Optim Eng (2009) 10: 1–17
DOI 10.1007/s11081-008-9045-3

Convex piecewise-linear fitting

Alessandro Magnani · Stephen P. Boyd

Received: 14 April 2006 / Accepted: 4 March 2008 / Published online: 25 March 2008

Abstract We consider the problem of fitting a convex piecewise-linear function, with some specified form, to given multi-dimensional data. Except for a few special cases, this problem is hard to solve exactly, so we focus on heuristic methods that find locally optimal fits. The method we describe, which is a variation on the K-means algorithm for clustering, seems to work well in practice, at least on data that can be fit well by a convex function. We focus on the simplest function form, a maximum of a fixed number of affine functions, and then show how the methods extend to a more general form.

Keywords Convex optimization · Piecewise-linear approximation · Data fitting

1 Convex piecewise-linear fitting problem

We consider the problem of fitting some given data

$$(u_1, y_1), \ldots, (u_m, y_m) \in \mathbf{R}^n \times \mathbf{R}$$

with a convex piecewise-linear function $f : \mathbf{R}^n \to \mathbf{R}$ from some set $\mathcal{F}$ of candidate functions. With a least-squares fitting criterion, we obtain the problem

$$\begin{array}{ll} \text{minimize} & J(f) = \displaystyle\sum_{i=1}^{m} (f(u_i) - y_i)^2 \\ \text{subject to} & f \in \mathcal{F}, \end{array} \tag{1}$$

A. Magnani · S.P. Boyd (✉)
Electrical Engineering Department, Stanford University, Stanford, CA 94305, USA
e-mail: boyd@stanford.edu

A. Magnani
e-mail: alem@stanford.edu

with variable f. We refer to $(J(f)/m)^{1/2}$ as the RMS (root-mean-square) fit of the function f to the data. The convex piecewise-linear fitting problem (1) is to find the function f, from the given family $\mathcal{F}$ of convex piecewise-linear functions, that gives the best (smallest) RMS fit to the given data.

Our main interest is in the case when n (the dimension of the data) is relatively small, say not more than 5 or so, while m (the number of data points) can be relatively large, e.g., 10^4 or more. The methods we describe, however, work for any values of n and m.

Several special cases of the convex piecewise-linear fitting problem (1) can be solved exactly. When $\mathcal{F}$ consists of the affine functions, i.e., f has the form $f(x) = a^T x + b$, the problem (1) reduces to an ordinary linear least-squares problem in the function parameters $a \in \mathbf{R}^n$ and $b \in \mathbf{R}$ and so is readily solved. As a less trivial example, consider the case when $\mathcal{F}$ consists of *all* piecewise-linear functions from $\mathbf{R}^n$ into $\mathbf{R}$, with no other constraint on the form of f. This is the *nonparametric* convex piecewise-linear fitting problem. Then the problem (1) can be solved, exactly, via a quadratic program (QP); see (Boyd and Vandenberghe 2004, Sect. 6.5.5). This nonparametric approach, however, has two potential practical disadvantages. First, the QP that must be solved is very large (containing more than mn variables), limiting the method to modest values of m (say, a thousand). The second potential disadvantage is that the piecewise-linear function fit obtained can be very complex, with many terms (up to m).

Of course, not all data can be fit well (i.e., with small RMS fit) with a convex piecewise-linear function. For example, if the data are samples from a function that has strong negative (concave) curvature, then no convex function can fit it well. Moreover, the best fit (which will be poor) will be obtained with an affine function. We can also have the opposite situation: it can occur that the data can be *perfectly* fit by an affine function, i.e., we can have $J = 0$. In this case we say that the data is *interpolated* by the convex piecewise-linear function f.

1.1 Max-affine functions

In this paper we consider the parametric fitting problem, in which the candidate functions are parametrized by a finite-dimensional vector of coefficients $\alpha \in \mathbf{R}^p$, where p is the number of parameters needed to describe the candidate functions. One very simple form is given by $\mathcal{F}^k_{\mathrm{ma}}$, the set of functions on $\mathbf{R}^n$ with the form

$$f(x) = \max\{a_1^T x + b_1, \ldots, a_k^T x + b_k\}, \tag{2}$$

i.e., a maximum of k affine functions. We refer to a function of this form as 'max-affine', with k terms. The set $\mathcal{F}^k_{\mathrm{ma}}$ is parametrized by the coefficient vector

$$\alpha = (a_1, \ldots, a_k, b_1, \ldots, b_k) \in \mathbf{R}^{k(n+1)}.$$

In fact, any convex piecewise-linear function on $\mathbf{R}^n$ can be expressed as a max-affine function, for some k, so this form is in a sense universal. Our interest, however, is in the case when the number of terms k is relatively small, say no more than 10, or a few 10s. In this case the max-affine representation (2) is compact, in the sense

that the number of parameters needed to describe f (i.e., p) is much smaller than the number of parameters in the original data set (i.e., $m(n+1)$). The methods we describe, however, do not require k to be small.

When $\mathcal{F} = \mathcal{F}_{\mathrm{ma}}^k$, the fitting problem (1) reduces to the nonlinear least-squares problem

$$\text{minimize} \quad J(\alpha) = \sum_{i=1}^{m} \left(\max_{j=1,\ldots,k} (a_j^T u_i + b_j) - y_i \right)^2, \tag{3}$$

with variables $a_1, \ldots, a_k \in \mathbf{R}^n$, $b_1, \ldots, b_k \in \mathbf{R}$. The function J is a piecewise-quadratic function of α. Indeed, for each i, $f(u_i) - y_i$ is piecewise-linear, and J is the sum of squares of these functions, so J is convex quadratic on the (polyhedral) regions on which $f(u_i)$ is affine. But J is not globally convex, so the fitting problem (3) is not convex.

1.2 A more general parametrization

We will also consider a more general parametrized form for convex piecewise-linear functions,

$$f(x) = \psi(\phi(x, \alpha)), \tag{4}$$

where $\psi : \mathbf{R}^q \to \mathbf{R}$ is a (fixed) convex piecewise-linear function, and $\phi : \mathbf{R}^n \times \mathbf{R}^p \to \mathbf{R}^q$ is a (fixed) bi-affine function. (This means that for each x, $\phi(x, \alpha)$ is an affine function of α, and for each α, $\phi(x, \alpha)$ is an affine function of x.) The simple max-affine parametrization (2) has this form, with $q = k$, $\psi(z_1, \ldots, z_k) = \max\{z_1, \ldots, z_k\}$, and $\phi_i(x, \alpha) = a_i^T x + b_i$.

As an example, consider the set of functions $\mathcal{F}$ that are sums of k terms, each of which is the maximum of two affine functions,

$$f(x) = \sum_{i=1}^{k} \max\{a_i^T x + b_i, c_i^T x + d_i\}, \tag{5}$$

parametrized by $a_1, \ldots, a_k$, $c_1, \ldots, c_k \in \mathbf{R}^n$ and $b_1, \ldots, b_k$, $d_1, \ldots, d_k \in \mathbf{R}$. This family corresponds to the general form (4) with

$$\psi(z_1, \ldots, z_k, w_1, \ldots, w_k) = \sum_{i=1}^{k} \max\{z_i, w_i\},$$

and

$$\phi(x, \alpha) = (a_1^T x + b_1, \ldots, a_k^T x + b_k, c_1^T x + d_1, \ldots, c_k^T x + d_k).$$

Of course we can expand any function with the more general form (4) into its max-affine representation. But the resulting max-affine representation can be very much larger than the original general form representation. For example, the function form (5) requires $p = 2k(n+1)$ parameters. If the same function is written out as a max-affine function, it requires 2^k terms, and therefore $2^k(n+1)$ parameters. The

hope is that a well chosen general form can give us a more compact fit to the given data than a max-affine form with the same number of parameters.

As another interesting example of the general form (4), consider the case in which f is given as the optimal value of a linear program (LP) with the right-hand side of the constraints depending bi-affinely on x and the parameters:

$$f(x) = \min\{c^T v \mid Av \leq b + Bx\}.$$

Here c and A are fixed; b and B are considered the parameters that define f. This function can be put in the general form (4) using

$$\psi(z) = \min\{c^T v \mid Av \leq z\}, \qquad \phi(x, b, B) = b + Bx.$$

The function ψ is convex and piecewise-linear (see, e.g., Boyd and Vandenberghe 2004); the function ϕ is evidently bi-affine in x and (b, B).

1.3 Dependent variable transformation and normalization

We can apply a nonsingular affine transformation to the dependent variable u, by forming

$$\tilde{u}_i = Tu_i + s, \quad i = 1, \ldots, m,$$

where $T \in \mathbf{R}^{n\times n}$ is nonsingular and $s \in \mathbf{R}^n$. Defining $\tilde{f}(\tilde{x}) = f(T^{-1}(x - s))$, we have $\tilde{f}(\tilde{u}_i) = f(u_i)$. If f is piecewise-linear and convex, then so is $\tilde{f}$ (and of course, vice versa). Provided $\mathcal{F}$ is invariant under composition with affine functions, the problem of fitting the data (u_i, y_i) with a function $f \in \mathcal{F}$ is the same as the problem of fitting the data $(\tilde{u}_i, y_i)$ with a function $\tilde{f} \in \mathcal{F}$.

This allows us to normalize the dependent variable data in various ways. For example, we can assume that it has zero (sample) mean and unit (sample) covariance,

$$\bar{u} = (1/m)\sum_{i=1}^{m} u_i = 0, \qquad \Sigma_u = (1/m)\sum_{i=1}^{m} u_i u_i^T = I, \tag{6}$$

provided the data u_i are affinely independent. (If they are not, we can reduce the problem to an equivalent one with smaller dimension.)

1.4 Outline

In Sect. 2 we describe several applications of convex piecewise-linear fitting. In Sect. 3, we describe a basic heuristic algorithm for (approximately) solving the max-affine fitting problem (1). This basic algorithm has several shortcomings, such as convergence to a poor local minimum, or failure to converge at all. By running this algorithm a modest number of times, from different initial points, however, we obtain a fairly reliable algorithm for least-squares fitting of a max-affine function to given data. Finally, we show how the algorithm can be extended to handle the more general function parametrization (4). In Sect. 4 we present some numerical examples.

1.5 Previous work

Piecewise-linear functions arise in many areas and contexts. Some general forms for representing piecewise-linear functions can be found in, e.g., Kang and Chua, Kahlert and Chua (1978, 1990). Several methods have been proposed for fitting general piecewise-linear functions to (multidimensional) data. A neural network algorithm is used in Gothoskar et al. (2002); a Gauss-Newton method is used in Julian et al., Horst and Beichel (1998, 1997) to find piecewise-linear approximations of smooth functions. A recent reference on methods for least-squares with semismooth functions is Kanzow and Petra (2004). An iterative procedure, similar in spirit to our method, is described in Ferrari-Trecate and Muselli (2002). Software for fitting general piecewise-linear functions to data include, e.g., Torrisi and Bemporad (2004), Storace and De Feo (2002).

The special case $n = 1$, i.e., fitting a function on $\mathbf{R}$, by a piecewise-linear function has been extensively studied. For example, a method for finding the minimum number of segments to achieve a given maximum error is described in Dunham (1986); the same problem can be approached using dynamic programming (Goodrich 1994; Bellman and Roth 1969; Hakimi and Schmeichel 1991; Wang et al. 1993), or a genetic algorithm (Pittman and Murthy 2000). The problem of simplifying a given piecewise-linear function on $\mathbf{R}$, to one with fewer segments, is considered in Imai and Iri (1986).

Another related problem that has received much attention is the problem of fitting a piecewise-linear curve, or polygon, in $\mathbf{R}^2$ to given data; see, e.g., Aggarwal et al. (1985), Mitchell and Suri (1992). An iterative procedure, closely related to the k-means algorithm and therefore similar in spirit to our method, is described in Phillips and Rosenfeld (1988), Yin (1998).

Piecewise-linear functions and approximations have been used in many applications, such as detection of patterns in images (Rives et al. 1985), contour tracing (Dobkin et al. 1990), extraction of straight lines in aerial images (Venkateswar and Chellappa 1992), global optimization (Mangasarian et al. 2005), compression of chemical process data (Bakshi and Stephanopoulos 1996), and circuit modeling (Julian et al. 1998; Chua and Deng 1986; Vandenberghe et al. 1989).

We are aware of only two papers which consider the problem of fitting a piecewise-linear convex function to given data. Mangasarian et al. (2005) describe a heuristic method for fitting a piecewise-linear convex function of the form $a + b^T x + \|Ax + c\|_1$ to given data (along with the constraint that the function underestimate the data). The focus of their paper is on finding piecewise-linear convex underestimators for known (nonconvex) functions, for use in global optimization; our focus, in contrast, is on simply fitting some given data. The closest related work that we know of is Kim et al. (2004). In this paper, Kim et al. describe a method for fitting a (convex) max-affine function to given data, increasing the number of terms to get a better fit. (In fact they describe a method for fitting a max-monomial function to circuit models; see Sect. 2.3.)

2 Applications

In this section we briefly describe some applications of convex piecewise-linear fitting. None of this material is used in the sequel.

2.1 LP modeling

One application is in LP modeling, i.e., approximately formulating a practical problem as an LP. Suppose a problem is reasonably well modeled using linear equality and inequality constraints, with a few nonlinear inequality constraints. By approximating these nonlinear functions by convex piecewise-linear functions, the overall problem can be formulated as an LP, and therefore efficiently solved.

As an example, consider a minimum fuel optimal control problem, with linear dynamics and a nonlinear fuel-use function,

$$\begin{array}{ll} \text{minimize} & \displaystyle\sum_{t=0}^{T-1} f(u(t)) \\ \text{subject to} & x(t+1) = A(t)x(t) + B(t)u(t), \quad t = 0, \ldots, T-1, \\ & x(0) = x_{\text{init}}, \quad x(T) = x_{\text{des}}, \end{array}$$

with variables $x(0), \ldots, x(T) \in \mathbf{R}^n$ (the state trajectory), and $u(0), \ldots, u(T-1) \in \mathbf{R}^m$ (the control input). The problem data are $A(0), \ldots, A(T-1)$ (the dynamics matrices), $B(0), \ldots, B(T-1)$ (the control matrices), x_{init} (the initial state), and x_{des} (the desired final state). The function $f : \mathbf{R}^m \to \mathbf{R}$ is the fuel-use function, which gives the fuel consumed in one period, as a function of the control input value. Now suppose we have empirical data or measurements of some values of the control input $u \in \mathbf{R}^m$, along with the associated fuel use $f(u)$. If we can fit these data with a convex piecewise-linear function, say,

$$f(u) \approx \hat{f}(u) = \max_{j=1,\ldots,k} (a_j^T u + b_j),$$

then we can formulate the (approximate) minimum fuel optimal control problem as the LP

$$\begin{array}{ll} \text{minimize} & \displaystyle\sum_{t=0}^{T-1} \tilde{f}(t) \\ \text{subject to} & x(t+1) = A(t)x(t) + B(t)u(t), \quad t = 0, \ldots, T-1, \\ & x(0) = x_{\text{init}}, \qquad x(T) = x_{\text{des}}, \\ & \tilde{f}(t) \geq a_j^T u(t) + b_j, \quad t = 0, \ldots, T-1, \ j = 1, \ldots, k, \end{array} \tag{7}$$

with variables $x(0), \ldots, x(T) \in \mathbf{R}^n$, $u(0), \ldots, u(T-1) \in \mathbf{R}^m$, and $\tilde{f}(0), \ldots, \tilde{f}(T-1) \in \mathbf{R}$.

2.2 Simplifying convex functions

Another application of convex piecewise-linear fitting is to simplify a convex function that is complex, or expensive to evaluate. To illustrate this idea, we continue our minimum fuel optimal control problem described above, with a piecewise-linear fuel use function. Consider the function $V : \mathbf{R}^n \to \mathbf{R}$, which maps the initial state x_{init} to its associated minimum fuel use, i.e., the optimal value of the LP (7). (This is the Bellman value function for the optimal control problem.) The value function is piecewise-linear and convex, but very likely requires an extremely large number of terms to be expressed in max-affine form. We can (possibly) form a simple approximation of V by a max-affine function with many fewer terms, as follows. First, we evaluate V via the LP (7), for a large number of initial conditions. Then, we fit a max-affine function with a modest number of terms to the resulting data. This convex piecewise-linear approximate value function can be used to construct a simple feedback controller that approximately minimizes fuel use; see, e.g., Bemporad et al. (2002).

2.3 Max-monomial fitting for geometric programming

Max-affine fitting can be used to find a max-monomial approximation of a positive function, for use in geometric programming modeling; see Boyd et al. (2006). Given data $(z_i, w_i) \in \mathbf{R}^n_{++} \times \mathbf{R}_{++}$, we form

$$u_i = \log z_i, \qquad y_i = \log w_i, \quad i = 1, \ldots, m.$$

(The log of a vector is interpreted as componentwise.) We now fit this data with a max-affine model,

$$y_i \approx \max\{a_1^T u_i + b_1, \ldots, a_k^T u_i + b_k\}.$$

This gives us the max-monomial model

$$w_i \approx \max\{g_1(z_i), \ldots, g_K(z_i)\},$$

where g_i are the monomial functions

$$g_j(z) = e^{b_i} z_1^{a_{j1}} \cdots z_n^{a_{jn}}, \quad j = 1, \ldots, K.$$

(These are not monomials in the standard sense, but in the sense used in geometric programming.)

3 Least-squares partition algorithm

3.1 The algorithm

In this section we present a heuristic algorithm to (approximately) solve the k-term max-affine fitting problem (3), i.e.,

$$\text{minimize} \quad J = \sum_{i=1}^{m} \left(\max_{j=1,\ldots,k} (a_j^T u_i + b_j) - y_i \right)^2,$$

with variables $a_1, \ldots, a_k \in \mathbf{R}^n$ and $b_1, \ldots, b_k \in \mathbf{R}$. The algorithm alternates between partitioning the data and carrying out least-squares fits to update the coefficients.

We let $P_j^{(l)}$ for $j = 1, \ldots, k$, be a partition of the data indices at the lth iteration, i.e., $P_j^{(l)} \subseteq \{1, \ldots, m\}$, with

$$\bigcup_j P_j^{(l)} = \{1, \ldots, m\}, \qquad P_i^{(l)} \cap P_j^{(l)} = \emptyset \quad \text{for } i \neq j.$$

(We will describe methods for choosing the initial partition $P_j^{(0)}$ later.)

Let $a_j^{(l)}$ and $b_j^{(l)}$ denote the values of the parameters at the lth iteration of the algorithm. We generate the next values, $a_j^{(l+1)}$ and $b_j^{(l+1)}$, from the current partition $P_j^{(l)}$, as follows. For each $j = 1, \ldots, k$, we carry out a least-squares fit of $a_j^T u_i + b_j$ to y_i, using only the data points with $i \in P_j^{(l)}$. In other words, we take $a_j^{(l+1)}$ and $b_j^{(l+1)}$ as values of a and b that minimize

$$\sum_{i \in P_j^{(l)}} (a^T u_i + b - y_i)^2. \tag{8}$$

In the simplest (and most common) case, there is a unique pair (a, b) that minimizes (8), i.e.,

$$\begin{bmatrix} a_j^{(l+1)} \\ b_j^{(l+1)} \end{bmatrix} = \begin{bmatrix} \sum u_i u_i^T & \sum u_i \\ \sum u_i^T & |P_j^{(l)}| \end{bmatrix}^{-1} \begin{bmatrix} \sum y_i u_i \\ \sum y_i \end{bmatrix}, \tag{9}$$

where the sums are over $i \in P_j^{(l)}$.

When there are multiple minimizers of the quadratic function (8), i.e., the matrix to be inverted in (9) is singular, we have several options. One option is to add some regularization to the simple least-squares objective in (8), i.e., an additional term of the form $\lambda \|a\|_2^2 + \mu b^2$, where λ and μ are positive constants. Another possibility is to take the updated parameters as the unique minimizer of (8) that is closest to the previous value, $(a_j^{(l)}, b_j^{(l)})$, in Euclidean norm.

Using the new values of the coefficients, we update the partition to obtain $P_j^{(l+1)}$, by assigning i to $P_s^{(l+1)}$ if

$$f^{(l)}(u_i) = \max_{s=1,\ldots,k} (a_s^{(l)T} u_i + b_s^{(l)}) = a_j^{(l)T} u_i + b_j^{(l)}. \tag{10}$$

(This means that the term $a_j^{(l)T} u_i + b_j^{(l)}$ is 'active' at the data point u_i.) Roughly speaking, this means that $P_j^{(l+1)}$ is the set of indices for which the affine function $a_j^T z + b_j$ is the maximum; we can break ties (if there are any) arbitrarily.

This iteration is run until convergence, which occurs if the partition at an iteration is the same as the partition at the previous iteration, or some maximum number of iterations is reached.

We can write the algorithm as

LEAST-SQUARES PARTITION ALGORITHM.

given partition $P_1^{(0)}, \ldots, P_K^{(0)}$ of $\{1, \ldots, m\}$, iteration limit $l_{\max}$

for $l = 0, \ldots, l_{\max}$

1. Compute $a_j^{(l+1)}$ and $b_j^{(l+1)}$ as in (9).
2. Form the partition $P_1^{(l+1)}, \ldots, P_k^{(l+1)}$ as in (10).
3. Quit if $P_j^{(l)} = P_j^{(l+1)}$ for $j = 1, \ldots, k$.

During the execution of the least-squares partition algorithm, one or more of the sets $P_j^{(l)}$ can become empty. The simplest approach is to drop empty sets from the partition, and continue with a smaller value of k.

3.2 Interpretation as Gauss-Newton method

We can interpret the algorithm as a Gauss-Newton method for the problem (3). Suppose that at a point $u \in \mathbf{R}^n$, there is a unique j for which $f(u) = a_j^T u + b_j$ (i.e., there are no ties in the maximum that defines $f(u)$). In this case the function f is differentiable with respect to a and b; indeed, it is locally affine in these parameter values. Its first order approximation at a, b is

$$f(u) \approx \hat{f}(u) = \tilde{a}_j^T u + \tilde{b}_j.$$

This approximation is exact, provided the perturbed parameter values $\tilde{a}_1, \ldots, \tilde{a}_k$, $\tilde{b}_1, \ldots, \tilde{a}_b$ are close enough to the parameter values $a_1, \ldots, a_k, b_1, \ldots, a_b$.

Now assume that for each data point u_i, there is a unique j for which $f(u_i) = a_j^{(l)T} u_i + b_j^{(l)}$ (i.e., there are no ties in the maxima that define $f(u_i)$). Then the first order approximation of $(f(u_1), \ldots, f(u_m))$ is given by

$$f(u_i) \approx \hat{f}(u_i) = \tilde{a}_{j(i)}^T u_i + \tilde{b}_{j(i)},$$

where $j(i)$ is the unique active j at u_i, i.e., $i \in P_j^{(l)}$.

In the Gauss-Newton method for a nonlinear least-squares problem, we form the first order approximation of the argument of the norm, and solve the resulting least-squares problem to get the next iterate. In this case, then, we form the linear least-squares problem of minimizing

$$\hat{J} = \sum_{i=1}^m \left(\hat{f}(u_i) - y_i\right)^2 = \sum_{i=1}^m \left(\tilde{a}_{j(i)}^T u_i + \tilde{b}_{j(i)} - y_i\right)^2,$$

over the variables $\tilde{a}_1, \ldots, \tilde{a}_k, \tilde{b}_1, \ldots, \tilde{b}_k$. We can re-arrange the sum defining J into terms involving each of the pairs of variables $a_1, b_1, \ldots, a_k, b_k$ separately:

$$\hat{J} = \hat{J}_1 + \cdots + \hat{J}_k,$$

where

$$\hat{J}_j = \sum_{i \in P_j^{(l)}} (\tilde{a}^T u_i + \tilde{b} - y_i)^2, \quad j = 1, \ldots, k.$$

Evidently, we can minimize $\hat{J}$ by separately minimizing each $\hat{J}_i$. Moreover, the parameter values that minimize $\hat{J}$ are precisely $a_1^{(l+1)}, \ldots, a_k^{(l+1)}, b_1^{(l+1)}, \ldots, b_k^{(l+1)}$. This is exactly the least-squares partition algorithm described above.

The algorithm is closely related to the k-means algorithm used in least-squares clustering (Gersho and Gray 1991). The k-means algorithm approximately solves the problem of finding a set of k points in $\mathbf{R}^n$, $\{z_1, \ldots, z_k\}$, that minimizes the mean square Euclidean distance to a given data set $u_1, \ldots, u_m \in \mathbf{R}^n$. (The distance between a point u and the set of points $\{z_1, \ldots, z_k\}$ is defined as the minimum distance, i.e., $\min_{j=1,\ldots,k} \|u - z_j\|_2$.) In the k-means algorithm, we iterate between two steps: first, we partition the data points according to the closest current point in the set $\{z_1, \ldots, z_k\}$; then we update each z_j as the mean of the points in its associated partition. (The mean minimizes the sum of the squares of the Euclidean distances to the point.) Our algorithm is conceptually identical to the k-means algorithm: we partition the data points according to which of the affine functions is active (i.e., largest), and then update the affine functions, separately, using only the data points in its associated partition.

3.3 Nonconvergence of least-squares partition algorithm

The basic least-squares partition algorithm need not converge; it can enter a (nonconstant) limit cycle. Consider, for example, the data

$$\begin{array}{lllll} u_1 = -2, & u_2 = -1, & u_3 = 0, & u_4 = 1, & u_5 = 2, \\ y_1 = 0, & y_2 = 1, & y_3 = 3, & y_4 = 1, & y_5 = 0, \end{array}$$

and $k = 2$. The data evidently cannot be fit well by any convex function; the (globally) best fit is obtained by the constant function $f(u) = 1$. For many initial parameter values, however, the algorithm converges to a limit cycle with period 2, alternating between the two functions

$$f_1(u) = \max\{u + 2, -(3/2)u + 17/6\}, \qquad f_2(u) = \max\{(3/2)u + 17/6, -u + 2\}.$$

The algorithm therefore fails to converge; moreover, each of the functions f_1 and f_2 gives a very suboptimal fit to the data.

On the other hand, with real data (not specifically designed to illustrate nonconvergence) we have observed that the least-squares partition algorithm appears to converge in most cases. In any case, convergence failure has no practical consequences since the algorithm is terminated after some fixed maximum number of steps, and moreover, we recommend that it be run from a number of starting points, with the best fit obtained used as the final fit.

3.4 Piecewise-linear fitting algorithm

The least-squares partition algorithm, used by itself, has several serious shortcomings. It need not converge, and when it does converge, it can (and often does) converge to a piecewise-linear approximation with a poor fit to the data. Both of these problems can be mitigated by running the least-squares partition algorithm multiple times, with different initial partitions. The final fit is taken to be the best fit obtained among all iterations of all runs of the algorithm.

We first describe a simple method for generating a random initial partition. We randomly choose points $p_1, \ldots, p_K$, and define the initial partition to be the Voronoi sets associated with these points. We have

$$P_j^{(0)} = \{i \mid \|u_i - p_j\| < \|u_i - p_s\| \text{ for } s \neq j\}, \quad j = 1, \ldots, K. \tag{11}$$

(Thus, $P_j^{(0)}$ is the set of indices of data points that are closest to p_j.) The seed points p_i should be generated according to some distribution that matches the shape of the data points u_i, for example, they can be chosen from a normal distribution with mean $\bar{u}$ and covariance Σ_u (see (6)).

The overall algorithm can be described as

PIECEWISE-LINEAR FITTING ALGORITHM.

given number of trials N_{trials}, iteration limit $l_{\max}$

for $i = 1, \ldots, N_{\text{trials}}$

1. Generate random initial partition via (11).
2. Run least-squares partition algorithm with iteration limit $l_{\max}$.
3. Keep track of best RMS fit obtained.

3.5 General form fitting

In this section we show the least-squares partition algorithm can be modified to fit piecewise-linear functions with the more general form (4),

$$f(x, \alpha) = \psi(\phi(x, \alpha)),$$

where ψ is a fixed convex piecewise-linear function, and ϕ is a fixed bi-affine function.

We described the least-squares partition algorithm in terms of a partition of the indices, according to which of the k affine functions is active at the point u_i. The same approach of an explicit partition will not work in the more general case, since the size of the partition can be extremely large. Instead, we start from the idea that the partition gives an approximation of $f(u_i)$ that is affine in α, and valid near $\alpha^{(l)}$. If there is no 'tie' at u_i (i.e., there is a unique affine function that achieves the maximum), then the affine approximation is exact in a neighborhood of the current parameter value $\alpha^{(l)}$.

We can do the same thing with the more general form. For each i, we find $a_i(\alpha)$ and $b_i(\alpha)$, both affine functions of α, so that

$$f(u_i, \alpha) \approx a_i(\alpha)^T u_i + b_i(\alpha)$$

for α near $\alpha^{(l)}$, the current value of the parameters. This approximation is exact in a neighborhood of $\alpha^{(l)}$ if $\psi(u_i, \alpha)$ is a point of differentiability of ψ. (For max-affine functions, this is the case when there is no 'tie' at u_i.) If it is not such a point, we can choose any subgradient model of $f(u_i, \alpha)$, i.e., any $a_i(\alpha)$ and $b_i(\alpha)$ for which

$$f(u_i, \alpha^{(l)}) = a_i(\alpha^{(l)})^T u_i + b_i(\alpha^{(l)}),$$

(the approximation is exact for $\alpha = \alpha^{(l)}$), and

$$f(u_i, \alpha) \geq a_i(\alpha)^T u_i + b_i(\alpha)$$

for all α. (In the case of max-affine functions, breaking any ties arbitrarily satisfies this condition.)

We then compute a new parameter value using a Gauss-Newton like method. We replace $f(u_i)$ in the expression for J with

$$\hat{f}^{(l)}(u_i, \alpha) = a_i(\alpha^{(l)})^T u_i + b_i(\alpha^{(l)}),$$

which is affine in α. We then choose $\alpha^{(l+1)}$ as the minimizer of

$$\hat{J} = \sum_{i=1}^{m} (\hat{f}^{(l)}(u_i) - y_i)^2,$$

which can be found using standard linear least-squares.

To damp this update rule, we can add a regularization term to $\hat{J}$, by choosing $\alpha^{(l+1)}$ as the minimizer of

$$\sum_{i=1}^{m} (\hat{f}^{(l)}(u_i) - y_i)^2 + \rho \|\alpha - \alpha^{(l)}\|^2,$$

where $\rho > 0$ is a parameter.

4 Numerical examples

In this section we show some numerical results, using the following data set. The dimension is $n = 3$, and we have $m = 11^3 = 1331$ points. The set of points u_i is given by $\mathcal{V} \times \mathcal{V} \times \mathcal{V}$, where $\mathcal{V} = \{-5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5\}$. The values are obtained as $y_i = g(u_i)$, where g is the (convex) function

$$g(x) = \log(\exp x_1 + \exp x_2 + \exp x_3).$$

We use the piecewise-linear fitting algorithm described in Sect. 3.4, with iteration limit $l_{\max} = 50$, and number of terms varying from $k = 0$ to $k = 20$. For $k = 0$, the fitting function is taken to be zero, so we report the RMS value of $y_1, \ldots, y_m$ as the RMS fit. For $k = 1$, the fit is the best affine fit to the data, which can be found using least-squares. Figure 1 shows the RMS fits obtained after $N_{\text{trials}} = 10$ trials (top curve), and after $N_{\text{trials}} = 100$ trials (bottom curve). These show that good fits

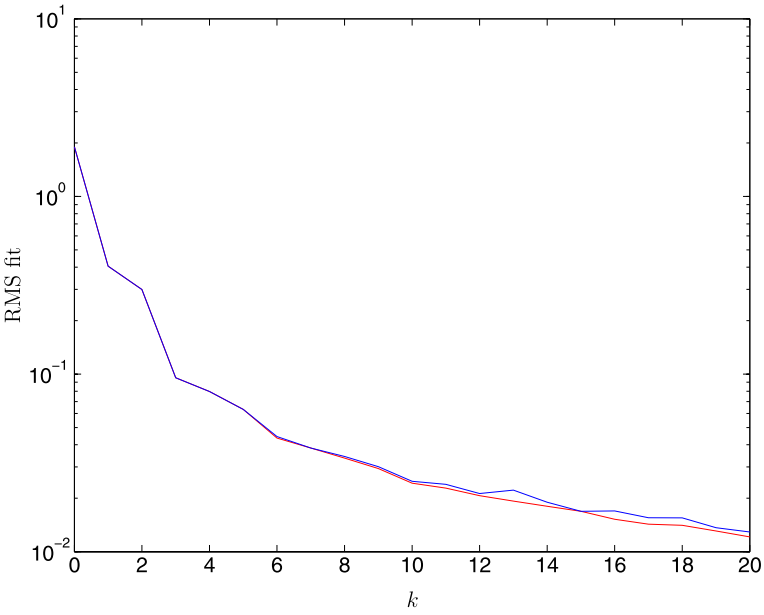

Fig. 1 Best RMS fit obtained with 10 trials (*top curve*) and 100 trials (*bottom curve*), versus number of terms k in max-affine function

are obtained with only 10 trials, and that (slightly) better ones are obtained with 100 trials.

To give an idea of the variation in RMS fit obtained with different trials, as well as the number of steps required for convergence (if it occurs), we fix the number of terms at $k = 12$, and run the least-squares partition algorithm 200 times, with a limit of 50 iterations, recording both the final RMS fit obtained, and the number of steps before convergence. (The number of steps is reported as 50 if the least-squares partition algorithm has not converged in 50 steps.) Figure 2 shows the histogram of RMS fit obtained. We can see that the fit is often, but not always, quite good; in just a few cases the fit obtained is poor. Evidently the best of even a modest number of trials will be quite good.

Figure 3 shows the distribution of the number of iterations of the least-squares partition algorithm required to converge. Convergence failed in 13 of the 200 trials; but in fact, the RMS fit obtained in these trials was not particularly bad. Typically convergence occurs within around 25 iterations.

In our last numerical example, we compare fitting the data with a max-affine function with k terms, and with the more general form

$$f(x) = \max_{i=1,\ldots,k/2} \left(a_i^T x + b_i\right) + \max_{i=k/2+1,\ldots,k} \left(a_i^T x + b_i\right),$$

parametrized by $a_1, \ldots, a_k \in \mathbf{R}^n$ and $b_1, \ldots, b_k \in \mathbf{R}$. (Note that the number of parameters in each function form is the same.) This function corresponds to the general

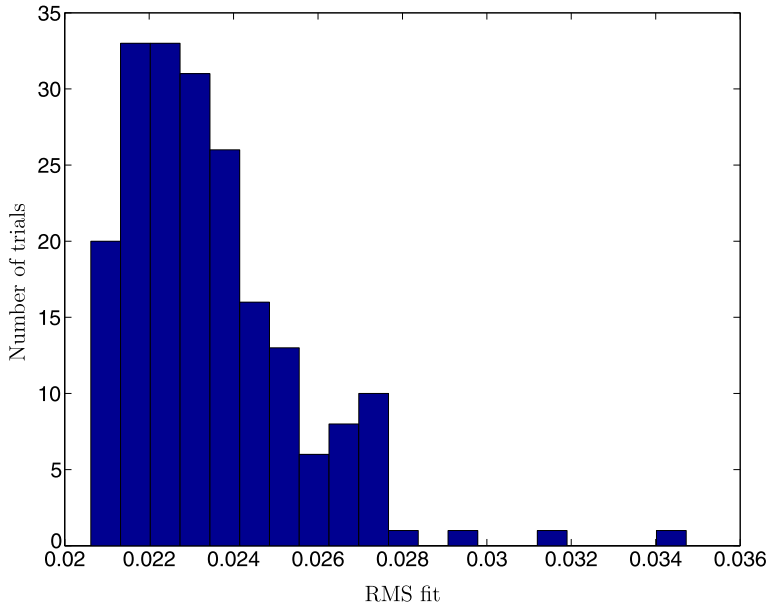

Fig. 2 Distribution of RMS fit obtained in 200 trials of least-squares partition algorithm, for $k = 12$, $l_{\max} = 50$

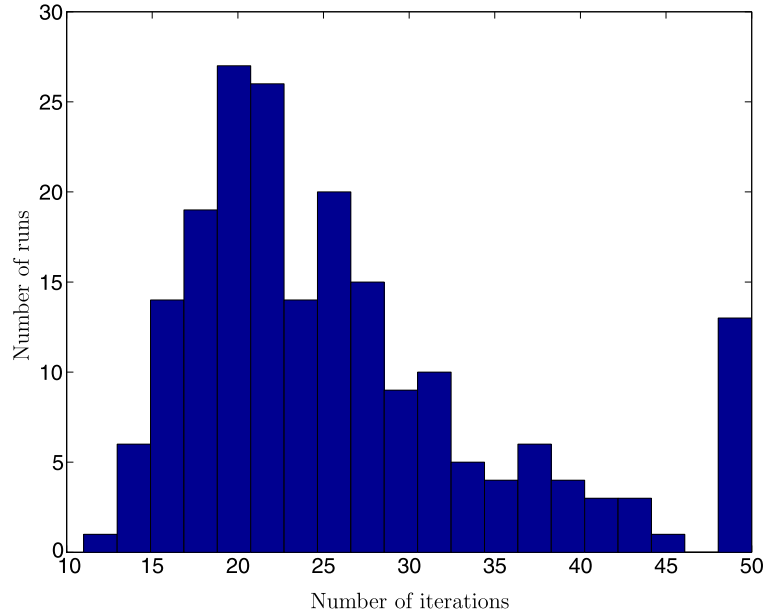

Fig. 3 Distribution of the number of steps required by least-squares partition algorithm to converge, over 200 trials. The number of steps is reported as 50 if convergence has not been obtained in 50 steps

form (4) with

$$\psi(z_1, \ldots, z_k) = \max_{i=1,\ldots,k/2} z_i + \max_{i=k/2+1,\ldots,k} z_i,$$

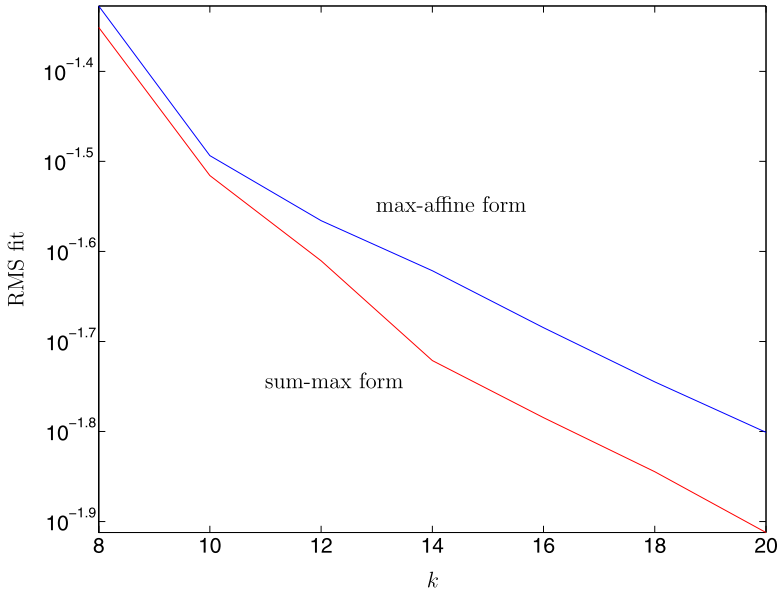

Fig. 4 Best RMS fit obtained for max-affine function (*top*) and sum-max function (*bottom*)

and

$$\phi(x, \alpha) = (a_1^T x + b_1, \ldots, a_k^T x + b_k).$$

We set the iteration limit for both forms as $l_{\max} = 100$, and take the best fit obtained in $N_{\text{trials}} = 10$ trials. We use the value $\rho = 10^{-5}$ for the regularization parameter in the general form algorithm. Figure 4 shows the RMS fit obtained for the two forms, versus k. Evidently the sum-max form gives (slightly) better RMS fit than the max-affine form.

5 Conclusions

We have described a new method for fitting a convex piecewise linear function to a given (possibly large) set of data (with a modest number of independent variables). The method is heuristic, since the algorithm can (and does) fail to converge to the globally optimal fit. Numerical examples suggest, however, that the method works very well in practice, on data that can be fit well by a convex function.

The method has many applications in practical optimization modeling. Data samples can be used to generate piecewise-linear convex functions, which in turn can be used to construct linear programming models.

Acknowledgement This work was carried out with support from C2S2, the MARCO Focus Center for Circuit and System Solutions, under MARCO contract 2003-CT-888.

We are grateful to Jim Koford for suggesting the problem.

References

Aggarwal A, Booth H, O'Rourke J, Suri S, Yap C (1985) Finding minimal convex nested polygons. In: Proceedings of the symposium on computational geometry. ACM Press, New York, pp 296–304

Bakshi B, Stephanopoulos G (1996) Compression of chemical process data by functional approximation and feature extraction. AIChE J 42(2):477–492

Bellman R, Roth R (1969) Curve fitting by segmented straight lines. Am Stat Assoc J 64:1079–1084

Bemporad A, Borrelli F, Morari M (2002) Model predictive control based on linear programming—the explicit solution. IEEE Trans Autom Control 47(12):1974–1985

Boyd S, Vandenberghe L (2004) Convex optimization. Cambridge University Press, Cambridge

Boyd S, Kim S, Vandenberghe L, Hassibi A (2006) A tutorial on geometric programming. Optim Eng

Chua L, Deng A (1986) Canonical piecewise-linear modeling. IEEE Trans Circuits Syst 33(5):511–525

Dobkin D, Levy S, Thurston W, Wilks A (1990) Contour tracing by piecewise linear approximations. ACM Trans Graph 9(4):389–423

Dunham J (1986) Optimum uniform piecewise linear approximation of planar curves. IEEE Trans Pattern Anal Mach Intell 8(1):67–75

Ferrari-Trecate G, Muselli M (2002) A new learning method for piecewise linear regression. In: Proceedings of the international conference on artificial neural networks. Springer, Berlin, pp 444–449

Gersho A, Gray R (1991) Vector quantization and signal compression. Kluwer Academic, Norwell

Goodrich M (1994) Efficient piecewise-linear function approximation using the uniform metric. In: Proceedings of the symposium on computational geometry. ACM Press, New York, pp 322–331

Gothoskar G, Doboli A, Doboli S (2002) Piecewise-linear modeling of analog circuits based on model extraction from trained neural networks. In: Proceedings of IEEE international workshop on behavioral modeling and simulation, pp 41–46

Hakimi S, Schmeichel E (1991) Fitting polygonal functions to a set of points in the plane. CVGIP: Graph Models Image Process 53(2):132–136

Horst J, Beichel I (1997) A simple algorithm for efficient piecewise-linear approximation of space curves. In: Proceedings of international conference on image processing. IEEE Computer Society, Los Alamitos

Imai H, Iri M (1986) An optimal algorithm for approximating a piecewise linear function. J Inf Process 9(3):159–162

Julian P, Jordan M, Desages A (1998) Canonical piecewise-linear approximation of smooth functions. IEEE Trans Circuits Syst 45(5):567–571

Kahlert C, Chua L (1990) A generalized canonical piecewise-linear representation. IEEE Trans Circuits Syst 37(3):373–383

Kang S, Chua L (1978) A global representation of multidimensional piecewise-linear functions with linear partitions. IEEE Trans Circuits Syst 25:938–940

Kanzow C, Petra S (2004) On a semismooth least squares formulation of complementarity. Optim Methods Softw 19(5):507–525

Kim J, Lee J, Vandenberghe L, Yang C (2004) Techniques for improving the accuracy of geometric-programming based analog circuit design optimization. In: Proceedings of the IEEE international conference on computer aided design, pp 863–870

Mangasarian O, Rosen J, Thompson M (2005) Global minimization via piecewise-linear underestimation. J Glob Optim 32

Mitchell J, Suri S (1992) Separation and approximation of polyhedral objects. In: Proceedings of the ACM-SIAM symposium on discrete algorithms. Society for industrial and applied mathematics, Philadelphia, pp 296–306

Phillips T, Rosenfeld A (1988) An ISODATA algorithm for straight line fitting. Pattern Recognit 7(5):291–297

Pittman J, Murthy C (2000) Fitting optimal piecewise linear functions using genetic algorithms. IEEE Trans Pattern Anal Mach Intel 22(7):701–718

Rives G, Dhome M, La Preste J, Richetin M (1985) Detection of patterns in images from piecewise linear contours. Pattern Recognit Lett 3:99–104

Storace M, De Feo O (2002) Piecewise-linear approximation of nonlinear dynamical systems

Torrisi F, Bemporad A (2004) HYSDEL—a tool for generating computational hybrid models for analysis and synthesis problems. IEEE Trans Control Syst Technol 12(2):235–249

Vandenberghe L, de Moor B, Vandewalle J (1989) The generalized linear complementarity problem applied to the complete analysis of resistive piecewise-linear circuits. IEEE Trans Circuits Syst 36(11):1382–1391

Venkateswar V, Chellappa R (1992) Extraction of straight lines in aerial images. IEEE Trans Pattern Anal Mach Intell 14(11):1111–1114

Wang D, Huang N, Chao H, Lee R (1993) Plane sweep algorithms for the polygonal approximation problems with applications. In: Proceedings of the international symposium on algorithms and computation. Springer, London, pp 515–522

Yin P (1998) Algorithms for straight line fitting using k-means. Pattern Recognit Lett 19(1):31–41