Why Skew-Normal Distributions and How They Are Related to ReLU Activation Function in Deep Learning

Damian Lorenzo Gallegas Espinosa, Olga Kosheleva, and Vladik Kreinovich

Abstract Normal distributions are ubiquitous, but many actual distributions are different from normal – for example, they are skewed. To describe such distributions, it is desirable to have a few-parametric family that extends the family of normal distributions. Several such families have been proposed. Empirically, the most effective among them is the family of so-called skew-normal distributions first proposed by A. Azzalini. In particular, this family is effective in econometrics. In this paper, we provide a theoretical explanation for this empirical success. This explanation is similar to an explanation of what ReLU activations functions are most effective in deep learning.

1 Why skew-normal distributions: a challenge

Normal distributions and why they are ubiquitous: a brief reminder. In order to understand our problem, let us first recall what are normal distributions and why they are ubiquitous.

In many practical situation, we have many small independent factors affecting the desired quantity. In such cases, according to the Central Limit Theorem (see, e.g., [4]), the resulting distribution is close to *Gaussian* (*normal*), i.e., a distribution with the following probability density function:

Damian Lorenzo Gallegas Espinosa
Department of Computer Science, University of Texas at El Paso, 500 W. University
El Paso, Texas 79968, USA, e-mail: dlgallegose@miners.utep.edu

Olga Kosheleva
Department of Teacher Education, University of Texas at El Paso, 500 W. University
El Paso, Texas 79968, USA, e-mail: olgak@utep.edu

Vladik Kreinovich
Department of Computer Science, University of Texas at El Paso, 500 W. University
El Paso, Texas 79968, USA, e-mail: vladik@utep.edu

$$f(x) = \frac{1}{\sqrt{2\pi}\cdot\sigma}\cdot\exp\left(-\frac{(x-m)^2}{2\sigma^2}\right),$$

where m is the distribution's mean and σ is its standard deviation.

In line with the Central Limit Theorem, many real-life distributions are well-described by normal distributions.

Comments.

- The formula for the normal distribution can be described in the following equivalent form:

  $$f(x) = f_0\left(\frac{x-m}{\sigma}\right),$$

  where

  $$f_0(x) = \frac{1}{\sqrt{2\pi}\cdot}\cdot\exp\left(-\frac{x^2}{2}\right)$$

  is the probability density function of the *basic* normal distribution, with mean 0 and standard deviation 1.
- A normal distribution is uniquely determined by its first two moments: the mean value m and the standard deviation σ. So, when the empirical distribution is normal, we can easily find the exact shape of this distribution:
  - first, based on the sample, we estimate the mean m and the standard deviation σ, and
  - then, we use the normal distribution with these values of the mean and the standard deviation.

  The resulting distribution is symmetric relative to the mean m – and, as a result, its third central moment is equal to 0.

Need to go beyond normal distributions. Although normal distributions are ubiquitous, not all empirical distributions are normal. Many empirical distributions are *skewed* (asymmetric), i.e., have non-zero third central moment.

It is therefore desirable to have a few-parametric generalization of the class of normal distributions that would allow to consider skewness.

Skew-normal distributions: a brief reminder. Many skew-related generalizations of the class of normal distributions are possible, and many such families have been proposed.

Empirically, one of such families – called *skew-normal* (see, e.g., [1]) – has been most successful. The basic skew-normal distribution has the form

$$s_0(x) = f_0(x)\cdot F_0(\alpha\cdot x),$$

where $F_0(x)$ is the cumulative distribution function corresponding to the basic normal distribution, and α is some real number. The probability density function $s(x)$ of a general skew-normal distribution can be obtained from the pdf of the basic one:

$$s(x) = s_0 \left(\frac{x-m}{\sigma} \right)$$

for some m and σ.

In particular, skew-normal distributions are very effective in econometrics; see, e.g., [3, 5, 7] and references therein.

Resulting challenge. How can we explain why this particular generalization of the family of normal distributions turned out to be the most successful?

What we do in this paper. In this paper, we provide a possible explanation for the success of skew-normal distribution.

It turns out that this explanation is similar to one of the possible explanations of another empirical success – the success of rectified linear activation functions in deep learning.

2 Why skew-normal: an explanation

We need a distribution that can be used in simulations. One of the main objectives of determining the actual distribution of the analyzed phenomena is that we will be able to simulate this phenomenon – by appropriate simulating this distribution.

From this viewpoint, it is necessary to recall how different probability distributions are simulated.

Two usual ways to simulate a distribution. Most computers have built-in random number generators – hardware or software implemented – that generate either the uniform distribution on the interval $[0,1]$ or the basic normal distribution, with mean 0 and standard deviation 1. To generate different distributions, we usually use these random number generators.

There are two main ways to do it:

- we can either apply some easy-to-compute function to the results of a random number generator,
- or we compute an easy-to-compute (e.g., linear) combination of several such results.

Let us briefly recall both approaches.

First way to simulate a distribution. Let us start with the technique of applying an easy-to-compute function to the results of the standard number generator. This is, for example, how computers simulate a generic normal distribution, with mean m and standard deviation σ:

- they use the standard random number generator to generate a random number X distributed according to the basic normal distribution, and
- then they compute $Y = m + \sigma \cdot X$.

One can easily check that the resulting random variable Y indeed has the desired distribution.

The use of this technique for simulating distributions other than normal comes from the easy-to-prove fact that all continuous distributions on a finite or infinite interval – i.e., all distributions with continuous probability density functions – can be obtained from each other by a continuous re-scaling of the interval. In precise terms:

- if X is a random variable that is distributed according to the first distribution,
- then, for an appropriate increasing function $g(x)$, the value $Y = g(X)$ is distributed according to the second distribution.

The desired function $g(x)$ is easy to find. Indeed:

- Let us denote the cumulative distribution function of the first distribution by

$$F_X(x) \stackrel{\text{def}}{=} \text{Prob}_X(X \le x).$$

- Let us denote the cumulative distribution of the second distribution by

$$F_Y(y) \stackrel{\text{def}}{=} \text{Prob}_Y(Y \le y).$$

Then, for any strictly increasing function $g(x)$, we have $X \le x \leftrightarrow g(X) \le g(x)$. Thus, the cumulative distribution function for $g(X)$ has the property

$$F_{g(X)}(g(x)) = \text{Prob}(g(X) \le g(x)) = \text{Prob}_X(X \le x) = F_X(x).$$

So, for any y, if we, as usual, denote the x for which $g(x) = y$ by $g^{-1}(y)$, then we get $F_{g(X)}(y) = F_X(g^{-1}(y))$. Thus, to get the desired distribution for $g(X)$, we need to make sure that we always have $F_Y(y) = F_X(g^{-1}(y))$. In other words, for $x = g^{-1}(y)$, for which $y = g(x)$, we need to have $F_Y(g(x)) = F_X(x)$. Thus, if we apply the inverse function F_Y^{-1} to both sides of this equality, we get the explicit formula for the desired re-scaling function:

$$g(x) = F_Y^{-1}(F_X(x)).$$

Second way to simulate a distribution. When all we have is a random number generator for the uniform distribution, then one of the possible ways to simulate a normal distribution is to use the above-mentioned Central Limit Theorem, according to which the distribution of the sum of a large number of independent random variables is close to normal. Thus, we can form the sum of several uniformly distributed random variables – and we will get a good simulation of a normal distribution.

For independent normal random variables, their sum – and, more generally, their linear combination is also normally distributed.

Main idea behind our explanation. We need a non-normal distribution. Since normal distributions are ubiquitous, all computer systems already have a normal distribution. So a natural idea is to apply some function to the normal random variable.

As we have mentioned, this does not restrict the class of distributions, since all continuous distributions can be obtained from each other by applying some function – this is one of the usual ways to simulate different distributions.

Applying a linear function will still keep the resulting distribution normal. So, to simulate a probability distribution that is different from normal, we need to us apply a nonlinear function.

From the practical viewpoint, the faster-to-compute this nonlinear function, the better. Thus, we need ot use the fastest-to-compute nonlinear functions.

So what are the fastest-to-compute nonlinear functions?

What are the fastest-to-compute nonlinear functions? To answer this question, let us recall which computer operations are the fastesr. In a computer, the fastest possible operations are:

- unary minus – it changes just one bit, the sign bit, and
- min and max – that require, on average, 2 bit operations.

Indeed:

- In 1/2 of the cases, the two numbers have different first bits, i.e., 0 and 1. So, by applying only one bit operation – namely, by comparing the first bits of the two numbers – we already know which number is larger.
- In 1/4 of the cases, the first bits are equal, but the second bits are different. In this case, we need 2 bit comparisons to decide which number is larger.
- In general, in $1/2^k$ cases, the first $k-1$ bits are equal, but the k-th bits are different. In this case, we need k bit comparisons to decide which number is larger.

So, the average number b of bit operations needed to compute min or max is equal to the following:

$$b = 1 \cdot \frac{1}{2} + 2 \cdot \frac{1}{2^2} + \ldots + (k-1) \cdot \frac{1}{2^{k-1}} + k \cdot \frac{1}{2^k} + (k+1) \cdot \frac{1}{2^{k+1}} + \ldots \qquad (1)$$

To compute the value b, let us divide both sides of the formula (1) it by 2. Then we get:

$$\frac{b}{2} = \frac{1}{2^2} + 2 \cdot \frac{1}{2^3} + \ldots + (k-1) \cdot \frac{1}{2^k} + k \cdot \frac{1}{2^{k+1}} + (k+1) \cdot \frac{1}{2^{k+2}} + \ldots \qquad (2)$$

If we subtract (2) from (1) term by term, subtracting the terms proportional to the same factor $1/2^k$, we will get

$$\frac{b}{2} = \frac{1}{2} + \frac{1}{2^2} + \ldots + \frac{1}{2^k} + \frac{1}{2^{k+1}} + \ldots \qquad (3)$$

The sum in the right-hand side is a geometric progression, so, in principle, we can use the known formula for its sum. But for simplicity, instead of using this general formula, let us use the same trick that we have just used. First, we divide both sides of the equality (3) by 2, resulting in the following:

$$\frac{b}{4} = \frac{1}{2^2} + \frac{1}{2^3} + \ldots + \frac{1}{2^k} + \ldots \tag{4}$$

If we subtract (4) from (3), we conclude that

$$\frac{b}{2} - \frac{b}{4} = \frac{b}{4} = \frac{1}{2}. \tag{5}$$

Thus, indeed

$$b = 4 \cdot \frac{1}{2} = 2. \tag{6}$$

In addition to the above two operations, we can also have constants like 0 that are also easy to generate.

Next in complexity are addition and subtraction. They require as many bit operations as there are bits – i.e., 64 on most computers. Multiplication and division require even more bit operations.

So, what are the fastest-to-compute nonlinear functions? The fastest functions are the ones that only use the fastest computer operations:

- unary minus that computes $-x$ and
- minimum and/or maximum.

The function $x \mapsto -x$ is linear. So, the only way to get a nonlinear functions is to apply minimum or maximum to x and $-x$. This leads either to $|x| = \max(x, -x)$ or to $-|x| = \min(x, -x)$.

Comment. It would be even faster to take $\max(0, x)$ or $\min(0, x)$ – this would require only one fastest computer operations – but this would not lead to a continuous random variable :-(

This explains the efficiency of skew normal distributions. Interestingly, if we apply each of these two functions: $|x|$ and $-|x|$, to the basic normal distribution, we get two particular cases of skew normal distributions.

And if we use the second way of simulating random distributions and consider all possible linear combinations of $|x|$ (for a normal random variable x) and independent normal random variables, then we get exactly all skew-normal distributions.

This explains the empirical success of this family of distributions.

3 How is this related to ReLU?

What we do in this section. In this section, following [6], we will show that similar ideas explain the empirical success of Rectified Linear (ReLU) activation functions in machine learning (see, e.g., [2]). Later in this section, we will explain what is an activation function in general and what is ReLU activation function.

What are neural networks: a brief reminder. In a neural network, we have several units (called neurons) that perform some transformations: they take several values

$v_1, \ldots, v_n$ and return the value

$$v = s(w_1 \cdot v_1 + \ldots + w_n \cdot v_n - w_0),$$

where:

- the values w_i are numerical constants, and
- the function $s(x)$ is a function known as *activation function*.

Some neurons process inputs to the algorithm, others process results of the previous neurons, etc. The output of one of the neurons is then returned as the computation result.

Need for non-linear neurons. Some neurons have a linear activation function. For such neurons, the dependence of v on v_i is linear.

However, If all neurons were linear, then we would only get linear functions, and many real-life dependencies are nonlinear. Thus, to be able to describe real-life dependencies, we need to add neurons with nonlinear activation functions.

Which nonlinear activation functions should we choose?

Which nonlinear activation functions should we choose? In many practical situations, we cannot compute the desired quantity since computations take too long. From this viewpoint, it is desirable to perform computations as fast as possible.

In relation to neurons, this means that we need to select the fastest-to-compute activations functions.

This explains ReLU activation function. Similar to the above analysis of rescaling functions $g(x)$, we can conclude that the fastest-to-compute are the activation functions that consist of the fastest computer operations – i.e., changing the sign, minimum, and maximum. Simuilar to the above case, we conclude that we should get one of the following functions:

$$|x| = \max(x, -x), \ \ -|x| = \min(x, -x), \ \ \max(0, x), \ \ \min(0, x),$$

$$-\max(0, x), \ \ -\min(0, x), \ \ \max(0, -x), \ \ \min(0, -x),$$

$$-\max(0, -x), \text{ and } -\min(0, -x).$$

In this case – in contrast to the case analyzed in the previous sections – all these options are possible. Of these options, the fastest are the ones that use only one fastest computer operations, i.e.,

$$\max(0, x) \text{ and } \min(0, x).$$

From the viewpoint of the future linear transformations in a neural network, the activation function $\min(0, x)$ is equivalent to $\max(0, x)$; indeed:

- we have since $\min(0, x) = -\max(0, -x)$ and,
- similarly, $\max(0, x) = -\min(0, -x)$.

Since, from the neural network viewpoint, these two functions are equivalent, we can therefore conclude that the fastest-to-compute activation function is

$$s(x) = \max(0, x).$$

This is exactly the rectified linear (ReLU) activation functions that is so effective in machine learning.

Acknowledgments

This work was supported in part by the National Science Foundation grants 1623190 (A Model of Change for Preparing a New Generation for Professional Practice in Computer Science), HRD-1834620 and HRD-2034030 (CAHSI Includes), EAR-2225395 (Center for Collective Impact in Earthquake Science C-CIES), and by the AT&T Fellowship in Information Technology. It was also supported by a grant from the Hungarian National Research, Development and Innovation Office (NRDI), and by the Institute for Risk and Reliability, Leibniz Universitaet Hannover, Germany.

The authors are greatly thankful to all the participants of the Workshop on Mathematics, Computer Science, and Computational Science organized by New Mexico State University and University of Texas at El Paso (Las Cruces, New Mexico, April 12, 2025), especially to Tonghui "Tony" Wang, for valuable discussions.

References

1. A. Azzalini, *The Skew-Normal and Related Families*, Cambridge University Press, Cambridge, UK, 2013.
2. I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, Cambridge, Massachusetts, 2016.
3. Z. Ma, T. Wang, S. T. B. Choy, Z. Wei, and X. Zhu, "Extending the A Priori Procedure for estimating location parameter under multivariate skew normal settings", In: N. N. Thach, V. Kreinovich, D. T. Ha, and N. D. Trung (eds.), *Optimal Transport Statistics for Economics and Related Topics*, Springer, Cham, Switzerland, 2024, pp. 107–116.
4. D. J. Sheskin, *Handbook of Parametric and Nonparametric Statistical Procedures*, Chapman and Hall/CRC, Boca Raton, Florida, 2011.
5. T. Tong, X. Chen, T. Wang, D. Trafimow, S. T. B. Choy, and C. Wang, "The A Priori Procedure (APP) for estimating standardized regression coefficients in multiple linear model with skew errors", In: V. Kreinovich, S. Sriboonchitta, and W. Yamaka (eds.), *Machine Learning for Econometrics and Related Topics*, Springer, Cham, Switzerland, 2024, pp. 143–160.
6. J. C. Urenda and V. Kreinovich, "Why Rectified Linear activation functions? Why max-pooling? a possible explanation", In: O. Castillo and P. Melin (eds.), *New Perspectives on Hybrid Intelligent System Design based on Fuzzy Logic, Neural Networks and Metaheuristics*, Springer, 2023, pp. 459–463.
7. Z. Wang, T. Wang, D. Trafimow, S. T. B. Choy, and X. Chen, "Extending a priori procedure for simultaneously estimating location and scale parameters in the context of skew normal distributions", In: N. N. Thach, N. D. Trung, D. T. Ha, and V. Kreinovich (eds.), *Artificial Intelligence*

and Machine Learning for Econometrics: Applications and Regulation (and Related Topics), Springer, Cham, Switzerland, to appear.