Fitting absolutely monotonic functions

Juan Casanova

December 22, 2025

This is the abstract.

Contents

1 Introduction

This paper has two purposes. First, to present a method for fitting absolutely monotonic functions to data. Second, to discuss in this context three areas of mathematics research that, surprisingly, are rarely discussed together. We summarize the relevant similarities and connections of each of them with the aforementioned method, and consequently between them, as well as how this relates to potential applications of this work and existing literature.

1.1 Absolutely monotonic functions

A smooth function $f : [a, b] \to \mathbb{R}$ (where b can be $+\infty$) is *absolutely monotonic* if all of its derivatives are non-negative at every point in the interval. A simple example of an absolutely monotonic function is the exponential e^x, and in fact in a very significant sense, it can be shown that all absolutely monotonic functions are essentially sums of exponentials, a key idea supporting this work. More accurately, in his classic work [Bernstein, 1929], among other things, Bernstein characterized all absolutely monotonic functions as weighted averages of exponential functions with positive exponents.

Theorem 1.1 ([Bernstein, 1929])**.** *Let $f : [a, b] \to \mathbb{R}$ (where a and b are both finite) be an absolutely monotonic function. Then, there is a non-negative finite Borel measure μ such that*

$$f(x) = \int_0^\infty e^{tx} d\mu(t) \tag{1}$$

for every $x \in [a, b]$.

There is a direct connection between absolutely monotonic functions and *completely monotonic functions*, which are smooth functions with alternating sign derivatives. Specifically, a function $f(x)$ is absolutely monotonic in $[a, b]$ if and only if the function $f(-x)$ is completely monotonic in $[-b, -a]$. For example, e^{-x} is a completely monotonic function, and similarly, Bernstein's theorem, when applied to completely monotonic functions, characterizes them as weighted averages of exponential functions with negative exponents (the Laplace transform of the non-negative measure μ).

In informal terms, one can describe the function e^x to be absolutely monotonic *towards the right* and completely monotonic *towards the left*, and the absolutely monotonic or completely monotonic views of functions are more related with what side of the function we are focusing on, and the corresponding properties that they have on them.

For various significant reasons, there has been substantially more work on completely monotonic functions than on absolutely monotonic functions. While it is tempting to argue that they are simply two sides of the same coin, their relation is limited by the nature of the result we are focusing on. More precisely, a lot of the work on completely monotonic functions considers them in the half-line $[0, \infty)$, which when mirrored results in the interval $(-\infty, 0]$ for absolutely monotonic functions. However, the interesting monotonic behaviour of absolutely monotonic functions is most relevant in intervals on the right side of the real line. Similarly, completely monotonic

functions involve small absolute values and controlled decay, whereas absolutely monotonic functions involve very large absolute values and unfettered growth. This means that the types of interesting behaviours and challenges in each of these two sides of the coin can be quite different in practice.

Absolutely monotonic and completely monotonic functions are particular cases of exponential sums. When both positive, negative, and complex exponents are combined, more general types of functions appear, including periodic functions and variations on these.

Absolutely monotonic functions form a *convex cone*. In particular, weighted sums of absolutely monotonic functions are absolutely monotonic.

Lemma 1.1. *If f, g are absolutely monotonic functions and $\alpha, \beta \in [0, +\infty)$, then $\alpha f + \beta g$ is an absolutely monotonic function.*

The proof is trivial by looking at the definition of absolute monotonicity in terms of non-negative derivatives and linearity of derivation.

This can naturally be extended to infinite sums of functions (of which the Bernstein theorem presents a particular case) over non-negative measures, but we will not use this result in this work so we skip the proof of this slightly more general result. Other than Bernstein's theorem, we will limit ourselves to finite sums of absolutely monotonic functions.

We do not concern ourselves too much with further theoretical properties of absolutely monotonic or completely monotonic functions in this work, but for the interested reader, [Koumandos, 2014] offers a modern summary of theoretical results and additional related families of functions, such as logarithmically completely monotonic functions, Bernstein functions, and Stieltjes functions.

[Rajba, 2011] is a particularly interesting piece of work partially bridging the gap between methods focused on convexity and absolute monotonicity. It presents an integral representation of *n-convex* functions, which are functions where a certain number of derivatives are positive. In other words, an absolutely monotonic function is a function that is n-convex for every n. The mathematics in this work and the representation it offers for n-convex functions are strongly reminiscent of the Bernstein theorem. In particular, there is a direct relationship between the terms in this work and the Taylor series of the exponential function, indicating that this integral representation is using "limited exponential functions" consisting of partial Taylor series of exponential functions, up to the derivative that is required. It is surprising,

then, that the work does not mention the Bernstein theorem or the words "absolutely monotonic" or "completely monotonic" whatsoever, or cite any work in this subject. More precisely, theorem 2.1 in [Rajba, 2011] states that any n-convex function f with regular derivatives in (a, b) is such that:

$$f(x) = \int_{(a,\epsilon]} (-1)^{n+1} \frac{[-(x-u)]_+^n}{n!} dg_{(n)-}(u) + \int_{[\epsilon,b)} \frac{(x-u)_+^n}{n!} dg_{(n)+}(u) + Q(x) \tag{2}$$

(with certain constraints on the terms in this equation).

Informally, one can draw a clear connection between this and absolutely monotonic and completely monotonic functions when we notice that $\frac{(x-u)_+^n}{n!}$ is the truncated Taylor series for e^x up to n, and therefore this equation is essentially separating the n-convex function f into a completely monotonic part (to degree n), an absolutely monotonic part (to degree n), and a remainder term $Q(x)$ with lesser degree.

While we do not use them in this paper, this work can be relevant in extending some of the notions here to less constrained cases like n-convex functions, or to better understand the methods and their limitations.

1.2 Fitting exponential sums

There is a large body of work on fitting exponential sums to data. A lot of this work is related to signal analysis and therefore is concerned with general exponential sums, including positive, negative, and complex terms; though practical work is often limited to discrete sums.

One of the most significant families of methods for fitting exponential sums are the so-called Prony methods. Prony methods are spectral methods that use the relation between exponential sums, uniform sampling, and difference equations that enable a certain type of frequency analysis using linear algebra. [Keller and Plonka, 2021] offers a modern survey of Prony methods, focusing on common principles, such that they are linear methods on the space of operators on functions, and that they rely on sampling at specific distances to capture periodic behaviours in the signals. A lot of work on Prony methods is related to the optimization of the linear algebra aspects of the algorithms.

Prony methods are more general in its function space, present more constraints to the data, and focus more on frequency analysis of a signal than on good numerical fitting of a real function. Their goal is the analysis of sig-

nals, often assumed to be in the $[0, +\infty)$ half-line, rather than the fitting of observed real data on the absolutely monotonic side of the function. Therefore, they are not particularly well suited for fitting absolutely monotonic functions, even if they could theoretically be applied. It could be interesting to apply some of these methods to examples on absolutely monotonic functions in bounded intervals by mirroring them and using Prony methods, to compare with the results in this paper, but we have decided not to pursue this avenue further.

Other methods have been explored for fitting exponential sums over time. [Kammler, 1976] presents a theoretically focused and exploratory analysis of using Chebyshev polynomials to approximate completely monotonic functions. [Smith et al., 1976] discusses several potential methods, also with a focus on completely monotonic functions, with limited success, and with significant computational constraints that are not applicable in the current time. [Evans et al., 1980] uses a least squares algorithm to fit completely monotonic functions, but also suffers from being computationally obsolete. It does note, however, that with a least squares approach, numerical aspects become critical, such as initialization of values or behaviour of the parameter space. We will discuss these aspects later in this work. In a slightly more modern survey, [Holmström and Petersson, 2002] discusses a wide array of methods for fitting exponential sums, including Prony methods, least squares approaches, and many others. None of these pay significant attention to absolutely monotonic functions.

There is definitely room for further exploring whether some of these methods may be applied successfully with a focus on absolutely monotonic functions, but work would be required to adapt them. We believe that between the focus on complex exponent signals, completely monotonic functions, the differences in error measurements between fitting a slow decay function and a fast growth function, and the outdated computational considerations of some of the older pieces of work, making any of these methods work successfully on absolutely monotonic functions would not be straightforward. Moreover, the method presented in §2 uses a least squares approach with a specific choice of representation of exponentials that improves numerical behaviour of the optimization algorithm, resulting in a very simple and relatively fast algorithm that gives significantly satisfying results. This suggests that there is no need to painfully adapt methods designed for other problems when a relatively simple one exists for this problem. It would, however, be interesting to verify this with an actual comparison.

1.3 Fitting monotonic and convex functions

The methods discussed in §1.2 are parametric or semi-parametric, using a predetermined family of functions and fitting using parameterizations. This ensures certain properties of the resulting fit by construction (complete monotonicity, periodic behaviour, etc). Another family of work focuses instead on fitting the data accurately while imposing the desired properties in less constructive ways. Specifically, we are talking about the wide range of work on fitting monotonic or convex functions in general.

Absolute monotonicity is a particular case of convexity, which is a particular case of monotonicity. Absolute monotonicity is a stronger constraint. Two important questions deserve to be explored. First, can methods to fit monotone or convex functions be extended to absolute monotonicity constraints? Second, since absolutely monotonic functions are convex, can methods constrainted to absolutely monotonic functions be used when the target constraint is weaker (convexity)?

All or most of the work in this area expresses the fit as an optimization problem (e.g. linear programming), that minimizes the error, and uses additional constraints on the optimization problem and/or steps in the algorithm to ensure the monotonicity/convexity of the result. They are all heavily focused on local properties of the functions and how to encode the constraints in this way, e.g. by looking at properties of derivatives/gradients/jacobians/hessian matrices.

We found two major families of research. First, work on imposing monotonicity constraints to splines. Second, fitting convex functions in a multivariate context.

A *spline* is a smooth piecewise polynomial. Spline interpolation is widely used because it is simple, very efficient, and produces smooth satisfying results for interpolation. However, while splines are smooth by design, imposing monotonicity in them is not easy. Work like [He and Shi, 1998, Zhang, 2004, Nagahara and Martin, 2013, Lu, 2015] considers a candidate spline interpolation, and defines a linear programming problem that minimizes its error with respect to the data. It then introduces the monotonicity constraint as additional constraints on the linear program. The results are satisfying but limited, because imposing monotonicity on splines can severely affect their ability to adequately approximate the data, or the constraints may not be strict and may result in functions that are not monotonic at every point. Moreover, extending this to absolutely monotonic functions seems extremely

hard, since new ways of representing the constraints on all derivatives of absolutely monotonic functions in a linear program would need to be devised, without significantly affecting the computational and approximation properties of the resulting algorithm.

Work like [Hannah and Dunson, 2013] considers the more general problem of fitting convex multivariate functions to data. Particularly due to the multivariate nature, this problem is significantly more complicated. Various techniques are used to balance the convexity constraints with the goal of a good fit to the data. For example, [Lachand-Robert and Oudet, 2005] uses a projection step as part of a gradient descent algorithm to ensure the solutions are convex, while [Aguilera and Morin, 2008] focuses on finding local characterizations of convexity and discretization of the hessian matrix of the target function, to ensure the convexity of the solution. These solutions are significantly computationally expensive, and struggle with balancing difficult constraints in a large solution space, due to their multivariate nature. Moreover, while convex but not absolutely monotonic functions are more rare, the application space is still more limited than our target problem.

It might be worth comparing the performance, in computational cost and goodness of fit, of these solutions to our proposed method for the problems in which they overlap (monotonic and convex functions). We expect these will be more cost for worse fit.

1.4 Why absolutely monotonic functions

Monotonic and convex functions are much more general than absolutely monotonic functions. Similarly, exponential sums appear in signal and frequency analysis, and completely monotonic functions can be relevant in economic modeling. These applications, and the lack of a comparatively as significant volume of applications for absolutely monotonic functions, explain the differences in the volume of work devoted to them. Why, then, do we want to fit absolutely monotonic functions? While this paper is focused on the method, we arrived at this problem from an applied setting. We wanted to use data to fit reward functions (for players in a video game). Absolute monotonicity is an extremely attractive constraint for a reward function, as it ensures not only that better results produce better rewards, but also that the returns increase, as it is often informally said, *exponentially*, or more accurately, absolutely monotonically. This means that the same comparative improvement in performance will always result in a larger improvement (and a faster increase in this improvement, and a faster increase on the speed of

this improvement, etc.) on the reward the higher the performance already is. In other words, small differences become much more noticeable near the top, which is very attractive for reward functions, for example in a competitive setting. Another way to look at this is that an absolutely monotonic function is a way to re-scale a performance value into a reward space that is "exponential", greatly accentuating differences at higher performance value while giving less relevance to differences at lower performance values.

Naturally, convex functions are often enough to represent this type of behaviour. While we have not thoroughly explored the economics literature in search for uses of convex functions, we conjecture that there might be some applications in economics in which convex functions are expected, that would benefit from using absolutely monotonic functions instead, by giving them stronger reward behaviours. Moreover, the method discussed in §2 is simple, fast, and successful enough that it might be used in place of other methods focused on convexity.

As a whole, we believe that any real-world modeling of a function that benefits from consistently accentuating differences at higher values, but fit this function using existing data, such as reward functions, could benefit from imposing an absolute monotonicity constraint on the result and using the fitting method described in this paper.

1.5 Summary

In the field of mathematical analysis there is a significant theoretical understanding of absolutely monotonic functions, completely monotonic functions, exponential sums in general, and other related families of functions. At the same time, several methods of fitting some families of exponential sums to data exist, coming primarily from the signal analysis world. This applied work often does not use the existing theoretical results to their full extent, and sometimes fails to recognize the connection between the different families of exponential sums and their properties. At the same time, another significant research field studies algorithms to fit monotonic and convex functions, without acknowledging their relation to absolutely monotonic functions, and therefore exponential sums, and therefore not attempting (semi-)parameteric approaches to this problem, that could benefit from easier to fulfill constraints, and better computational performance and fitness.

In the remainder of this paper we discuss our proposed, simple method for fitting absolutely monotonic functions that sits in the middle of all of this,

and that could help as an initial step towards briding these gaps. Regardless of this new method, we believe it would be in the best interest of researchers in each of these fields to explore the results in each other's fields and consider how they might improve their approaches or reconsider some assumptions or choices that they make.

2 Method

In this section we describe the method that we use to solve the following problem:

Problem 2.1. *Given a set of data points $\mathcal{D} = \{(x_i, y_i)\}$, we wish to fit an absolutely monotonic function f to minimize the error between the $\{y_i\}$ and $\{f(x_i)\}$.*

Note that we do not prescribe the measure of error in the general definition of the problem, as that can be arbitary. We also are not looking at the strict definition of *minimize* but rather a best effort approach to it. In general, this is the larger problem we are trying to address.

However, in this work we use a semi-parametric least squares optimization, which means that we are minimizing the mean squared error, and we use an iterative method to find a best effort solution within a prescribed family of functions. Further details are described in the remainder of this section. This is a consequence of our chosen method.

2.1 Domain

Since the data $\mathcal{D}$ is finite, there will be bounds on the $\{x_i\}$

$$\begin{aligned} a &= \min x_i \\ b &= \max x_i \end{aligned} \tag{3}$$

Therefore, we can safely constrain ourselves to absolutely monotonic functions $f : [a, b] \to f([a, b])]$. Moreover, we can reduce the problem to absolutely monotonic functions $\bar{f} : [0, 1] \to f([a, b])$ through a simple affine transformation. By transforming the $\{x_i\}$ into $\{\bar{x}_i = \frac{x_i - a}{b-a}\}$, if we have a solution $\bar{f} : [0, 1] \to f([a, b])$ for the problem associated to $\{\bar{x}_i\}$, we can define

$$f(x) = \bar{f}(\frac{x - a}{b - a}) \tag{4}$$

and we have:

$$\begin{aligned} f(a) &= \bar{f}(\tfrac{a-a}{b-a}) = \bar{f}(0) \\ f(b) &= \bar{f}(\tfrac{b-a}{b-a}) = \bar{f}(1) \end{aligned} \tag{5}$$

(correspondence of bounds)

$$f(x_i) = \bar{f}(\frac{x_i - a}{b-a}) = \bar{f}(\bar{x_i}) \tag{6}$$

(correspondence of data points)

$$f^{(n)}(x) = (\frac{1}{b-a})^n \bar{f}^{(n)}(\frac{x-a}{b-a}) \tag{7}$$

(f is absolutely monotonic if and only if $\bar{f}$ is absolutely monotonic, since $b \geq x \geq a$.

Therefore, for the remainder of this work, we only consider the problem of fitting absolutely monotonic functions $f : [0,1] \to f([0,1])$ with all data points such that $x_i \in [0,1]$.

2.2 Unit exponentials

One of the key novel insights of this work is the definition of what we call *unit exponentials*. Unit exponentials are rescalings of exponential functions with positive exponents designed to have more regular numerical behaviours that make them both easier to understand in the context of absolutely monotonic fitting, and improve the computational properties of the fitting algorithm.

Definition 2.1 (Unit exponentials)**.** *We define the parametric* unit exponentials *family of absolutely monotonic functions, with parameter $t > 0$:*

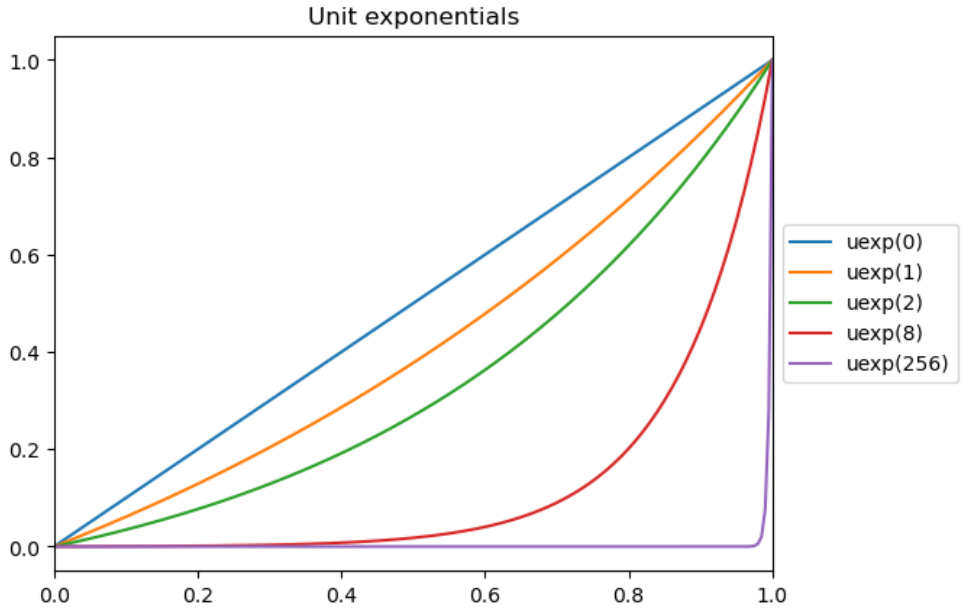
$$uexp_t(x) = \frac{e^{tx} - 1}{e^t - 1} \tag{8}$$

The most immediate advantage of unit exponentials is that they are all pinned to the $[0,1]$ interval:

$$\begin{aligned} \mathrm{uexp}_t(0) &= \tfrac{e^0-1}{e^t-1} = 0 \\ \mathrm{uexp}_t(1) &= \tfrac{e^t-1}{e^t-1} = 1 \end{aligned} \tag{9}$$

This makes them extremely convenient for the problem of fitting absolutely monotonic functions $f : [0,1] \to [0,1]$. We note that $\lim_{t \to 0} \mathrm{uexp}_t(x) = x$,

Figure 1: Some unit exponentials



and so we can smoothly extend the definition to $\text{uexp}_0(x) = x$

However, their attractiveness only begins here. One of the most interesting properties of unit expontentials is that their value for any $x \in [0, 1]$ always corresponds to the proportion of the total growth between 0 and 1 that *all* of its derivatives have manifested between 0 and x:

Theorem 2.1. *For every $t > 0$, every $n \geq 0$, and every $x \in [0, 1]$.*

$$\frac{uexp_t{}^{(n)}(x) - uexp_t{}^{(n)}(0)}{uexp_t{}^{(n)}(1) - uexp_t{}^{(n)}(0)} = uexp_t(x) \tag{10}$$

While unit exponentials always go between 0 and 1, their derivatives do not necessarily do so. They are always greater than 0 at 0 and grow to very high values near 1, with no bound for unbounded t. However, the theorem establishes a link between the function itself and how fast **all** of the derivatives grow, displaying a very significant and regular coordination among all derivatives for unit exponentials.

There are two useful lemmas that we will use to prove this theorem among other things.

Lemma 2.1. *For every $t > 0$ and $n > 0$,*

$$uexp_t{}^{(n)}(x) = \frac{t^n e^{tx}}{e^t - 1} \tag{11}$$

Proof. This is a simple inductive application of the derivation of $\text{uexp}_t(x) = \frac{e^{tx}-1}{e^t-1}$. □

Lemma 2.2. *For every $t > 0$ and $n > 0$,*

$$\begin{aligned} uexp_t{}^{(n)}(0) &= \tfrac{t^n}{e^t-1} \\ uexp_t{}^{(n)}(1) &= \tfrac{t^n e^t}{e^t-1} \\ uexp_t{}^{(n)}(1) - uexp_t{}^{(n)}(0) &= t^n \end{aligned} \tag{12}$$

Proof. The first two lines are a direct application of the previous lemma, but we will write the derivation of the last line explicitly, as it is the most interesting part of this lemma, providing a direct expression for the range of the n-th derivative of uexp_t depending cleanly on t.

$$\text{uexp}_t{}^{(n)}(1) - \text{uexp}_t{}^{(n)}(0) = \frac{t^n e^t}{e^t - 1} - \frac{t^n}{e^t - 1} = \frac{t^n(e^t - 1)}{e^t - 1} = t^n \tag{13}$$

□

Note that his last line also holds for $n = 0$, that is, it is also the case that $\text{uexp}_t(1) - \text{uexp}_t(0) = t^0 = 1$.

We are now in a position to prove the theorem.

(Proof of the theorem). We first note that for $n = 0$ the theorem is trivial since $\text{uexp}_t(0) = 0$ and $\text{uexp}_t(1) - \text{uexp}_t(0) = 1$, so it follows directly from the equation.

$$\frac{\text{uexp}_t(x) - \text{uexp}_t(0)}{\text{uexp}_t(1) - \text{uexp}_t(0)} = \frac{\text{uexp}_t(x) - 0}{1} = \text{uexp}_t(x) \tag{14}$$

For $n > 0$, we begin by simplifying the denominator to t^n as per the lemma,

$$\frac{\text{uexp}_t{}^{(n)}(x) - \text{uexp}_t{}^{(n)}(0)}{\text{uexp}_t{}^{(n)}(1) - \text{uexp}_t{}^{(n)}(0)} = \frac{\text{uexp}_t{}^{(n)}(x) - \text{uexp}_t{}^{(n)}(0)}{t^n} \tag{15}$$

We can then replace the two values in the numerator,

$$\frac{\text{uexp}_t{}^{(n)}(x) - \text{uexp}_t{}^{(n)}(0)}{t^n} = \frac{1}{t^n} \cdot (\frac{t^n e^{tx}}{e^t - 1} - \frac{t^n}{e^t - 1}) =$$
$$= \frac{1}{t^n} \cdot \frac{t^n(e^{tx} - 1)}{e^t - 1} = \frac{e^{tx} - 1}{e^t - 1} = \text{uexp}_t(x) \tag{16}$$

□

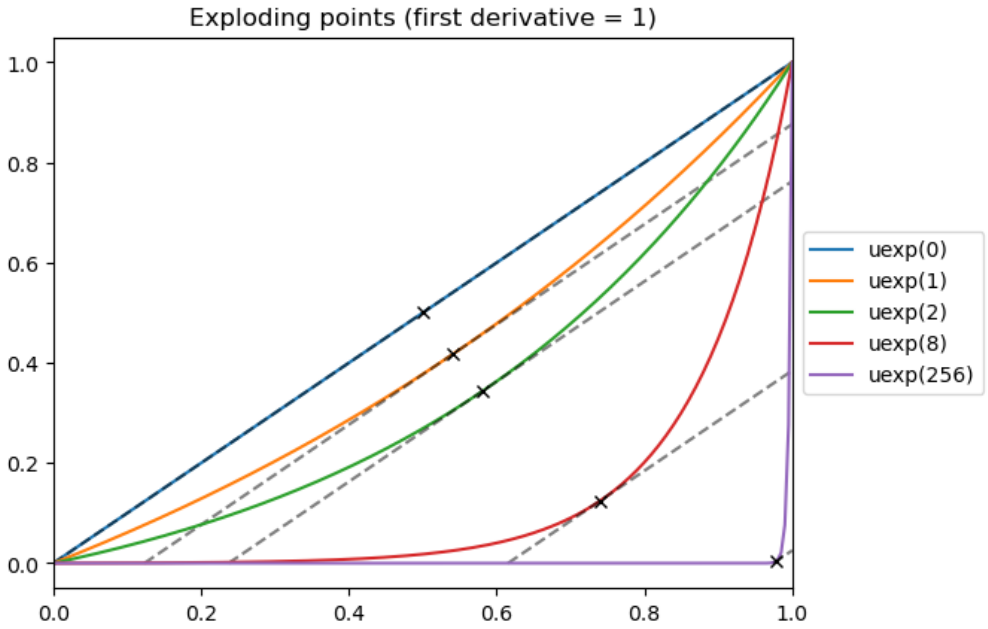
By the middle point theorem, for any $t > 0$, there will always be a point $x_1(t) \in [0, 1]$ where the first derivative of uexp_t will be exactly 1. We will call this the *exploding point* of uexp_t. All unit exponentials go underneath the straight line $\text{uexp}_0(x) = x$, with lower initial first derivative and higher final first derivative the higher the t. At some point, the first derivative will be exactly 1, which can be identified to be as the point in which the unit exponential "kicks off" and begins its exponential growth. The location of the exploding point with respect to t is interesting as it approximately separates the area where the unit exponential is flat and where it is steep. This point will always be $x_1(t) \geq 0.5$ for $t > 0$, and will get progressively closer to 1 as t increases. See figure 2 for a visualization of the exploding points of various unit exponentials.

It is, in fact, more interesting or useful to consider instead the distance between $x_1(t)$ and 1. Write $1 - \bar{x}_1(t) = x_1(t)$. We can calculate this value analytically:

$$\begin{aligned}
\text{uexp}_t{}'(x_1(t)) = \text{uexp}_t{}'(1 - \bar{x}_1(t)) = \frac{te^{t(1-\bar{x}_1(t))}}{e^t - 1} &= 1 \\
te^{t(1-\bar{x}_1(t))} &= e^t - 1 \\
\ln t + t(1 - \bar{x}_1(t)) &= \ln(e^t - 1) \\
1 - \bar{x}_1(t) &= \frac{\ln(e^t - 1) - \ln t}{t} \\
\bar{x}_1(t) &= \frac{t + \ln t - \ln(e^t - 1)}{t}
\end{aligned} \tag{17}$$

An interesting way to think about this gap between the exploding point and 1 is to see how it changes when we multiply t by a constant $k > 1$. Therefore, we can compare $\bar{x}_1(t)$ with $\bar{x}_1(kt)$. The latter will be smaller, as a larger t moves the exploding point closer to 1, but by how much? For large

Figure 2: Exploding points



enough t, multiplying t by $k > 1$ reduces the gap between the exploding point and 1 to approximately $1/k$. That is, $\frac{\bar{x}_1(t)}{\bar{x}_1(kt)} \simeq k$. This is very useful in the implementation of an iterative process to find a semiparametric fitting of an absolutely monotonic function as a sum of unit exponentials, as we can use a multiplying factor k to obtain meaningful variation in the parameters of the unit exponentials that are independent of the current values of t (see §2.3).

We can show this numerically by plotting $\frac{\bar{x}_1(t)}{\bar{x}_1(kt)}$ for various values of k. See figure 3. Note that the convergence is notoriously slow (logarithmic), but it stabilises relatively quickly in a relatively constant range. It can be proven to be exactly k in the limit (though this is not very useful because we will not use such large t).

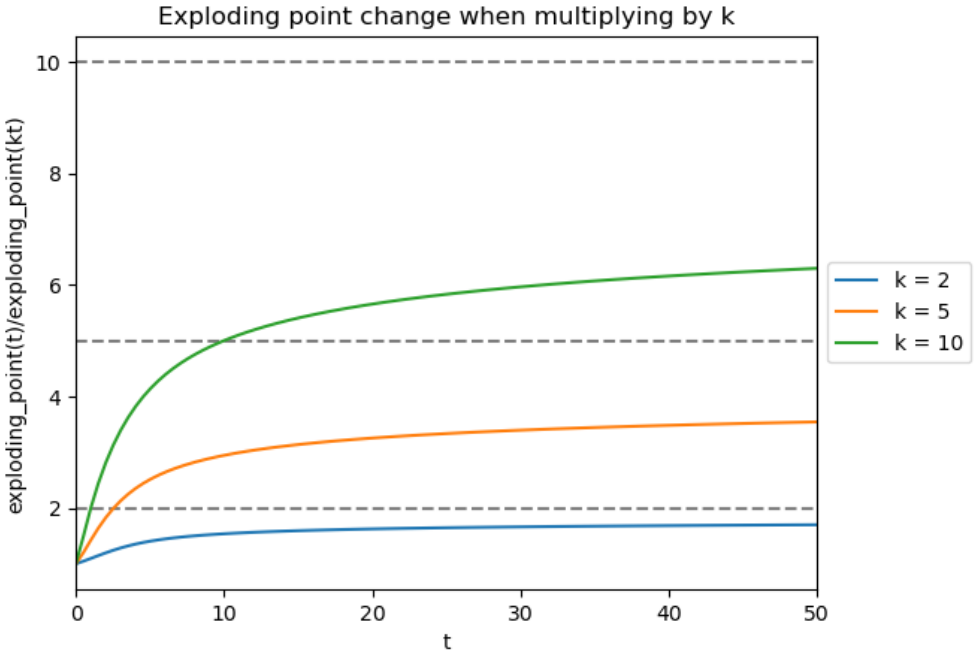
Theorem 2.2.

$$\lim_{t\to\infty} \frac{\bar{x}_1(t)}{\bar{x}_1(kt)} = k \tag{18}$$

Proof. First note that

$$t - \ln(e^t - 1) = t - \ln(e^t(1 - e^{-t})) = t - t - \ln(1 - e^{-t}) = -\ln(1 - e^{-t}) \tag{19}$$

Figure 3: Exploding point change when multiplying



With this in mind, simplify the original quotient.

$$\frac{\bar{x}_1(t)}{\bar{x}_1(kt)} = \frac{t + \ln t - \ln(e^t - 1)}{kt + \ln kt - \ln(e^{kt} - 1)} \cdot \frac{kt}{t} = k \cdot \frac{\ln t - \ln(1 - e^{-t})}{\ln kt - \ln(1 - e^{-kt})} = k \cdot \frac{\ln t - \ln(1 - e^{-t})}{\ln k + \ln t - \ln(1 - e^{-kt})} \tag{20}$$

The latter terms go to 0 as $t \to \infty$, and so we have

$$\lim_{t\to\infty} \frac{\bar{x}_1(t)}{\bar{x}_1(kt)} = \lim_{t\to\infty} k \cdot \frac{\ln t}{\ln k + \ln t} = k \cdot 1 = k \tag{21}$$

□

One last but critical element to explain how we use unit exponentials to fit absolutely monotonic functions is to show how the Bernstein theorem (theorem 1.1) applies to them.

Theorem 2.3. *Let $f : [0, 1] \to \mathbb{R}$ be an absolutely monotonic function. Then,*

there is a non-negative finite Borel measure μ such that

$$f(x) - f(0) = \int_0^\infty uexp_t(x)d\mu(t) \tag{22}$$

for every $x \in [0, 1]$

Proof. Start from Bernstein's theorem. Then there exists a non-negative finite Borel measure μ such that $f(x) = \int_0^\infty e^{tx}d\mu(t)$. But then,

$$f(x) = \int_0^\infty e^{tx}d\mu(t) = \int_0^\infty (e^{tx}-1+1)d\mu(t) = \int_0^\infty (e^{tx}-1)d\mu(t)+\int_0^\infty 1d\mu(t) = \\ = \int_0^\infty (e^t - 1)\frac{e^{tx}-1}{e^t-1}d\mu(t) + \int_0^\infty 1d\mu(t) \tag{23}$$

Since $e^t - 1$ is non-negative and integrable, we can define the non-negative finite Borel measure $d\nu(t) = (e^t - 1)d\mu(t)$ that is absolutely continuous with respect to μ, and write:

$$f(x) = \int_0^\infty \frac{e^{tx}-1}{e^t-1}d\nu(t) + \int_0^\infty 1d\mu(t) = \int_0^\infty \text{uexp}_t(x)d\nu(t) + \int_0^\infty 1d\mu(t) \tag{24}$$

Now consider

$$f(0) = \int_0^\infty \text{uexp}_t(0)d\nu(t)+\int_0^\infty 1d\mu(t) = 0+\int_0^\infty 1d\mu(t) = \int_0^\infty 1d\mu(t) \tag{25}$$

and so

$$f(x)-f(0) = \int_0^\infty \text{uexp}_t(x)d\nu(t)+\int_0^\infty 1d\mu(t)-\int_0^\infty 1d\mu(t) = \int_0^\infty \text{uexp}_t(x)d\nu(t) \tag{26}$$

□

We can therefore attempt to approximate any absolutely monotonic function f for which $f(0) = 0$ with a finite weighted sum of unit exponentials. Our goal now becomes, given the normalized dataset $\mathcal{D}$ with all x values in $[0, 1]$, find a finite weighted sum of unit exponentials that explains the values in $\mathcal{D}$ as well as possible. By construction, this finite weighted sum will be absolutely monotonic, and because of the theorem, we know that if we use a large enough number of well selected unit exponentials, we should be able to approximate the values as well as any absolutely monotonic function could.

2.3 Least squares optimization

In this section we present a simple parametric method to fit a positive sum of unit exponentials $\sum \text{uexp}_{t_i}$ to $\mathcal{D}$. We predefine the number of unit exponentials we will use n, but allow flexibility in both the weight w_i of each of them, and their parameters t_i. This creates a parameter space of size $2n$ that we can search using nonlinear least squares.

However, it is not reasonable to fit the t_i directly using nonlinear least squares, due to the difference in result becoming smaller with the same difference in the t_i the larger the t_i are, which also means the upper bounds for the t_i parameters would need to be very high for high representation power. For example, uexp_{100} and uexp_{101} are much more similar between them than uexp_2 and uexp_1 are. Moreover, if the t_i are each left to freely roam the entire parameter space, they may overlap or close to overlap with other t_i, which in turn generates redundancy in the parameter space. For example, 2uexp_2 is the same as $1\text{uexp}_2 + 1\text{uexp}_2$ (same parameter chosen twice), or very similar to $1.3\text{uexp}_{2.1} + 0.7\text{uexp}_{1.9}$. In other words, using the t_i directly for a nonlinear least squares optimization is ineffective.

Instead, we use the property, described earlier, that multiplying a t_i by a factor k approximately reduces the distance of the exploding point from 1 by $1/k$, to use these factors as parametes instead. Specifically, we define the set of parameters $\{k_i\} \subset (k_\downarrow, k^\uparrow)$ and initial $k_0 \in (0, k^\uparrow)$ for some chosen hyperparameter bounds $k_\downarrow, k^\uparrow$, and

$$t_i = \prod_{j=0}^{i-1} k_j \tag{27}$$

For example, the parameters $\{k_0 = 0.5, k_1 = 3, k_2 = 5, k_3 = 2\}$ results in $\{t_1 = 0.5, t_2 = 1.5, t_3 = 7.5, t_4 = 15\}$.

This creates relative spacing between exploding points of the unit exponentials proportional to the k_i, while making all k_i live in the same bounds $(k_\downarrow, k^\uparrow)$, which makes nonlinear least squares optimization more effective for fitting absolutely monotonic functions.

In terms of the bounds for the weight parameters w_i, we use the property that $\text{uexp}_t(1) = 1$ to realize that $w_i\text{uexp}_t(1) = w_i$, consider the maximum $M = \max_{\mathcal{D}}\{y\}$ in the data being fitted, and allow some extra room in case the corresponding x_i is far from 1. We allow the w_i to go up to rM, for $r \geq 1$ (e.g. $r = 3$), so that the bounds for the w_i are $(0, rM)$.

As a result, the problem can be summarized as follows:

- Input data $\mathcal{D}$ with maximum $M = \max_{\mathcal{D}}\{y\}$.
- Hyperparameter $n \geq 1$ (integer), the number of unit exponentials. Increasing it allows for more complex shape functions but increases the number of parameters in the optimization problem. Tested value: $n = 7$
- Hyperparameters $1 < k_{\downarrow} < k_{\uparrow}$ (real), the bounds of the proportional increase in the exponents of subsequent unit exponentials. Increasing $k_{\downarrow}$ ensures more difference between terms in the sum, while decreasing it allows more precision. Similarly, increasing $k^{\uparrow}$ allows more difference between subsequent unit exponentials, while decreasing it forces more precision. Tested values: $k_{\downarrow} = 1.25, k_{\uparrow} = 2.25$.
- Hyperparameter $r \geq 1$ (real), the proportion of extra room given to the weights. Increasing it allows for larger functions that might represent the data better, but increases the size of the parameter space. Tested value: $r = 3$.
- Parameters $\{w_i\}$ for $i = 1..n$, with each $w_i \in (0, rM)$. Tested initial value: 0.5.
- Parameter $k_0 \in (0, k^{\uparrow})$. Tested initial value: 0.5.
- Parameters $\{k_j\}$ for $j = 1..n-1$, with each $k_j \in (k_{\downarrow}, k^{\uparrow})$. Tested initial value: 2.

We use a nonlinear least squares optimization algorithm to find the $\{w_i\}$, $\{k_j\}$ that minimize the sum mean squared error

$$\sum_{(x,y)\in\mathcal{D}} (y - f(x))^2 \tag{28}$$

where

$$\begin{aligned} t_i &= \prod_{j=0}^{i-1} k_j \\ f(x) &= \sum_{i=1..n} w_i \mathrm{uexp}_{t_i}(x) \end{aligned} \tag{29}$$

2.4 Evaluation

In terms of evaluation metrics, because we are using nonlinear least squares and there is no statistical interpretation to our approach, the most reasonable metric is mean squared error (MSE). We normalize it for each test case to the value $f(1)$ of the fitted source function at 1 (NMSE).

More interestingly, there is the question of which datasets to test it on. In all cases we generate the dataset from an underlying function. We however have variation in multiple dimensions. We measure the NMSE on the same points used to fit the function, which we call fit NMSE, and then on a uniform sample of 200 points (without noise), which we call source NMSE. The difference between these two allows us to analyse to what degree the specific sampling affects the goodness of fit, with a high difference indicating that absolutely monotonic functions are too flexible, with a fit that works well for the samples but not in other parts of the support. For each test case, we run the fit 100 times with different noise and point samples, and calculate the average and worst source NMSE.

- **Number of samples** - In each test case we generate N data points from the source function to fit the function. We try with a low ($N = 10$), medium ($N = 25$) and high ($N = 200$) sample size. It is worth noting that the sample x values are uniformly distributed in $(0, 1)$. Due to the nature of absolutely monotonic functions, x values closer to 1 are harder to fit because the functions grow more quickly near them, so uniform sampling can make the fit quality vary more depending on the specific sample.

- **Noise** - It is interesting to see the effect of random noise on the method. As such, we have test cases with no random noise and with gaussian noise with standard deviation of different sizes, relative to the value of the function at 1 (the maximum): small gaussian noise ($\sigma = 0.01 \cdot f(1)$) and high gaussian noise ($\sigma = 0.1 \cdot f(1)$). Note that with high noise, the fit NMSE can be lower than the source NMSE, as the method can compensate for the noise in the fit over multiple samples.

- **Function** - We try the fit with various functions with different properties:

  - **Absolutely monotonic functions on** $(-\infty, a]$ - The cases for which the method should be best suited and which should show small error.

    - **Discrete sum of unit exponentials** - The fit can potentially be exact, and any error in the result is due to the approximation method rather than the problem setup. We try with $f_1(x) = 0.3\text{uexp}_{2.5}(x) + 2.6\text{uexp}_{10.35}(x) + 1.24\text{uexp}_{53.4}(x)$.
    - **Hyperexponentials** - Bernstein's theorem still applies to them but only up to an (arbitrary) limited interval - any fit will still fail if we extend it far enough towards positive infinity. The method should be able to fit them perfectly since our data is constrained, but the behaviour near the top of the interval might be complicated. We normalize them so that they go between 0 and 1 to avoid numerical errors with very large numbers. We try with $f_2(x) = \frac{e^{e^{e^x}} - e}{e^e - e}$ as a simple case and $f_3(x) = \frac{e^{e^{e^{e^x}}} - e^e}{e^{e^e} - e^e}$.
- **Absolutely monotonic functions on** $[0, 1]$ - Bernstein's theorem no longer applies to them so some error is to be expected.
    - **Absolutely monotonic polynomials** - These have a finite number of non-zero derivatives, and do not match exactly with a finite sum of unit exponentials. We try with $f_4(x) = x^2$ and $f_5(x) = x^5 + 2x^4 + x^3 + x$.
- **Convex functions that are not absolutely monotonic** - It is interesting to evaluate how well the method works when the absolutely monotonic assumption is untrue for the underlying function, but it has a convex shape. Some error is to be expected, but we wish to evaluate how good this method can be as an approximation for convex functions.
    - **Polynomial** - We try with $f_6(x) = 3x^2 - x^3$. This polynomial is convex in $(0, 1)$ but becomes concave exactly at 1. We also try with $f_7(x) = x^4 - 4x^3 + 6x^2$, which is convex for $x > 0$, and has a positive fourth derivative, but its third derivative is negative in $(0, 1)$.
    - **Trigonometric function** - We try with $f_8(x) = \sin(\frac{\pi}{2}x - \frac{\pi}{2}) + 1$, which is convex in $(0, 1)$ but becomes concave exactly at 1.
    - **Piecewise function** - We try with $f_9(x) = 4x^2\{x \in [0, 0.5]\}|4x - 1\{x \in [0.5, 1]\}$. This function is convex everywhere, continuous, has a continuous first derivative, but its second derivative jumps at 0.5 from 8 to 0.

TODO: Discussion about comparing with other libraries. Check ChatGPT discussion. It suggested cvxpy.

3 Results

4 Conclusions

References

[Aguilera and Morin, 2008] Aguilera, N. E. and Morin, P. (2008). Approximating optimization problems over convex functions. *Numerische Mathematik*, 111:1–34.

[Bernstein, 1929] Bernstein, S. (1929). Sur les fonctions absolument monotones. *Acta Mathematica*, 52(1):1–66.

[Evans et al., 1980] Evans, J. W., Gragg, W. B., and LeVeque, R. J. (1980). On least squares exponential sum approximation with positive coefficients. *Mathematics of Computation*, 34(149):203–211.

[Hannah and Dunson, 2013] Hannah, L. A. and Dunson, D. B. (2013). Multivariate convex regression with adaptive partitioning. *J. Mach. Learn. Res.*, 14(1):3261–3294.

[He and Shi, 1998] He, X. and Shi, P. (1998). Monotone b-spline smoothing. *Journal of the American Statistical Association*, 93(442):643–650.

[Holmström and Petersson, 2002] Holmström, K. and Petersson, J. (2002). A review of the parameter estimation problem of fitting positive exponential sums to empirical data. *Appl. Math. Comput.*, 126(1):31–61.

[Kammler, 1976] Kammler, D. W. (1976). Chebyshev approximation of completely monotonic functions by sums of exponentials. *SIAM Journal on Numerical Analysis*, 13(5):761–774.

[Keller and Plonka, 2021] Keller, I. and Plonka, G. (2021). Modifications of prony's method for the recovery and sparse approximation with generalized exponential sums. In Fasshauer, G. E., Neamtu, M., and Schumaker, L. L., editors, *Approximation Theory XVI*, pages 123–152, Cham. Springer International Publishing.

[Koumandos, 2014] Koumandos, S. (2014). *On Completely Monotonic and Related Functions*, pages 285–321. Springer New York, New York, NY.

[Lachand-Robert and Oudet, 2005] Lachand-Robert, T. and Oudet, E. (2005). Minimizing within convex bodies using a convex hull method. *SIAM Journal on Optimization*, 16(2):368–379.

[Lu, 2015] Lu, M. (2015). Spline estimation of generalised monotonic regression. *Journal of Nonparametric Statistics*, 27(1):19–39.

[Nagahara and Martin, 2013] Nagahara, M. and Martin, C. F. (2013). Monotone smoothing splines using general linear systems. *Asian Journal of Control*, 15(2):461–468.

[Rajba, 2011] Rajba, T. (2011). New integral representations of nth order convex functions. *Journal of Mathematical Analysis and Applications*, 379(2):736–747.

[Smith et al., 1976] Smith, M. R., Cohn-Sfetcu, S., and Buckmaster, H. A. (1976). Decomposition of multicomponent exponential decays by spectral analytic techniques. *Technometrics*, 18(4):467–482.

[Zhang, 2004] Zhang, J.-T. (2004). A simple and efficient monotone smoother using smoothing splines. *Journal of Nonparametric Statistics*, 16(5):779–796.