Optimization and Engineering manuscript No.
(will be inserted by the editor)

Fitting geometric programming models to data

Warren Hoburg · Philippe Kirschen · Pieter Abbeel

Received: date / Accepted: date

Abstract Motivated by practical applications in engineering, this article considers the problem of approximating a set of data with a model that is compatible with geometric programming (GP). Starting with well-established methods for fitting max-affine functions, it is shown that improved fits can be obtained using an extended function class based on the softmax of a set of affine functions. The softmax is generalized in two steps, with the most expressive function class using an implicit representation that allows fitting algorithms to locally tune softness. Each of the proposed function classes is directly compatible with the posynomial constraint forms in geometric programming. Through a limit analysis, max-monomial fitting and posynomial fitting are shown to correspond to fitting special cases of the proposed implicit softmax function class. The fitting problem is formulated as a nonlinear least squares regression, solved locally using a Levenberg-Marquardt algorithm. Practical implementation considerations are discussed. The article concludes with several numerical examples from aerospace engineering and electrical engineering.

Keywords Convex optimization · Convex regression · Geometric programming

W. Hoburg · P. Kirschen
Department of Aeronautics and Astronautics, Massachusetts Institute of Technology
77 Massachusetts Avenue, Cambridge, MA 02139, USA
E-mail: whoburg@mit.edu · kirschen@mit.edu

P. Abbeel
Computer Science Department, University of California, Berkeley, CA 94720, USA
E-mail: pabbeel@cs.berkeley.edu

1 Introduction

This article presents methods for fitting geometric programming (GP) models to data. To provide context, it begins with an overview of GP, followed by a motivation for fitting models that can be used in GP.

1.1 Overview of geometric programming

First introduced in 1967 by Duffin, Peterson, and Zener [10], a geometric program (GP)[1] is a specific type of constrained, nonlinear optimization problem that becomes convex after a logarithmic change of variables. Despite significant work on early applications in structural design, network flow, and optimal control [3,24], reliable and efficient numerical methods for solving GPs were not available until the 1990's [20]. GP has recently undergone a resurgence as researchers have discovered promising applications in statistics [5], digital circuit design [6], antenna optimization [2], communication systems [7], aircraft design [13], and other engineering fields.

Modern GP solvers employ primal-dual interior point methods [19] and are extremely fast. A typical sparse GP with tens of thousands of decision variables and one million constraints can be solved on a desktop computer in minutes [4]. Furthermore, these solvers do not require an initial guess and guarantee convergence to a *global* optimum, whenever a feasible solution exists.

These performance benefits are possible because geometric programs represent a restricted subset of nonlinear optimization problems. In particular, the objective and constraints can only be composed of *monomial* and *posynomial* functions.

Monomials In GP, a monomial is a function $h(\mathbf{u}) : \mathbb{R}^n_{++} \to \mathbb{R}_{++}$ of the form

$$h(\mathbf{u}) = c \prod_{j=1}^{n} u_j^{a_j}, \tag{1}$$

where $\mathbf{a} \in \mathbb{R}^n$, $c \in \mathbb{R}_{++}$ and $\mathbf{u} = (u_1, \ldots, u_n)$. Since the exponents a_i in (1) may be negative and non-integer, expressions like $\frac{u_1 u_2^{0.7} \sqrt{u_3}}{u_4}$ are monomials.

Posynomials Like monomials, posynomials are functions $g(\mathbf{u}) : \mathbb{R}^n_{++} \to \mathbb{R}_{++}$. A posynomial has the form

$$g(\mathbf{u}) = \sum_{k=1}^{K} c_k \prod_{j=1}^{n} u_j^{a_{jk}}, \tag{2}$$

[1] The "GP" acronym is overloaded, referring both to geometric programs - the class of optimization problem discussed in this article - and geometric programming - the practice of using such programs to model and solve optimization problems.

where $\mathbf{a}_k \in \mathbb{R}^n$, and $c_k \in \mathbb{R}_{++}$. Thus, a posynomial is simply a sum of monomial terms, and all monomials are also posynomials (with just one term). The expression $0.23+u_1^2+0.1u_1u_2^{-0.8}$ is an example of a posynomial in $\mathbf{u} = (u_1, u_2)$, whereas $2u_1 - u_2^{1.5}$ is not a posynomial because negative leading coefficients c_k are not allowed.

1.1.1 Geometric programs in standard form

GPs are often written in the standard form

$$\begin{aligned} \text{minimize} \quad & g_0(\mathbf{u}) \\ \text{subject to} \quad & g_i(\mathbf{u}) \leq 1, \;\; i = 1, \ldots, n_i, \\ & h_i(\mathbf{u}) = 1, \;\; i = 1, \ldots, n_e, \end{aligned} \tag{3}$$

where each g_i is a posynomial, and each h_i is a monomial.

The expression *GP-compatible* will be used in this article to refer to constraints that can be written in the form of either the monomial or posynomial constraints of Problem (3). It will also refer to objective functions that can be written as posynomials.

1.1.2 Geometric programs in convex form

GPs are of great interest because they become *convex* optimization problems under a logarithmic change of variables. In particular, the transformation

$$\mathbf{x} = \log \mathbf{u} \tag{4}$$

converts monomial equality constraints $h(\mathbf{u}) = 1$ to the form

$$\log h(e^{\mathbf{x}}) = \mathbf{a}^{\mathrm{T}}\mathbf{x} + b = 0, \tag{5}$$

and posynomial inequality constraints $g(\mathbf{u}) \leq 1$ to the form

$$\log g(e^{\mathbf{x}}) = \log \sum_{k=1}^{K} \exp(\mathbf{a}_k^{\mathrm{T}}\mathbf{x} + b_k) \leq 0. \tag{6}$$

By inspection, (5) is affine in $\mathbf{x}$, and it is straightforward to show that (6) is convex[2] in $\mathbf{x}$, since log-sum-exp functions are known to be convex, and convexity is preserved under affine transformations [5]. Thus, (3) is transformed into a convex optimization problem by the variable change (4).

This article refers to functions as *log-convex* if they are convex after both the independent and dependent variables are transformed into logarithmic space by the variable change (4).

[2] A function $f(\mathbf{x})$ is convex if the property $f(\theta\mathbf{x}_1 + (1-\theta)\mathbf{x}_2) \leq \theta f(\mathbf{x}_1) + (1-\theta)f(\mathbf{x}_2)$ holds for all $\theta \in [0, 1]$ and $\mathbf{x}_1$, $\mathbf{x}_2$ in the domain of f.

1.2 Motivation for fitting geometric programming models to data

The strengths of GP – its speed, its guarantee of global optimality, and its robustness – make a case for formulating nonlinear optimization problems as GPs, whenever possible. Although physical models for practical engineering problems can sometimes be written directly in GP-compatible form, the GP formulation can prove too restrictive.

Fortunately, certain models that cannot be written directly in GP-compatible form can still be well approximated by a monomial or posynomial, provided they exhibit a reasonable degree of log-convexity. Examples of such models include analytic expressions that are log-convex but are not posynomials, such as $\exp(x)$ or $-\log(1-x)$; empirical data sets; and "black box" simulations.

GP-compatible function fitting allows relationships such as these to be used in a GP, either as constraints or as an objective function. This creates an opportunity to incorporate a much broader range of models into the GP realm; the purpose of this article is to demonstrate methods for doing so.

1.3 Overview of the GP-compatible fitting procedure

In *GP-compatible fitting*, one is interested in approximating a set of data points

$$(\mathbf{u}_i, w_i) \in \mathbb{R}^n_{++} \times \mathbb{R}_{++}, \;\; i = 1 \ldots m,$$

with monomial or posynomial constraints. The multivariate data set $(\mathbf{u}_i, w_i)$ is the input to the fitting procedure and typically captures a relationship between some variables of interest. There are two restrictions on the input data: 1) the data must be strictly positive because of the log transformation, and 2) the data should be well-approximated by a log-convex function, i.e. once log-transformed, the data should exhibit a reasonable degree of convexity.

The fitting procedure can be summarized in three steps:

1. Apply the log transformation $(\mathbf{x}_i, y_i) = (\log \mathbf{u}_i, \log w_i)$ to the original data.
2. Fit a *convex* (but not GP-compatible) function to the transformed data.
3. Transform the resulting function back to the original space to obtain a GP-compatible function.

The output of the procedure is a single multivariate GP-compatible function (as opposed to a set of functions or constraints). This function can easily be incorporated into a GP either in the form of a constraint or an objective function. As Section 4 will discuss, the convex functions used for step two are specifically designed to have parallel interpretations as GP-compatible functions. From a purely mechanical perspective, this means that GP-compatible functions are actually a "free" by-product of step two.

Note that one could conceivably fit monomials and posynomials directly to the original data, without doing any logarithmic transformations. It turns out, however, that fitting convex functions to log-transformed data tends to better capture relationships for data that spans multiple orders of magnitude.

2 Current methods for fitting convex functions to data

To obtain GP-compatible function fits, we first address the problem of fitting functions that are convex but not necessarily GP-compatible. Consider the problem of fitting a multivariate function $f(\mathbf{x})$ to a set of m data points

$$(\mathbf{x}_i, y_i) \in \mathbb{R}^n \times \mathbb{R}, \;\; i = 1 \dots m.$$

With the restriction that f be convex, the fitting problem is

$$\underset{f \in \mathcal{F}}{\text{minimize}} \quad ||Y - f(X)|| \tag{7}$$

where $X \in \mathbb{R}^{m \times n}$ is a matrix of the $\mathbf{x}_i$ (independent data), $Y \in \mathbb{R}^m$ is a vector of the y_i (dependent data), f is the function being fitted[3], $\mathcal{F}$ is a set of convex functions, and $||\cdot||$ is some norm. This article takes $||\cdot||$ to be the 2-norm, but the work extends to other fitting criteria. Clearly, the norm (residual) cannot be small if the data is not well-approximated by any convex function.

As with any regression problem, the specific choice of function class $\mathcal{F}$ can dramatically influence the quality of the resulting model with respect to the chosen loss function. One common choice is $\mathcal{F}_{\text{MA}}^K$, the set of *max-affine* functions with K terms:

$$f_{\text{MA}}(\mathbf{x}) = \max_{k=1\dots K} \left[b_k + \mathbf{a}_k^{\text{T}} \mathbf{x} \right], \tag{8}$$

where $b_k \in \mathbb{R}$ and $\mathbf{a}_k \in \mathbb{R}^n$ are the model parameters. The total number of model parameters is $n_p = K(n+1)$. It is well known that the supremum over a set of affine functions is a convex function, thus (8) is convex in $\mathbf{x}$.

This choice is motivated by the fact that any convex function can be expressed as the point-wise supremum over a (generally infinite) set of affine functions [5]. That is, $\mathcal{F}_{\text{MA}}^\infty$ can approximate *any* convex function to arbitrary precision. Max-affine functions are also appealing from a practical perspective: they transform to monomial constraint sets (compatible with geometric programming) under the variable change (4).

Methods for fitting max-affine functions are well established. In 2008, Magnani and Boyd [17] proposed a least-squares partition algorithm that works well in practice. They also discussed a more general bi-affine function parameterization. In 2010, Kim et al. [15] used a similar method to fit max-monomial models for circuit design. Hannah and Dunson [12] fit max-affine functions using a statistically consistent adaptive partitioning approach that refines accuracy for increasing K by generating candidate partition splits and selecting splits greedily. They also describe an ensemble version of their method that avoids instability in max-monomial fitting [11].

Although max-affine functions are a natural and popular choice, a large number of affine terms K may be necessary to achieve a desired level of accuracy. This article argues for choosing f from a more general class of convex

[3] The mapping f is overloaded: $f(\mathbf{x})$ refers to function evaluation at a single point ($f : \mathbb{R}^n \to \mathbb{R}$), whereas $f(X)$ refers to a vectorized version ($f : \mathbb{R}^{m \times n} \to \mathbb{R}^m$).

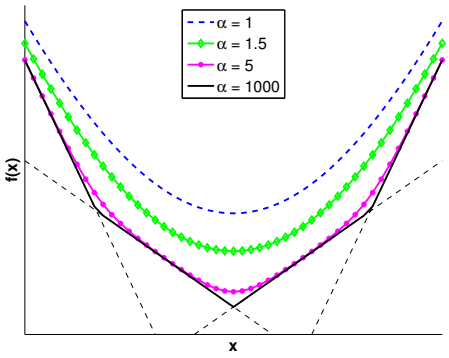


Fig. 1 Influence of α on softmax-affine functions. Each of the softmax-affine functions plotted above shares the same $K = 4$ affine terms (the thin dashed lines), but has a different α. The solid curve corresponds to a max-affine function; the dashed curve corresponds to a softmax-affine function with $\alpha = 1$, and one can interpolate smoothly among these by varying α. While this figure illustrates the situation in $\mathbb{R}^1$, the limiting cases extend to arbitrary dimensions.

functions. Previous work on convex regression has improved fitting algorithms and guaranteed important statistical properties like consistency, often for the special case of max-affine functions. In contrast, the emphasis here is on the mathematical structure of the functions being fitted – the choice of $\mathcal{F}$.

3 Proposed convex function classes

This section introduces two convex function classes: the *softmax-affine* class, $\mathcal{F}_{\text{SMA}}^K$, and the *implicit softmax-affine* class, $\mathcal{F}_{\text{ISMA}}^K$. These can be thought of as successive generalizations of $\mathcal{F}_{\text{MA}}^K$, a relationship discussed later in this section.

3.1 Softmax-affine functions

The proposed softmax-affine class $\mathcal{F}_{\text{SMA}}^K$ consists of functions of the form

$$f_{\text{SMA}}(\mathbf{x}) = \frac{1}{\alpha} \log \sum_{k=1}^{K} \exp\left(\alpha(b_k + \mathbf{a}_k^{\mathrm{T}}\mathbf{x})\right). \tag{9}$$

The addition of the scalar parameter $\alpha \in \mathbb{R}_{++} \cup \{+\infty\}$ brings the total number of parameters to $n_p = K(n+1)+1$. Since log-sum-exp functions are convex and convexity is preserved under positive scaling and affine transformations [5], SMA functions are guaranteed to be convex in $\mathbf{x}$ for any $\alpha > 0$.

Limiting behavior As depicted in Figure 1, one can view α as a smoothing parameter that controls the sharpness of the softmax over the K affine planes. In the limit of infinite sharpness,

$$\lim_{\alpha \to +\infty} \frac{1}{\alpha} \log \sum_{k=1}^{K} \exp\left(\alpha(b_k + \mathbf{a}_k^{\mathrm{T}}\mathbf{x})\right) = \max_{k=1\ldots K} \left[b_k + \mathbf{a}_k^{\mathrm{T}}\mathbf{x}\right]. \tag{10}$$

According to Equation 10, softmax-affine functions become max-affine functions in the limit $\alpha \to +\infty$. This limiting behavior implies that for a given data set (X, Y) and number of affine terms K, there always exist SMA functions with at least as small a residual as the best possible max-affine fit.

3.2 Implicit softmax-affine functions

The proposed implicit softmax-affine class, $\mathcal{F}_{\text{ISMA}}^{K}$, expresses the relationship between $\mathbf{x}$ and y via the zero of an *implicit* function

$$\tilde{f}_{\text{ISMA}}(\mathbf{x}, y) = \log \sum_{k=1}^{K} \exp\left(\alpha_k(b_k + \mathbf{a}_k^{\mathrm{T}}\mathbf{x} - y)\right), \tag{11}$$

where each $\alpha_k \in \mathbb{R}_{++} \cup \{+\infty\}$.

Proposition 1 *For all* $\mathbf{x}$*, there exists a unique* y *such that* $\tilde{f}_{\text{ISMA}}(\mathbf{x}, y) = 0$.

Proof For all $\mathbf{x}$, $y \mapsto \tilde{f}_{\text{ISMA}}(\mathbf{x}, y)$ is a continuous monotone strictly decreasing function of y, since increasing y decreases every term in the K-term sum. Moreover, there exists some γ^- such that $\tilde{f}_{\text{ISMA}}(\mathbf{x}, \gamma^-) > 0$, and some γ^+ such that $\tilde{f}_{\text{ISMA}}(\mathbf{x}, \gamma^+) < 0$. Thus by the intermediate value theorem the function must have a unique zero crossing between γ^- and γ^+.

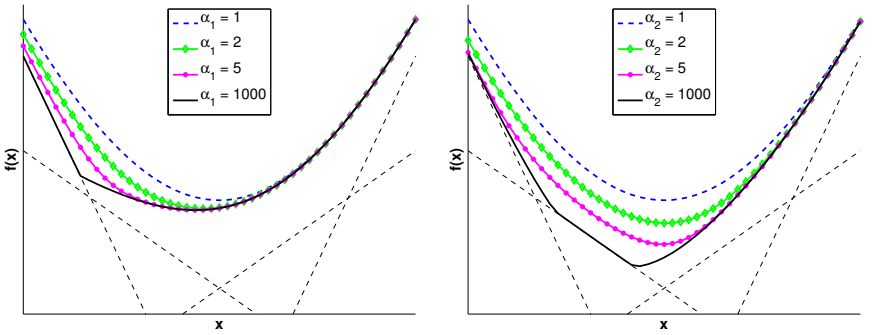
Based on Proposition 1, f_{ISMA} is defined such that for all $\mathbf{x}$,

$$\tilde{f}_{\text{ISMA}}(\mathbf{x}, f_{\text{ISMA}}(\mathbf{x})) = 0. \tag{12}$$

That is, the predicted value $\hat{y} = f_{\text{ISMA}}(\mathbf{x})$ is the unique value $\hat{y}$ such that $\tilde{f}_{\text{ISMA}}(\mathbf{x}, \hat{y}) = 0$.

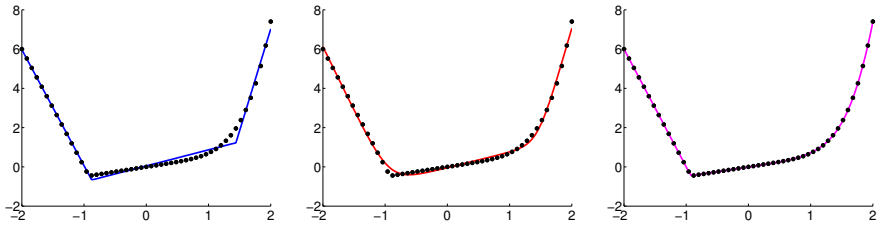
Proposition 2 *The function* $\mathbf{x} \mapsto f_{\text{ISMA}}(\mathbf{x})$ *is convex.*

Proof Consider any two points $(\mathbf{x}_1, y_1)$ and $(\mathbf{x}_2, y_2)$ that both solve $\tilde{f}_{\text{ISMA}}(\mathbf{x}, y) = 0$. By convexity of the log-sum-exp function and preservation of convexity under affine mappings, $\tilde{f}_{\text{ISMA}}(\theta\mathbf{x}_1 + (1-\theta)\mathbf{x}_2, \theta y_1 + (1-\theta)y_2) \leq 0 \ \forall \ \theta \in [0, 1]$. But for some $\hat{y}$, $\tilde{f}_{\text{ISMA}}(\theta\mathbf{x}_1 + (1-\theta)\mathbf{x}_2, \hat{y}) = 0$. Since $\tilde{f}_{\text{ISMA}}$ is monotone decreasing in y, $\hat{y} \leq \theta y_1 + (1-\theta)y_2$. Thus the function value $\hat{y}$ at any weighted combination of $\mathbf{x}$ points is less than or equal to a weighted combination of the y values – exactly the definition of convexity.



(a) Varying α_1 (leftmost plane) (b) Varying α_2

Fig. 2 Influence of individual softness parameters α_k on ISMA functions. Each of the functions above shares the same $K = 4$ affine terms (the thin dashed lines). Setting all of the softness parameters α_k to 1 results in the top curve (dashed line) in each figure. Varying just one of the four softness parameters then gives intuition about its effect. This figure illustrates the situation in $\mathbb{R}^1$, but the qualitative behavior extends to arbitrary dimensions.



(a) MA: `RMS = 0.1908` (b) SMA: `RMS = 0.1266` (c) ISMA: `RMS = 0.0104`

Fig. 3 This toy fitting problem illustrates how ISMA functions can significantly outperform SMA (and therefore also MA) functions. All fits used $K = 3$ affine terms. Here the data are samples of the convex function $y = \max\left(-6x - 6, \frac{1}{2}x, \frac{1}{5}x^5 + \frac{1}{2}x\right)$, which has a sharp kink near $x = -0.8$, but gradual curvature elsewhere. The best fit is an ISMA function, which can conform to the data by adjusting softness locally.

ISMA functions have individual softness parameters $\alpha_k > 0$ for each of the K affine terms in the model, bringing the total number of model parameters to $n_p = K(n+2)$. Figure 2 illustrates the influence of these softness parameters, and Figure 3 shows how they can improve fitting performance.

Setting all the α_k parameters to the same value α, one recovers the softmax-affine function class. This implies that the implicit softmax-affine class subsumes the softmax-affine class, and therefore also the max-affine class. As a result, for a given fitting problem and choice of K, there always exists some setting of the α_k parameters for which the ISMA class performs as well as or better than the best model in each of the other function classes.

Evaluating ISMA functions No explicit formula is available for evaluating ISMA functions. Instead, Algorithm 1 gives a Newton-Raphson [25] method for solving $\tilde{f}_{\text{ISMA}}(\mathbf{x}, y) = 0$. The breakdown $y = s + t$ avoids numerical is-

Algorithm 1 Evaluate $y = f_{\text{ISMA}}(\mathbf{x})$

$z_k \leftarrow b_k + \mathbf{a}_k^{\mathrm{T}}\mathbf{x}, \;\; k = 1 \ldots K$
$s \leftarrow \max_k z_k$
$t \leftarrow 0$
repeat
  $f \leftarrow \log \sum_{k=1}^{K} \exp\left(\alpha_k(z_k - s - t)\right)$
  $J \leftarrow -\frac{\sum_k \alpha_k \exp(\alpha_k(z_k - s - t))}{\sum_k \exp(\alpha_k(z_k - s - t))}$
  $t \leftarrow t - f/J$
until $|f| <$ `machine_precision`
return $y = s + t$

sues associated with computing ratios and logarithms of large exponentials. In practice, Algorithm 1 reliably converges to machine precision in approximately 5-10 Newton iterations.

4 Transformation to GP-compatible functions

Currently, a common approach to fit GP-compatible functions is to approximate y as a max-affine function of $\mathbf{x}$, and convert the resulting affine functions back to equivalent monomial constraints on the GP. Since GPs also admit posynomial constraints, another approach is to directly fit w as a posynomial function of $\mathbf{u}$. This approach has been used by a number of authors [8,16,22]. Typically mixed results are reported, with max-monomial models performing better on data with sharp kinks, and posynomial models performing better on smoother data [4,17]. As discussed shortly, one benefit of SMA and ISMA functions is the unification of these two approaches.

4.1 GP interpretation of proposed convex function classes

Each of the convex function classes in this article has a parallel interpretation as a GP-compatible function. This makes SMA and ISMA functions natural choices for GP-compatible fitting.

4.1.1 Max-affine functions as max-monomial functions

Recall that under the GP log transformation, monomial functions become affine. Thus, a max-affine model for y,

$$\hat{y} = f_{\text{MA}}(\mathbf{x}), \tag{13}$$

corresponds exactly to a max-monomial model for w,

$$\hat{w} = \max_{k=1\ldots K} \left[e^{b_k} \prod_{i=1}^{n} u_i^{a_{ik}} \right], \tag{14}$$

which is easily converted to a GP-compatible set of K monomial inequality constraints on $(\mathbf{u}, w)$,

$$\frac{e^{b_k}}{w}\prod_{i=1}^{n} u_i^{a_{ik}} \leq 1, \;\; k = 1 \ldots K. \tag{15}$$

Because they can be fit using established max-affine methods, max-monomial functions are currently a common choice for GP modeling [11,15].

4.1.2 SMA functions as posynomial functions

Consider the GP log transformation applied to (9). A model,

$$\hat{y} = f_{\text{SMA}}(\mathbf{x}), \tag{16}$$

corresponds to a posynomial model[4] for w^α,

$$w^\alpha = \sum_{k=1}^{K} e^{\alpha b_k}\prod_{i=1}^{n} u_i^{\alpha a_{ik}}, \tag{17}$$

which corresponds to a posynomial constraint,

$$\frac{1}{w^\alpha}\sum_{k=1}^{K} e^{\alpha b_k}\prod_{i=1}^{n} u_i^{\alpha a_{ik}} \leq 1. \tag{18}$$

For the special case $\alpha = 1$, SMA functions reduce to posynomials in convex form (see Equation (6)). That is, the model $\hat{y} \geq f_{\text{SMA}}(\mathbf{x}; \alpha = 1)$ corresponds to a posynomial constraint on $(\mathbf{u}, w)$,

$$\frac{1}{w}\sum_{k=1}^{K} e^{b_k}\prod_{i=1}^{n} u_i^{a_{ik}} \leq 1. \tag{19}$$

While the distinction between (18) and (19) appears subtle, it has important implications for GP modeling. In particular, in existing literature, posynomial fitting often refers to fitting functions of the form $\hat{w} = g(\mathbf{u})$, which corresponds to searching for softmax-affine models, but with the restriction $\alpha = 1$. Better fits can be obtained by searching for models of a more expressive posynomial form $\hat{w}^\alpha = g(\mathbf{u})$.

The softmax-affine function class includes max-affine functions ($\alpha = +\infty$) and posynomial functions ($\alpha = 1$) as special cases. Softmax-affine functions therefore unify max-monomial and posynomial fitting, eliminating the need to search over multiple model classes.

[4] Equation (17) can also be interpreted as a generalized posynomial[4,14] model for w, $w = \left(\sum_{k=1}^{K} e^{\alpha b_k}\prod_{i=1}^{n} u_i^{\alpha a_{ik}}\right)^{\frac{1}{\alpha}}$.

4.1.3 ISMA functions as implicit posynomial functions

Consider the GP log transformation applied to (11). An ISMA model,

$$\hat{y} = f_{\text{ISMA}}(\mathbf{x}) \tag{20}$$

corresponds exactly to a posynomial model,

$$\sum_{k=1}^{K} \frac{e^{\alpha_k b_k}}{w^{\alpha_k}} \prod_{i=1}^{n} u_i^{\alpha_k a_{ik}} = 1 \tag{21}$$

which, in turn, corresponds to a posynomial constraint on $(\mathbf{u}, w)$,

$$\sum_{k=1}^{K} \frac{e^{\alpha_k b_k}}{w^{\alpha_k}} \prod_{i=1}^{n} u_i^{\alpha_k a_{ik}} \leq 1. \tag{22}$$

By allowing a different α_k in each monomial term, ISMA functions are even more expressive than SMA or MA models.

5 Fitting model parameters

This section addresses the problem of fitting the proposed function classes to data. Data fitting and nonlinear regression are well-established fields supported by an extensive literature. Nevertheless, there is value in describing a practical end-to-end fitting procedure for the specific cases of SMA and ISMA functions.

5.1 Fitting approach overview

Given m data points $(\mathbf{x}_i, y_i) \in \mathbb{R}^n \times \mathbb{R}$, a least squares fitting objective is

$$\underset{\beta}{\text{minimize}} \ \sum_{i=1}^{m} \left(f(\mathbf{x}_i; \beta) - y_i\right)^2, \tag{23}$$

where f is an instance of one of the convex function classes already presented, and $\beta \in \mathbb{R}^{n_p}$ is a vector that contains the parameters a, b, and α for the chosen function class. In general, (23) is a nonlinear least squares problem. The quantity $\mathbf{r}(\beta) = f(X; \beta) - Y$ is the vector of residuals at each data point, and the goal is to find a set of parameters β for which $\mathbf{r}(\beta)^{\mathrm{T}}\mathbf{r}(\beta)$ is minimized.

Consider some initial set of parameters β_0, and corresponding residual $\mathbf{r}(\beta_0)$. For a small change in parameters δ, the new residual is approximately

$$\mathbf{r}(\beta_0 + \delta) \approx \mathbf{r}(\beta_0) + \mathbf{J}\delta, \tag{24}$$

where $\mathbf{J} \in \mathbb{R}^{m \times n_p}$ is $\partial f / \partial \beta$, the Jacobian of f. This suggests a first-order approximation of (23), rewritten in terms of the parameter update δ:

$$\begin{aligned} \underset{\delta}{\text{minimize}} \quad & \delta^{\mathrm{T}} \mathbf{J}^{\mathrm{T}} \mathbf{J} \delta + 2 \delta^{\mathrm{T}} \mathbf{J}^{\mathrm{T}} \mathbf{r} + \mathbf{r}^{\mathrm{T}} \mathbf{r} \\ \text{subject to} \quad & \delta \in \mathit{\Delta}, \end{aligned} \tag{25}$$

where $\mathit{\Delta}$ represents a trust region, imposed to keep the approximation (24) valid. Most popular algorithms for nonlinear least squares alternate between solving some form of the trust region subproblem (25), and updating β, $\mathbf{r}$, and $\mathbf{J}$ for those steps that achieve an acceptable improvement in residual. The high-level approach is sketched in Algorithm 2.

Algorithm 2 Nonlinear least squares fitting

```
β ← β_0
δ = 0
Δ ← initial trust region parameters
repeat
  r_trial ← f(X; β + δ) − Y
  if trial point acceptable then
    β ← β + δ
    r ← r_trial
    J ← ∂f/∂β
    Δ ← keep or expand trust region
  else
    Δ ← constrict trust region
  end if
  δ ← trust-region-subproblem(J, r, Δ)
until no further improvement
return r, β
```

5.2 Trust region subproblem

This section describes two formulations of the trust region subproblem (25).

5.2.1 Gauss-Newton update

Gauss-Newton methods [21] find the δ that minimizes the quadratic objective (25), with no trust region bounds. The optimal step is the solution to a set of linear equations,

$$\mathbf{J}^{\mathrm{T}} \mathbf{J} \delta = -\mathbf{J}^{\mathrm{T}} \mathbf{r}. \tag{26}$$

If the computed step does not achieve a satisfactory reduction in residual, a line search is typically used to refine the step length. The least squares partition algorithm due to Magnani and Boyd [17] can be viewed as a Gauss-Newton update for the specific case of max-affine fitting.

5.2.2 Levenberg-Marquardt algorithms

Levenberg-Marquardt (LM) algorithms [18,23] are similar to Gauss-Newton methods, but with trust region bounds on the parameter update. Instead of constraining δ to lie within the bounds of an explicit trust region, LM algorithms construct the trust region implicitly through a quadratic penalty on δ. The advantage of this formulation is that the step is simply the solution to a linear system,

$$\left(\mathbf{J}^{\mathrm{T}}\mathbf{J} + \lambda \mathrm{diag}(\mathbf{J}^{\mathrm{T}}\mathbf{J})\right)\delta = -\mathbf{J}^{\mathrm{T}}\mathbf{r}, \tag{27}$$

where λ controls the magnitude of the quadratic penalty on δ. Various update schemes for λ exist; in general λ should be increased when the step fails to decrease the residual sufficiently, kept constant when the penalty term is not appreciably affecting the solution, and decreased otherwise.

5.2.3 Comments on scaling to large problems

From a computational perspective, solving the trust region sub-problem is the dominant cost in the nonlinear least squares solution algorithm [1]. The overall cost of fitting one of the proposed functions to a data set scales as the cost of solving the trust region sub-problem, times the number of iterations needed for convergence, times the number of random restarts used to avoid poor local optima. The number of iterations and necessary number of random restarts are user-chosen parameters of the fitting algorithm for which rigorous bounds are not available.

The computational cost of each LM or trust region sub-problem iteration is comparable to solving a least squares problem involving $\mathbf{J}$, which grows as $\mathcal{O}(mn_p^2)$ [5]. The number of parameters n_p is $\mathcal{O}(Kn)$ for all three function classes, thus the total computational cost of each LM or trust region sub-problem is $\mathcal{O}(mK^2n^2)$.

Although the authors have not attempted to improve upon this scaling, for large problems better performance may be possible by exploiting sparsity. In particular, even for the smoothed SMA and ISMA function classes, the parameters $\mathbf{a}_k, b_k$ of each affine plane typically have relatively little effect on the function value at many of the data points, which introduces a nearly-sparse structure in $\mathbf{J}$. The limiting case is the max-affine function class, for which the fitting problems for each affine plane actually decouple. The Magnani-Boyd partition algorithm leverages this by solving K independent least squares problems, each with an average of m/K data points, for a total cost of $\mathbf{O}(mn^2)$. Thus iterative methods that can exploit near-sparsity in $\mathbf{J}$ may be capable of significantly reducing the effect of K on overall computational cost, even for SMA and ISMA models.

5.3 Model Jacobians

This section lists the partial derivatives needed to construct Jacobians for all the convex function classes presented in Section 3. The parameterization in terms of $\log \alpha$ instead of α implicitly constrains $\alpha > 0$ and results in better numerical conditioning.

5.3.1 Max-affine functions

The partial derivatives for max-affine functions are defined assuming that there are no ties in the maximum that defines $f_{\text{MA}}(x)$ [17]. For a practical implementation it suffices to break ties arbitrarily.

$$\frac{\partial f_{\text{MA}}}{\partial b_i} = \begin{cases} 1, \, i = \text{argmax}_k \;\; b_k + \mathbf{a}_k^{\text{T}}\mathbf{x} \\ 0, \text{ otherwise} \end{cases} \tag{28}$$

$$\frac{\partial f_{\text{MA}}}{\partial \mathbf{a}_i} = \begin{cases} \mathbf{x}^{\text{T}}, \, i = \text{argmax}_k \;\; b_k + \mathbf{a}_k^{\text{T}}\mathbf{x} \\ 0, \text{ otherwise} \end{cases} \tag{29}$$

5.3.2 Softmax-affine functions

$$\frac{\partial f_{\text{SMA}}}{\partial b_i} = \frac{\exp(\alpha(b_i + \mathbf{a}_i^{\text{T}}\mathbf{x}))}{\sum_{k=1}^{K} \exp(\alpha(b_k + \mathbf{a}_k^{\text{T}}\mathbf{x}))} \tag{30}$$

$$\frac{\partial f_{\text{SMA}}}{\partial \mathbf{a}_i} = \frac{\mathbf{x}^{\text{T}} \cdot \exp(\alpha(b_i + \mathbf{a}_i^{\text{T}}\mathbf{x}))}{\sum_{k=1}^{K} \exp(\alpha(b_k + \mathbf{a}_k^{\text{T}}\mathbf{x}))} \tag{31}$$

$$\frac{\partial f_{\text{SMA}}}{\partial (\log \alpha)} = \frac{\sum_{k=1}^{K} (b_k + \mathbf{a}_k^{\text{T}}\mathbf{x}) \exp(\alpha(b_k + \mathbf{a}_k^{\text{T}}\mathbf{x}))}{\sum_{k=1}^{K} \exp(\alpha(b_k + \mathbf{a}_k^{\text{T}}\mathbf{x}))} - f_{\text{SMA}} \tag{32}$$

5.3.3 Implicit softmax-affine functions

$$\frac{\partial f_{\text{ISMA}}}{\partial b_i} = \frac{\alpha_i \exp(\alpha_i(b_i + \mathbf{a}_i^{\text{T}}\mathbf{x} - f_{\text{ISMA}}))}{\sum_{k=1}^{K} \alpha_k \exp(\alpha_k(b_k + \mathbf{a}_k^{\text{T}}\mathbf{x} - f_{\text{ISMA}}))} \tag{33}$$

$$\frac{\partial f_{\text{ISMA}}}{\partial \mathbf{a}_i} = \frac{\mathbf{x}^{\text{T}}\alpha_i \cdot \exp(\alpha_i(b_i + \mathbf{a}_i^{\text{T}}\mathbf{x} - f_{\text{ISMA}}))}{\sum_{k=1}^{K} \alpha_k \exp(\alpha_k(b_k + \mathbf{a}_k^{\text{T}}\mathbf{x} - f_{\text{ISMA}}))} \tag{34}$$

$$\frac{\partial f_{\text{ISMA}}}{\partial (\log \alpha_i)} = \frac{\alpha_i(b_i + \mathbf{a}_i^{\text{T}}\mathbf{x} - f_{\text{ISMA}}) \exp(\alpha_i(b_i + \mathbf{a}_i^{\text{T}}\mathbf{x} - f_{\text{ISMA}}))}{\sum_{k=1}^{K} \alpha_k \exp(\alpha_k(b_k + \mathbf{a}_k^{\text{T}}\mathbf{x} - f_{\text{ISMA}}))} \tag{35}$$

5.4 Parameter initialization

This section describes suitable initial parameter vectors β_0 for each of the convex function classes.

5.4.1 Max-affine initialization

The parameter initialization for a max-affine function follows Magnani and Boyd [17]. They pick K data points $\{\bar{\mathbf{x}}_1, \bar{\mathbf{x}}_2, \ldots, \bar{\mathbf{x}}_K\}$ at random without replacement, and partition the data set according to which $\bar{\mathbf{x}}$ is closest (in Euclidean distance, for example). Each partition is then fit locally using least squares, thereby forming an initial set of max-affine parameters.

In practice, one or more of the randomly-sampled partitions may contain very few data points, resulting in a singular local fitting problem. In the authors' implementation, rank-deficient initial partitions are expanded until all the local fits become well-posed.

5.4.2 Softmax-affine initialization

Since softmax-affine functions represent max-affine functions in the limit $\alpha \to +\infty$, max-affine functions provide a convenient initialization for SMA fitting. In particular, given a max-affine fit $\beta_{\text{MA}} = (\mathbf{a}_{\text{MA}}, \mathbf{b}_{\text{MA}})$, a natural softmax-affine initialization might be

$$\beta_{\text{SMA}} = (\mathbf{a}_{\text{MA}}, \mathbf{b}_{\text{MA}}, \alpha_0 = 100)$$

Too large an initial α_0 will cause the α-gradient (32) to be extremely small for all data points. Therefore, a line search over α is implemented to find an α_0 that has non-zero $\mathbf{J}_\alpha$ at some data points.

5.4.3 Implicit softmax-affine initialization

Given the parameters $\beta_{\text{SMA}} = (\mathbf{a}_{\text{SMA}}, \mathbf{b}_{\text{SMA}}, \alpha_{\text{SMA}})$ of a SMA fit, a suitable ISMA initialization is

$$\beta_{\text{ISMA}} = (\mathbf{a}_{\text{SMA}}, \mathbf{b}_{\text{SMA}}, [\alpha_{\text{SMA}}, \alpha_{\text{SMA}}, \ldots, \alpha_{\text{SMA}}]).$$

Alternatively, one can initialize ISMA fitting directly from a randomly-seeded MA fit:

$$\beta_{\text{ISMA}} = (\mathbf{a}_{\text{MA}}, \mathbf{b}_{\text{MA}}, [100, 100, ..., 100]).$$

The implementation used for this work tries both initializations for every random seed, and selects the result with the smaller residual.

5.5 Avoiding numerical overflow

Many `exp` terms appear in both the convex function definitions and their Jacobians. When exponentiated, a modestly large argument can quickly overwhelm the representation capability of floating point arithmetic. One must therefore use caution when implementing these equations in software. Two common situations occur:

Ratios of sums of exponentials To handle these, note that

$$\frac{ce^{\alpha p}}{\sum_{i=1}^{N} e^{\alpha p_i}} = \frac{ce^{\alpha(p-s)}}{\sum_{i=1}^{N} e^{\alpha(p_i-s)}}, \tag{36}$$

i.e. one can simply subtract some s from all the exponents in the numerator and denominator, such that the largest argument to `exp` is small - say, 0.

Log sum exp terms To handle these, note that

$$\frac{1}{\alpha}\log\sum_{i=1}^{N} e^{\alpha p_i} = s + \frac{1}{\alpha}\log\sum_{i=1}^{N} e^{\alpha(p_i-s)}, \tag{37}$$

for some s chosen to keep the maximum exponential small.

6 Numerical examples and comparisons

Throughout this section, RMS error on the residuals, $\hat{y} - y$, is defined as

$$\text{RMS error } \equiv \sqrt{\frac{1}{m}\sum_{i=1}^{m}(f(\mathbf{x}_i) - y_i)^2}. \tag{38}$$

Absolute error on $(\mathbf{x}, y)$ is closely related to percentage error on $(\mathbf{u}, w)$, since

$$\frac{\hat{w} - w}{w} \approx \log\frac{\hat{w}}{w} = \hat{y} - y. \tag{39}$$

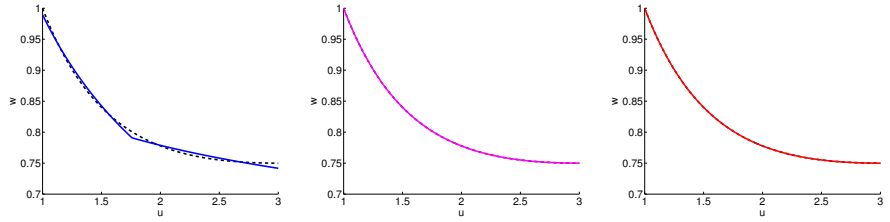
6.1 Example: local convex fitting of a scalar function

Suppose that the scalar relationship

$$w(u) = \frac{u^2 + 3}{(u+1)^2}, \ 1 \leq u \leq 3 \tag{40}$$

expresses an important relationship between two variables (u, w) in a GP. Can this relationship be encoded as a GP-compatible constraint?

Importantly, the expression $(u^2+3)/(u+1)^2$ is not log-convex for all $u > 0$ (a simple log-log plot verifies this). Thus there is no hope of algebraically manipulating (40) into a posynomial constraint. Nevertheless, (40) *is* log-convex over the domain of interest, $1 \leq u \leq 3$. This suggests sampling the function and fitting one of the proposed convex function classes to the 'data'. For this problem a value of $K = 2$ is used. Figure 4 shows the resulting fits, with SMA and ISMA functions significantly outperforming MA functions. The fitted models are listed in Table 1.



(a) MA: `RMS = 5.24e-3` (b) SMA: `RMS = 2.30e-5` (c) ISMA: `RMS = 7.48e-6`

Fig. 4 Generalized posynomial fitting of the function $w = (u^2+3)/(u+1)^2$ over $1 \le u \le 3$. The function was sampled at 501 logarithmically-spaced points, which were log-transformed and fitted with MA, SMA, and ISMA functions, each with $K = 2$ affine terms. Converting these functions back to to the original (u, w) space turned them into max-monomial and posynomial models.

Table 1 Convex fits for the fitting problem in Section 6.1.

Function class	Fitted function	RMS log error
MA	$\hat{w} = \max\left(0.846u^{-0.12}, 0.989u^{-0.397}\right)$	5.24×10^{-3}
SMA	$\hat{w}^{3.44} = 0.154u^{0.584} + 0.847u^{-2.15}$	2.30×10^{-5}
ISMA	$1 = 0.0658\dfrac{u^{0.958}}{\hat{w}^{3.79}} + 0.934\dfrac{u^{-1.22}}{\hat{w}^{2.03}}$	7.48×10^{-6}

6.2 Performance on randomly-generated convex functions

To test the proposed function classes and fitting algorithms on a wider range of data sets, a convex function generator was created to produce random instances of convex functions using a stochastic grammar of convexity-preserving operations and composition rules. This generator was used to produce 20 convex functions $\mathbb{R}^2 \to \mathbb{R}$, and 20 more convex functions $\mathbb{R}^5 \to \mathbb{R}$. For example, the first $\mathbb{R}^2 \to \mathbb{R}$ function drawn was

$$\begin{aligned} f_1(x_1, x_2) = &\max(\log(\exp(-0.32724x_1 + 0.89165x_2) + \exp(0.44408x_1 - 0.91182x_2 + ... \\ &\max(-0.10307x_1 - 2.799x_2 + (0.62101x_1 - 1.5075x_2)^2 + 0.2417x_1 + ... \\ &0.54935x_2 - 0.2298x_1 - 0.57922x_2 + (0.42162x_1 + 1.0877x_2)^2, \\ &\log(\exp((0.17694x_1 + 3.4663x_2)^2) + \exp(\max(\log(\exp((-0.4385x_1 + ... \\ &0.34322x_2)^2) + \exp(\max(-0.83768x_1 - 1.3075x_2, 0.64915x_1 - 0.83147x_2))), \\ &1.5667x_1 + 0.8465x_2))) - 0.39754x_1 + 0.25429x_2))), (1.2951x_1 + 2.7681x_2)^2). \end{aligned}$$

Input data $\mathbf{x}$ for each function consisted of 1000 points drawn from a multivariate Gaussian with zero mean and identity covariance. For each of the fitting problems, the fitting process was restarted from 10 random initializa-

tions, and the best fit was chosen. Tables 2 and 3 report the average fitting error and time across the 20 randomly-generated fitting problems considered in $\mathbb{R}^2$ and $\mathbb{R}^5$ respectively. These results show that SMA and ISMA functions provide consistent benefits in fitting error, when compared with max-affine functions as a baseline.

Table 2 Fitting error comparison for 20 randomly-generated functions $\mathbb{R}^2 \to \mathbb{R}$. Results are reported across the 20 fitting problems considered. The code ran on a quad-core Intel Core i7 CPU @ 2.8GHz with 4GB memory.

	RMS error as percentage of MA error with K=2 (worst-case, **average**, best-case)			Average fitting time (s) per random restart		
K	MA	SMA	ISMA	MA	SMA	ISMA
2	(100.0, **100.0**, 100.0)	(93.4, **74.8**, 56.0)	(93.0, **74.5**, 55.8)	0.18	0.28	0.48
3	(77.0, **69.8**, 59.1)	(24.3, **13.0**, 5.4)	(24.1, **13.3**, 6.9)	0.23	0.36	0.90
4	(53.6, **48.6**, 39.3)	(17.2, **10.4**, 4.9)	(10.7, **7.7**, 3.7)	0.24	0.55	0.95
5	(42.9, **37.9**, 26.7)	(10.3, **7.0**, 3.5)	(9.0, **5.5**, 3.0)	0.26	0.54	1.10
6	(36.0, **30.6**, 22.1)	(9.0, **6.0**, 3.3)	(7.1, **4.5**, 2.7)	0.28	0.53	1.56
7	(29.0, **25.7**, 18.9)	(8.5, **5.2**, 2.8)	(6.4, **3.6**, 1.6)	0.29	0.92	1.44
8	(26.0, **22.6**, 17.3)	(7.7, **4.6**, 2.4)	(5.5, **3.1**, 1.6)	0.32	0.85	1.27
9	(23.6, **19.7**, 16.1)	(6.4, **4.0**, 2.3)	(5.0, **2.7**, 1.5)	0.38	1.12	1.55
10	(19.6, **17.3**, 14.0)	(5.9, **3.6**, 2.1)	(4.3, **2.4**, 1.1)	0.38	1.22	1.66

Table 3 Fitting error comparison for 20 randomly-generated functions $\mathbb{R}^5 \to \mathbb{R}$.

	RMS error as percentage of MA error with K=2 (worst-case, **average**, best-case)			Average fitting time (s) per random restart		
K	MA	SMA	ISMA	MA	SMA	ISMA
2	(100.0, **100.0**, 100.0)	(98.3, **90.2**, 58.6)	(97.6, **89.6**, 58.3)	0.54	0.66	1.40
3	(86.2, **80.1**, 60.3)	(78.2, **61.1**, 32.5)	(77.5, **60.4**, 32.3)	0.52	0.74	2.04
4	(76.9, **66.7**, 45.3)	(56.9, **41.6**, 21.6)	(56.2, **40.7**, 21.1)	0.62	0.91	2.07
5	(67.6, **58.1**, 40.4)	(41.8, **29.5**, 16.7)	(41.1, **28.6**, 14.7)	0.86	0.99	1.98
6	(59.6, **51.6**, 33.7)	(33.6, **21.5**, 13.7)	(30.2, **19.5**, 13.1)	0.86	1.43	2.38
7	(55.5, **46.4**, 26.8)	(31.9, **19.2**, 12.6)	(22.7, **15.8**, 10.7)	1.15	1.60	2.89
8	(52.1, **41.2**, 19.6)	(28.6, **17.6**, 12.1)	(18.7, **13.7**, 9.0)	1.23	1.78	3.00
9	(48.3, **38.3**, 17.1)	(24.1, **15.9**, 10.8)	(17.0, **12.0**, 7.9)	1.38	2.39	4.24
10	(43.7, **35.6**, 15.7)	(21.8, **14.4**, 9.8)	(15.8, **11.0**, 6.9)	1.56	3.23	5.05

6.3 Example: power modeling for circuit design

Consider a power dissipation model,

$$P = V_{\text{dd}}^2 + 30V_{\text{dd}}e^{-(V_{\text{th}}-0.06V_{\text{dd}})/0.039}, \tag{41}$$

Table 4 RMS log errors for the circuit design power model in Section 6.3. For $K = 2$ through $K = 4$, SMA functions outperformed MA functions by a factor of 2.5 to 44.

K	MA	SMA	ISMA
1	0.0904	0.0904	0.0904
2	0.0229	0.0092	0.0091
3	0.01260	0.00037	0.00034
4	0.00760	0.00017	0.00014

which is relevant for GP-based circuit design. This example comes from [6], and is also studied in [11]. The power relationship (41) is to be modeled as a posynomial constraint for the domain $1.0 \leq V_{\mathrm{dd}} \leq 2.0$ and $0.2 \leq V_{\mathrm{th}} \leq 0.4$.

Starting with 1000 samples of $\mathbf{u} = (V_{\mathrm{dd}}, V_{\mathrm{th}})$, uniformly drawn from the region of interest, each sample is associated with the corresponding $w = P(\mathbf{u})$. After applying the log transformation $(\mathbf{x}, y) = (\log \mathbf{u}, \log w)$, MA, SMA, and ISMA functions are fit to the transformed data set. Table 4 gives the resulting RMS log errors. Of course, any of the resulting models is easily converted to a posynomial constraint on V_{dd}, V_{th}, and P. For example, the $K = 3$ SMA model corresponds to a posynomial model,

$$P^{2.14} \geq 1.09 V_{\mathrm{dd}}^{4.27} V_{\mathrm{th}}^{0.112} + 7.79 \times 10^{-5} \frac{V_{\mathrm{dd}}^{4.75}}{V_{\mathrm{th}}^{6.44}} + 1.36 \times 10^{-7} \frac{V_{\mathrm{dd}}^{8.94}}{V_{\mathrm{th}}^{8.89}}, \qquad (42)$$

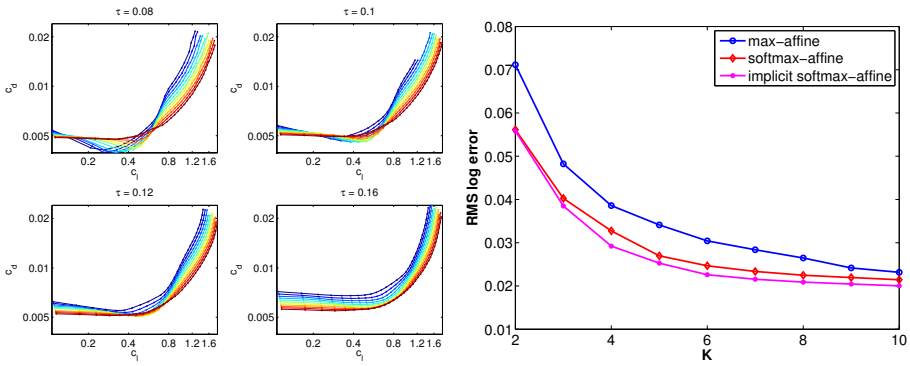
which is readily substituted directly into any GP.

6.4 Example: profile drag modeling for aircraft design

In aerospace engineering, *profile drag* on a wing arises due to viscous skin friction in the boundary layer surrounding the airfoil. For a given airfoil cross section shape, the profile drag coefficient (c_d) depends on Reynolds number (Re), airfoil thickness ratio (τ), and lift coefficient (c_l). No analytical expression is available for this relationship, but it can be simulated using the viscous airfoil analysis program XFOIL [9].

The data for this example consists of 3073 samples, $((\mathrm{Re}, \tau, c_l), c_d)_i$, generated for Re ranging from 10^6 to 10^7, τ ranging from 8% to 16%, and c_l ranging from 0.01 to stall. As before, a log transformation, $(\mathbf{x}, y) = (\log(\mathrm{Re}, \tau, c_l), \log c_d)$ is applied. Algorithm 2 then fits MA, SMA, and ISMA functions for a range of K values.

As shown in Figure 5, for a given number of terms K, an ISMA model provides the best fit, followed by SMA, and finally by the MA fit. A related interpretation is that ISMA models can provide a desired level of maximum RMS error with no more, and often fewer, terms than the best MA model. Each of the fitted models is directly compatible with geometric programming, and can therefore represent profile drag relations in practical aircraft design problems [13].



(a) Profile drag data (b) GP-compatible fitting error

Fig. 5 Approximating profile drag for symmetric NACA airfoils. Here the data consists of 3073 samples of profile drag coefficient c_d as a function of lift coefficient (c_l), Reynolds number (Re), and airfoil thickness coefficient (τ). Each model class was fit for a range of K, with implicit softmax-affine functions achieving the best fit. For each K, 20 random restarts (initializations) were used, and, of these, the best fit achieved was chosen.

7 Conclusions

This article discusses methods for fitting GP-compatible functions to data. Specifically, the focus of this work is on fitting special classes of convex functions to logarithmically transformed data. Two convex function classes are introduced: softmax-affine, and implicit softmax-affine. In Section 3, it is shown that these functions form a hierarchy, with the most general ISMA function class able to reproduce the other classes of convex functions for special parameter settings. One can therefore conclude that the best ISMA fit is always at least as good (and often much better than) the best fit from the more commonly used max-affine class.

Both of the proposed function classes have parallel interpretations as GP-compatible posynomials. Indeed, the proposed approach unifies max-affine fitting, max-monomial fitting, and posynomial fitting as special cases of SMA and ISMA fitting. The most general ISMA function class leverages the full expressive power of GP, by using an implicit representation corresponding to a posynomial constraint $g([\mathbf{u}, w]) \leq 1$.

While the focus of this article is on the form of the function classes and their connection to GP, the ingredients of a practical fitting algorithm that can be used to fit these functions to data are also presented.

Acknowledgements The authors thank Aude Hofleitner and Timothy Hunter for their thorough and insightful comments on the draft. This work was supported by a National Science Foundation Graduate Research Fellowship.

References

1. Agarwal, S., Mierle, K., Others: Ceres solver. http://ceres-solver.org
2. Babakhani, A., Lavaei, J., Doyle, J., Hajimiri, A.: Finding globally optimum solutions in antenna optimization problems. IEEE International Symposium on Antennas and Propagation (2010)
3. Beightler, C.S., Phillips, D.T.: Applied geometric programming, vol. 150. Wiley New York (1976)
4. Boyd, S., Kim, S.J., Vandenberghe, L., Hassibi, A.: A tutorial on geometric programming. Optimization and Engineering (2007)
5. Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press, New York, NY, USA (2004)
6. Boyd, S.P., Kim, S.J., Patil, D.D., Horowitz, M.A.: Digital circuit optimization via geometric programming. Operations Research **53**, 899–932 (2005)
7. Chiang, M.: Geometric programming for communication systems. Commun. Inf. Theory **2**, 1–154 (2005). DOI 10.1516/0100000005. URL http://portal.acm.org/citation.cfm?id=1166381.1166382
8. Daems, W., Gielen, G., Sansen, W.: Simulation-based generation of posynomial performance models for the sizing of analog integrated circuits. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on **22**(5), 517–534 (2003)
9. Drela, M.: Xfoil subsonic airfoil development system (2000). Open source software available at http://web.mit.edu/drela/Public/web/xfoil/
10. Duffin, R.J., Peterson, E.L., Zener, C.: Geometric programming: theory and application. Wiley New York (1967)
11. Hannah, L., Dunson, D.: Ensemble methods for convex regression with applications to geometric programming based circuit design. arXiv preprint arXiv:1206.4645 (2012)
12. Hannah, L.A., Dunson, D.B.: Multivariate convex regression with adaptive partitioning. arXiv preprint arXiv:1105.1924 (2011)
13. Hoburg, W., Abbeel, P.: Geometric programming for aircraft design optimization. AIAA Journal **52** (2014)
14. Kasamsetty, K., Ketkar, M., Sapatnekar, S.S.: A new class of convex functions for delay modeling and its application to the transistor sizing problem [cmos gates]. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on **19**(7), 779–788 (2000)
15. Kim, J., Vandenberghe, L., Yang, C.K.K.: Convex piecewise-linear modeling method for circuit optimization via geometric programming. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on **29**(11), 1823 –1827 (2010). DOI 10.1109/TCAD.2010.2053151
16. Li, X., Gopalakrishnan, P., Xu, Y., Pileggi, T.: Robust analog/rf circuit design with projection-based posynomial modeling. In: Proceedings of the 2004 IEEE/ACM International conference on Computer-aided design, pp. 855–862. IEEE Computer Society (2004)
17. Magnani, A., Boyd, S.P.: Convex piecewise-linear fitting. Optimization and Engineering **10**, 1–17 (2009). DOI 10.1007/s11081-008-9045-3
18. Marquardt, D.W.: An algorithm for least-squares estimation of nonlinear parameters. Journal of the Society for Industrial and Applied Mathematics **11**(2), pp. 431–441 (1963). URL http://www.jstor.org/stable/2098941
19. Mehrotra, S.: On the implementation of a primal-dual interior point method. SIAM Journal on optimization **2**(4), 575–601 (1992)
20. Nesterov, Y., Nemirovsky, A.: Interior-point polynomial methods in convex programming, volume 13 of studies in applied mathematics. SIAM, Philadelphia, PA (1994)
21. Nocedal, J., Wright, S.J.: Numerical optimization. Springer Science+ Business Media (2006)
22. Oliveros, J., Cabrera, D., Roa, E., Van Noije, W.: An improved and automated design tool for the optimization of cmos otas using geometric programming. In: Proceedings of the 21st annual symposium on Integrated circuits and system design, pp. 146–151. ACM (2008)

23. Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical recipes 3rd edition: The art of scientific computing. Cambridge university press (2007)
24. Wilde, D.: Globally optimal design. Wiley interscience publication. Wiley (1978). URL http://books.google.com/books?id=XYBRAAAAMAAJ
25. Ypma, T.J.: Historical development of the newton-raphson method. SIAM review **37**(4), 531–551 (1995)