NIST Technical Note 2152

Neural Networks for Classifying Probability Distributions

Siham Khoussi Alan Heckert Abdella Battou Saddek Bensalem

This publication is available free of charge from: https://doi.org/10.6028/NIST.TN.2152



NIST Technical Note 2152

Neural Networks for Classifying Probability Distributions

Siham Khoussi Abdella Battou Advanced Network Technologies Division Information Technology Laboratory

> Alan Heckert Statistical Engineering Division Information Technology Laboratory

> Saddek Bensalem University of Grenoble Alpes (UGA) Grenoble, France

This publication is available free of charge from: https://doi.org/10.6028/NIST.TN.2152

April 2021



U.S. Department of Commerce Gina M. Raimondo, Secretary

National Institute of Standards and Technology James K. Olthoff, Performing the Non-Exclusive Functions and Duties of the Under Secretary of Commerce for Standards and Technology & Director, National Institute of Standards and Technology Certain commercial entities, equipment, or materials may be identified in this document in order to describe an experimental procedure or concept adequately. Such identification is not intended to imply recommendation or endorsement by the National Institute of Standards and Technology, nor is it intended to imply that the entities, materials, or equipment are necessarily the best available for the purpose.

> National Institute of Standards and Technology Technical Note 2152 Natl. Inst. Stand. Technol. Tech. Note 2152, 20 pages (April 2021) CODEN: NTNOEF

> > This publication is available free of charge from: https://doi.org/10.6028/NIST.TN.2152

Abstract

Probability distribution fitting of an unknown stochastic process is an important preliminary step for any further analysis in science or engineering. However, it requires some background in statistics, prior considerations of the process or phenomenon under study and familiarity with several distributions. As such, this paper presents an alternative approach which doesn't require prior knowledge of statistical methods nor previous assumption on the available data. Instead, using Deep Learning, the best candidate distribution is extracted from the output of a neural network that was previously trained on a large suitable database in order to classify an array of observations into a matching distributional model. We find that our classifier can perform this task comparably to using maximum likelihood estimation with an Anderson-Darling goodness of fit test.

Key words

Deep Learning; neural networks; distribution fitting; data normalization.

Table of Contents

1	Introduction	1				
2	Related work	2				
3	Methodology	3				
	3.1 Data collection	3				
	3.2 Neural networks	5				
	3.3 Evaluation	7				
4	Results	7				
5	Conclusion	11				
6	Limitations	13				
7	Future work	13				
Re	eferences					

List of Tables

Table 1 Confusion matrix for the large category: NN vs maximum likelihood/Anderson-	
Darling (MLE-AD)	11
Table 2 Confusion matrix for the moderate category: NN vs maximum likelihood/Anders	on
Darling (MLE-AD)	12
Table 3 Confusion matrix for the small category: NN vs maximum likelihood/Anderson-	
Darling (MLE-AD)	12

List of Figures

Traditional approach of conducting distribution fitting	1
DEX mean plot for small sample sizes	9
DEX mean plot for moderate sample sizes	10
DEX mean plot for large sample sizes	10
	Traditional approach of conducting distribution fitting DEX mean plot for small sample sizes DEX mean plot for moderate sample sizes DEX mean plot for large sample sizes

1. Introduction

Many problems in both science and engineering require fitting a distributional model to a uni-variate dataset. That is, data consisting of a set of N empirical observations obtained by measuring a certain system, provided that the measurements are independent and come from the same distribution.

Distributional modeling can be used in several contexts. Statistical tests typically depend on certain assumptions with regards to the underlying distribution (e.g., many tests are based on the assumption of normality). Appropriate distributional models are also used to more accurately assess uncertainty and, in particular, to assess the uncertainty over the full range of the data. Although a poorly chosen distributional model may suffice for measuring and assessing the uncertainty of averages, this will not be the case for tails of the distribution. In many applications (e.g., reliability), accurate assessment of the tail behavior is more critical than the average. Obtaining an appropriate distributional model can be an exhaustive process that takes time, patience and requires previous knowledge of statistics and is, therefore, a difficult task for some analysts.



Fig. 1. Traditional approach of conducting distribution fitting

Figure 1 presents the traditional approach to obtaining an appropriate distributional model. This approach consists of four steps: The first step is to assess whether the data is in fact independent. This can be done via standard statistical tests for randomness or auto-correlation (e.g., runs test, Ljung-Box test). It can also be assessed graphically (e.g., a lag plot or an auto-correlation plot). The second step is to identify potential distributional models. Typically, histograms or kernel density plots are used to help identify the basic shape of the underlying distribution as well as certain properties such as the skewness and the presence of multiple modes in the data. Identifying good potential models from these plots typically requires some degree of statistical knowledge, experience and familiarity with

several distributions. The third step is to estimate the parameters of the chosen distribution via methods like maximum likelihood. The fourth and final step is to assess the goodness of fit of the proposed distributional model via one of the many goodness of fit tests such as the Anderson-Darling test, the Kolmogorov-Smirnov test, the Cramer-Von Mises test or information criteria such as AIC and BIC.

This research focuses on the use of Deep Learning (DL) to assist in identifying the best distributional model among a fixed set of candidate models (step 2) in order to help analysts who are not equipped with sufficient statistical background easily map a set of empirical observations obtained from an experiment to an appropriate distributional model. DL is a sub-field of machine learning that applies neural network architectures to learn features of the object to be classified. It has become more popular in recent years due to its high accuracy, its capacity to deal with massive data and larger neural networks as well as its capability to deduce data features automatically. Although DL neural networks take a longer duration to train and usually require high performance servers with graphic processing units (GPU) which are expensive, it is still considered an efficient and effective approach for object classification.

The approach proposed here exploits the strengths of Deep Learning for classification of distributional models. We restrict ourselves to the case of continuous measurements where the data is not binned, censored or truncated. Moreover, we do not consider pathological distributions (e.g. Cauchy distribution). In this paper, we train a feed forward neural network to recognize patterns in an input dataset then predict the "best" candidate model from nine commonly used distributions that are widely encountered. The distributions considered in this study are: uniform, normal, logistic, exponential, half normal, half logistic, gumbel max, gumbel min and double exponential. Note that once the DL has identified the "best" distributional model (step 2 according to Figure 1), the parameter estimation and the goodness of fit assessment still need to be applied using traditional statistics. Although, in this study a limited list of distributions is examined, it is covering the most encountered models. Moreover, this paper is intended to serve as a proof of concept to prove the viability of our method then increase the number of the distributions later on.

In this paper, we present similar related works in section 2. We explain our methodology and evaluation metrics in section 3. The results of our analysis are shown in section 5. We also discuss the limitations of this work in section 6 and finally preview our ongoing and future research in Section 7.

2. Related work

There have been several attempts to automate the distributional modeling process by creating software and packages that let the user know the "best" candidate distribution that matches their inputs. Generally, most of them use traditional statistical techniques to preprocess data and run some goodness of fit tests in order to rank and identify a good representation of the data. Similar tools and packages include [1], [2] and [3], etc. All of them have the same goal which is to guide and help analysts regardless of their knowledge of statistics, pin down the best candidate distribution matching their data and avoid using the wrong distribution while saving them time.

Other researchers in the literature investigated the use of neural network for conducting parameter estimation of probability density functions which corresponds to step 3 from Figure 1. Similar papers include [4], [5] and [6]. Others have discussed conditional density estimation using artificial intelligence such as [7] and [8]. Additionally, [9] presented the best practices for conditional density estimation for finance applications, specifically, and using neural networks. Finally, [10] used an ensemble of mixture density networks to predict the probability density function of the surf height in order to know if it will fall within a given 'surfable' range. However, at the time of this writing, there has been no work involving the use of neural network to tackle step 2 of Figure 1 and create a classifier for distributional models based on a set of independent empirical observations. In fact, this task is the most important since one it is completed and the exact distribution model has been identified, it becomes extremely easy to estimate the parameters of the distribution and formulate the probability density function (PDF).

3. Methodology

3.1 Data collection

In this paper, we collected data in two stages. The first stage was to create the datasets for training and validating the neural networks.

Probability distributions are characterized by three types of parameters: a location, a scale and one or more shape parameters. The standard form of the distribution is the case where the location parameter is zero and the scale parameter is one. Given a graph of the standard form of the probability density function, the effect of a non-zero location parameter is to shift the graph left (for negative location values) or right (for positive location values) on the horizontal axis. The effect of a scale parameter is to either stretch the graph on the horizontal axis (for scale parameters greater than one) or to compress the graph on the horizontal axis (for scale parameters less than one). Chapter one from the NIST/SEMATECH e-Handbook [11] shows examples of this for the normal distribution. The relationship of the probability density function of a general form of the distribution (i.e., location and scale not equal to zero and one) is:

$$f(x;a,b) = \frac{f(\frac{(x-a)}{b};0,1)}{b}$$
(1)

where a and b are the location and scale parameters, respectively. Any parameter that is not either a location or scale (or a parameter that is a function of the location and scale parameters only) is considered a shape factor. Shape parameters allow a distribution to take a variety of different shapes. By shape, we mean properties such as skewness and kurtosis (peakedness). Location and scale parameters have no effect on these properties. For this study, we only considered distributions without shape parameters and keep the ones with the location and scale factors. Since the shape of the distribution does not depend on the location and scale parameters, the training data only utilized the standard form of the distributions (location = 0, and scale = 1).

For each of the nine distributions considered in this study, 10,000 datasets were generated for the standard form of the distributions at different sample sizes (30, 50, 100, 250, 500, 750, 1,000 and 10,000). The random numbers were generated with the Dataplot software [12]. And a congruential-Fibonnaci [13] generator was used with a different seed for each distribution/sample size configuration. For each set of random numbers, a kernel density plot was generated with Dataplot using the Silverman algorithm [14]. The kernel density plot is a graphical estimate of the underlying probability density function and is considered a typical technique by statisticians to conduct exploratory analysis of data in order to identify the shape of the underlying distributional model. Furthermore, the intuition behind the choices of the sample sizes is based on the fact that the smoothness of the kernel density plot increases with the increase of the number of available data points. In this study, several experiments were conducted to assess and select the correct sample sizes prior to training the neural networks that will eventually determine if the underlying distribution can be recognized from the kernel density plot.

The kernel density estimate, $f_n(x)$, of a set of *n* points from a density *f* is defined as:

$$f_n(x) = \frac{\sum_{j=1}^n K\{\frac{(x-X_j)}{h}\}}{nh}$$
(2)

where K is the kernel function and h is the smoothing parameter or window width. The Silverman algorithm uses a Gaussian kernel function. This down weights points smoothly as the distance from x increases. We used Silverman's default recommendation for the h parameter:

$$0.9\min(s, \mathrm{IQ}/1.34)n^{-1/5}$$
 (3)

with *s*, IQ, and *n* denoting the sample standard deviation, the sample interquartile range and the sample size, respectively. The kernel density plot was generated at 256 points. The input for the neural networks is the y-axis coordinates of the kernel density plot. The implicit x-axis coordinates are 1, 2, ..., 256.

The second stage was to create datasets for testing. Typically, real world data will have location and scale values, so we generated random numbers for each distribution with several different location and scale parameters. Specifically, datasets were generated with sample sizes of 50, 100, 250, 500, 750, 1,000 and 10,000. For each sample size, datasets were generated with location values: 20, 60 and 100 and scale values: 10, 30 and 50. This adds up to a total of nine different combinations of location and scale parameters. Then, we generated 1,000 datasets for each distribution/sample size/location/scale combination.

As with the training data, kernel density plots were generated for each dataset. And the algorithm used for creating the kernel density plots in the training data, was also used for this set.

One question of interest is whether an appropriate normalization can address the issues introduced by the location and scale parameters. The training and the validation data was generated for standard forms of the distributions (location parameter = 0, scale parameter = 1) while the testing data was generated with non-standard values of the location and scale parameters. The y-coordinate (height) of the kernel density plots is used for the input to the neural networks (NN) which gives an implicit x-coordinate scale of 1 to 256. The location parameter does not change the height of the kernel density. However, the scale parameter does change the height of the kernel density plot.

Using an implicit x-axis scale of 1 to 256 for both the training and testing data should minimize the effects of the location parameter. However, since the scale parameter changes the height of the kernel density plot, there is a need to transform the kernel density heights so that the testing data can be more effectively compared to the training and validation data.

In this study, we experimented with several transformation algorithms on both the training/validation and the testing data. But, only two yielded promising results:

1. The U-score, also referred to as the Min-Max normalization. The u-score algorithm transforms the kernel density heights to a (0,1) scale according to the following mathematical formulation:

$$u_score = \frac{x - min(x)}{max(x)) - min(x)}$$
(4)

where x is the original value, u_score is the normalized value, min(x) and max(x) are respectively the minimum and maximum values of each dataset x.

2. The kernel density normalization transforms the kernel density heights to integrate to 1 on the 1 to 256 x-coordinate scale:

$$k_score = \frac{x}{\sum_{i=1}^{256} x_i}$$
(5)

where x is the original value and k-score is the normalized value.

3.2 Neural networks

An initial attempt at producing a single solid neural network model to classify an arbitrary number of data points (N) to a matching probability density function (PDF) has not yielded promising results especially when N is small. The intuition behind the misclassification could be interpreted as follows: our approach relies on building a kernel density estimator (KDE) from a set of independent empirical observations. This KDE tends to be noisy for small N and becomes increasingly smooth as N gets larger. Thus, models trained on the larger sample sizes perform poorly on the smaller sample sizes and models trained on smaller sample sizes perform poorly for larger sample sizes.

To improve performance, we consider 20+ models. The data collected is generated using eight sample sizes: N=30, N=50, N=100, N=250, N=500, N=750, N=1,000 and

N=10,000 and each model is trained on a specific sample size range. The idea is to evaluate the models individually and collectively to deduce which ones work best for small, moderate and large sample sizes. Examples of the considered models include: Model 1 ($N \in [30, 100]$), Model 2 ($N \in [100, 750]$), Model 3 ($N \in [750, 10000]$), etc.

All the models have the same input and output layers. However, their hidden layers differ in size and width. The input layer has 256 unit representing the Y-Axis coordinates for a kernel density plot whereas the output layer has 9 points which refer to the one hot encoding of the 9 distributional models considered in this paper. For each interval, we started with a very simple Feed Forward Neural Network (FNN) that overfits. We then proceeded to handle the overfiting by tuning FNN parameters to achieve the lowest loss and highest accuracy. We found that the following work best:

- 1. 20% of the training data was allocated for the validation;
- 2. All models use Softmax as the activation function for the output layer and Relu for the hidden layers;
- 3. The choice of the loss function was Categorical cross-entropy (CAT) for larger intervals and Mean squared error (MSE) for smaller intervals;
- 4. The ADAM optimizer was used when the loss function was set to CAT and RMSprop for MSE;
- 5. The learning rate is set to 1e-6 or 1e-5 in most cases;
- 6. The batch size is set to 200 for most models;
- 7. Each model was run for an average epoch of 500;
- 8. The weights were initialized using the 'He uniform' distribution;
- 9. The bias was enabled in the hidden layers and disabled in the output layer;
- 10. The depth of each NN model was 40, while the width was either 512 or 1024 nodes per layer;
- 11. Early stopping was deployed;
- 12. Regularized and dropout were used.

The models were all implemented using Python and Keras with Tensorflow as a backend and the experiment was run on our testbed with with 1 GPU and 40 CPU cores.

3.3 Evaluation

After the training and validation steps of the neural networks were complete, additional datasets were used to test the models (stage 2 from section 3.1). Furthermore, to determining the viability and effectiveness our approach, the models' accuracy results were compared against the results of a conventional statistical approach as follows:

- The data is fit to each distribution using maximum likelihood (MLE). The one exception is that the half-logistic distribution is fit using the method of moments.
- After estimating the parameters with maximum likelihood, the distributions are ranked based on the Anderson-Darling (AD) goodness of fit statistic [15]. The AD test is a refinement of the Kolmogorov-Smirnov (KS) statistic that puts more weight in the tails of the distribution. The AD test is generally considered to have more power than the the KS test.

Given an ordered set of data Y_i and a cumulative distribution function F, the Anderson-Darling test statistic is defined as

$$A^{2} = -N - \sum_{i=1}^{N} \frac{(2i-1)}{N} [\ln F(Y_{i}) + \ln (1 - F(Y_{N+1-i}))]$$
(6)

There are a variety of estimation methods and goodness of fit statistics that could be used for this approach. However, the combination of MLE estimation and ranking by the AD goodness of fit test provides a reasonable benchmark for assessing the results of the neural networks. Moreover, it is important to mention that both approaches (neural networks and MLE-AD) were compared based on the same data.

4. Results

Our primary metric of success was the percentage of times that the correct distribution was accurately identified. For the mis-classified cases, we also identify which distributions were chosen instead. The following factors are examined while analyzing the results:

- There are two different normalization algorithms considered. These are referred to as the u-score and the kernel density normalization, respectively (section 3.1);
- There were eight different sample sizes used for the testing datasets. We grouped these into three categories: "small", that is datasets with 30 or 50 or 100 observations; "moderate", that is datasets with 100, 250, 500 or 750 observations; and "large", that is datasets with 750, 1,000 or 10,000 observations. Note that these categories contain overlaps in order to create three intervals: small [30, 100], moderate [100, 750] and large [750, 10000]. These intervals are useful because real world data is not confined to these particular eight sizes;

- This study considered nine distinct distributions. These distributions allow for location and scale parameters, but none of them have shape parameters;
- There were 20+ training models considered;
- There were nine combinations of location/scale parameters for each distribution/sample size cell.

As a first step, we generated Design of Experiments (DEX) mean plots [16] as shown in Figures 2, 3 and 4. DEX mean plots are useful for showing the most important factors. We generated separate DEX plots for the small, moderate and large sample sizes. Initial DEX plots included all of the training models. However, due to space limitations, we only show plots with the most effective training models for each sample category. Some initial conclusions from these DEX mean plots are:

- When we look at the location/scale factors for the plots that only include the best performing training models, the effect is negligible. For this reason, in subsequent analysis, the data for all nine location/scale combinations are aggregated into a single value.
- The u-score and kernel density normalization methods have similar performance.
- For each sample size category, there is significant variability in the performance of the different training models. The best training models are different for the three sample size categories, but for a given sample size category there are several training models that have similar performance. According to Figures 2, 3 and 4, we highlight the following best three models: ¹:
 - 1. For a small sample size, we choose the training model '30-50-100' and the u-score normalization. Table 3 shows the confusion matrix for this model compared against MLE-AD (rounded to two decimal places);
 - For moderate and large sample sizes, we choose the training model '100-250-500-750-1000-10000' and the kernel normalization. Tables 2 and 1, respectively show the confusion matrix for this model compared against MLE-AD for the moderate and the large categories (rounded to two decimal places);
- There are performance differences between the distributions. Specifically, the halflogistic and and the logistic have significantly poorer performance than the other distributions. This is not surprising as these have similar shapes to the half-normal and normal distributions, respectively.

¹uniform, 2: normal, 3: logistic, 4: exponential, 5: double-exponential, 6: half-normal, 7: half-logistic, 8: gumbel-min, 9: gumbel-max

As expected, performance improves as the sample size increases. According to the DEX mean plots, for the small category, the overall performance was approximately 70%, for the moderate category the overall performance was close to 85%, and for the large category the overall performance was about 98%. The latter means that with a large number of observations, our NN can identify "the correct" distribution.



Fig. 2. DEX mean plot for small sample sizes



Fig. 3. DEX mean plot for moderate sample sizes



Fig. 4. DEX mean plot for large sample sizes

		Selected distribution								
True Distribution	Approach	1	2	3	4	5	6	7	8	9
	NN	100.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Uniform	MLE-AD	99.99	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	NN	0.00	96.77	3.20	0.00	0.00	0.00	0.00	0.01	0.02
Normal	MLE-AD	0.00	96.38	3.62	0.00	0.00	0.00	0.00	0.00	0.00
	NN	0.00	1.87	96.94	0.00	1.13	0.00	0.00	0.03	0.03
Logistic	MLE-AD	0.00	0.79	99.08	0.00	0.12	0.00	0.00	0.00	0.01
	NN	0.00	0.00	0.00	98.93	0.00	0.00	1.07	0.00	0.00
Exponential	MLE-AD	0.00	0.00	0.00	99.93	0.00	0.00	0.06	0.01	0.00
	NN	0.00	0.00	0.30	0.00	99.70	0.00	0.00	0.00	0.00
Double Exponential	MLE-AD	0.00	0.00	0.34	0.00	99.64	0.00	0.00	0.01	0.00
	NN	0.00	0.00	0.00	0.00	0.00	96.45	3.55	0.00	0.00
Half Normal	MLE-AD	0.00	0.00	0.00	0.00	0.00	99.57	0.34	0.00	0.09
	NN	0.00	0.00	0.00	2.32	0.00	2.47	95.21	0.00	0.00
Half Logistic	MLE-AD	0.00	0.00	0.00	0.81	0.00	5.70	93.48	0.00	0.01
	NN	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00	0.00
Gumbel Min	MLE-AD	0.00	0.00	0.00	0.00	0.01	0.00	0.00	99.99	0.00
	NN	0.00	0.00	0.00	0.01	0.00	0.00	0.01	0.00	99.98
Gumbel Max	MLE-AD	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	99.99

 Table 1. Confusion matrix for the large category: NN vs maximum likelihood/Anderson-Darling (MLE-AD)

For evaluation purposes, we also provide confusion matrices for MLE-AD in the same tables as the output of the Neural Networks (NN) (tables 1, 2 and 3). Theses tables indicates that neural networks perform comparably to MLE-AD and give better performance in a majority, but not all, of the cases, than the MLE-AD method. In fact:

- For the small category, DL outperforms MLE-AD for 6 out of 9 distributions and they perform essentially the same for the half-logistic distribution;
- For the moderate category, DL outperforms MLE-AD for 6 out of the 9 distributions and they perform essentially the same for the Gumbel max distribution;
- For the large category, ML-AD performs slightly better for 3 distributions, DL performs slightly better for one distribution and for the remaining 5 distributions they perform essentially the same.

5. Conclusion

In this empirical study, we investigated the use of neural networks for distributional models classification. Given a set of independent empirical observations obtained from an unknown process or phenomenon, we show that a neural network classifier is capable of identifying which distributional model is best fitted for the input data.

		Selected distribution								
True Distribution	Approach	1	2	3	4	5	6	7	8	9
	NN	99.95	0.03	0.00	0.00	0.00	0.00	0.00	0.02	0.01
Uniform	MLE-AD	73.78	15.48	0.07	0.00	0.00	0.94	0.00	5.30	4.43
	NN	0.10	91.01	7.08	0.00	0.34	0.00	0.00	0.74	0.72
Normal	MLE-AD	0.00	79.27	18.55	0.00	0.38	0.00	0.00	0.93	0.86
	NN	0.01	16.42	77.15	0.00	4.44	0.00	0.00	0.96	1.03
Logistic	MLE-AD	0.00	8.71	85.67	0.00	4.52	0.00	0.00	0.54	0.55
	NN	0.00	0.00	0.00	86.62	0.00	1.30	12.07	0.00	0.00
Exponential	MLE-AD	0.00	0.00	0.00	87.98	0.00	0.43	10.89	0.00	0.70
	NN	0.00	0.53	10.19	0.00	88.66	0.00	0.00	0.32	0.31
Double Exponential	MLE-AD	0.00	0.09	10.57	0.00	89.04	0.00	0.00	0.13	0.17
	NN	0.03	0.00	0.00	0.29	0.00	88.85	9.24	0.00	1.58
Half Normal	MLE-AD	0.00	0.24	0.10	0.00	0.00	87.50	3.00	0.00	9.15
	NN	0.00	0.00	0.00	8.55	0.00	15.93	74.96	0.00	0.57
Half Logistic	MLE-AD	0.00	0.01	0.01	5.18	0.00	20.69	68.84	0.00	5.27
	NN	0.03	1.02	0.39	0.00	0.07	0.00	0.00	98.49	0.00
Gumbel Min	MLE-AD	0.00	0.59	1.60	0.00	0.20	0.00	0.00	97.62	0.00
	NN	0.02	0.85	0.27	0.03	0.06	0.57	0.26	0.00	97.94
Gumbel Max	MLE-AD	0.00	0.54	1.46	0.00	0.20	0.13	0.01	0.00	97.66

Table 2. Confusion matrix for the moderate category: NN vs maximum

 likelihood/Anderson-Darling (MLE-AD)

Table 3. Confusion matrix for the small category: NN vs maximum likelihood/Anderson-Darling (MLE-AD)

		Selected distribution								
True Distribution	Approach	1	2	3	4	5	6	7	8	9
	NN	97.44	0.89	0.00	0.01	0.00	0.73	0.02	0.58	0.33
Uniform	MLE-AD	3.72	52.00	0.64	0.00	0.00	2.68	0.01	21.72	19.24
	NN	1.99	64.13	17.94	0.00	4.04	0.46	0.00	5.89	5.54
Normal	MLE-AD	0.00	51.72	31.96	0.00	3.07	0.00	0.00	6.81	6.44
	NN	0.37	28.42	38.84	0.00	19.99	0.20	0.00	6.14	6.03
Logistic	MLE-AD	0.00	19.74	57.32	0.00	13.28	0.00	0.00	4.81	4.86
	NN	0.02	0.00	0.00	79.29	0.00	5.96	14.58	0.00	0.16
Exponential	MLE-AD	0.00	0.11	0.09	30.44	0.00	1.58	51.80	0.00	15.98
	NN	0.03	4.41	14.17	0.01	75.39	0.03	0.01	2.96	2.98
Double Exponential	MLE-AD	0.00	1.51	30.37	0.00	63.49	0.00	0.00	2.18	2.45
	NN	1.04	0.22	0.01	4.67	0.01	69.24	19.06	0.00	5.76
Half Normal	MLE-AD	0.00	2.80	1.22	0.00	0.08	36.36	13.61	0.02	45.91
	NN	0.13	0.01	0.01	24.99	0.00	29.99	41.84	0.00	3.03
Half Logistic	MLE-AD	0.00	0.49	0.39	3.79	0.03	19.34	41.84	0.01	34.10
	NN	0.70	4.82	1.65	0.00	1.43	0.00	0.00	91.39	0.01
Gumbel Min	MLE-AD	0.00	4.29	6.81	0.00	1.67	0.00	0.00	87.21	0.03
	NN	0.68	4.57	1.58	0.21	1.40	6.55	3.14	0.02	81.84
Gumbel Max	MLE-AD	0.00	4.07	6.44	0.00	1.74	0.69	0.46	0.05	86.54

We chose two neural networks models depending on the number of available data points (small, moderate and large) and apply a suitable normalization technique (kernel density normalization or u-score normalization) then run the points through the neural networks to predict the "best fitted" distributional model.

We validated the results by comparing them to a traditional statistical approach: parameter estimation by maximum likelihood with subsequent goodness of fit ranking by Anderson-Darling (MLE-AD) and showed that this approach outperforms MLE-AD in a majority of cases.

6. Limitations

In this study we proposed the use of deep learning to build a classifier for distributional modeling. This classifier takes as input a set of data points and provides a distribution label that matches one of nine most common distributions.

This approach is not a complete replacement of the traditional statistical methodology that statisticians have been following to analyze and fit the data. However, it is an alternative to step 2 from Figure 1 (Exploratory analysis) which usually requires a good background of statistical knowledge as well as a familiarity with several distributions to be able to recognize a good potential distribution from a set of empirical observations. This is typically done via histograms or kernel density plots to help pin down the basic shape of the underlying distribution and find properties such as the skewness and the presence of multiple modes in the data.

Moreover, this paper considers uni-variate non-censored and non-truncated data and doesn not consider families of distributions with the shape parameters or noisy data which is generally a mixture distributions. In this research, We consider a limited number of distributions that correspond to the most commonly used models that are widely encountered. The reason behind our decision is that we hope to provide an initial working prototype that can prove the viability and applicability of our methodology.

7. Future work

In future work, we will extend the training set beyond the nine currently supported distributions. In particular, this will include commonly used families of distributions such as the Weibull, lognormal and gamma distributions. These families can generate a variety of shapes based on the value of their shape parameter. For this reason, we plan to incorporate the ability to make more specific classifications (e.g., distinguish between a Weibull or a lognormal distribution) and compare this to approaches such as the likelihood ratio test [17], [18].

Furthermore, we are currently building a tool that automates the distributional fitting process for uncensored and unbinned uni-variate data that deploys our trained neural networks to identify the best candidate model from the distributions presented in this study.

This tool takes empirical observation of any size and computes the kernel density estimation on behalf of the user, then runs the classifier to predict the best fit distribution.

In addition to that, we are also deploying traditional statistical techniques to estimate the parameters of the fitted distribution and assess it's goodness of fit by comparing the predictions to Anderson Darling, Kolmogorov-Smirnov and the probability plot correlation coefficient tests as well as information criteria (AIC, BIC).

Moreover, we include in the tool an interactive module to clean and pre-process the data before starting the NN classifier. This module will contain a step by step guide to help users identify and eventually remove outliers prior to running the neural network classification. It is very important because outlier identification is for the purpose of identifying bad data in the sense of being erroneous (e.g., data is mis-coded or there is an assignable cause for why the observation is in error). Furthermore, statistical classification of an observation as an outlier is dependent on the underlying distribution of the data, which is what we are trying to determine, so simply being an "extreme" observation is not sufficient justification for removing it.

References

- [1] Delignette-Muller M, Dutang C (2015) fitdistrplus: An R Package for Fitting Distributions. *Journal of Statistical Software* 64(4):1–34.
- [2] Law AM, McComas MG (2011) How the ExpertFit distribution-fitting software can make your simulation models more valid. *Proceedings of the Winter Simulation Conference* 1:199–204.
- [3] Schittkowski K (2002) EASY-FIT: a software system for data fitting in dynamical systems. *Structural and Multidisciplinary Optimization* 23:153–169.
- [4] Reyneri L, Colla V, Vannucci M (2011) Estimate of a probability density function through neural networks 6691(1):57–64.
- [5] Likas A (2001) Probability density estimation using artificial neural networks. *Computer Physics Communications* 135:167–175.
- [6] Nakamura Y, Hasegawa O (2017) Non parametric density estimation based on selforganizing incremental neural network for large noisy data. *IEEE Transactions on Neural Networks and Learning Systems* 28(1):8–17.
- [7] Kobyz GV, Zamyatin AV (2015) Conditional probability density estimation using artificial neural network :441–445.
- [8] Wesley Tansey KP, Scott JG (2016) Better Conditional Density Estimation for Neural Networks Available at https://arxiv.org/abs/1606.02321.
- [9] Rothfuss J, Ferreira F, Walther S, Ulrich M (2019) Conditional Density Estimation with Neural Networks: Best Practices and Benchmarks Available at http://arxiv.org/ abs/1903.00954.
- [10] Carney CP Michael, Dowling J, Lee C (2005) Predicting probability distributions for surf height using an ensemble of mixture density networks :113–120.
- [11] Filliben JJ (2003) NIST/SEMATECH e-Handbook of Statistical Methods,

Chapter 1 (National Institute of Standards and Technology), . Available at http://web.archive.org/web/20191213225442/https://www.itl.nist.gov/div898/handbook/eda/section3/eda364.htm.

- [12] Filliben JJ, Heckert AN (1978) Dataplot (National Institute of Standards and Technology), Available at http://web.archive.org/web/20190819195854/https://www.itl.nist.gov/div898/software/dataplot/.
- [13] David Kahaner GEFSNMAM Cleve B Moler (1988) Numerical Methods and Software, Available at https://books.google.com/books/about/Numerical{_}Methods{_} and{_}Software.html?id=jipEAQAAIAAJ.
- [14] Silverman BW (1982) *Kernel Density Estimation Using the Fast Fourier Transform*, . Available at https://rss.onlinelibrary.wiley.com/doi/epdf/10.2307/2347084.
- [15] Stephens MA (1974) EDF Statistics for Goodness of Fit and Some Comparisons. Journal of the American Statistical Association 69(347):730.
- [16] Filliben JJ (2003) Mean Plot (National Institute of Standards and Technology),
 Available at http://web.archive.org/web/20180217195200/http://www.itl.nist.gov/ div898/handbook/eda/section3/dexmeanp.htm.
- [17] Dumonceaux R, Antle CE, Haas G (1973) Likelihood Ratio Test for discrimination between two models with unknown scale and location parameters. *Technometrics* 15(1):19.
- [18] Dumonceaux R, Antle CE (1973) Discrimination Between the Log-Normal and the Weibull Distributions. *Technometrics* 15(4):923–926.