Inverted Exponential Distribution

Zachary Weaver

April 6, 2024

Contents

1 Introduction

A less frequently encountered distribution of data that arises naturally is exponentially rising data, and as such, there isn't a well-known parametric distribution that describes this type of data. In this paper, we derive a parametric distribution to fit exponentially rising data for any closed continuous interval with finite Lebesgue measure on the real line and compare it to a kernel density estimation to show that kernel density estimation doesn't do as well to capture the behavior of the underlying distribution. This contrasts the standard exponential distribution as we are attempting to model exponential rise rather than decay starting at an arbitrary point in the real line.

2 Formulating the Probability Density Function

We now start with the derivation. Consider the following figure of sample data that we want to build a parametric distribution for.

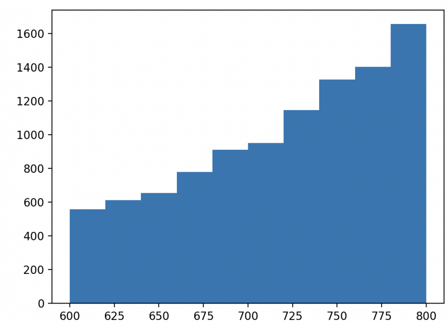

Figure 1: An example of exponentially rising data on the interval [600, 800]

There can be varying shapes to this distribution, some with sharper or softer rises. To begin, let's take a look at the standard exponential distribution.

$$f(x; \lambda) = \lambda e^{-\lambda x} \tag{1}$$

This is the classic well-known exponential decay model. However, we need to invert this to model exponentially rising data. We will flip λ in the exponent to be positive.

$$f(x; \lambda) = \lambda e^{\lambda x} \tag{2}$$

As is, this distribution cannot move anywhere. Thus, we introduce a location parameter, θ, to do so.

$$f(x; \lambda, \theta) = \lambda e^{\lambda(x-\theta)} \tag{3}$$

To turn this into a proper probability density function, we need to define this such that the integral over the domain is one. By definition, this data rises over a closed, finite Lebesgue-measurable interval, meaning it's defined over some interval $[a, b]$ for $a, b \in \mathbb{R}$ and $b > a$. In this case, our lower bound is set by θ. We integrate this function to obtain the normalizing factor.

$$\int_\theta^b f(x; \lambda, \theta) dx = \int_\theta^b \lambda e^{\lambda(x-\theta)} dx \tag{4}$$

$$= \lambda e^{-\lambda\theta} \int_\theta^b e^{\lambda x} dx \tag{5}$$

$$= \lambda e^{-\lambda\theta} \left[\frac{1}{\lambda} e^{\lambda x} \right]_{\theta}^{b} \tag{6}$$

$$= \lambda e^{-\lambda\theta} \left[\frac{1}{\lambda} e^{\lambda b} - \frac{1}{\lambda} e^{\lambda\theta} \right] \tag{7}$$

$$= \lambda e^{-\lambda\theta} \left[\frac{e^{\lambda b} - e^{\lambda\theta}}{\lambda} \right] \tag{8}$$

$$= \lambda e^{-\lambda\theta} \left[\frac{e^{\lambda b} - e^{\lambda\theta}}{\lambda} \right] \tag{9}$$

$$= e^{-\lambda\theta} \left[e^{\lambda b} - e^{\lambda\theta} \right] \tag{10}$$

$$= e^{\lambda(b-\theta)} - e^{\lambda(\theta-\theta)} \tag{11}$$

$$= e^{\lambda(b-\theta)} - 1 \tag{12}$$

We can now divide our original function by this normalizing factor to convert it to a proper probability density function such that, once integrated over $[\theta, b]$, will be equal to one.

$$f(x; \lambda, \theta, b) = \frac{\lambda e^{\lambda(x-\theta)}}{e^{\lambda(b-\theta)} - 1} \tag{13}$$

We define any $x \notin [\theta, b]$ to be 0.

3 Parameter Estimation

According to equation 13, there are three total parameters to estimate: λ, θ and b. Notice that θ is the lower bound of the domain and b is the upper bound. These can be naively estimated as the sample minimum and sample maximum respectively. This could be sensitive to outliers and perhaps treated better, but this section will mainly focus on estimating λ, the shape parameter of this distribution.

We will approach this with maximum likelihood. Consider the joint probability density function which we'll call the likelihood.

$$L(x; \lambda, \theta, b) = \prod_{i=1}^{n} f(x_i; \lambda, \theta, b) \tag{14}$$

$$= \prod_{i=1}^{n} \frac{\lambda e^{\lambda(x_i-\theta)}}{e^{\lambda(b-\theta)} - 1} \tag{15}$$

Due to the complexities of taking derivatives of this, a common trick is to take the logarithm of this product as monotonically increasing functions (such as a logarithm) preserve extrema. It's easy to see this as if $f(x) < f(y)$ then $\log f(x) < \log f(y)$ since the logarithm is monotonically increasing. This implies that a local minimum/maximum is preserved under logarithm. For the remainder of this paper, we notate log as the natural logarithm. That is, a logarithm with base e.

$$\log(L(x;\lambda,\theta,b)) = \log\left(\prod_{i=1}^{n} \frac{\lambda e^{\lambda(x_i-\theta)}}{e^{\lambda(b-\theta)}-1}\right) = \sum_{i=1}^{n} \log\left(\frac{\lambda e^{\lambda(x_i-\theta)}}{e^{\lambda(b-\theta)}-1}\right) \tag{16}$$

Due to properties of logarithms, the logarithm of a product can be expressed as the sum of individual logarithms. We will also use the fact that the logarithm of a ratio, $\log(\frac{x}{y})$, can be expressed as the difference of logarithms, $\log(x) - \log(y)$.

$$= \sum_{i=1}^{n} \log\left(\lambda e^{\lambda(x_i-\theta)}\right) - \sum_{i=1}^{n} \log\left(e^{\lambda(b-\theta)}-1\right) \tag{17}$$

$$= \sum_{i=1}^{n} \log(\lambda) + \lambda \sum_{i=1}^{n} (x_i-\theta) - \sum_{i=1}^{n} \log\left(e^{\lambda(b-\theta)}-1\right) \tag{18}$$

Note that the first and third terms are just constants, so $\sum_{i=1}^{n} c = nc$.

$$= n\log(\lambda) + \lambda \sum_{i=1}^{n} (x_i-\theta) - n\log\left(e^{\lambda(b-\theta)}-1\right) \tag{19}$$

With this, we can define our gradient by taking the partial derivative with respect to λ, our parameter of interest.

$$\frac{\partial}{\partial\lambda}\log(L(x;\lambda,\theta,b)) = \frac{n}{\lambda} + \sum_{i=1}^{n} (x_i-\theta) - \frac{n(b-\theta)e^{\lambda(b-\theta)}}{e^{\lambda(b-\theta)}-1} \tag{20}$$

Finally, we can achieve our estimate by finding the root of this gradient.

$$\hat{\lambda} = \frac{\partial}{\partial\lambda}\log(L(x;\lambda,\theta,b)) \overset{\text{set}}{=} 0 \tag{21}$$

As of this writing, no analytical solution has been found or proven to exist or not exist, but can be numerically approximated.

4 Comparison to Kernel Density Estimation

Kernel density estimation is a popular approach to estimating complex distributions where the parametric form is either unknown or difficult to obtain. Here, we compare kernel density estimation against estimating the parameters for the inverted exponential distribution on data generated by a few known theoretical inverted exponential distributions by varying shape parameters: $f(x; 0.001, 300, 900)$, $f(x; 0.003, 300, 900)$, $f(x; 0.005, 300, 900)$, $f(x; 0.007, 300, 900)$

and $f(x; 0.01, 300, 900)$. Note that due to how the distribution is defined, the values for λ will always be relatively small, otherwise overflows will occur, so we test the range $\lambda \in [0.001, 0.01]$ and should reflect what most "real world" data should follow (higher values of λ will cause the tail end to spike pretty significantly.)

For each experiment, we will sample 30 random points from the given theoretical distribution and fit both a kernel density estimate and estimate the parameters for the inverted exponential and use the symmetric form of KL-Divergence to evaluate which distribution "fits" better on 1000 evenly-spaced points (using numpy [2]) in the interval $[300, 900]$. We will repeat this experiment 250 times and measure the proportion of times that KDE or the estimated inverse exponential was a closer fit based on which KL-Divergence value was smaller as well as measure the average improvement for each setting.

For kernel density estimation, we will use Gaussian kernels with the bandwidth estimated by Scott's rule [1].

We will denote the kernel density estimate as $\hat{K}(x)$ and the estimated inverted exponential as $\hat{f}(x)$.

Given a set of evenly-spaced points $x_i \in [300, 900]$, we define the symmetric KL-Divergence as follows.

$$SKL(\hat{f}) := \sum_i \hat{f}(x_i)\log\left(\frac{\hat{f}(x_i)}{f(x_i)}\right) + f(x_i)\log\left(\frac{f(x_i)}{\hat{f}(x_i)}\right) \tag{22}$$

$$SKL(\hat{K}) := \sum_i \hat{K}(x_i)\log\left(\frac{\hat{K}(x_i)}{f(x_i)}\right) + f(x_i)\log\left(\frac{f(x_i)}{\hat{K}(x_i)}\right) \tag{23}$$

Whichever value is smaller is a "better" fit.

Below are visualizations of the theoretical distribution at different parameter values.

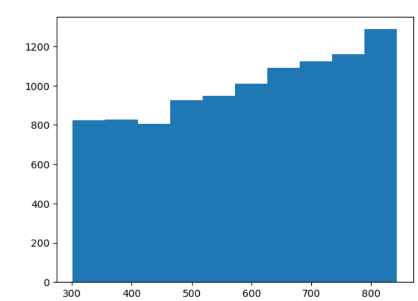

Figure 2: The theoretical inverse exponential distribution f(x; 0.001, 300, 900)

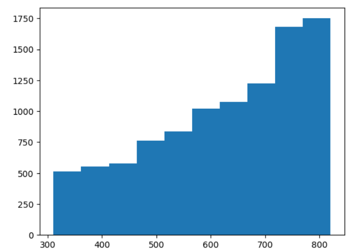

Figure 3: The theoretical inverse exponential distribution f(x; 0.003, 300, 900)

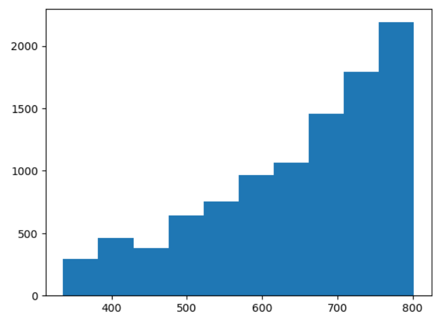

Figure 4: The theoretical inverse exponential distribution f(x; 0.005, 300, 900)

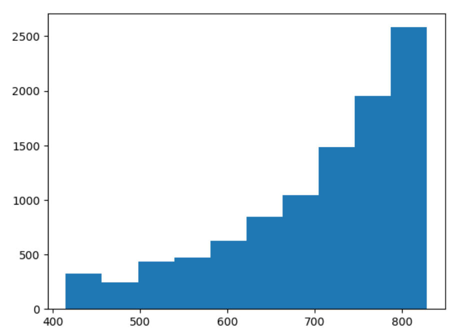

Figure 5: The theoretical inverse exponential distribution f(x; 0.007, 300, 900)

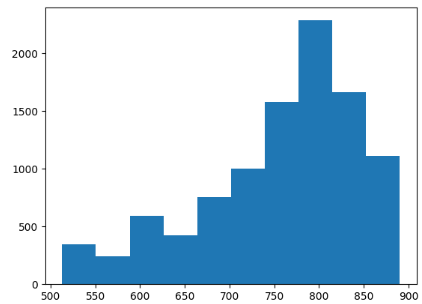

Figure 6: The theoretical inverse exponential distribution f(x; 0.01, 300, 900) - this parameter value is relatively high so it starts exhibiting odd behavior

After sampling 30 observations and evaluating the divergence metrics for 250 iterations across the various shape parameters, these are the results:

N	λ	# $\hat{K}(x)$	# $\hat{f}(x)$	$\hat{f}(x)$ %	Avg. SKL($\hat{f}$)	Avg. SKL($\hat{K}$)	$\hat{f}(x)$ % Improvement
30	0.001	45	205	82%	0.156	0.264	40.74%
30	0.003	32	218	87.2%	0.360	0.610	40.94%
30	0.005	50	200	80%	1.023	1.357	24.55%
30	0.007	38	212	84.8%	0.895	1.288	30.51%
30	0.01	109	141	56.4%	1.449	1.159	-25.21%

Table 1: A table of density estimation methods and the count of iterations where they had a smaller divergence metric (higher count is better) along with the average KL-Divergence score (lower is better.) NOTE: the % improvement score is calculated with the un-rounded average values.

As we can see in the table, this parametric estimation works pretty well in most cases but there is an apparent diminishing return. In particular, the estimates start to weaken somewhere in $\lambda \in (0.007, 0.01]$. Even though we technically had slightly more cases where the parametric estimate was "better", on average it performed 25.21% worse probably due to some particularly bad samples that were drawn that are unreliable with a sample size of 30 for this value of λ.

To see the impact of sample size, the simulation was re-ran, but instead of 30 samples, we now draw 150 samples and run the same 250 experiments. Below are the results.

N	λ	# $\hat{K}(x)$	# $\hat{f}(x)$	$\hat{f}(x)$ %	Avg. SKL($\hat{f}$)	Avg. SKL($\hat{K}$)	$\hat{f}(x)$ % Improvement
150	0.001	0	250	100%	0.052	0.235	78.02%
150	0.003	0	250	100%	0.214	0.673	67.27%
150	0.005	0	250	100%	0.670	1.669	59.90%
150	0.007	1	249	99.6%	0.626	1.411	55.69%
150	0.01	7	243	97.2%	0.742	1.129	34.23%

Table 2: The same experiment as before but using 150 samples instead of 30 to measure the impact of sample size.

There is a very clear impact of sample size, the performance has significantly improved with more samples and, looking at the average divergence scores, notice that the average scores for kernel density didn't change much in contrast to the inverted exponential - it seems KDE hit a limit of performance pretty quickly whereas inverted exponential was able to extract more information. It's unknown at this time what the "performance cap" in terms of sample size is for inverted exponential. It's clear from here that more samples will be required to reliably model larger values of λ.

5 Conclusion

We've identified a candidate parametric probability distribution to model a special case of data that happens to follow an exponential rise over an arbitrary continuous interval. We have derived the gradient that can be optimized and compared the performance of this parametrization against the popular kernel density estimate using various values of λ at sample sizes of 30 and 150. Even at 30 samples, the model reliably outperforms kernel density estimation for values of $\lambda <= 0.007$ but diminished somewhere in $\lambda \in (0.007, 0.01]$.

The performance of inverted exponential drastically improved when moving from 30 to 150 samples and was able to more reliably predict the larger values of $\lambda > 0.007$. Also observed when increasing sample size, kernel density didn't see any performance gains in terms of the average divergence score - it hit its performance cap relatively quickly, but the inverted exponential was able to extract more information with the increased samples and significantly outperformed KDE in every case tested.

It was also observed by visualization the distribution starts to behave oddly starting around $\lambda >= 0.01$ but may not really occur in practice since one of the other smaller values of λ should sufficiently capture the shape.

6 Implementation

A Python implementation was created to support fitting, sampling, integrating and computing other statistical properties with the help of SciPy [3] as a back-end. The package is up on PyPi under the name invexpo (https://pypi.org/project/invexpo/) with the source code located at the following GitHub repository: https://github.com/Kiyoshika/inverse-exponential

The code used to run the simulation (section 4) is also provided in the repository linked above if you want to audit the results, reproduce or further the experimentation.

References

[1] D.W. Scott. *Multivariate Density Estimation: Theory, Practice, and Visualization*. John Wiley & Sons, 1992. ISBN: 9780471547709.

[2] Charles R. Harris et al. "Array programming with NumPy". In: *Nature* 585 (2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2.

[3] Pauli Virtanen et al. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2. URL: https://doi.org/10.1038/s41592-019-0686-2.