FEDERAL UNIVERSITY OF CEARÁ

CENTRE OF SCIENCE

STATISTICS AND APPLIED MATHEMATICS DEPARTMENT

GRADUATE PROGRAM IN MODELING AND QUANTITATIVE METHODS

MODELING AND QUANTITATIVE METHODS GRADUATE DEGREE

FÁBIO HEMERSON ARAÚJO DE SOUZA

MACHINE LEARNING ALGORITHMS TO SOLVE STATISTICAL PROBLEMS

FORTALEZA

2020

FÁBIO HEMERSON ARAÚJO DE SOUZA

MACHINE LEARNING ALGORITHMS TO SOLVE STATISTICAL PROBLEMS

Master thesis presented to Graduate Program in Modeling and Quantitative Methods in Centre of Science of Federal University of Ceará, as a partial requirement for obtaining a master's degree in Modeling and Quantitative Methods. Emphasis field: Modeling and Quantitative Methods

Advisor: Prof. Dr. Luis Gustavo Bastos Pinho
Co-advisor: Profa. Dra. Silvia Maria de Freitas

FORTALEZA
2020

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

S238m Souza, Fábio Hemerson Araújo de.
Machine learning algorithms to solve statistical problems / Fábio Hemerson Araújo de Souza. – 2020.
45 f. : il. color.

Dissertação (mestrado) – Universidade Federal do Ceará, Centro de Ciências, Programa de Pós-Graduação em Modelagem e Métodos Quantitativos, Fortaleza, 2020.
Orientação: Prof. Dr. Luis Gustavo Bastos Pinho.
Coorientação: Profa. Dra. Sílvia Maria de Freitas.

1. Artificial Intelligence. 2. Learning. 3. Algorithms. 4. PIPE. 5. Network. I. Título.
CDD 510

FÁBIO HEMERSON ARAÚJO DE SOUZA

MACHINE LEARNING ALGORITHMS TO SOLVE STATISTICAL PROBLEMS

Master thesis presented to Graduate Program in Modeling and Quantitative Methods in Centre of Science of Federal University of Ceará, as a partial requirement for obtaining a master's degree in Modeling and Quantitative Methods. Emphasis field: Modeling and Quantitative Methods

Approved on: 20/02/2020.

COMMITTEE MEMBERS

to my beloved parents Inácio and Ecila

and

my future wife Rebeca.

ACKNOWLEDGMENTS

My parents for the immeasurable understanding about my all my choices.

Rebeca Dieb Holanda Silva, for her comprehension at all times that I was absent and for unconditional support she always gave me in all my choices.

My advisor Prof.Dr.Luis Gustavo Bastos Pinho.

Especial professors like João Maurício Araújo Mota, Gualberto Segundo Agamez Montalvo, Juvêncio Santos Nobre, Maria Jacqueline Batista, Silvia Maria de Freitas, Maria do Carmo Soares Lima.

My friends Alexandre Magno Cavalcante Sucupira, Alexandre Farias Monte Monteiro, Alex Newman Veloso dos Santos, Anny Suellen Gomes da Silva, Andreia El Haber Limão, André Erasmo de Almeida, Elícius Feijó Cordeiro, Gabriel Garcez Barros Sousa, José Roberval Cândido Júnior, Rodolfo Jordan Domingos Quintela, Thaina Soares Silva e Victor Diego de Almeida.

“I believe observation and knowledge must precede action.” (STRANGE, Stephen Vicent).

RESUMO

Antes de sua popularidade o campo da inteligência artificial começou a se espalhar pelo mundo como uma ferramenta computacional de um futuro distante. Em 1950, as Redes Neurais Artificiais (RNA) começaram a ser desenvolvidas e todos os algoritmos de inteligência computacional seguiram o mesmo caminho, fazendo com que esse futuro ficasse cada vez mais próximo. Atualmente, o aprendizado de máquina, um ramo da IA, é usado em muitos campos e processos diferentes, como marketing e vendas, inteligência de negócios, pesquisa e desenvolvimento, cadeia de suprimentos, estoques financeiros, recursos humanos, saúde, etc. O amadurecimento da área de IA trás consigo uma base probabilística que pode ser usada para resolver alguns problemas em estatística. Neste trabalho fazemos uso de técnicas e algoritmos de aprendizado de máquina para resolver dois problemas estatísticos propostos. No capítulo 1, o problema é encontrar uma aproximação para a função de distribuição acumulada para distribuição Normal. Esta expressão precisa ser matemática e computacionalmente mais simples do que outras aproximações encontradas em palestras e artigos de estatística e um possível uso dessa expressão é em sala de aula em disciplinas de introdução à estatística. No capítulo 2 abordamos um problema de distribuição de identificabilidade usando algoritmo de aprendizado de máquina e uma estrutura para computação matemática chamada textit Tensorflow e uma biblioteca de abstração para rotinas de aprendizagem profunda chamada textit Keras, ambas escritas em Python. O objetivo principal aqui é construir uma estrutura que possa ser capaz de capturar características de uma amostra fornecida pelo usuário e classificar a distribuição original dessa amostra. Os resultados foram promissores com uma precisão superior a 95 % para cada distribuição usada nos exemplos.

Palavras-chave: Inteligência artificial. Aprendizado do computador. Algoritmos. PIPE. Redes neurais.

ABSTRACT

The field of artificial intelligence before its popularity, begin to be spread around the world as a computational tools from a distant future. In 1950, Artificial Neural Network (ANN) begin to be developed and all compute intelligence algorithms follow the same way , making that future be more closer now a days. Currently, machine learning, an AI branch, is used in many different fields and processes such as marketing and sales, business intelligence, research and development, supply chain, financial stocks, human resources, healthcare, etc. The maturation of AI field carries itself a probabilistic bases that can be used to solve some problems in statistics. In this work we make use of machine learning techniques and algorithms to solve two proposed statistical problems. In chapter 1, the issue is to find an approximation to normal cumulative distribution function. This expression needs to be mathematically and computationally simpler than other approximations founded in statistics lectures and papers and one possible use of this expression is in introductory statistics classrooms. Chapter 2 we address an identifiability distribution problem using machine learning algorithm and a framework for mathematical computation called *Tensorflow* and an abstraction library for deep learning routines called *Keras*, both of them written in Python. The main goal here is construct a structure that can be able to capture features from a sample provided by the user and classify the parent distribution of this sample. The results were promising with a accuracy greater then 95% for each distribution used for examples.

Keywords: Artificial inteligence. Machine learning. Algorithms. PIPE. Neural Network.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

AI	Artificial Intelligence
AIC	Akaike Information Criterion
ANN	Artificial Neural Network
BIC	Bayesian Information Criterion
CDF	Cumulative Distribution Function
DNA	DeoxyiboNucleic Acid
FDL	Fitness Dependent Learning
GMM	Generalized Moment Methods
GMME	Generalized Moment Methods Estimator
KL	Kullback-Liebler
MAE	Mean Absolute Error
MaxAE	Max Absolute Error
MLE	Maximum Likelihood Estimation
MLP	Multilayer Perceptron
MM	Moment Methods
MME	Moment Methods Estimator
MRI	Magnetic Ressonance Image
MSE	Mean Squared Error
NN	Neural Network
PDF	Probability Density Function
PIPE	Probabilistic Incremental Program Evolution
PMF	Probability Mass Function
PPT	Probabilistic Prototype Tree
SEV	Small Extracellular Vesicles
TEM	Transmition Electron Microscopy

SUMMARY

1 INTRODUCTION

The first part of this paper works on usage of normal distribution density function is at the core of many first semester courses on Introduction to Statistics for many majors in Engineering and Sciences. It also appears in other subjects as solution to certain equations that model certain phenomena. Usually, the probabilities under this model are presented in a probability table that the students have to consult during exercises and exams. Although this method works well, there is, in our view, a more efficient way to teach how to obtain probabilities under the standard normal distribution without relying on personal computers.

We present an approximate expression for the cumulative distribution function of the standard normal distribution. The purpose of our approximate expression is to be used in classroom as an alternative to the typical normal distribution probability tables both in introductory courses in Statistics and other subjects. The motivation for this is twofold. First, the probability table is one extra sheet of paper that students have to print and carry with them for months. Usually they get lost and students get caught in an exam or class without a table. Second, the normal distribution is widely used in so many different fields and by so many different professionals that it is worthwhile to have a small formula that is easy to use and to remember. It can be used with a simple hand held calculator for quick assessments.

It is not rare to find graduate students from several fields relying on the normal distribution table for their research when a computer is available. This suggests that the normal distribution table may be seen by many students and professionals as a first option when using the normal probability model even though they are not in a classroom environment. We believe that having an approximate expression will reinforce to the students that the normal distribution cumulative probabilities come from calculus, they may not be easily obtained by hand but can easily be calculated by a computer.

Our approximation was obtained by using an algorithm called Probabilistic Incremental Program Evolution (PIPE) proposed by Salustowicz and Schmidhuber (1997). This algorithm belongs to the class of Estimation of Distribution algorithms. Given a task and an optimal criterion, PIPE searches an space of possible solutions to the task by following a set of probability rules. These probability rules are updated over iterations of the algorithm such that better solutions will have a larger likelihood of being visited.

The second part focus on one major concern in practical uses of statistical models capacity of choosing a plausible and adequate distribution to the data at hand. Ideally, the model is created from the underlying knowledge of the problem. After designing the model and fitting it to the data, some diagnostic methods are used to evaluate how well the model fits the data.

One of the simplest methods for univariate problems is using the likelihood function of the model, defined as the product of the probability density function evaluated in all points of the samples seen as a function of the parameters, that is

$$\ell(\boldsymbol{\theta}; x_1, x_2, \ldots, x_n) = \prod_{i=1}^{n} f(x_i; \boldsymbol{\theta}), \tag{1}$$

where $\boldsymbol{\theta}$ is a parameter vector and $x_1, x_2, \ldots, x_n$ are observed sample points. This measures how likely are samples from a random variable with probability density function $f(x; \boldsymbol{\theta})$ to be similar to the observed sample.

Other quantities, such as the AIC (Akaike Information Criterion) and BIC (Bayesian Information Criterion), are penalized versions of the likelihood, where the penalty is due to the number of parameters. A lower value of AIC and BIC is desirable. Under mild conditions the BIC provides consistent choices given that the true model for the data is among the models compared. Another popular measure is the Kolmogorov-Smirnoff distance, defined as the largest absolute deviation between the theoretical and the empirical cumulative distribution function.

Marshall et al. (2001) investigate weather or not the true model that generates data is preferred by the likelihood and Kolmogorov-Smirnoff distance. A Monte Carlo simulation is used to access the probability of correct choice. This is done for different sample sizes and parameter values. The general conclusion of Marshall et al. (2001) is that the data will, in fact, recognize its parent distribution, but that depends on the sample size and values of the parameters.

For some sample sizes and parameters configurations the chance of the correct model being the one pointed out by the selection methods is very low. In many situations involving the gamma and Weibull distributions there is about 50% chance that the methods will choose a wrong distribution. That raises important questions on the reliability of these methods specially for small to moderate sample sizes. As the authors point out, the methods work nicely for large sample sizes but these are not the absolute prevalent situation.

Here we investigate the use of neural networks to identify the parent distribution of the data from sample moments. We use an increasingly popular framework, at the time of writing, for machine learning named Keras. We show empirical evidence that suggests there are non linear patterns in the sample moments that may be exploited to identify the parent distribution with a much higher accuracy then compared to the likelihood based methods from Marshall et al.

(2001). The results have two main implications. First, it suggests the use of deep neural networks to model selection tasks in the univariate context. There is, however, the need for further studies to understand or formulate theoretical guarantees before using deep neural networks for this end in practical and possibly sensible applications. As a secondary implication, the results suggest that there still is room for new selection methods that are based solely on the sample moments. One possible direction to such end is the use of maximum entropy characterizations (JAYNES, 1957), which usually involve some constraints on the moments of the distributions.

2 NORMAL CUMULATIVE DISTRIBUTION FUNCTION APPROXIMATION

In this chapter, we present an approximate expression for the cumulative distribution function of the standard normal distribution that is both easy to remember and easy to use. This approximation was obtained by an estimation of distribution algorithm.

2.1 Existing approximations for the normal distribution cumulative distribution function

Let Z be a random variable with standard normal distribution and cumulative distribution function given by

$$P(Z \leq z) = \Phi(z) = \int_{-\infty}^{z} \frac{1}{\sqrt{2\pi}} e^{-\frac{z^2}{2}} dx. \tag{2}$$

Pólya's approximation (PÓLYA, 1945) is one of the most cited approximations for the normal cumulative distribution function with the purpose of providing convenient calculations. It is given by

$$\widehat{\Phi}(z) = \frac{1}{2}\left\{1 + \left[1 - \exp\left(\frac{-2z^2}{\pi}\right)\right]^{\frac{1}{2}}\right\} \tag{3}$$

The result from Pólya's work involves elegant series expansions and is related to other works such as those developed by Neyman and Pearson on hypothesis testing. It is actually an upper bound for $\Phi(z)$. The maximum absolute error for the approximation over the range (-5,5) is 0.003.

Cadwell (1951) improves this inequality bound by a geometric manipulation of an integral used by Pólya. His approximation is given by

$$\frac{1}{2}\left\{1 + \left[1 - \exp\left(\frac{-2z^2}{\pi} - \frac{2(\pi-3)z^4}{3\pi^2}\right)\right]^{\frac{1}{2}}\right\}. \tag{4}$$

The maximum absolute error for Cadwell's approximation is 0.006 for $z \in (-5,5)$.

Another approximation is seen in Hoyt (1968), which is a very short paper presenting a connection between sums of independent and identically distributed uniform random variables and the normal distribution by the central limit theorem. It is intended to be an exercise in an undergraduate probability class and not a viable, practical and accurate approximation such as the other ones presented here, but it is a very interesting one nonetheless. Also, it relies on a different argument compared to the others. Hoyt (1968) observes that if $X_1, X_2, \cdots, X_n$ are independent and identically distributed uniformly in the interval $(-1,1)$, then, by the central limit theorem,

$$T_n = (S_n - 0.5n)\sqrt{\frac{12}{n}}, \tag{5}$$

$S_n = X_1 + X_2 + \cdots + X_n$, converges in distribution to a $N(0,1)$ random variable. For $n = 3$, T_n ranges from -3 and 3, which is the most used interval for the normal distribution. Also, the probability density function of T_3 has inflection points at -1 and 1, as does the normal distribution probability density function. For these similarities, the probability density function of T_3 can be used as to obtain an approximation for $\Phi(z)$. The exact probability density function of T_3 is

$$g(t) = \begin{cases} \dfrac{(3-t)^2}{8} & \text{if } |t| < 1, \\ \dfrac{(3-|t|)^2}{16}, & \text{if } 1 < |t| < 3, \\ 0 & \text{elsewhere.} \end{cases} \tag{6}$$

From it, simple integration provides an approximation for $\Phi(z)$.

Lin (1989) gives an approximation in the form $\widehat{\Phi}(z) = 1 - \frac{1}{2}\exp\{az + bz^2\}$ by fitting $az + bz^2$ to $\log\Phi(z)$ using least squares. He finds $a = -0.416$ and $b = -0.717$. This is remarkably simple as well and the relative error for the most used values is less than 1%.

The same author proposes another approximation using a logistic function. In Lin (1990) it is presented

$$\widehat{\Phi}(z) = \left[1 + \exp\left(\frac{4.2\pi z}{9 - z}\right)\right]^{-1}. \tag{7}$$

This formula has errors as small as the previous one except at some tail values. For $\widehat{\Phi}(0.99975)$ and $\widehat{\Phi}(0.99999)$ the author reports relative errors of 3% and 8%, respectively.

Bowling et al. (2009) propose yet another approximation based on logistic functions. Their approximation is given by

$$\widehat{\Phi}(z) = 1 - \left[1 + \exp\left(-0.07056z^3 + 1.5976z\right)\right]^{-1}. \tag{8}$$

Dombi and Jonas (2018), based on continuous-valued logic operators, obtain

$$\widehat{\Phi}(z) = \left[1 + \exp\left(-2\sqrt{\frac{2}{\pi}}z\right)\right]^{-1}, \tag{9}$$

which has a maximum absolute error of 0.0177. This result appears also in Tocher (1963).

Next, we describe our approximation.

2.2 Proposed approximation

We claim that for classroom and non-critical situations where a computer is not available the following approximation can be used instead of the normal probability table:

$$\widetilde{\Phi}(z) = \begin{cases} \dfrac{1}{0.19^z + 1}, & 0 < z \leq 1 \\ 0.63^{\exp(-z^{1.6})}, & z > 1 \end{cases} \tag{10}$$

For the negative arguments of Φ we use the symmetry of the function as it is done with the probability table. Table 1 shows a typical normal distribution table presented in most introductory texts in Statistics. Table 2 shows the same table filled with our approximation. Lastly, Table 3 shows the absolute errors of our approximation calculated as $|\widetilde{\Phi}(z) - \Phi(z)|$.

Table 1 – Values for normal cumulative distribution function

	0.00	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09
0.0	0.5000	0.5040	0.5080	0.5120	0.5160	0.5199	0.5239	0.5279	0.5319	0.5359
0.1	0.5398	0.5438	0.5478	0.5517	0.5557	0.5596	0.5636	0.5675	0.5714	0.5753
0.2	0.5793	0.5832	0.5871	0.5910	0.5948	0.5987	0.6026	0.6064	0.6103	0.6141
0.3	0.6179	0.6217	0.6255	0.6293	0.6331	0.6368	0.6406	0.6443	0.6480	0.6517
0.4	0.6554	0.6591	0.6628	0.6664	0.6700	0.6736	0.6772	0.6808	0.6844	0.6879
0.5	0.6915	0.6950	0.6985	0.7019	0.7054	0.7088	0.7123	0.7157	0.7190	0.7224
0.6	0.7257	0.7291	0.7324	0.7357	0.7389	0.7422	0.7454	0.7486	0.7517	0.7549
0.7	0.7580	0.7611	0.7642	0.7673	0.7704	0.7734	0.7764	0.7794	0.7823	0.7852
0.8	0.7881	0.7910	0.7939	0.7967	0.7995	0.8023	0.8051	0.8078	0.8106	0.8133
0.9	0.8159	0.8186	0.8212	0.8238	0.8264	0.8289	0.8315	0.8340	0.8365	0.8389
1.0	0.8413	0.8438	0.8461	0.8485	0.8508	0.8531	0.8554	0.8577	0.8599	0.8621
1.1	0.8643	0.8665	0.8686	0.8708	0.8729	0.8749	0.8770	0.8790	0.8810	0.8830
1.2	0.8849	0.8869	0.8888	0.8907	0.8925	0.8944	0.8962	0.8980	0.8997	0.9015
1.3	0.9032	0.9049	0.9066	0.9082	0.9099	0.9115	0.9131	0.9147	0.9162	0.9177
1.4	0.9192	0.9207	0.9222	0.9236	0.9251	0.9265	0.9279	0.9292	0.9306	0.9319
1.5	0.9332	0.9345	0.9357	0.9370	0.9382	0.9394	0.9406	0.9418	0.9429	0.9441
1.6	0.9452	0.9463	0.9474	0.9484	0.9495	0.9505	0.9515	0.9525	0.9535	0.9545
1.7	0.9554	0.9564	0.9573	0.9582	0.9591	0.9599	0.9608	0.9616	0.9625	0.9633
1.8	0.9641	0.9649	0.9656	0.9664	0.9671	0.9678	0.9686	0.9693	0.9699	0.9706
1.9	0.9713	0.9719	0.9726	0.9732	0.9738	0.9744	0.9750	0.9756	0.9761	0.9767
2.0	0.9772	0.9778	0.9783	0.9788	0.9793	0.9798	0.9803	0.9808	0.9812	0.9817
2.1	0.9821	0.9826	0.9830	0.9834	0.9838	0.9842	0.9846	0.9850	0.9854	0.9857
2.2	0.9861	0.9864	0.9868	0.9871	0.9875	0.9878	0.9881	0.9884	0.9887	0.9890
2.3	0.9893	0.9896	0.9898	0.9901	0.9904	0.9906	0.9909	0.9911	0.9913	0.9916
2.4	0.9918	0.9920	0.9922	0.9925	0.9927	0.9929	0.9931	0.9932	0.9934	0.9936
2.5	0.9938	0.9940	0.9941	0.9943	0.9945	0.9946	0.9948	0.9949	0.9951	0.9952
2.6	0.9953	0.9955	0.9956	0.9957	0.9959	0.9960	0.9961	0.9962	0.9963	0.9964
2.7	0.9965	0.9966	0.9967	0.9968	0.9969	0.9970	0.9971	0.9972	0.9973	0.9974
2.8	0.9974	0.9975	0.9976	0.9977	0.9977	0.9978	0.9979	0.9979	0.9980	0.9981
2.9	0.9981	0.9982	0.9982	0.9983	0.9984	0.9984	0.9985	0.9985	0.9986	0.9986
3.0	0.9987	0.9987	0.9987	0.9988	0.9988	0.9989	0.9989	0.9989	0.9990	0.9990

Source: Author

Table 2 – Approximate values using PIPE approximation

	0.00	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09
0.0	0.5000	0.5042	0.5083	0.5125	0.5166	0.5207	0.5249	0.5290	0.5332	0.5373
0.1	0.5414	0.5455	0.5497	0.5538	0.5579	0.5620	0.5660	0.5701	0.5742	0.5782
0.2	0.5823	0.5863	0.5903	0.5943	0.5983	0.6023	0.6063	0.6103	0.6142	0.6181
0.3	0.6220	0.6259	0.6298	0.6337	0.6375	0.6414	0.6452	0.6490	0.6527	0.6565
0.4	0.6602	0.6639	0.6676	0.6713	0.6750	0.6786	0.6822	0.6858	0.6894	0.6929
0.5	0.6964	0.6999	0.7034	0.7069	0.7103	0.7137	0.7171	0.7204	0.7238	0.7271
0.6	0.7304	0.7336	0.7368	0.7401	0.7432	0.7464	0.7495	0.7526	0.7557	0.7588
0.7	0.7618	0.7648	0.7678	0.7707	0.7736	0.7765	0.7794	0.7822	0.7851	0.7878
0.8	0.7906	0.7933	0.7961	0.7987	0.8014	0.8040	0.8066	0.8092	0.8118	0.8143
0.9	0.8168	0.8192	0.8217	0.8241	0.8265	0.8289	0.8312	0.8335	0.8358	0.8381
1.0	0.8403	0.8426	0.8447	0.8469	0.8490	0.8512	0.8533	0.8553	0.8574	0.8594
1.1	0.8614	0.8634	0.8653	0.8672	0.8691	0.8710	0.8729	0.8747	0.8765	0.8783
1.2	0.8800	0.8818	0.8835	0.8852	0.8869	0.8885	0.8902	0.8918	0.8934	0.8950
1.3	0.8965	0.8980	0.8995	0.9010	0.9025	0.9040	0.9054	0.9068	0.9082	0.9096
1.4	0.9109	0.9123	0.9136	0.9149	0.9162	0.9174	0.9187	0.9199	0.9211	0.9223
1.5	0.9235	0.9247	0.9258	0.9270	0.9281	0.9292	0.9303	0.9313	0.9324	0.9334
1.6	0.9345	0.9355	0.9365	0.9374	0.9384	0.9394	0.9403	0.9412	0.9421	0.9430
1.7	0.9439	0.9448	0.9457	0.9465	0.9473	0.9482	0.9490	0.9498	0.9506	0.9513
1.8	0.9521	0.9528	0.9536	0.9543	0.9550	0.9557	0.9564	0.9571	0.9578	0.9585
1.9	0.9591	0.9598	0.9604	0.9610	0.9616	0.9623	0.9629	0.9634	0.9640	0.9646
2.0	0.9652	0.9657	0.9663	0.9668	0.9673	0.9678	0.9684	0.9689	0.9694	0.9698
2.1	0.9703	0.9708	0.9713	0.9717	0.9722	0.9726	0.9731	0.9735	0.9739	0.9743
2.2	0.9748	0.9752	0.9756	0.9760	0.9763	0.9767	0.9771	0.9775	0.9778	0.9782
2.3	0.9785	0.9789	0.9792	0.9796	0.9799	0.9802	0.9805	0.9808	0.9812	0.9815
2.4	0.9818	0.9821	0.9823	0.9826	0.9829	0.9832	0.9835	0.9837	0.9840	0.9843
2.5	0.9845	0.9848	0.9850	0.9852	0.9855	0.9857	0.9860	0.9862	0.9864	0.9866
2.6	0.9868	0.9871	0.9873	0.9875	0.9877	0.9879	0.9881	0.9883	0.9885	0.9887
2.7	0.9888	0.9890	0.9892	0.9894	0.9895	0.9897	0.9899	0.9901	0.9902	0.9904
2.8	0.9905	0.9907	0.9908	0.9910	0.9911	0.9913	0.9914	0.9916	0.9917	0.9918
2.9	0.9920	0.9921	0.9922	0.9924	0.9925	0.9926	0.9927	0.9928	0.9930	0.9931
3.0	0.9932	0.9933	0.9934	0.9935	0.9936	0.9937	0.9938	0.9939	0.9940	0.9941

Source: Author

Table 3 – Absolute error for PIPE approximation

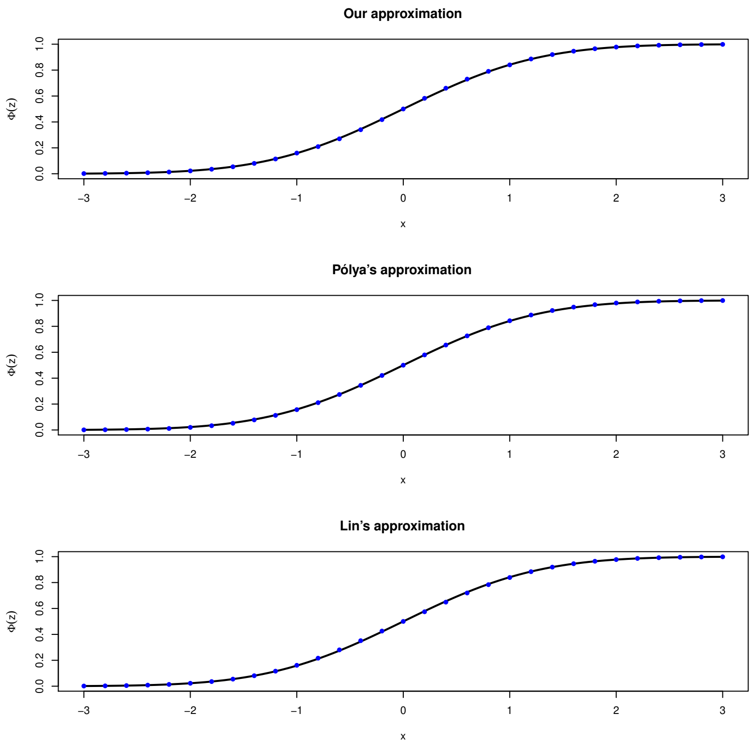
	0.00	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09
0.0	0.0000	0.0002	0.0003	0.0005	0.0006	0.0008	0.0010	0.0011	0.0013	0.0014
0.1	0.0016	0.0017	0.0019	0.0020	0.0022	0.0023	0.0025	0.0026	0.0028	0.0029
0.2	0.0030	0.0031	0.0033	0.0034	0.0035	0.0036	0.0037	0.0038	0.0039	0.0040
0.3	0.0041	0.0042	0.0043	0.0044	0.0045	0.0045	0.0046	0.0047	0.0047	0.0048
0.4	0.0048	0.0048	0.0049	0.0049	0.0049	0.0050	0.0050	0.0050	0.0050	0.0050
0.5	0.0050	0.0050	0.0049	0.0049	0.0049	0.0049	0.0048	0.0048	0.0047	0.0047
0.6	0.0046	0.0045	0.0045	0.0044	0.0043	0.0042	0.0042	0.0041	0.0040	0.0039
0.7	0.0038	0.0036	0.0035	0.0034	0.0033	0.0032	0.0030	0.0029	0.0028	0.0026
0.8	0.0025	0.0023	0.0022	0.0020	0.0018	0.0017	0.0015	0.0014	0.0012	0.0010
0.9	0.0008	0.0007	0.0005	0.0003	0.0001	0.0001	0.0003	0.0004	0.0006	0.0008
1.0	0.0010	0.0012	0.0014	0.0016	0.0018	0.0020	0.0022	0.0024	0.0026	0.0028
1.1	0.0030	0.0031	0.0033	0.0035	0.0037	0.0039	0.0041	0.0043	0.0045	0.0047
1.2	0.0049	0.0051	0.0053	0.0054	0.0056	0.0058	0.0060	0.0062	0.0063	0.0065
1.3	0.0067	0.0069	0.0070	0.0072	0.0074	0.0075	0.0077	0.0079	0.0080	0.0082
1.4	0.0083	0.0085	0.0086	0.0088	0.0089	0.0090	0.0092	0.0093	0.0094	0.0096
1.5	0.0097	0.0098	0.0099	0.0100	0.0101	0.0103	0.0104	0.0105	0.0106	0.0107
1.6	0.0107	0.0108	0.0109	0.0110	0.0111	0.0112	0.0112	0.0113	0.0114	0.0115
1.7	0.0115	0.0116	0.0116	0.0117	0.0117	0.0118	0.0118	0.0119	0.0119	0.0119
1.8	0.0120	0.0120	0.0120	0.0121	0.0121	0.0121	0.0121	0.0121	0.0121	0.0122
1.9	0.0122	0.0122	0.0122	0.0122	0.0122	0.0122	0.0121	0.0121	0.0121	0.0121
2.0	0.0121	0.0121	0.0121	0.0120	0.0120	0.0120	0.0119	0.0119	0.0119	0.0118
2.1	0.0118	0.0118	0.0117	0.0117	0.0116	0.0116	0.0115	0.0115	0.0114	0.0114
2.2	0.0113	0.0113	0.0112	0.0112	0.0111	0.0111	0.0110	0.0109	0.0109	0.0108
2.3	0.0107	0.0107	0.0106	0.0105	0.0105	0.0104	0.0103	0.0103	0.0102	0.0101
2.4	0.0100	0.0100	0.0099	0.0098	0.0097	0.0097	0.0096	0.0095	0.0094	0.0094
2.5	0.0093	0.0092	0.0091	0.0090	0.0090	0.0089	0.0088	0.0087	0.0087	0.0086
2.6	0.0085	0.0084	0.0083	0.0083	0.0082	0.0081	0.0080	0.0079	0.0079	0.0078
2.7	0.0077	0.0076	0.0075	0.0075	0.0074	0.0073	0.0072	0.0071	0.0071	0.0070
2.8	0.0069	0.0068	0.0068	0.0067	0.0066	0.0065	0.0065	0.0064	0.0063	0.0062
2.9	0.0062	0.0061	0.0060	0.0060	0.0059	0.0058	0.0057	0.0057	0.0056	0.0055
3.0	0.0055	0.0054	0.0053	0.0053	0.0052	0.0051	0.0051	0.0050	0.0049	0.0049

Source: Author

These errors seem acceptable for classroom usage when a computer is not available.

Figure 1 shows some of the approximations compared to our own and the true value of $\Phi(z)$. In Table 4 we compare the errors of these approximations to our own. The mean absolute error (MAE) and maximum absolute error (MaxAE) were evaluated for set of points $\{-5.00, -4.99, -4.98, \ldots, 4.98, 4.99, 5.00\}$.

Figure 1 – True values of the normal cdf (solid line) and approximate values (dots)



Source: Author

Table 4 – Mean absolute error and maximum absolute error for some approximations

Approximation	Expression	MAE	MaxAE
Our approximation	$\begin{cases} \frac{1}{0.19^z+1}, & \text{if } 0 \le z \le 1 \\ 0.63^{\exp(-z^{1.6})}, & \text{if } 1 < z \end{cases}$	0.00086	0.00498
Pólya (1945)	$\frac{1}{2}\left\{1+\left[1-\exp\left(\frac{-2z^2}{\pi}\right)\right]^{\frac{1}{2}}\right\}$	0.00101	0.00315
Cadwell (1951)	$\frac{1}{2}\left\{1+\left[1-\exp\left(\frac{-2z^2}{\pi}-\frac{2(\pi-3)z^4}{3\pi^2}\right)\right]^{\frac{1}{2}}\right\}$	0.00204	0.00647
Lin (1990)	$1-\left[1+\exp\left(\frac{4.2\pi z}{9-z}\right)\right]^{-1}$	0.00109	0.00669
Bowling (2009)	$1-\left[1+\exp\left(-0.07056z^3+1.5976z\right)\right]^{-1}$	0.10459	0.69678
Dombi and Jonas (2018)	$\left[1+\exp\left(-2\sqrt{\frac{2}{\pi}}z\right)\right]^{-1}$	0.00703	0.01767

Source: Author

Our approximation achieves the best mean absolute error and the second best maximum absolute error over the entire range, requires less inputs from the user of a pocket calculator and is not difficult to remember. For $z \to -\infty$ and $z \to \infty$ it approaches 0 and 1, as it is desirable for an approximation of $\Phi(z)$. The inconvenience is that our approximation is defined piece-wise, but even so, the derivatives of both expressions at $z = 1$ are very close in value. They are respectively 0.222 and 0.229. That seems sufficiently smooth for classroom use. The gap at $z = 1$ is approximately 0.004.

This approximation was obtained by using an probability estimation algorithm to automatically search for a good approximation in a very large set of possible candidates as explained in the next section.

2.3 An example of use for the approximation outside Statistics

The approximation we propose can be used in other subjects other than introductory courses in Statistics. Here is an example of how it can be used in an Engineering classroom. It is not expected that students in an Physics or Engineering classroom will carry statistical tables at all times. Cooray and Ananda (2012) discuss the mechanics of fatigue induced cracks in some materials. A generalization of the half-normal distribution is proposed by them as follows.

Brittle materials may exhibit delayed fractures due to environmental stress such as corrosion, moisture, heat and other factors. Glass and ceramics are two of such materials (WACHMAN, 1996). Let $\{\tau\}$ be the stress tensor given by

$$\boldsymbol{\tau} = \begin{bmatrix} \tau_{xx} & \tau_{xy} & \tau_{xz} \\ \tau_{yx} & \tau_{yy} & \tau_{yz} \\ \tau_{zx} & \tau_{zy} & \tau_{zz} \end{bmatrix} \tag{11}$$

where τ_{ij} represents the stress acting on a pĺane normal to direction i and towards direction j. A mode I fracture happens from a flaw in a plane that is normal to the tensile strength. The length of the crack evolves over time according to the equation

$$\frac{dL}{dt} = AK_I^B, \tag{12}$$

where A and B are constants and K_I is the mode I fracture intensity factor, which is related to the asymptotic behavior of τ_{yy}, the component of the stress that is parallel to the y axis. In an homogeneous material under an homogeneous stress $\tau_{yy} = \tau$ it can be shown that $K_I = \tau D L^{1/2}$, where D depends on the crack geometry. This yields $L = \sigma t^{\alpha}$, where $\alpha = 2/(2-B)$, as solution to Equation 12.

The fatigue failure happens as a dominant crack grows over a critical length l_c. The length of those sub-critical cracks are usually modeled by a half-normal distribution whose cdf is given by

$$P(L < l) = 2\Phi(l) - 1,\ l > 0. \tag{13}$$

Using $L = \tau\sigma t^{\alpha}$ yields $P(T < t) = 2\Phi\left[\left(\frac{t}{\sigma}\right)^{\alpha}\right] - 1$ for the distribution of the random time T until failure of the material. This distribution was called Generalized Half-Normal distribution by Cooray and Ananda (2012). From this point on the discussion would be disrupted if there was a need for computer or normal probabilities tables. With the formula proposed here, we can easily obtain several quantities regarding this application without any special preparation. Some of these quantities are, for instance, threshold values for the time until failure of the material.

For instance, an approximation for $P(T < t)$ using (10) is given by

$$P(T < t) = \begin{cases} \dfrac{2}{1 + 0.19^{\left(\frac{t}{\sigma}\right)^{\alpha}}} - 1, & 0 < t \le \sigma \\ 2 \cdot 0.63^{\exp\left[-\left(\frac{t}{\sigma}\right)^{1.6\alpha}\right]} - 1, & t > \sigma. \end{cases} \tag{14}$$

The quantile function can be obtained by inverting the previous expression. For a quantile t_q such that $P(T < t_q) = q$ we have

$$t_q = \begin{cases} \sigma\left[\dfrac{\log\left(\dfrac{1-q}{1+q}\right)}{\log(0.19)}\right]^{1/\alpha}, & 0 \le q \le \dfrac{81}{119} \\ \sigma\left[\log\left(\dfrac{\log(0.63)}{\log\left(\dfrac{1+q}{2}\right)}\right)\right]^{1/1.6\alpha}, & \dfrac{81}{119} < q \le 1. \end{cases} \tag{15}$$

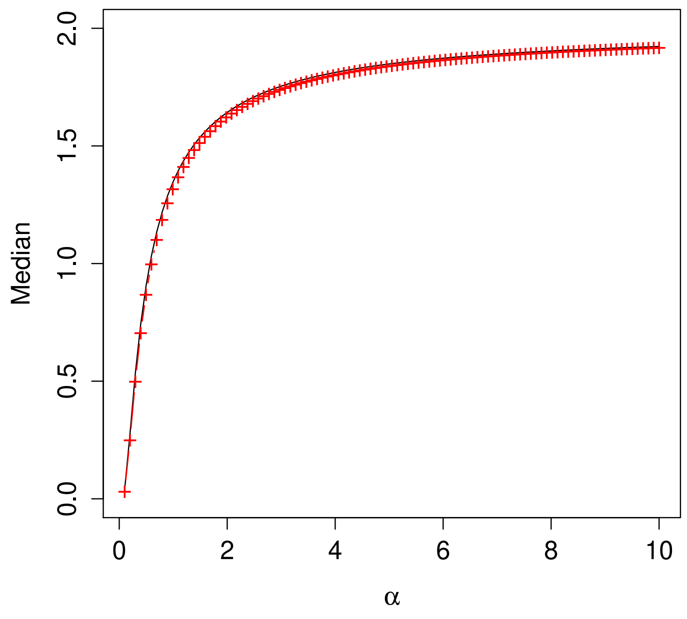
To illustrate the behavior of our approximation we provide, in Figure 2, the comparison of the approximate and real values of the median for varying values of α and $\sigma = 1$.The true value of the median is given by

$$t_{0.5} = \sigma\left[\Phi^{-1}\left(\frac{3}{4}\right)\right]^{1/\alpha}$$

. Using (10) yields

$$t_{0.5} = \sigma\left[-\frac{\log(3)}{\log(0.19)}\right]^{1/\alpha}$$

Figure 2 – True median values (solid line). Approximated values (+)



Source: Author

To test our approximation even further, consider a system of n components each of which is subject to the same stress of the previous situation. If these components are arranged in a parallel fashion the system works until all of them fail. In this case, the time until failure is the maximum of the set of times until failure of each component. It has cdf given by

$$P(T_{max} < t) = \left(2\Phi\left[\left(\frac{t}{\sigma}\right)^{\alpha}\right] - 1\right)^{n}. \tag{16}$$

If the components are arranged in a serial configuration, meaning that the system will fail if any of the components fail, the time until failure will have cdf given by

$$P(T_{min} < t) = 1 - \left(2 - 2\Phi\left[\left(\frac{t}{\sigma}\right)^{\alpha}\right]\right)^{n}. \tag{17}$$

The quantile function for the maximum and the minimum are respectively given by

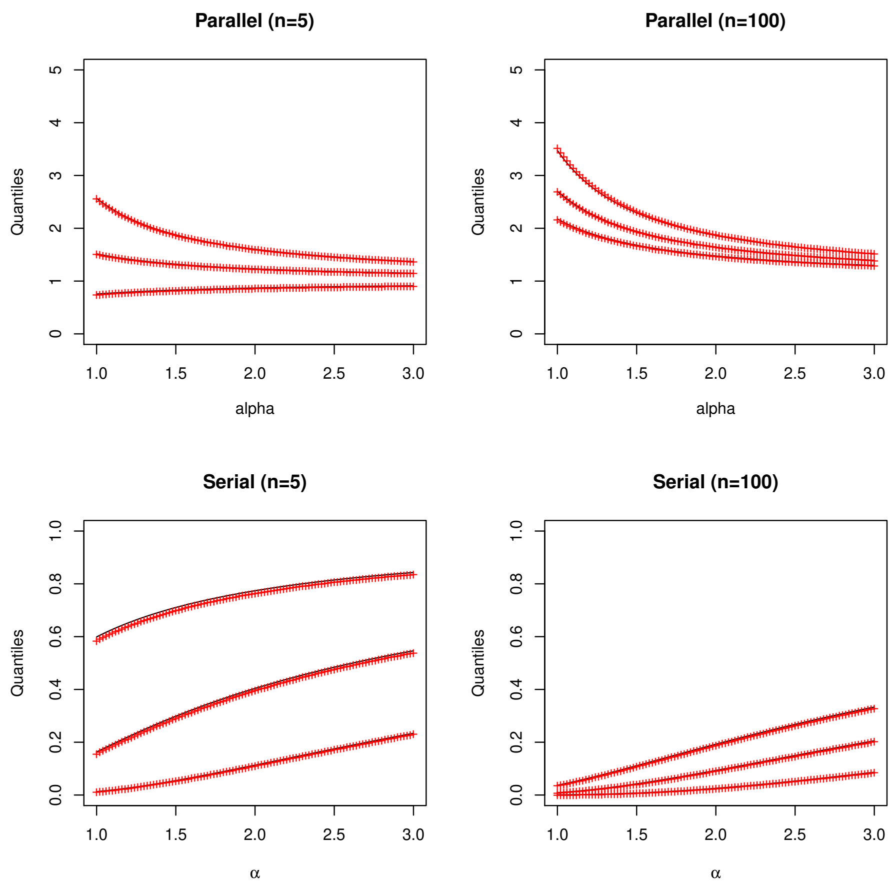
$$x_q = \begin{cases} \sigma \left[\dfrac{\log\left(\dfrac{1-q^{1/n}}{1+q^{1/n}}\right)}{\log(0.19)} \right]^{1/\alpha}, & 0 \le q \le \left[\dfrac{81}{119}\right]^n \\ \sigma \left[\log\left(\dfrac{\log(0.63)}{\log\left(\dfrac{1+q^{1/n}}{2}\right)} \right) \right]^{1/1.6\alpha}, & \left[\dfrac{81}{119}\right]^n < q \le 1, \end{cases} \tag{18}$$

and

$$x_q = \begin{cases} \sigma \left[\dfrac{\log\left(\dfrac{(1-q)^{1/n}}{2-(1-q)^{1/n}}\right)}{\log(0.19)} \right]^{1/\alpha}, & 0 \le q \le 1-\left[\dfrac{38}{119}\right]^n \\ \sigma \left[\log\left(\dfrac{\log(0.63)}{\log\left(\dfrac{2-(1-q)^{1/n}}{2}\right)} \right) \right]^{1/1.6\alpha}, & 1-\left[\dfrac{38}{119}\right]^n < q \le 1. \end{cases} \tag{19}$$

Figure 3 illustrates the error in our approximation for the 5% and 95% quantiles and the median of both scenarios for varying values of α, $\sigma = 1$, $n = 5$ and $n = 100$.

Figure 3 – True quantile values (solid line). Approximation quantile values of a GHN random variables set (+)



Source: Author

The approximation was good enough to be used for solving classroom problems not only regarding direct applications of the normal distribution but also in at least one other field where the normal cdf may present itself as part of the expressions needed. Besides this use for teaching purposes it can also be used in situation such as quantile regression involving some extensions of the normal distribution as a fast first analysis of the problem.

2.4 The PIPE algorithm

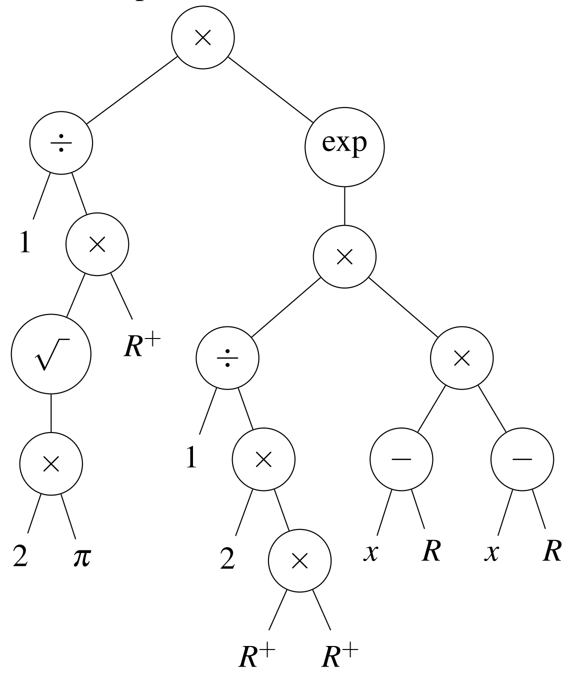
Our approximation was obtained by using an algorithm known as the Probabilistic Incremental Program Evolution (PIPE) proposed by Salustowicz and Schmidhuber (1997). The PIPE algorithm is capable of producing programs according to a set of probability rules. These rules are improved over iterations so that the generated programs are more likely to solve a given problem. In this section we follow closely the explanation in the original paper.

A *program* is a set of instructions given in a certain order. Each of these instructions may depend on a (possibly empty) set of terminal symbols, which usually denote constants or user inputs. Let $F = \{f_1, f_2, \dots, f_k\}$ be a set of k functions and $T = \{t_1, t_2, \dots, t_l\}$ be a set of l terminals. For instance, to write a program that calculates the value of the probability density function (pdf) for the normal or exponential distributions at a point x and a given set of parameters, it is sufficient to take $F = \{-, \times, \div, exp, \sqrt{}\}$ and $T = \{x, \pi, 2, 1, -1, R, R^+\}$, where $\div$ the protected division (does not allow division by zero), x represents an user input, R represents a real constant and R^+ represents a positive real constant. The normal distribution pdf can be described in the below example

$$(1 \div (\sqrt{2 \times \pi} \times R^+) \times \exp((-1 \div (2 \times R^+ \times R^+)) \times (x - R) \times (x - R)).$$

Each program can be represented by an n-ary tree, where n is the maximum possible of arguments for a function in F. For the normal distribution example we may use the tree in Figure 4. The tree representing a program is not unique unless we specify a set of rules for parsing a program to a tree, however this is negligible for our purpose.

Figure 4 – Tree representation of the normal distribution pdf



Source: Author

Programs can be created randomly by traversing a structure called Probabilistic Prototype Tree (PPT). The PPT is a n-ary tree with n, again, representing the maximum arity of an instruction in F. The node at depth $d \geq 0$ and horizontal position $w \geq 0$ (width) is represented by $N_{d,w}$. Each node contains a probability vector $\mathbf{P}_{d,w}$ whose entries are $\mathbf{P}_{d,w}(I)$ for each $I \in F \cup T$ such that

$$\sum_{I \in F \cup T} P_{d,w}(I) = 1,\ \forall N_{d,w}. \tag{20}$$

That is, each node has the probability distribution of the possible instructions in the programs at the respective node of their tree representation. The PPT is traversed in a depth first fashion from left to right, starting at $N_{0,0}$. For each accessed node, an instruction I is selected with probability $P_{d,w}(I)$ and denoted $I_{d,w}$. If $I_{d,w} \in F$, then a subtree is created for each argument of $I_{d,w}$. If $I_{d,w} \in T$ then it is replaced by an instance $V_{d,w}(I_{d,w})$ of that terminal. This instance equals $I_{d,w}$ if $P_{d,w}(I_{d,w})$ is greater than a certain threshold T_I and equals a randomly generated number $R_{d,w}$ otherwise. For each terminal instruction $I \in T$ there corresponds a threshold T_I and these are not changed throughout the iterations. The authors in Salustowicz and Schmidhuber (1997) also consider the *g*rowing and *p*runing of the PPT to reduce the memory requirements of the algorithm. Initially there is only the node $N_{0,0}$. If $I_{d,w} \in F$ is chosen and the subtree for its arguments are missing in the PPT, then additional nodes are created (growing). Conversely, if the probability of accessing a certain node in the PPT is too small, the node is deleted from the PPT (pruning).

PIPE has two learning mechanics: elitist learning and generation-based learning. These two mechanics alternate until a stopping criterion is met. Generation-based learning comprises five distinct phases.

1. **Program Population Creation.** A population of programs is created according to the rules mentioned earlier. These programs are enumerated as PROG$_j$, $0 < j \leq PS$, with PS denoting the population size. Probabilities in each node are initialized in a random way but maintaining their sum equal to 1.
2. **Population evaluation.** Each $PROG_j$ in the population is evaluated regarding a certain fitness function. This is a numeric value assigned by a function FIT($PROG_j$). The programs are ranked in ascending order of those values. The best program in the current population is denoted $PROG_b$ while the best program found so far is denoted $PROG^{el}$.
3. **Learning from the population.** The probabilities in each node of the PPT are modified as to increase the likelihood of $PROG_b$ being generated. The following steps can be stored as the content of an *adaptPPTtowards*($PROG_b$) routine at the time of the implementation

by the reader. This routine can also be found in our implementation. First $P(PROG_b)$ is obtained as $\prod P_{d,w}(I_{d,w})$ for each instruction $I_{d,w}$ used in the production of $PROG_b$. A target probability is calculated as

$$P_{TARGET} = P(PROG_b) + lr[1 - P(PROG_b)] \times \frac{\varepsilon + FIT(PROG^{el})}{\varepsilon + FIT(PROG_b)} \tag{21}$$

in which the constant lr denotes the learning rate of the algorithm, and ε is a user-defined positive real constant. The fraction in the right hand side of the equation implements the fitness-dependent-learning (fdl). If ε is chosen such that $\varepsilon \ll FIT(PROG^{el})$ then generations with lower quality (higher fitness values) programs do not influenciate much the learning process, allowing for the use of smaller populations. Once P_{TARGET} is obtained, all the probabilities $P_{d,w}(I_{d,w})$ for the instructions used in $PROG_b$ are increased iteratively:

```
1 REPEAT UNTIL P(PROG$_b$)>P$_{TARGET}$:
2   FOR EACH INSTRUCTION $I_{d,w}$ in PROG$_b$:
3     $P_{d,w}(I_{d,w}) := P_{d,w}(I_{d,w}) + c^{lr} \cdot lr \cdot (1 - P_{d,w}(I_{d,w}))$,
```

where c^{lr} denotes a constant that influences the number of iterations and the precision. The choice of this constant is subjective. Lower values will imply more iterations and more precision while higher values will do the opposite. Then, each terminal used in the construction of PROB$_b$ is stored in the respective node of the PPT, that is, I$_{d,w}$:= V$_{d,w}$(I$_{d,w}$) for each terminal instruction I$_{d,w}$ used in PROG$_b$.

4. **PPT Mutation.** In this step, the nodes acessed during the production of $PROG_b$ are mutated with a probability P_{M_P} given by

$$P_{M_P} = \frac{P_M}{(l+k)\sqrt{|PROG_b|}}, \tag{22}$$

where P_M is a user defined parameter controlling the overall probability of mutation. The previous formula is empirically justified in Salustowicz and Schmidhuber (1997). If a node is to be mutated, the probability $P_{d,w}(I_{d,w})$ is changed to $P_{d,w}(I_{d,w}) + mr \cdot (1 - P_{d,w}(I_{d,w}))$, in which mr represents a mutation rate. Notice that this change is small if $P_{d,w}(I_{d,w})$ is already large. After the mutation step every modified node is normalized to keep the sum of probabilities equal to 1.

5. **PPT pruning.** If $P_{d,w}(I_{d,w})$ becomes too small for a certain node $N_{d,w}$ and instruction $I_{d,w} \in F$ then the sub-trees corresponding to the possible arguments of $I_{d,w}$ are deleted from the PPT.

After the generation-based learning, elitist learning takes place by repeating the previous procedure using $PROG^{el}$ instead of $PROG_b$. However, during the elitist learning mutation is not performed. The PPT is then pruned accordingly.

The python's PIPE implementation for this task is composed of three files. The file *pipeffr.py* contains the algorithm and is the core engine algorithm. Here, *ffr* stands for "for functional regression"as it can be readily used to approximate other functions with very little effort. The file *myFunctions.py* contains implementations of the mathematical functions we wish to use that are not available in popular python modules. In the investigations, the set of parameters for the PIPE algorithm are shown in Table 5. We provided a set of 1000 pairs $(\Phi(z), z)$ to the algorithm, with z in the set of equally spaced points in a $[0, 5]$ interval as input. An analysis of the execution time of our implementation is available in Table 6.

Table 5 – PIPE's Setup Parameters

Parameter	Value
Population size (PS)	10000
Learning rate (lr:	0.01
ε	0.01
Generations	500
c^{lr}	0.5
P_M	3
Mutation rate (mr)	0.8
Height of the PPT	4

Source: Author

Table 6 – Statistics for elapsed time per sample size (100 times execution)

Sample Size	Mean Time (sec)	Std Deviation Time (sec)
10	138,71	11,02
20	166,66	28,76
30	225,22	49,04
40	267,43	65,52
50	317,72	73.96
100	559,68	109,87
150	850,18	178,08
250	1130.03	263.95
500	2312.08	288.42
1000	4257.26	958.78

Source: Author

The code and instructions on how to use it for other problems are available under request.

2.5 Chapter final remarks

In this paper we presented an approximate expression for the normal cumulative distribution function that is both easy to use and to remember. It achieves a very low absolute error that is better, on average, than the ones reported for other existing approximations. This is intended for classroom support and quick use when out in the field or for any non-critical use. We provided an example of situation outside of the introductory courses in Statistics where this approximation may be used to allow easier calculations of some properties of a system. Many others may arise from talking about income distribution (by means of the log-normal distribution) and fluid dynamics as in gaussian diffusion models.

In our experience, we also feel that the normal table ends up being understood as the only way to obtain normal probabilities even by some skilled professionals from other areas. We also suggest de-emphasizing the use of the normal distribution as a general tool for solving every problem, which is the prevalent perception of the role of the normal distribution by the students in many fields. These approximations were presented to our students and we found a very good acceptance.

We intend to provide, in future investigations, other approximations for the other distributions used in introductory courses in Statistics. So far, we did not have much success searching for an approximate expression for the Student's t distributions as our approximations were not as good as the existing ones.

3 NEURAL NETWORKS FOR MODEL SELECTION PROBLEMS

In this chapter we revisit a problem presented in Marshall et al. (2001). We attempt to find evidence of the existence of techniques that may provide better model selection than the standard techniques in main literatures.

3.1 Model selection methods

In Elderton and Johnson (1969) there is a discussion and general account of the history of the use of frequency curves to model data. Among the models in chapter 4 of Elderton and Johnsons (1969) there are the probability distributions from the so called Pearson system, developed by Karl Pearson. The Pearson system is a class of distributions obtained by noticing common features in data sets such as the relative position of the mean and median. Most of the distributions of the Pearson system were presented in Pearson (1985) and Pearson (1901). Many modern distributions were present in this system. That is not by chance. The distributions in the Pearson system were designed for specific scenarios that could be observed from measures of skewness and kurtosis in empirical plots. The system is majorly indexed by the third and fourth moments of the distributions. Among the distributions in the Pearson system are the normal distribution (Pearson type V), gamma distribution (Pearson type III), inverse-gamma distribution (Pearson type V), beta-prime distribution (Pearson type VI) and even Student's t (Pearson type IV) much before William Sealy Gosset's paper published under the pseudonym of Student which gave the distribution its name.

Nowadays, the choice of a particular distribution to model data is largely ad-hoc or the normal distribution is invoked through some version of the central limit theorem (often through distorted or misinterpreted assumptions). Models of higher complexity, in number of parameters, can be quite flexible in modeling a diverse spectrum of data but may lack a theoretical connection to the underlying mechanism that generates the data. Even so, it is undeniable that ad hoc choices of models will greatly speed up the time to deploy a model to production. The collection of papers on newly created distributions, especially by means of additional parameters using composition of existing cumulative distribution functions, grows larger and the growth seems to be accelerating. The collective of these distributions represents an alternative to non-parametric models. However, it might be of considerable difficulty to choose the appropriate distribution from such a large pool of often similar choices.

Since this shift from careful design of probability distributions specific to certain situations to a faster framework of statistical modeling, model selection methods have improved in demand.

There are three popular techniques for model selection: choosing the distribution with higher likelihood among the candidates, choosing the distribution with the lower Kolmogorov-Smirnoff distance or the one with the lowest AIC or BIC (or similar measures) scores.

For using the likelihood, one can rely on the generalized likelihood ratio test as well for extra certainty. The Kolmogorov-Smirnoff distance is the statistic used for the test of the same name. It has been suggested the use of the Kolmogorov-Smirnoff statistic as a criteria for model selection (MARSHALL et al., 2001).

To penalize overparametrized models, the AIC assigns to the fit of the model to the data the value $-2\widehat{\ell}+2k$, where $\widehat{\ell}$ represents the logarithm of the maximum value of the likelihood of the model and k represents the number of parameters of the model. The BIC has a penalty factor of $\log(n)k$

Overfitting, in a broad sense, is not restricted to having a high number of parameters. One example of a one parameter function that overfits data to an arbitrary degree can be seen in Piantadosi (2018). It makes use of power of 2 arithmetic and theory on iterating functions to choose the correct value of a parameter. In the examples of Piantadosi (2018) the choice of parameter alters the scatterplot of the function's values from the drawing of an elephant to a signature.

3.2 Results from Marshall and Olkin (2001)

The main concern of Marshall and Olkin (2001) is whether or not data will recognize its parent distribution, as state previously. They initially point out the issue of models having no physical considerations regarding the nature of the data, as opposed to the Pearson system, for instance, as seen in Section 2.

Regarding mode selection techniques, they used only the maximum likelihood values of the models and the Kolmogorov-Smirnoff distance in their analysis. Based on these two criteria they attempt to answer two specific questions:

1. How successful are these methods in identifying the actual parent parametric distribution when given one or more alternative distributions?
2. How large must be the sample size to make the correct selection with a given probability?

They observe that some parametric distribution families are “richer” than others, in the following sense. A dataset generated by a Weibull distribution is likely to be well fitted by a gamma distribution. On the other hand, data from a gamma distribution is not likely to be better fitted by a Weibull distribution. We will denote this relationship by saying the gamma model is more flexible than the Weibull family.

One possible criticism to such caution on model selection is that if a model is more flexible than other it might be enough to use the most flexible one in practical applications. This is not true at all, especially if tail behavior has a critical importance. An example of such situation appears in insurance modelling, where it is reasonable to sacrifice the goodness of fit of the majority of the data in order to accurately predict tail behavior.

More details on this matter can be seen in, for instance, Bain and Engelhardt (1980), focusing on the Gamma and Weibull models, Kappenman (1982), Taylor and Jakeman (1985), Fearn and Nebenzahl (1991).

The probability density function of the three distributions considered in Marshall and Olkin (2001) are given in below table.

Table 7 – Probability density functions in Marshall and Olkin (2001)

Family	Probability density function
Weibull	$\alpha\lambda(\lambda x)^{\alpha-1}$
Gamma	$\Gamma(\nu)^{-1}\lambda^{\nu}x^{\nu-1}exp^{-\lambda x}$
MO-Exp	$\gamma\lambda exp^{-\lambda x}\left(1-(1-\gamma)exp^{-\lambda x}\right)^{-2}$

Source: Author

The authors show that for their investigations the scale parameter is not important. For the three distributions, they set the scale parameter such that the expected value of the random variable with such distribution equals 1.

The reason for choosing these distributions is that they are similar in some aspects. They all include the exponential distribution as special cases, they all have monotone hazard rates, and they all have one parameter after setting the scale parameter accordingly.

For the experiment of choosing one distribution to the date, Marshall and Olkin proceed as follows. The families are fitted to the data and the Kolmogorov-Smirnoff distance is calculated for every fitted distribution. The one with the lower distance is chosen. Taylor and Jakeman (1985) used this hybrid method. In one of their examples, they showed that if the exponential, gamma and Weibull models are to compete on modeling exponential data, then using this method provides more correct choices then using only the maximum likelihood alone. Similar works on discrimination procedures can be seen in Pandy et al. (1991), Dyer (1973), McDonald et al. (1995), Cox (1961) (using Bayesian methods), Siswadi and Quesenberry (1982), Hogg et al. (1972), Atikinson (1970) and Olkin and Spielgman (1987).

For each combination of sample sizes and a selected set of parameters, Marshall and Olkin (2001) generated data from each distribution, 100.000 samples for each configuration, to estimate the probability of correctly choosing the parent model. The results from Mashall and Olkin (2001) are summarizes as follows.

For the Weibull data, the correct selection probability is less if against a gamma alternative if compared to a Marshall-Olkin-exponential alternative if the shape parameter is away from 1. closer to 1, the Marshall-Olkin-exponential is preferred compared to the gamma. For sample sizes lower than 400 and shape parameter between 0.9 and 1.1, the Weibull distribution is the least likely to be chosen.

When the gamma distribution is the parent distribution, for shape parameter values above 1, the Weibull distribution is more likely to be chosen than the Marshall-Olkin-exponential distribution when only one alternative is given. For the two alternatives given at the same time, the Weibull is more likely to be chosen over the Marshall-Olkin-exponential distribution for shape parameter values away from 1. The opposite is true for values near 1. When the shape parameter is near 1, the Marshall-Olkin-expoential distribution is mode likely to be chosen over the gamma for reasonable sample sizes.

Finally, for the Marshall-Olkin-exponential data, with one alternative, the gamma distribution is not as likely as the Weibull. One curious fact is that when the Weibull distribution is presented as the alternative to the parent Marshall-Olkin-exponential, the Kolmogorov-Smirnoff distance is mode likely to point out the correct distribution than the likelihood value for sample sizes bellow 600.

3.3 A Brief Introduction to Deep Neural Networks and Keras Python Library

Machine learning is the are of Science consisting of studies on methods and algorithms that perform a given task without being explicitly programmed to do so. These methods and algorithms rely on patterns and rule based inference.

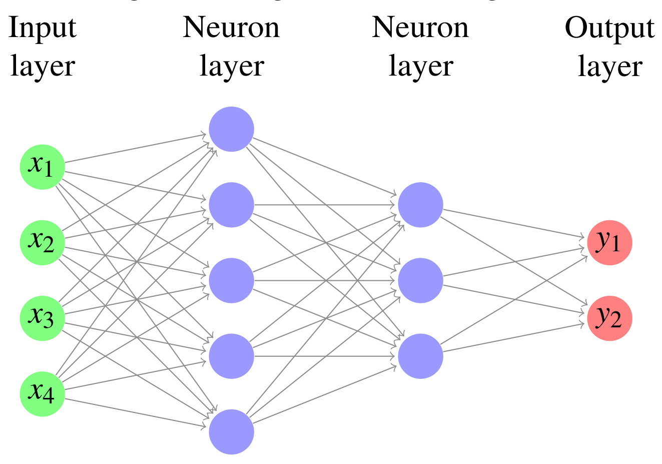
There is a vast spectrum of algorithms in machine learning. The previous chapter dealt with the PIPE algorithm. For this chapter, neural networks are at focus. A neural network is an organized sequence of neurons layers. Each neuron is a unit that receives the output of the neurons of the previous layers and processes these inputs to produce an output. The neurons from the first layer receive the data as input while the ones in the final layer produce the output of the network. By presenting a series of input and output examples it is possible to adjust the connections between layers to obtain accurate predictions of linear and non-linear patterns in the data.

A neural network, loosely speaking, is a system of inputs and outputs vaguely based on some biological process. Mathematically, let g be a function, $\boldsymbol{x}$ and $\boldsymbol{y}$ vectors such that $g(\boldsymbol{x}) = \boldsymbol{y}$. Suppose that g is not easy or practical to be computed. A MLP can be used to approximate $g(\boldsymbol{x})$. This process is made in, usually, three steps.

1. Example collection: We provide some examples of pairs $(\boldsymbol{x},\boldsymbol{y})$, such that $\boldsymbol{y} = g(\boldsymbol{x})$.
2. Learning or training: The network is trained to correctly assign each $\boldsymbol{x}$ to a value close to its respective $\boldsymbol{y}$.
3. Validation: During the training phase, it is possible to overfit the data. That means the network will perform incredibly good in the example set but may perform very poorly for $\boldsymbol{x}$ outside the example set. The validation phase checks if there is evidence of overfitting.

A neuron is the basic element of a MLP. It receives a value v and returns $\phi(v)$, where $\phi(\cdot)$ is called activation function and usually has range in $[0,1]$ or $[-1,1]$. Neurons are organized in layers. The first layer of neurons is proceeded by a layer of inputs of the network. Between every input and neuron there is a weighted link called synapsis. The weight between neuron j and input i is denoted ω^1_{ij}. The input of this neuron j is $v_j = \sum_{i=1}^{n} \omega_{ij} x_i$, where n is the number of inputs of the network. Refer to Figure 5.

Figure 5 – A general MLP diagram



Source: Author

Th output of the jth neuron in the first layer will be denoted $\theta_{1j} = \phi(v_j)$. The outputs will be the inputs of the next neurons layer, and so on. Finally, the last layer will have as many neurons as $g(\boldsymbol{x})$ has entries. The jth output of the last layer will be denoted θ_j. It is usual to make $x_1 = 1$. A popular choice for $\phi(v)$ is $\phi(v) = (1 + \mathrm{e}^{-v})^{-1}$, which is known as the sigmoid function.

The learning process consists in adjusting the weights of the network to make its output close to $\boldsymbol{y}$ for each corresponding $\boldsymbol{x}$. A popular method of doing so is the back-propagation algorithm. Consider the cost function

$$C = \frac{1}{2} \sum_j (\theta_j - y_j)^2,\ y_j \in \boldsymbol{y}. \tag{23}$$

The weights can be adjusted in an iterative fashion aiming at reducing the value of the cost function. The gradient of C regarding the weights ω_{ij} can be shown to equal

$$\frac{\partial C}{\partial \omega_{ij}} = \delta_j\, \theta_i, \text{ with } \delta_j = \begin{cases} (\theta_j - y_j)\theta_j\,(1-\theta_j), \text{ if } j \text{ is in the output layer,} \\ \left(\sum_k \delta_j\, \omega_{jk}\, \theta_j\,(1-\theta_j)\right), \text{ if } j \text{ is not in the output layer,} \end{cases}$$

where the summation carries over all k neurons in the proceeding layer j. Consider θ_j as the jth input of the network when adjusting the first layer of neurons. The above formula is valid only when $\phi(v)$ is the sigmoid function. Some common stopping criteria are limiting the number of interactions, stopping when the change in C is lower than a certain threshold, stopping when the percent change in C is small enough.

An epoch is the number of iterations to update the weights once for every pair (x_i, y_i). After each epoch, it is a good ideia to do an overfitting check. This is done by evaluating the performance of the network in a different set than the one used to train the network. When the performance in this test set begins to fall, stop network training.

Deep neural networks are network architectures able to extract higher level concepts from data such as the comprehension on what is a digit, an animal or a mathematical formula from the input. The input might be given as a picture of the object whose features are to be discovered, an audio or text description or measurements from equipments such as MRI readings. Deep neural networks have been used also to extract sentiment from text, a technique known as sentiment analysis, or to classify texts by authors from their respectives writing styles.

To achieve such higher level concept comprehension, the architectures of such methods, i. e., the layers structures and how they will be connected might be considerably tricky both from computational and mathematical points of view. To implement such methods from ground up requires a hefty amount of knowledge in programming languages, scientific programming and mathematical computation. This leaves a huge gap between these methods and the areas they could be useful in. To bring these methods closer to practitioners, frameworks for deep learning have been proposed. Frameworks such as *Tensorflow*, *Theano* and *Pytorch* greatly speed up the time to use deep neural networks. The framework we used in this work was *Tensorflow* with a python library called *Keras* and runs atop of other frameworks.

This package was produced to allow fast implementations of such methods and algorithms, indeed, more of an interface than a framework of machine learning. Another useful layer of abstraction in the use of these models is the *Anaconda* Python distribution. The joint use of these tools is, at the time of writing, the fastest way to deploy deep learning capabilities in the typical personal computer setups.

These frameworks allowed many researchers of different fields to quickly use these methods and this allowed significant advances in many areas such as medical research. The evidence for this is in the large number of published papers in well established journals such as Nature Communications and similar. A quick search on its front page reveals, for instance, the works of Godec et al. (2019) on the analysis of biomedical images for breast cancer detection, sub cellular protein localization, plant disease detection and much more. Johnson et al. (2019) uses *Keras* (among other frameworks) to rain a model to estimate ages from brain structural MRI. The difference between chronological age and predicted age from the brain image is a predictor of brain deterioration and diseases. This study also unveiled two associated sequence variants on DNA, one is related to reduced sulcal width and the other to reduced white matter surface area. Some of these problems could be modeled by more traditional statistical methods but the mathematical knowledge threshold for doing so is much higher. This study utilizes transfer learning, which is training a model on a data set and utilizing it as a starting point in another problem that may share similarities with the previous one. We will discuss possible impacts of transfer learning on Statistical problems later on. Gomez-de-Mariscal et al. (2019) used *Keras* to achieve segmentation of small extracellular vesicles (sEV) in transmission electron microscopy (TEM). This is considered a difficult task since sEV's are of the nano scale and image preparation introduces a lot of statistical noise to the data. Their model outperformed two state-of-the-art models for this task. Park et al. (2019) developed automatic detection of pulmonary abnormalities using *Keras*.

Other papers are available on, for instance, new drugs research or new materials research. the main goal of this section is to present clear evidence that machine learning methods may provide useful tools for investigations in Statistics as well. In the next section we present one example of such investigation based on the ideas of Marshall and Olkin (2001) and partially in Jaynes (1954) on the maximum entropy characterizations of parametric families of distributions.

3.4 Our proposal

In this chapter, we propose to revisit the investigations from Marshall and Olkin (2001) from the point of view of neural networks for classification. Our intention for pursuing this direction is described as follows. Marshall and Olkin (2001) presented evidence that the likelihood and Kolmogorov-Smirnoff distance are not absolutely reliable for pairing data to its parent distribution. The neural networks, especially those easily scalable from *Keras* framework, may be able to find hidden patterns in the data that may provide efficient ways of classifying correctly according to the parent distribution. The natural candidate features of the data to hold information about the distributions are the sample moments. As the data grows in sample size, the sample moments approach the theoretical moments in probability. Given enough sample moments, it is our expectation that the networks may be able to correctly assign data to its parent distribution. One further evidence for our intuition is the maximum entropy characterization from Jaynes (1954). This characterization is based on constraints of the form $\mathbb{E}[g_i(X)] = a_i$, for $i = 1, 2, \ldots, k$, for a given k, real functions $g_i(\cdot)$ and real numbers a_i. For a set of such constraints, the maximum entropy distribution is the one satisfying the constraints and the one whose probability density function is a solution to $f(x) = \arg\max \mathscr{H}(g)$, over a set of probability density functions, where $\mathscr{H}(g) = \mathbb{E}[-\log(g(X))]$ is the Shannon entropy associated to $g(x)$. The maximum entropy characterization may be understood as the distribution that encompasses all the information known about the random variable (stated as the constraints) assuming nothing else. Thus, patterns in moments may identify uniquely the distribution. For instance, if the constraints are $\mathbb{E}[X] = \mu$ and $\mathbb{E}[(X-\mu)^2] = \sigma^2$, for real numbers μ and $\sigma > 0$, the maximum entropy distribution is the normal distribution. This can be verified by using the Gibbs inequality:

$$\int_{-\infty}^{+\infty} \log\left(\frac{f(x)}{g(x)}\right) f(x)dx > 0, \tag{24}$$

where the left hand side of the inequality is known is the Kullback-Leibler divergence or relative entropy.

Goodfellow et al. (2016) define this divergence is stated as followed. If we have two separate probability distributions $\mathbb{P}(X)$ and $\mathbb{Q}(X)$ over the same random variable X, we can measure how different these two distributions are using

$$D_{KL}(\mathbb{P}||\mathbb{Q}) = \mathbb{E}_X[\log \mathbb{P}(X) - \log \mathbb{Q}(X)]. \tag{25}$$

One most useful properties of KL divergence is that is non-negative. The divergence is 0 if and only if $\mathbb{P}(X)$ and $\mathbb{Q}(X)$ are the same distribution for discrete variables and almost equal in everywhere to continuous case. There is a misconception about the use of KL divergence. Sometimes it is used as a distance but like some sort of distance but it lacks symmetric property: $D_{KL}(\mathbb{P}||\mathbb{Q}) \neq D_{KL}(\mathbb{Q}||\mathbb{P})$.

Based on the suspicions, we used a *Keras* neural network with 3 fully connected layers, i. e., every neuron from the previous layer is connected to each other neuron in the next layer. We used the first 10 sample moments from artificially generated data sets as input. The output is a label indicating the parent distribution of the generated data. The simulated sample sizes were 100, 150, 200, 250, 500 and 1000. The shape parameter of these artificial samples were uniformly generated on $(1,5]$ interval. For each distribution, each sample size and each parameter value we generated 50.000 samples, and calculated the sample moments. This is the training data set. The output in training sample was coded as a dummy variable, the first 3 layers have 150 neurons each with a sigmoid activation function on each neuron. The output layer has 3 neurons, each neuron representing one of the candidate parent distributions. Unlike in Marshall and Olkin (2001), we set the scale parameter to 1 for all the distributions and don't necessarily have the expected value of the distributions equal to 1. This is reasonable since the training data has all the inputs normalized, i. e., each collection of sample moments is re-centered and re-scaled by the *Keras* functions for computational reasons.

The neural network was set to train for at most 10 epochs in batches of size 300. That is, the network process 200 samples from the training set at a time to make neuron weight adjustments. And early stopping criteria was used to stop the training if network accuracy reaches 95% or more. That is supposed to avoid overfitting and also because we observed a fast deterioration of the network performance above that level. The loss function we used was the categorical cross entropy function implemented in *Keras*. This function is used for multi-class classification tasks and its a modified version of cross entropy losses functions family and have a closely relationship with KL divergence.

Cross-entropy is frequently used to measure difference between two pdf. The grounded truth distribution is the one that our neural network is trying to predict and exposed in terms of a one-hot probability distribution. This measure is calculated as described in Goodfellow et al. (2016):

$$H(P,Q) = \mathbb{E}(X) \ln \mathbb{P}(X). \tag{26}$$

The categorical version also called Softmax Loss is a junction of softmax activation with cross-entropy loss. In this loss function, we will train a neural network to output a probability over the C classes of the problem and its defined by

$$\sigma(\boldsymbol{x})_i = \frac{\exp(x_i)}{\sum_j^K \exp(x_j)}, \quad for\ i = 1,2,\cdots,K\ and\ \boldsymbol{x} = (x_1,x_2,\cdots,x_k) \in \mathbb{R}^K \tag{27}$$

$$H_{categorical} = -\sum_i^C t_i \ln(\sigma(\boldsymbol{x})_i) \tag{28}$$

The test data set where we verify the capabilities of the trained network was generated in the exact same way as the training data set. The confusion matrix for the network classification is shown in Table 8.

Table 8 – Confusion matrix for the classification of the generated data in the test set

True distribution	Weibull	Gamma	MOE
Weibull	92,23%	6,52%	0,16%
Gamma	1,31 %	98,68%	0,00%
MOE	0,18 %	0,00%	99,82%

Source: Author

These results are overall superior than the ones obtained by main literature techniques used in Marshall and Olkin (2001).

3.5 Chapter final remarks

The main implications of these findings are the evidence of room for developing model selection methods based on sample moments and patterns in the theoretical moments. The maximum entropy characterization may be one starting point for developing these techniques. It is possible that this has to be developed on a per family basis, and in that case the neural network approach seems more efficient. This is the second implication, the use of neural networks for model selection. However, it is a considerable mathematical challenge to obtain theoretical guarantees on the soundness of the such method. To promote a method based on neural networks that works for a large set of probability distributions will most likely require a much larger and deeper neural network. Incremental changes are possible for the development of such network over time in a collaborative fashion by using transfer learning techniques. A network trained for classification of a smaller set of distributions may be used as starting point to train a larger network that classifies for more alternative distributions. Once trained this network can be readily used in model selection tasks. We, however, advise strongly that model designing takes place before using a model selection tool in an ad-hoc manner blindly. It is always desirable to have a model devised from the underlying aspects of the data than choosing from a pool of models possibly without theoretical connection to the data even if it means having a poorer fit to the data when inference is a goal. For predictions of non-critical applications it might be reasonable to choose the faster route.

4 FINAL REMARKS

In this text, we attempt to solve two problems of theoretical order in Statistics by using machine learning algorithms. For the two problems the results were considerably positive.

The problem of approximating the normal distribution cumulative distribution function has been approached by many authors since it began to be of importance, that is, since the gaussian kernel appeared as solution to many situations. While our method is not a substitute to proper mathematical evaluation and a combination of algebraic insights and first or second order approximations are still preferable, the use of an algorithm such as PIPE may provide solutions to these approximation problems. Our investigation regarding other important distributions, such as the Student's t, did not provide much success due to a problem related to the choice of the fitness function. The mean square error is very useful to penalize deviations from the average. For the datasets we trained on, the algorithm was overfitting the distributions with the degrees of freedom near the average value and the performance suffered on the rest of the data. Asymmetric choices of the fitness function did not provide good results when compared to the already existing approximations.

For the second problem, the main result is actually the evidence of possible developments on the designing of new model choice methods based on the sample moments. There results are largely superior to the ones using classical methods such as those on Marshall et al. (2001). Another possible development is the designing and training of a neural network model for the task of model selection. This will require a large computational capability for the training, given the number of different distributions and the amount of data necessary. It also would require a careful design of guidelines on updating the network when distributions are added to it. This would be preferably done in a collaborative fashion. Thus, concerns on stability and version control would arise. As for possible new tests, we emphasize the use of the maximum entropy characterization from the works of Jaynes.

Machine learning methods are often seen among practitioners and academics as a rival to Statistical methods. While it might be true for prediction tasks, where machine learning has the upper hand, statistical models offer more interpretability. Nevertheless, machine learning tools may be used to boost some areas of research in Statistics and related subjects. Also, major attempts are in course for more interpretable machine learning models.

REFERENCES

ATKISON, A. C. A method for discriminating between models. **Journal of the Royal Statistical Society**: Series B (Methodological), [*S.I.*], v. 32, n. 3, p. 323–353, set.1970.

BAIN, Lee J.; ENGELHARDT, Max. Probability of correct selection of weibull versus gamma based on livelihood ratio. **Communications in Statistics, Theory and Methods**, [*S.I.*], v. 9, n.4, p. 375–381, 1980.

BOWLING, Shannon R.; KHASAWNEH, M. T.; KAEWKUEKOOL, S.; CHO, B. R. A Logistic approximation to the cumulative normal distribution. **Journal of Industrial Engineering and Management**, [*S.I.*], v. 2, n. 1, p. 114–127, 2009.

CADWELL, J. H. The bivariate normal integral. **Biometrika**, [*S.I.*], v. 38, n. 3-4, p. 475–479, 1951.

COORAY, Kahadawala; ANANDA, Malwane M. A. A generalization of the half-normal distribution with applications to lifetime data. **Communications in Statistics, Theory and Methods**, [*S.I.*], v. 37, n. 9, p. 1323–37, 2008.

COX, D. R. Tests of separate families of hypotheses. In: BERKELEY SYMPOSIUM ON MATHEMATICAL STATISTICS AND PROBABILITY, 4., 1961, Berkeley. **Proceedings** [...]. Berkeley: University of California Press, 1961. p. 105-123.

DOMBI, József; JÓNÁS, Tamás. Approximations to the normal probability distribution function using operators of continuous-valued logic. **Acta Cybernetica**, [*S.L.*], v. 23, n. 3, p. 829–852, 2018.

DYER, Alan R. Discrimination procedures for separate families of hypotheses. **Journal of the American Statistical Association**, [*S.L.*], v. 68, n. 344, p. 970–974, 1973.

ELDERTON, W. P.; JOHNSON, N. L. **Systems of frequency curves**. Cambridge: University Press, 1969.

FEARM, D. H.; NEBENZAHL, E. On the maximum likelihood ratio method of deciding between the weibull and gamma distributions. **Communications in Statistics - Theory and Methods**, [*S.L.*], v. 20, n. 2, p. 579–593, 1991.

GODEC, Primož et al. Democratized image analytics by visual programming through integration of deep models and small-scale machine learning. **Nature Communications**, [*S.L.*], v. 10, n. 4.551, out. 2019. DOI: https://doi.org/10.1038/s41467-019-12397-x. Disponível em: https://www.nature.com/articles/s41467-019-12397-x.

GÓMEZ-DE-MARISCAL, Estibaliz et al. Deep learning-based segmentation of small extracellular vesicles in transmission electron microscopy images. **Scientific Reports**, [*S.L.*], v. 9, n. 13.211, set. 2019. DOI: https://doi.org/10.1038/s41598-019-49431-3. Disponível em: https://www.nature.com/articles/s41598-019-49431-3.

GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. **Deep learning: adaptive computation and machine learning**. Massachusetts: The MIT Press, 2016.

HOGG, Robert V. et al. On the selection of the underlying distribution and adaptive estimation. **Journal of the American Statistical Association**, [*S.L.*], v. 67, n. 339, p. 597-600, 1972.

HOYT, John P. The Teacher's Corner: A simple spproximation to the standard normal probability density function. **The American Statistician**, [*S.L.*], v. 22, n. 2. p. 25–26, 1968.

JAYNES, E. T. Information theory and statistical mechanics. **Physics Reviews**, [*S.L.*], v.106, n. 4, p. 620–30, 1957.

JONSSON, B. A. et al. Brain age prediction using deep learning uncovers associated sequence variants". **Nature Communications**, [*S.L.*], v. 10, n. 5.909, nov. 2019. DOI: https://doi.org/10.1038/s41467-019-13163-9. Disponível em: https://www.nature.com/articles/s41467-019-13163-9.

KAPPENMAN, Russell F. On A method for selecting a distributional model. **Communications in Statistics - Theory and Methods**, [*S.L.*], v.11, n. 6, p. 663–672, 1982.

LIN, Jinn-Tyan. A simpler logistic approximation to the normal tail probability and its inverse. **Journal of the Royal Statistical Society**. Series C (Applied Statistics), [*S.L.*]: Wiley, v. 39, n. 2, p. 255–257. 1990.

LIN, Jinn-Tyan. Approximating the normal tail probability and its inverse for use on a pocket calculator. **Journal of the Royal Statistical Society**. Series C (Applied Statistics), [*S.L.*]: Wiley, v. 38, n. 1, p. 69–70, 1989.

MARSHALL, A. W.; MEZA, J. C.; OLKIN, I. Can data recognize its parent distribution?. **Journal of Computational and Graphical Statistics**, [*S.L.*], v. 10, n. 3, p. 555–580, 2001.

OLKIN, Ingram; SPIEGELMAN, Clifford H.. A semiparametric approach to density estimation. **Journal of the American Statistical Association**, [S.I.], v. 82, n. 399, p. 858–65, set. 1987.

PANDEY, M.; FERDOUS, Jannatual; UDDIN, Md Borhan. Selection of probability distribution for life testing data. **Communications in Statistics - Theory and Methods**, [S.I.], v. 20, n. 4, p.1373–1388, 1991.

PARK, Beomhee et al. A curriculum learning strategy to enhance the accuracy of classification of various lesions in Chest-PA X-ray screening for pulmonary abnormalities. **Scientific Report**, [*S.L.*], v. 9, n. 15352, 2019. DOI: https://doi.org/10.1038/s41598-019-51832-3. Disponível em: https://www.nature.com/articles/s41598-019-51832-3.

PEARSON, Karl. Contributions to the mathematical theory of evolution – II: skew variation in homogeneous material. **Philosophical Transactions of the Royal Society of London**. [S.I.], London, v. 186, p. 343–414, 1895.

PEARSON, Karl.Mathematical contributions to the theory of evolution – X: supplement to a memoir on skew variation. **Philosophical Transactions of the Royal Society of London**: Series A, Containing Papers of a Mathematical or Physical Character, [S.I.], London, v. 68, p. 442–450, 1901.

PIANTADOSI, Steven T. One parameter is always enough. **AIP Advances**, [S.I.], v. 8, n. 9, set. 2018. Disponível em: https://aip.scitation.org/doi/10.1063/1.5031956.

PÓLYA, G. Remarks on computing the probability integral in one and two dimensions. In: BERKELEY SYMPOSIUM ON MATHEMATICAL STATISTICS AND PROBABILITY, [1.], 1949, Berkeley. **Proceedings** [...]. Berkeley: University of California Press, 1949. p. 63-78.

SALUSTOWICZ, Rafal, SCHMIDHUBER, Jurgen. Probabilistic incremental program evolution: stochastic search through program space. 1997, in. SOMEREN, Van; WIDNER, G. Machine Learning: ECML-97 – **Lecture Notes in Computer Science. Berlin, Heidelberg**: Springer. 1997. p. 213 – 220.

SISWADI, C. P. Quesenberry. Selecting among weibull, lognormal and gamma distributions using complete and censored smaples. **Naval Research Logistics Quarterly**, [*S.L.*], v. 29, n. 4, p. 557–569, 1982.

TAYLOR, John A.; JAKEMAN, Anthony J. Identification of a distributional model. **Communications in Statistics - Simulation and Computation**, [*S.L.*], v. 14, n. 2, p. 497–508, 1985.

TOCHE, K. D. **The art of simulation**. London: English Universities Press. 1963.

WACHTMAN, J. B. **Mechanical properties of ceramics**. New York: Wiley, 1996.

ANEXO A – PIPE AND KERAS NEURAL NETWORK SOURCE CODE

You can find the code developed in this text at shorturl.at/cfgmF