




Generating Random Variates via Kernel Density Estimation and Radial Basis Function Based Neural Networks

Cristian Candia-García¹(✉), Manuel G. Forero²(✉) ,
and Sergio Herrera-Rivera²

¹ Faculty of Engineering, Escuela Colombiana de Ingeniería Julio Garavito,
Bogotá, Colombia

`cristian.candia@mail.escuelaing.edu.co`

² Faculty of Engineering, Universidad de Ibagué, Ibagué, Colombia

`{manuel.forero, sergio.herrera}@unibague.edu.co`

Abstract. When modeling phenomena that cannot be studied by deterministic analytical approaches, one of the main tasks is to generate random variates. The widely-used techniques, such as the inverse transformation, convolution, and rejection-acceptance methods, involve a significant amount of statistical work and do not provide satisfactory results when the data do not conform to the known probability density functions. This study aims to propose an alternative nonparametric method for generating random variables that combines kernel density estimation (KDE), and radial basis function based neural networks (RFBNNs). We evaluate the method's performance using Poisson, triangular, and exponential probability density distributions and assessed its utility for unknown distributions. The results show that the model's effectiveness depends substantially on selecting an appropriate bandwidth value for KDE and a certain minimum number of data points to train the algorithm. The proposed method enabled us to achieve an R^2 value between 0.91 and 0.99 for analyzed distributions.

Keywords: General regression neural network · Probabilistic neural network · Kernel density estimation · Random variable · Probability distribution

1 Introduction

Computational models are a widely-used alternative method for solving problems that cannot be studied by deterministic analytical approaches [1]. This has led to the development of a fairly small number of density functions to describe how values are distributed over the sample spaces of a large number of real phenomena. However, preparing and statistically analyzing the data to take advantage of these distributions requires significant effort, and does not produce good results when the system analyzed depends on random variables that do not follow known probability density functions, leading us to look for unconventional alternatives that can reproduce the stochasticity of real systems [2].

Estimating random variable distributions has played an important role in several recent studies, studying monthly rainfall and water flow to determine drought indicators [3], predicting crime based on Twitter messages [4], and studying trends in the marine duck populations along the Atlantic coast of the United States [5].

For more than 40 years, nonparametric probability density estimation techniques, such as the Kolmogorov–Smirnov and chi-squared tests, have been the most widely-used density estimation methods, because they do not depend on the explicit form of the distribution or its parameter values, as parametric techniques do [6, 7]. However, these tests only suggest how to adjust the data when working with known distributions, and they are also sensitive to common errors in interpreting the p -value [8].

Since its introduction in 1956, KDE has become one of the most widely-used nonparametric density estimation methods [9, 10]. Over time, various authors have extensively modified the original technique in order to reduce its sensitivity to the choice of the kernel function and bandwidth [11]. Most recent methods suggest using maximum likelihood algorithms, with maximum entropy [12] and histogram trend filters [13]. These nonparametric techniques have proved useful for analyzing phenomena that do not conform to known distributions, such as wind speeds [14], crime prediction using social network data [4], and smart sensor-based electricity readings [15].

Once a given random variable’s distribution has been established, the subsequent problem consists of generating numerical values that follow the same distribution. The most common conventional random variable generation techniques are the inverse transformation, the acceptance-rejection, and convolution methods. However, these all have issues in terms of the calculation speed, computational resources required, and effort needed to prepare and statistically analyze the data [1]. Some researchers have presented universal methods of generating random variables by means of generalized acceptance-rejection algorithms [16], multilayer neural networks [17], transforming random variables to generate continuous distribution families [18], and specialized algorithms for producing particular distributions such as geometric [19] distributions.

This paper presents a new nonparametric approach that combines KDE with RFBNNs to generate random variable values regardless of their probability distributions and whether they are discrete or continuous variables, thus reducing the dependence on goodness of fit tests, for both data that follows a known distribution and those that are distributed atypically.

2 Kernel Density Estimation

KDE is a common nonparametric technique for estimating the probability density functions of random variables. Given a set of n independent observations X_1, X_2, \dots, X_n , represented by the same probability density function f , this model estimates the probability density function (PDF) f_n associated with these observations as follows:

$$f_n(x) = n^{-1}h^{-1} \sum_{j=1}^n K\{h^{-1}(x - X_j)\}. \quad (1)$$

Here, K is the kernel function, which weights the result by the proximity of x to the sampled points, and h is the bandwidth, which defines the size of the kernel function’s weighting window.

KDE’s accuracy depends on the kernel function K and bandwidth h used. The value of K does not significantly affect the model’s statistical efficiency, but it does impact the calculation speed for large data sets [10]. In contrast, the bandwidth h is a sensitive parameter that governs the model’s overall behavior, so it is essential to select an optimal value for it when estimating the PDF [10, 11].

3 Generalized Regression Neural Networks and Probabilistic Neural Networks (RFBNNs)

GRNNs and PNNs are RFBNNs introduced by Donald Specht between 1990 and 1991. In case of PNNs, Specht demonstrated that the Bayes–Parzen classifier can be split into many simple processes and hence implemented as a multilayer neural network [20], also showed that the GRNNs can be implemented for any regression problem in which an assumption of linearity is no justified [21]. In general, the structure for GRNNs and PNNs can be summarized as shown in Fig. 1:

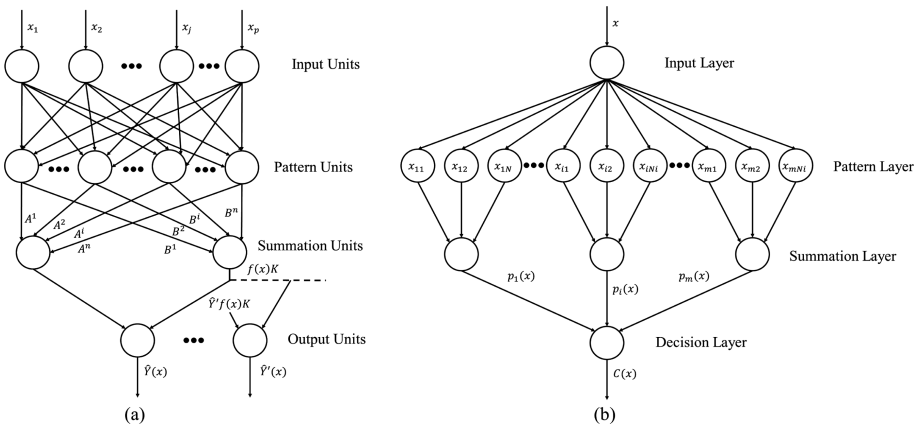


Fig. 1. Neural Networks structure for (a) GRNN (b) PNN. (Source: adapted from [20, 21])

In both cases the input layer distributes the input to the neurons of the next, or pattern, layer, when it receives a pattern x , pattern layer neuron x_{ij} calculates its output, that is later distributed in the units of sum that determine the output according to some weights or defined relation, finally the result obtained is used to estimate the classification generated in PNNs case, or the continuous value in GRNNs case [20, 21].

One important characteristic of this type of networks is that no iterative training is required; instead, the parameters are saved and used to make predictions. This makes it a computationally lightweight algorithm, which is significant when handling large amounts of data [22].

4 Combining KDE and RFBNNs

Figure 2 gives an overview of the proposed method. It starts by estimating the shape of the sample data’s PDF using KDE. Here we used the Epanechnikov weighting function and established an appropriate bandwidth for each data set using a local search procedure, starting from the reference bandwidth value proposed by Silver [10]. Once PDF’s shape is estimated, we compute the CDF’s shape using a numerical approximation of PDF’s area under curve by trapezoidal Riemann Sum. While in the analytical case the highest probability of CDF must be equal to one, in the estimation case, the highest value of the estimated CDF is better when is closer to one.

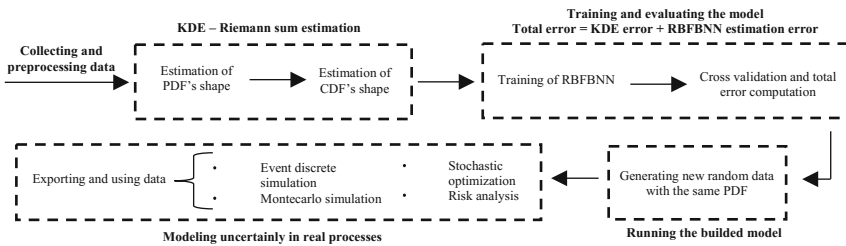


Fig. 2. Overview of the proposed method. (Source: the authors)

The computed points from CDF estimation are used to train a GRNN in case of continuous variable or a PNN in case of discrete variable. This produces a model that enables random values to be generated according to the same distribution as the sample data via an inverse transform procedure by replacing the CDF with the RFBNN.

5 Evaluation

For this study, a computer with an Intel Core i7 2.60 GHz processor and 8 GB of RAM was used. All the calculations were carried out using Python 3.2.6.

The GRNNs and PNNs implemented using the NeuPy library, were trained on 70% of the input data set, with the remaining 30% reserved for the subsequent validation step. For this evaluation, we used the learning curve algorithm from the scikit-learn library. This method evaluates the neural network’s accuracy by varying the training data set and performing repeated cross-validation, preserving the 70/30 split for each training data subset and using R^2 metric.

To evaluate the total error of our model (Eq. (2)), we used three probability distributions with known CDFs, so that their inverse transforms could be computed analytically for a given set of uniform random values, enabling us to calculate different errors between the analytic values obtained from the inverse transformations and the values generated by the RFBNN. The mixture of normal distributions data set was used as an illustration of applicability of proposed model to generate random variates from unknown distributions.

$$\text{Total error} = \text{KDE error} + \text{Riemann sum error} + \text{RFBNN error} \quad (2)$$

5.1 Data Sets Used

To evaluate the proposed model, we used four different data sets, of 600 samples each, generated using SciPy Python’s library. The first three data sets used the Poisson, triangular, and exponential distributions, while the fourth was a mixture of three different normal distributions, contributing 200 samples each. Table 1 lists the parameters used for each distribution, given according to SciPy’s nomenclature.

Table 1. Details of the data sets used.

Distribution	Type	Parameters	Total Samples
Poisson	Discrete	$\mu = 10, \text{loc} = 0$	600
Triangular	Continuous	$c = 1, \text{loc} = 0, \text{scale} = 1$	600
Exponential	Continuous	$\text{scale} = 20$	600
Mixture of normal	Continuous	$\text{loc } 1 = 15, \text{scale } 1 = 2$ $\text{loc } 2 = 25, \text{scale } 2 = 2$ $\text{loc } 3 = 35, \text{scale } 3 = 3$	600

6 Results and Discussion

6.1 PDFs and CDFs Estimations

Figure 3 shows histograms of the 600 samples from each probability distribution, together with the estimated PDFs and corresponding CDFs.

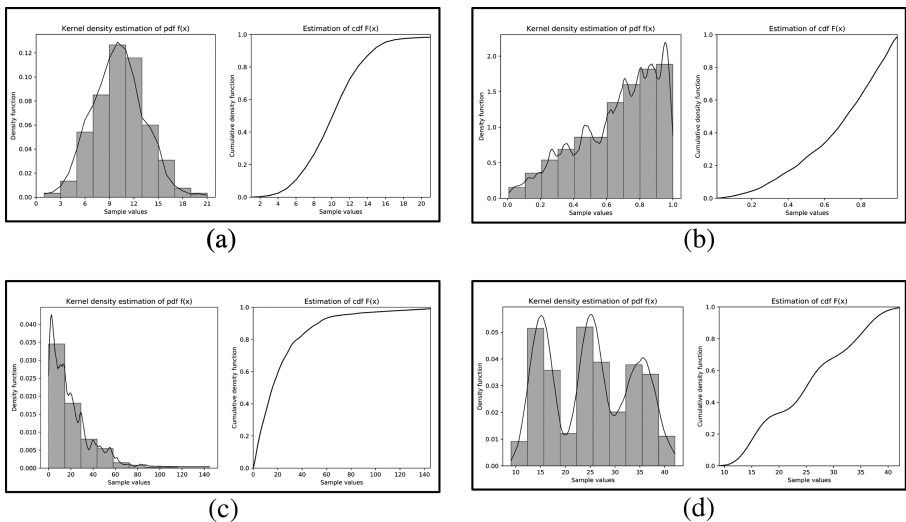


Fig. 3. KDE of the PDFs and CDFs for the (a) Poisson, (b) triangular, (c) exponential, and (d) mixed normal distribution data sets.

The bandwidth computed with the local search procedure shown in Table 2, allows good fit for PDF and CDF especially for Poisson and mixture of normal distributions, where the smoothed curves are closer to the histogram representations. In the case of triangular and exponential distributions, the KDE exhibits non-smoothed shapes in comparison with the histograms for both distributions, which is an evidence of histogram’s width class sensitivity in PDF’s shape estimation.

Table 2. Overall errors for the proposed method on each of known distributions.

Distribution	Kernel bandwidth	R ²	MAE	MSE	Explained variance
Poisson	1.1903	0.9824	0.1741	0.1763	0.9827
Triangular	0.0451	0.9980	0.0084	0.0001	0.9984
Exponential	2.9018	0.9118	3.6563	64.0301	0.9241

Otherwise, the learning curves shown in Fig. 4 for each data set, reflect the impact of the training data variation on the RBFNN’s R², allowing us to establish that, for the triangular and Poisson distributions, a training set of size approximately 150 was sufficient for good fitting, while, for the exponential distribution, the R² increases considerably above 300 samples, meaning that distributions with extreme values with low probability of occurrence, require a larger number of sample data for good CDF estimation.

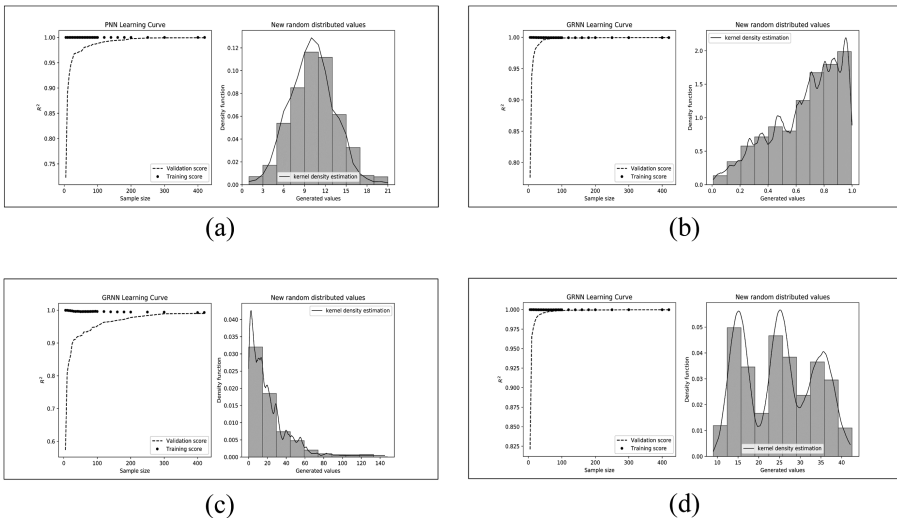


Fig. 4. Learning curves and estimated PDFs for the (a) Poisson, (b) triangular, (c) exponential, and (d) mixture normal distribution data sets.

Figure 4 also shows the histograms and estimated PDFs for 10,000 new random values generated by each of the RFBNN models, suggesting graphically that the original and generated data follow identical distributions.

6.2 Precision of the Method

We determined the method's overall precision by considering the MSE, mean absolute error (MAE), R^2 , and explained variance, calculated by comparing the analytic results with the KDE-based CDFs for each of the known probability distributions and the values generated by the RFBNNs, as discussed in Sect. 5.

The results shown in Table 2 indicate that the accuracy was generally good for the Poisson and triangular distributions, but the correlation and explained variance are notably reduced for the exponential distribution. This is probably due to the fact that exponential distribution includes extreme values that are unlikely to appear in the training data and therefore do not feature in the estimated PDFs and CDFs, weakening the GRNN's and PNN's ability to generate accurate results.

In case of mixture of normal distributions data set, where analytical CDF is unknown, we only analyzed KDE adjust between original data set and the histogram for 10,000 new random values generated using the proposed method, finding good fitting as shown in Fig. 4.

7 Conclusions

In this paper, we have proposed a nonparametric model for generating random variable values that has considerable advantages in terms of reducing the amount of preparatory and statistical analysis work required to represent the stochasticity of real phenomena in computer simulations. The integration of KDE and RFBNNs enable us to replicate known and unknown random variate distributions without needed of goodness of test fit procedures, improving the model's applicability when the data do not conform to the known probability density functions as shown before in the case of mixture of normal distributions.

One of the main weaknesses of our model is the strict need to establish a suitable KDE bandwidth, to prevent errors propagating to neural networks training process and hence guarantee the CDF curves are well-adjusted, since this involves a local search procedure to establish an adequate bandwidth value with an associated computational cost. This weakness is especially evident in distributions where unlikely values will generally not appear in the sample data, and thus will not be reflected in the KDE and the RFBNN's prediction, meaning a greater amount of data will be needed for training.

References

1. Banks, J. (ed.): Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice, 1 edn. Wiley-Interscience, New York/Norcross (1998)
2. Krishnamoorthy, K.: Handbook of Statistical Distributions with Applications, 2nd edn. CRC Press, Boca Raton (2016)

3. Svensson, C., Hannaford, J., Prosdocimi, I.: Statistical distributions for monthly aggregations of precipitation and streamflow in drought indicator applications. *Water Resour. Res.* **53**(2), 999–1018 (2017)
4. Gerber, M.S.: Predicting crime using Twitter and kernel density estimation. *Decis. Support Syst.* **61**(Suppl. C), 115–125 (2014)
5. Zipkin, E.F., Leirness, J.B., Kinlan, B.P., O’Connell, A.F., Silverman, E.D.: Fitting statistical distributions to sea duck count data: implications for survey design and abundance estimation. *Stat. Methodol.* **17**(Suppl. C), 67–81 (2014)
6. Berkson, J.: Some difficulties of interpretation encountered in the application of the chi-square test. *J. Am. Stat. Assoc.* **33**(203), 526–536 (1938)
7. Massey, F.J.: The Kolmogorov-Smirnov test for goodness of fit. *J. Am. Stat. Assoc.* **46**(253), 68–78 (1951)
8. Gutiérrez, M., Agustín, P., Gómez-Restrepo, C.: Beyond p value. *Rev. Colomb. Psiquiatr.* **38**(3), 574–586 (2009)
9. Rosenblatt, M.: Remarks on some nonparametric estimates of a density function. *Ann. Math. Stat.* **27**(3), 832–837 (1956)
10. Silverman, B.W.: Algorithm AS 176: kernel density estimation using the fast fourier transform. *J. R. Stat. Soc. Ser. C Appl. Stat.* **31**(1), 93–99 (1982)
11. Heidenreich, N.-B., Schindler, A., Sperlich, S.: Bandwidth selection for kernel density estimation: a review of fully automatic selectors. *AStA Adv. Stat. Anal.* **97**(4), 403–433 (2013)
12. Agarwal, R., Chen, Z., Sarma, S.V.: A novel nonparametric maximum likelihood estimator for probability density functions. *IEEE Trans. Pattern Anal. Mach. Intell.* **39**(7), 1294–1308 (2017)
13. Padilla, O.H.M., Scott, J.G.: Nonparametric density estimation by histogram trend filtering. [arXiv:150904348](https://arxiv.org/abs/150904348) *Stat.*, September 2015
14. Xu, X., Yan, Z., Xu, S.: Estimating wind speed probability distribution by diffusion-based kernel density method. *Electr. Power Syst. Res.* **121**, 28–37 (2015)
15. Arora, S., Taylor, J.W.: Forecasting electricity smart meter data using conditional kernel density estimation. *Omega* **59**(Part A), 47–59 (2016)
16. Barabesi, L., Pratelli, L.: Universal methods for generating random variables with a given characteristic function. *J. Stat. Comput. Simul.* **85**(8), 1679–1691 (2015)
17. Magdon-Ismail, M., Atiya, A.: Density estimation and random variate generation using multilayer networks. *IEEE Trans. Neural Netw.* **13**(3), 497–520 (2002)
18. Alzaatreh, A., Lee, C., Famoye, F.: A new method for generating families of continuous distributions. *METRON* **71**(1), 63–79 (2013)
19. Bringmann, K., Friedrich, T.: Exact and efficient generation of geometric random variates and random graphs. In: Fomin, F.V., Freivalds, R., Kwiatkowska, M., Peleg, D. (eds.) *ICALP 2013. LNCS*, vol. 7965, pp. 267–278. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39206-1_23
20. Specht, D.F.: Probabilistic neural networks. *Neural Netw.* **3**(1), 109–118 (1990)
21. Specht, D.F.: A general regression neural network. *IEEE Trans. Neural Netw.* **2**(6), 568–576 (1991)
22. Pedregosa, F., et al.: Scikit-learn: machine learning in python. *J. Mach. Learn. Res.* **12**, 2825–2830 (2012)