Automatic Distribution Fitting of Tabular Data using the Simulated Annealing Algorithm

David Krongauz (ID. 316440510), Teddy Lazebnik (ID. 208490722)

Abstract

Allocating a parametric distribution to features in tabular data is a well-known problem for all data analysis as this provides a wide range of abilities such as better explainability, generation of new samples, and better picking of classification and regressions models using meta-learning. In this study, we propose a simulated annealing approach to allocate a parametric distribution for features, obtaining the optimal distribution and its parameters. We show that the proposed approach performs on average similar to the commonly used brute force with maximum likelihood approach even for simple distributions such as uniform and normal distributions.

Keywords: directed search, feature engineering.

1 Introduction

A central problem in data analysis in general and in the development of machine learning (ML) solutions, in particular, is to identify and quantify the behavior of features of the data [1]. One method to better grasp the behavior of a feature is by finding the parametric distribution that yields it [2].

Given a parametric distribution to feature, one can more easily define erroneous data. For example, by looking at the distribution one can detect outliers as the rule of "too far from the mean" is analytically defined for most of the parametric distributions. Moreover, a computation of how noisy the data is can be achieved by computing the coefficient of determination between the parametric (analytical) and the sample (numerical) data. A higher coefficient of determination determines a less noisy signal.

These capabilities are later used for a wide range of data analysis and ML-related tasks such as finding the most useful features for obtaining the optimal results in a classification or regression task [3]. This paper proposed a simulated annealing approach to fit a parametric distribution to each feature in a tabular state by defining a discrete-continuous search space of the distribution functions and their parameters.

The paper is organized as follows: Section 2 presents a short literature review of methods for fitting parametric distribution to features, common parametric distributions, and the simulated annealing algorithm. In Section 3, we propose our novel simulated annealing based search approach for fitting parametric distribution to features. Section 4 describes the results of an empirical evaluation of the proposed approach and comparison to other methods. Section 5 concludes the paper.

2 Related work

Distributions related to engineering and technology, which attempt to model, for instance, the lifetime or time to failure of equipment, as well as in biology and pharmaceutics, have blossomed in recent years, driven by the fast increasing availability of sensor data and other large sources of quantifiable information. It is common to

fit the different features constricting these datasets using the uniform and Gaussian (normal) distributions [4] since many data science methods rely on normally distributed data or residuals such as the K-Means and the Gaussian Mixture Model (GMM).

Nonetheless, to properly model real-world random processes, one needs to be able to identify and evaluate alternative random models due to the fact that multiple biological, social, engineering processes are not normally or uniformly distributed [5]. For example, the popular SciPy library (written in Python) contains 123 different distributions in which 103 are continuous and 19 are discrete [6].

Several algorithms have been proposed to tackle the challenge of fitting the optimal parametric distribution for features over the years [7, 8]. For instance, Nayar et al. [2] proposed a computer vision based method, treating the tabular data as a n-dimensional image. The authors used the geometrical topology associated with each distribution to test the fitness of each distribution to the data in a two-step procedure: finding candidates for the distribution with random parameter values and fine-tuning the distribution's parameter values. Contrary, our approach assumes treats the distribution as a functional property rather than a geometrical configuration.

Another example is the approach proposed by Perkowski et al. [9] which assumes the distribution of each feature can be computed using the decomposition of its function to a sequence of basic distribution functions. One such sequence is identified, one can treat it as a clustering problem in order to obtain the optimal distribution. Afterward, the parameters of the obtained distribution can be obtained by reverse-engineering the values for the decommission representation using any numerical method such as the Newton-Raphson algorithm [10]. In contrast, our approach assumes that the distribution and its parameters are cons tracing a discrete-continuous search space.

2.1 Simulated Annealing

SA has been widely adopted for multiple optimization tasks. For instance, [11] applied SA to the problem of drawing graphs "nicely." The authors deal with general undirected graphs with straight-line edges, and employs several simple criteria for the aesthetic quality of the result, concluding that the algorithm is flexible, in that the relative weights of the criteria can be changed. [12] used SA in the context of spatial decision support systems to find the best policy to transparent and to support the design and evaluation of resource allocation alternatives. The authors show that SA is able to find close-to-optimal policies for high dimensionality cases while multi-criteria decision-making (MCDM) techniques run into numerical problems.

We provide below an overview of the components constricting the Simulated Annealing (SA) algorithm. In particular, one can divide the algorithm into six sections:

- A finite set S.
- A real-valued cost function J defined on S. Let $S^* \subset S$ be the set of global minima of the function J, assumed to be a proper subset of S.
- For each $i \in S$, a set $S(i) \subset S/\{i\}$ is the set of neighbors of i.
- For every i, a collection of positive coefficients $q_{i,j}$ such that $j \in S(i)$ iff $i \in S(j)$.
- A monotonically decreasing function $T : \mathbb{N} \to \mathbb{R}^+$, called the *temperature function* such that at some step t the system's temperature is defined by $T(t)$.
- An initial condition (state) $x(0) \in S$.

Given the above elements, the SA algorithm consists of a discrete-time inhomogeneous Markov chain $x(t)$. The SA algorithm works as follows. If the current state $x(t)$ is equal to i, choose a neighbor j of i at random; the probability that particular $j \in S(i)$ is selected is equal to q_{ij}. Once j is chosen, the next state $x(t+1)$ is determined based on the following logic. IF $J(j) \leq J(i)$ then $X(t+1) = j$. If this condition does not satisfies, $x(t+1) = j$ with probability

$$e^{-\frac{(J(j)-J(i))}{T(t)}}$$

and $x(t+1) = i$ otherwise.

3 Solution

Our solution is based on the SA algorithm which has several customization's. First, we define the state space S to be a discrete-continuous space in which the discrete part is the types of parametric distributions (D) and the continuous is the union of all the parameters of these parametric distributions. Formally, $S := D \times \mathbb{R}^{\Sigma_{d \in D}(p(d))}$ where $p(d)$ is a functional that gets a parametric distribution function and returns the number of parameters it uses. Second, the cost function J is defined to be the mean squared error between the theoretical distribution parmaters and the empirical data parameters performed on a given state $i \in S$ and the feature's values. Third, the neighbors of a state $i \in S$ (e.g., $S(i)$) are defined to be all the other parametric distributions with the same parameter values. For the edge cases where more parameters are used in the neighbor state (j) than the current state (i), they can obtain any value. In a similar manner, in the case where fewer parameters are used in the neighbor state (j) than the current state (i), they can be any subset of the parameters of i. Forth, the q_{ij} are assumed to be uniformly distributed between all $j \in S(i)$. Finally, the initial condition $x(o)$ is chosen at random. The algorithms stop at the t_f iteration, such that t_f satisfies $T(t_f) = \epsilon$, where $\epsilon > 0$ is a real value as small as needed which indicates the minimal meaningful probability value to transform between two states i and j.

4 Experimental Evaluation

In order to evaluate the performance of the proposed approach, we computed both the fitting of the distribution to a known original distribution source and the computation time. First, we describe the experiment's setup including the comparison method and the *in silico* generated data. Second, we compare our proposed approach to the leading brute force with maximum likelihood.

4.1 Setup

Since each fitting method can result in an error while approximating the original distribution that generated real data, we decided to randomly pick distributions, sample them in order to obtain a dataset. That way, we store as a meta-data the original distribution of each feature in the dataset, allowing, later on, to accurately evaluate and compare the performance of each approach.

In practice, we generate 50 datasets with a random number of rows and columns ranging between [50, 5000] and [5, 15], respectively. For each column, we chose in random (with a uniform distribution) one distribution and its parameters.

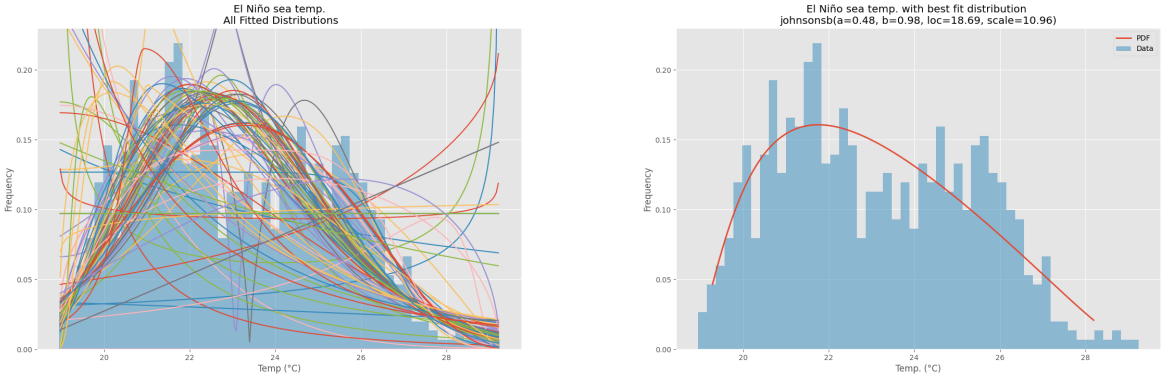
For the proposed approach, we find the optimal hyperparameters by conducting a grid search on the average fitting score over all datasets. The implementation is written in Python3.7. On the other hand, the *naive* approach of the brute force with maximum likelihood is implemented in C++ and wrapped with a Python3

interface using the Scipy [6] library. We constrain the search space of the naive approach to the uniform and normal distributions (rather than the 123 available distributions) to fairly compare between the two approaches.

Of note, all the experiments were conducted on a PC with the Windows 10 professional operation system with Intel i7 (generation 8) and 8 gigabit RAM. At the time of the experiments, only the operating system and the test run of the device in order to avoid computation time error due to other processes running in parallel.

4.2 Results

Before presenting the results, we illustrate that the fitting of distribution of data is a hard task, as shown in Fig. 1 where the *naive* approach is used. In particular, Fig. 1a shows the fitting of 108 continuous distributions on the a dataset representing the sea water temperature during "El Nino" phenomenon, revealing that while a large number of distributions not even close to represent the data, there are multiple candidates with similar performance and picking the best one can be tricky. Indeed, as shown in Fig. 1b, the best distribution is shown but one can see that it only roughly representing the data, missing the gap in 23.5 (C) .



(a) All distributions.

(b) Best distributions.

Figure 1: A fitting of 108 continuous distribution (available in the Scipy library) on the *El Nino* see temperature dataset using the *naive* approach.

The results of the experiment are shown in Table 1 as mean ± standard deviation of the $n = 50$ datasets.

Dataset	**Naive time [s]**	**SA time [s]**	**Naive performance [1]**	**SA performance [1]**
Uniform	0.000228 ± 0.005491	0.000141 ± 0.010347	0.000000 ± 0.000000	0.017732 ± 0.020400
Normal	0.000110 ± 0.004287	0.000123 ± 0.005396	0.031791 ± 0.073414	3.991825 ± 2.509279
Combined	0.000226 ± 0.003748	0.000196 ± 0.004598	0.021556 ± 0.104323	1.834477 ± 2.653128

Table 1: A summary of the computation time of the *naive* approach and the proposed SA approach. The results are shown as mean ± standard deviation.

5 Conclusion

In this work, we proposed a SA approach for fitting parametric distributions for tabular data. We compared the proposed approach to the commonly used brute force with maximum likelihood (e.g., naive) approach, revealing a similar performance in in the computation time metric, with both methods finishing within fractions of a second. However, the Naive approach outperformed the SA algorithm accuracy of the fitting both for

Uniform and Normal distribution type data-sets, as shown in Table 1. We believe that one reason of the worse performance of the SA approach is the discrete nature of the search domain, that combines discrete distribution types with continues parameter fitting. Furthermore, we encountered a major challenge in choosing the suitable *"goodness-of-fit"* function that represent the energy in SA. Initially, we chose the Kolmogorov-Smirnov test, that can be used as a *"goodness-of-fit* test, comparing the equality of a data sample to a presumed theoretical probability distribution. However, the p-value returned by test was not sensitive enough, with value equals exactly zero, unless the distribution parameters were highly fitting the data. The test statistic D, representing the largest difference between the cumulative distribution functions of the theoretical distribution function and the data sample, was some what better however the SA still did not converge to the correct distribution for most conducted runs. As a result, we had to switch to the SSE method comparing the function parameters to the sample parameters directly. In addition the naive approach would be a better choice as it theoretically guaranteed to obtain the best distribution while the SA approach does not due to its stochastic nature.

As future work, one can extend the proposed experiment by taking into consideration a wider range of both continuous and discrete distributions. In this case, an implementation of both approaches in the same programming language would result in a more accurate computing time comparison. Moreover, in the proposed work we used the standard temperature function which can be altered to a more appropriate function that takes advantage of the space of distributions.

References

[1] L. C. Molina, L. Belanche, and A. Nebot. "Feature selection algorithms: a survey and experimental evaluation". In: *2002 IEEE International Conference on Data Mining*. 2002, pp. 306–313.

[2] S.K. Nayar, S. Baker, and H. Murase. "Parametric feature detection". In: *Proceedings CVPR IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 1996, pp. 471–477. DOI: 10.1109/CVPR.1996.517114.

[3] G. Chandrashekar and F. Sahin. "A survey on feature selection methods". In: *Computers Electrical Engineering* 40.1 (2014), pp. 16–28.

[4] S. Nadrajah. "A generalized normal distribution". In: *Journal of Applied Statistics* 32.7 (2005), pp. 685–694. DOI: 10.1080/02664760500079464.

[5] L. G. M. Baas Becking and E. F. Drion. "On the origin of frequency distributions in biology". In: *Acta Biotheoretica* 1 (1936), pp. 133–150. DOI: 10.1007/BF02147636.

[6] P. Virtanen, R. Gommers, and T.E. et al. Oliphant. "SciPy 1.0: fundamental algorithms for scientific computing in Python". In: *Nat Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

[7] L. Yang and M. Wang. "Sample-Optimal Parametric Q-Learning Using Linearly Additive Features". In: *Proceedings of the 36th International Conference on Machine Learning* 97 (2019), pp. 6995–7004.

[8] D. R. Cox and C. Kartsonaki. "The fitting of complex parametric models". In: *Proceedings of the 36th International Conference on Machine Learning* 99.3 (2012), pp. 741–747.

[9] M. A. Perkowski and S. Grygiel. "A SURVEY OF LITERATURE ON FUNCTION DECOMPOSITION VERSION". In: *Portland State University* (1995), pp. 53–59.

[10] T. J. Ypma. "Historical Development of the Newton–Raphson Method". In: *SIAM Rev* 37.4 (1995), pp. 531–551.

[11] R. Davidson and D. Harel. “Drawing graphs nicely using simulated annealing”. In: *ACM Transactions on Graphics* 15.4 (1996), pp. 301–331.

[12] J. C. J. H. Aerts and G. B. M. Heuvelink. “Using simulated annealing for resource allocation”. In: *ACM Transactions on Graphics* 16.6 (2002), pp. 571–587.