CDF of PITs and the CDF of the standard uniform distribution is obvious in Fig. 4(a) where estimations are derived by the MLE method. In contrast, the empirical CDF of PITs and the CDF of the standard uniform distribution are hardly distinguishable in Fig. 4(b) in which the estimations are obtained by the proposed ANN approach.

Similar conclusions are obtained using Silverman Kernel. Table 2 indicates that, the distribution of PITs generated from the new technique is much closer to the standard uniform distribution compared to that of the MLE method when using Silverman Kernel.

Table 1
Performance Comparison of the MLE approach vs the proposed ANN method using Gaussian Kernel.

	$\widehat{h}$	$\widehat{\omega}$	KS Test p-value
MLE			
$m = 1$	0.4232	0.9580	0.0865
$m = 10$	0.4195	0.9585	0.0865
$m = 20$	0.4219	0.9581	0.0865
$m = 30$	0.4233	0.9576	0.0865
$m = 40$	0.4246	0.9572	0.0865
$m = 50$	0.4266	0.9569	0.0868
$m = 60$	0.4299	0.9567	0.0814
$m = 70$	0.4318	0.9563	0.0762
$m = 80$	0.4349	0.9561	0.0713
$m = 90$	0.4365	0.9560	0.0713
$m = 100$	0.4778	0.9307	0.0330
Proposed ANN			
	0.0604	0.9871	0.9995

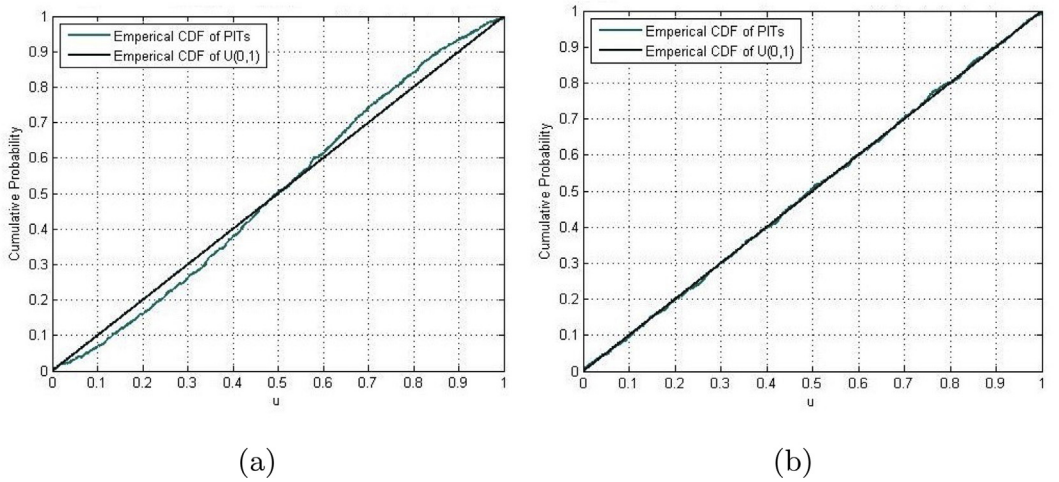

Fig. 4. A Comparison of the Empirical CDF of PITs and CDF of $U(0,1)$ (a) by MLE with $m = 50$, $(h^*, \omega^*) = (0.4266, 0.9569)$, KS test p-value $= 0.0868$ (b) by the proposed ANN method, $(h^*, \omega^*) = (0.0604, 0.9871)$, KS test p-value $= 0.9995$.

4.5. Real-life application: NASDAQ stock returns

A sample of 3459 NASDAQ stock returns presenting prices from June 15th, 2004, to March 11th, 2018, is used to illustrate the performance of the new proposed method in real-life. NASDAQ stock returns dataset can be obtained from Yahoo-Finance.[19] Suppose y_t represents daily adjusted close prices; that is the close price adjusted for dividends and splits. Then, following the procedure given in Harvey and Oryshchenko (2012), returns can be computed as $\Delta\ln(y_t)$. Fig. 5(a) shows the y_t along with the $\Delta\ln(y_t)$.

Comparing the results of Tables 1 and 2, the Gaussian kernel is performing slightly better since the KS test p-value is higher. Thus, the Gaussian Kernel is used in this real-life application. It is noteworthy to mention that the choice of bandwidth is more important than the choice of kernel.[5] To impellent the proposed method, the following steps are taken:

- Target parameters' interval for bandwidth and discount parameters is selected to be $(0.001, 0.5)$ and $(0.8, 0.99)$, respectively, after computing E for 341 combinations of (h, ω).
- 500 random pairs of (h, ω) generated where $h \in (0.001, 0.5)$, and $\omega \in (0.8, 0.99)$. Then, their E_k is calculated, given the instructions in Algorithm 1.

Table 2
Performance Comparison of the MLE approach vs the proposed ANN method using Silverman Kernel.

	$\widehat{h}$	$\widehat{\omega}$	KS Test p-value
MLE			
$m = 1$	0.3149	0.9508	0.5646
$m = 10$	0.3100	0.9508	0.5807
$m = 20$	0.3633	0.9055	0.3994
$m = 30$	0.3636	0.9052	0.3994
$m = 40$	0.3634	0.9052	0.3994
$m = 50$	0.3636	0.9051	0.3994
$m = 60$	0.3640	0.9052	0.3860
$m = 70$	0.3642	0.9047	0.3994
$m = 80$	0.3646	0.9046	0.3994
$m = 90$	0.3644	0.9047	0.3994
$m = 100$	0.3647	0.9045	0.4797
Proposed ANN			
	0.0762	0.9567	0.9824

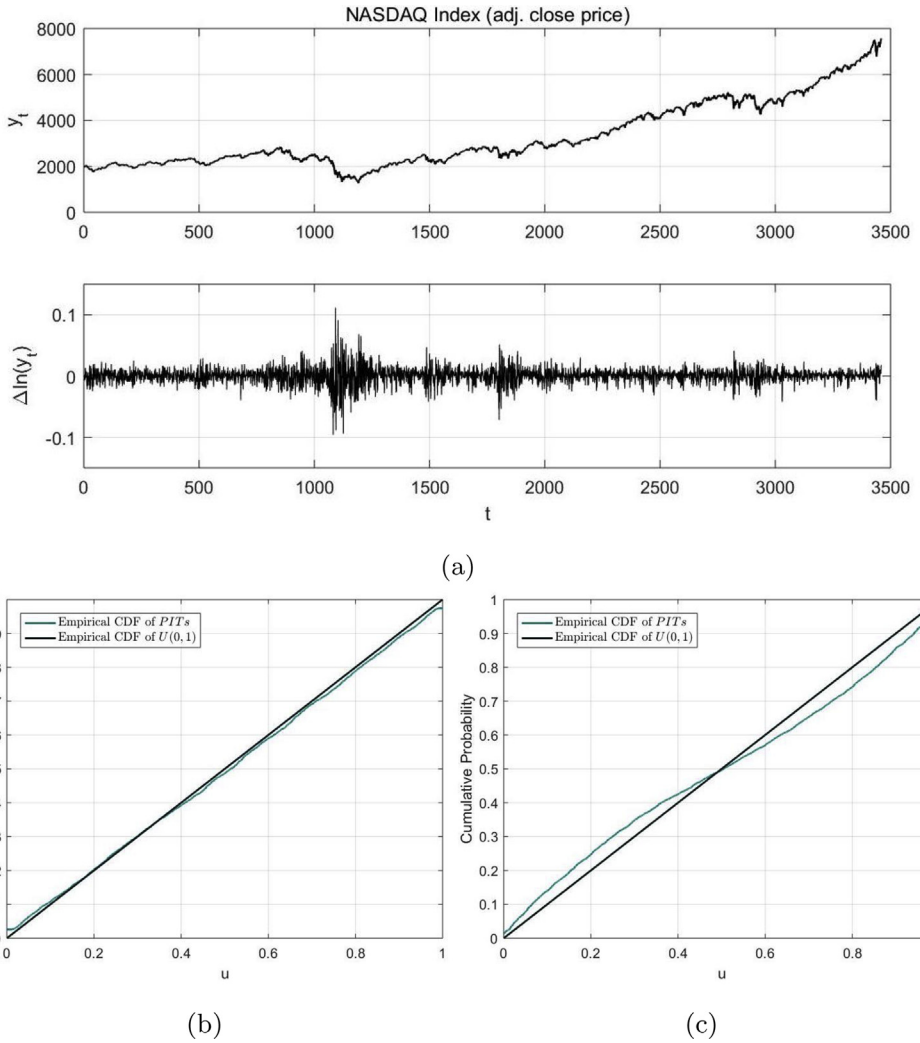

Fig. 5. (a) NASDAQ daily adjusted close stock prices and returns (b) Comparison of the Empirical CDF of PITs and CDF of $U(0,1)$ by the proposed ANN Method, $(h^*, \omega^*) = (0.0087, 0.9486)$ (b) Comparison of the Empirical CDF of PITs and CDF of $U(0,1)$ by common ML method with $m = 50$, $(h^*, \omega^*) = (0.4635, 0.9712)$.

- An Artificial Neural Network with Levenberg-Marquardt optimization algorithm is run using (ω_k, h_k) as the inputs, and E_k as the outputs, $k = 1, 2, \ldots, 500$.
- The optimal pair of (h, ω) is selected using the trained Neural Network. Performance of the optimal pair is then compared to estimations found from ML procedure using PIT criterion.

The estimations by the new algorithm are $(h^*, \omega^*) = (0.0087, 0.9486)$. The performance is evaluated by the uniformity of PITs in Fig. 5(b). Clearly, the empirical CDF is very close to a 45-degree line. Furthermore, the p-value for KS test is 0.1996, indicating that the PITs are uniformly distributed. The estimations using ML method were $(h^*, \omega^*) = (0.4635, 0.9712)$ using $m = 50$ and $(h^*, \omega^*) = (0.4649, 0.9711)$ using $m = 100$. In both cases, the p-value for KS test was less than 0.0001, indicating that the PITs are not uniformly distributed. This can be see from Fig. 5(c) as well. The new proposed method is apparently performing much better than the ML method proposed by Harvey and Oryshchenko (2012).

4.6. Conclusion

In this article, we extended the research of Perez (2012) by evaluating the performance of the probability density estimation for different kernel functions combined with various preset numbers. Utilizing the Uniformity of Probability Integral Transforms (PITs) criterion, we found that the Silverman Kernel has a better performance than the commonly used Gaussian Kernel when using ML method to estimate model parameters (h, ω) in the simulated datasets.

Furthermore, we developed a new method to estimate the parameters of the Time Dependent Kernel Density Estimation (TDKDE). The new estimation procedure resolves the existing issue with the Maximum Likelihood Estimation method by providing stable kernel estimates. More importantly, the new method improves the performance of the estimations evaluated by the uniformity of Probability Integral Transforms (PITs). The analytics could be applied to both stationary and non-stationary time series which gives this method an advantage over classical Time Series models.

Future works may involve performance comparison of the new proposed method in other kernel functions with different combinations of kernel functions, estimations of the discount and bandwidth parameters. Furthermore, the computing time can be investigated as a criterion to measure the performance of alternative estimation procedures.

Conflicts of interest

All authors have none to declare.

Appendix A. Supplementary data

Supplementary data related to this article can be found at https://doi.org/10.1016/j.jfds.2018.04.002.

Appendix

Table 3
The Classical Kernel Density Functions

Kernel	Kernel Density Function	Support
Gaussian	$K(u) = \frac{1}{\sqrt{2\pi}} e^{-\frac{u^2}{2}}$	$(-\infty, \infty)$
Epanechnikov	$K(u) = \frac{3}{4}(1 - u^2)$	$[-1, 1]$
Uniform	$K(u) = \frac{1}{2}$	$[-1, 1]$
Triangular	$K(u) = (1 - \lvert u \rvert)$	$[-1, 1]$
Triweight	$K(u) = \frac{35}{32}(1 - u^2)^3$	$[-1, 1]$
Tricube	$K(u) = \frac{70}{81}(1 - \lvert u \rvert^3)^3$	$[-1, 1]$

(*continued on next page*)

Table 3 (*continued*)

Kernel	Kernel Density Function	Support
Biweight	$K(u) = \frac{15}{16}(1-u^2)^2$	$[-1,1]$
Cosine	$K(u) = \frac{\pi}{4}\cos\left(\frac{\pi}{2}u\right)$	$[-1,1]$
Silverman	$K(u) = \frac{1}{2}e^{-\frac{\lvert u\rvert}{2}}\sin\left(\frac{\lvert u\rvert}{\sqrt{2}}+\frac{\pi}{4}\right)$	$(-\infty,\infty)$

References

1. Miladinovic B. *Kernel Density Estimation of Reliability with Applications to Extreme Value Distribution*. PhD Dissertation. University of South Florida; 2008.
2. Watada J. A kernel density estimation-maximum likelihood approach to risk analysis of portfolio. In: *IEEE 8th International Symposium on Intelligent Signal Processing (WISP), Portugal*. November 2013.
3. Van Es B, Spreij P, Van Zanten H. Nonparametric methods for volatility density estimation. In: *Advanced Mathematical Methods for Finance*. 2009:293−312.
4. Pitt D, Guillen M, Bolancé C. Estimation of parametric and nonparametric models for univariate loss distributions in financean approach using R. *J Financ Educ*. 2016;42(1):154−175.
5. Harvey A, Oryshchenko V. Kernel density estimation for time series data. *Int J Forecast*. 2012;28(1):3−14.
6. Wang J, Hu J, Ma K. Wind speed probability distribution estimation and wind energy assessment. *Renew Sustain Energy Rev*. 2016;60:881−899.
7. Liu T, Qiu N, Gu W. A stock trading strategy based on time-varying quantile theory. *J Adv Comput Intell Intell Inf*. 2015;19(3):417−422.
8. Perez A. Comments on Kernel density estimation for time series data. *Int J Forecast*. 2012;28(1):15−19.
9. Frost D, Tapamo JR. Detection and tracking of moving objects in a maritime environment using level set with shape priors. *Eurasip J Image Video Process*. 2013:42.
10. Zambom AZ, Dias R. A Review of kernel density estimation with applications to econometrics. *Int Econom Rev*. 2012;5(1):20−42.
11. Izenman A. *Modern Multivariate Statistical Techniques*. vol. 1. New York: Springer; 2008.
12. Guidoum AC. *Kernel Estimator and bandwidth Selection for Density and its Derivatives*. R package version 1.0.1. 2013.
13. Giraitis L, Kapetanios G, Price S. Adaptive forecasting in the presence of recent and ongoing structural change. *J Econom*. 2013;177(2):153−170.
14. Hagan MT, Demuth HB, Beale MH, Jesús OD. In: *Neural Network Design*. 2nd ed. Martin T. Hagan Publishing; 2014.
15. Baliyan A, Gaurav K, Mishra SK. A review of short term load forecasting using artificial neural network models. *Procedia Computer Science*. 2015;48:121−125.
16. Yu H, Wilamowski BM. *Levenberg-Marquardt training*. In: *Industrial Electronics Handbook*. vol. 5(1). 2011:1−16.
17. Diebold FX, Gunther TA, Tay AS. Evaluating density forecasts. *Int Econ Rev*. 1997;39(1):863−883.
18. Wuertz D, Chalabi Y, Luksan L. Parameter estimation of ARMA models with GARCH/APARCH errors. *J Stat Software*. 2006;55(2):28−33.
19. *NASDAQ GIDS Delayed Price*. Yahoo Finance; 2016. Accessed May 2016 https://finance.yahoo.com/quote/IXIC/history.