Isotone Optimization in R: Pool-Adjacent-Violators Algorithm (PAVA) and Active Set Methods

Jan de Leeuw
University of California,
Los Angeles

Kurt Hornik
WU Wirtschafts-
universität Wien

Patrick Mair
WU Wirtschafts-
universität Wien

Abstract

This introduction to the R package **isotone** is a (slightly) modified version of de˜Leeuw *et˜al.* (2009), published in the *Journal of Statistical Software*.

In this paper we give a general framework for isotone optimization. First we discuss a generalized version of the pool-adjacent-violators algorithm (PAVA) to minimize a separable convex function with simple chain constraints. Besides of general convex functions we extend existing PAVA implementations in terms of observation weights, approaches for tie handling, and responses from repeated measurement designs. Since isotone optimization problems can be formulated as convex programming problems with linear constraints we then develop a primal active set method to solve such problem. This methodology is applied on specific loss functions relevant in statistics. Both approaches are implemented in the R package **isotone**.

Keywords:˜isotone optimization, PAVA, monotone regression, active set, R.

1. Introduction: History of monotone regression

In monotone (or isotone) regression we fit an increasing or decreasing function to a set of points in the plane. This generalization of linear regression has a fairly complicated history, which we now discuss.

On June 4, 1958, Constance van Eeden defended her dissertation at the University of Amsterdam. The dissertation *Testing and Estimating Ordered Parameters of Probability Distributions* (van Eeden 1958) summarized and extended four articles published in 1956–1957 in *Indagationes Mathematicae*. Van Eeden's promotor Van Dantzig said in his speech at the promotion

> As often happens, just before you were able to obtain a final version of your result, a publication of an American professor appeared, which treated the same problem, while in a second publication by him, joint with four co-authors, the special case of a complete order was investigated in more detail.

He referred to Brunk (1955) and Ayer *et˜al.* (1955), which both appeared in the *Annals of Mathematical Statistics*. To complicate the situation, there were also the papers by Miles (1959) and Bartholomew (1959a,b), which can be thought of as yet another independent discovery. Some of the interconnectivity, and some of the early history, is reviewed in the

excellent book by the four B's˜(Barlow *et˜al.* 1972).

Of the classical treatments of monotone regression, Van Eeden's is the most general one. Moreover she very clearly separates the optimization problem, treated in full detail in Chapter˜1 of her dissertation, from the statistical applications. Of course we must realize that this work predates the advent of convex analysis by about 15 years, while it was done at the time that statisticians started showing interest in the (Karush-)Kuhn-Tucker conditions and in quadratic programming. In Van Eeden's setup, the problem is to minimize a strictly unimodal function $f(x_1, \ldots, x_n)$ under the bound constraints $\alpha_i \leq x_i \leq \beta_i$ and the partial order constraints $\sigma_{ij}(x_i - x_j) \geq 0$. Here the σ_{ij} are either zero or ± 1, and we suppose the system of inequalities defined by the bound and order constraints is consistent. By careful classical analysis, Van Eeden derives her algorithms, and shows how they specialize if the objective function is separable and/or a sum of squares, and if the order constraints define a complete order.

The next major contributions were by Robertson and Dykstra, summarized in the book by˜Robertson *et˜al.* (1988). At the same time, starting with˜Barlow and Brunk (1972) and˜Dykstra (1981), the isotonic regression problem was embedded more and more in quadratic and convex programming frameworks by, among others,˜Best and Chakravarti (1990) and˜Best *et˜al.* (2000). Recent major contributions, relying heavily on mathematical programming results, are˜Strömberg (1991), Ahuja and Orlin (2001), and˜Hansohm (2007). Extensions to particular partial orders defined by the variables of a multiple regression are in˜Dykstra and Robertson (1982) and˜Burdakov *et˜al.* (2004).

In this paper we present the R (R Development Core Team 2009) package **isotone** which implements PAVA and active set algorithms for solving monotone regression and more general isotone optimization problems. The package is available from the Comprehensive R Archive Network at `http://CRAN.R-project.org/package=isotone`.

2. A general isotone optimization framework

2.1. Basics of monotone regression

In simple regression we assume a linear relationship between a predictor $z = (z_1, \ldots, z_i, \ldots z_n)$ and a response $y = (y_1, \ldots, y_i, \ldots y_n)$. Note that for the predictors we use z instead of the common notation x since later on we embed this algorithm into a convex programming problem where the target variables are typically denoted by x. However, the loss function to be minimized is a least squares problem of the form

$$f(\alpha, \beta) = \sum_{i=1}^{n} w_i \left(y_i - \alpha - \beta z_i\right)^2 \rightarrow \min! \tag{1}$$

where α and β are the regression parameters and w_i some optional observation weights. Extensions can be formulated in terms of polynomial or other nonlinear parametric regression specifications. In many situations the researcher has no information regarding the mathematical specification of the true regression function. Rather, she can assume a particular shape which can be characterized by certain order restrictions. Typically, this involves that the y_i's increase with the ordered z_i's. Such a situation is called *isotonic regression*; the decreas-

ing case *antitonic regression*. The corresponding umbrella term for both cases is *monotonic regression* (for a compact description, see de Leeuw 2005).

Suppose that Z is the finite set $\{z_1, z_2, \ldots, z_n\}$ of the ordered predictors with no ties, i.e., $z_1 < z_2 < \cdots < z_n$. Let y be again the observed response vector and $x = (x_1, \ldots, x_i, \ldots x_n)$ the unknown response values to be fitted. The least squares problem in monotonic regression can be stated as

$$f(x) = \sum_{i=1}^{n} w_i \left(y_i - x_i\right)^2 \rightarrow \text{min!} \tag{2}$$

which has to be minimized over x under the inequality restrictions $x_1 \leq x_2 \leq \cdots \leq x_n$ for isotonic regression and $x_1 \geq x_2 \geq \cdots \geq x_n$ for the antitonic case. These restrictions avoid that we always get perfect fit. The basic theorem the isotonic solution of (2) is based on, is that if $y_i \geq y_{i+1}$, then the fitted values $\hat{x}_{i+1} := \hat{x}_i$. Correspondingly, the antitonic solution requires $y_i \leq y_{i+1}$.

For a non-strict partial order of the predictors, i.e., $z_1 \leq z_2 \leq \cdots \leq z_n$, several algorithms for the treatment of ties can be considered (Kruskal 1964; de Leeuw 1977). The *primary* approach partitions the index set $\{1, 2, \ldots, n\}$ into a number of tie blocks $I_1, I_2, \ldots, I_k$ with $k \leq n$. In case of a tie $z_i = z_{i'}$ this approach implies that x_i does not necessarily equal $x_{i'}$. It forces only that the following monotonicity condition holds for the tied observations i and i': If $y_i \leq y_{i'}$ then $x_i \leq x_{i'}$. In practical applications this approach is mostly used. The *secondary* approach is more restrictive and requires $x_i = x_{i'}$ for the tie i and i', regardless which y-values were observed. The *tertiary* approach defined in de Leeuw (1977) abandons the monotonicity condition from the primary approach. For each tie block $I_1, I_2, \ldots, I_k$ the unit of analysis are the weighted means $\overline{x}_{I_1}, \overline{x}_{I_2}, \ldots, \overline{x}_{I_K}$. The tertiary approach requires only that these means are monotonic across tie blocks.

2.2. Generalization to ℓ_p and multiple measurements

Now we extend this classical isotonic regression problem in terms of non-least squares loss functions and repeated observations. We establish the optimization problem $\mathcal{P}_0$ which includes a convex function of the form

$$h_i(y_{ij}, x_i) = |y_{ij} - x_i|^p. \tag{3}$$

The building blocks of $\mathcal{P}_0$ are a (possibly infinite) open real interval $\mathcal{I}$, and n real-valued convex functions f_i defined on $\mathcal{I}$. Problem $\mathcal{P}_0$ is to minimize the *separable* function of n variables of the form

$$f(x) = \sum_{i=1}^{n} \sum_{j=1}^{m_i} w_{ij} h_i\left(y_{ij}, x_i\right) \rightarrow \text{min!} \tag{4}$$

$$\text{subject to } x_1 \leq x_2 \leq \cdots \leq x_n. \tag{5}$$

over all *non-decreasing* x, i.e., all $x \in \mathcal{I}^n$. The response vectors of length m_i for each observation i we denote as $\mathbf{y}_i$ which can be caused, e.g., by repeated observations y_{ij}.

Common special cases of (3) are $p = 2$ and $p = 1$:

$$h_i(y_{ij}, x_i) = (y_{ij} - x_i)^2, \tag{6a}$$

$$h_i(y_{ij}, x_i) = |y_{ij} - x_i|. \tag{6b}$$

Least-squares estimation for the first problem results in estimates that approximate the conditional mean of the response given predictor value z_i. If weights w_i are involved, we get the weighted mean. If we estimate the second equation within a quantile regression framework (see Koenker 2005), the estimates approximate the (weighted) median. The general quantile regression specification is

$$h_i(y_{ij}, x_i) = a \max(y_{ij} - x_i, 0) + b \max(x_i - y_{ij}, 0). \tag{7}$$

with $a, b > 0$. The corresponding proportions are $a/(a+b)$ and $b/(a+b)$. Note that for $a = b$ Equation˜7 reduces again to the weighted median. Let us refer to the expression for the conditional (weighted) mean and (weighted) quantiles as *solver* $s(x_i)$. This function plays a central role for the parameter updates in PAVA described in the next section. Note that in our general framework the user can specify other convex functions and their corresponding solvers. As we will see in Section˜5.1 the `gpava()` function takes a (user-defined) solver as argument which is sufficient for defining the PAVA optimization problem.

2.3. Isotone optimization as a convex programming problem

The whole theory in this section is based on the fact that isotone optimization can be formulated as convex programming problem with inequality constraints. For instance, the least squares problem in Equation˜1 is one particular example of such a convex program. The isotonicity condition is contained in the side constraints. These inequalities defining isotonicity can be written in matrix form as $Ax \geq 0$ with $x \in \mathbb{R}^n$ as the target variable to be optimized. A is a $m \times n$ matrix in which each row corresponds with an index pair i, j such that $i \succeq j$. Such a row of length n has element i equal to $+1$, element j equal to -1, and the rest of the elements equal to zero. Formally, such isotone optimization problems, written as convex programming problem with linear inequality constraints look as follows:

$$\begin{aligned} f(x) &\rightarrow \min! \\ \text{subject to } Ax &\geq 0 \end{aligned} \tag{8}$$

The constraints in isotone regression are a total order. We now consider the more general case of partial orders. In order to eliminate redundancies we include a row in A for a pair (i, j) iff i *covers* j, which means that $i \succeq j$ and there is no $k \neq i, j$ such that $i \succeq k \succeq j$. This specification allows us to specify isotonicity patterns in a flexible manner. Some examples are given in Figure˜1 by means of *Hasse diagrams* (also known as *cover graphs*).

The Lagrangian associated with problem˜8 is

$$L(x, \lambda) = f(x) - \lambda^\top Ax \tag{9}$$

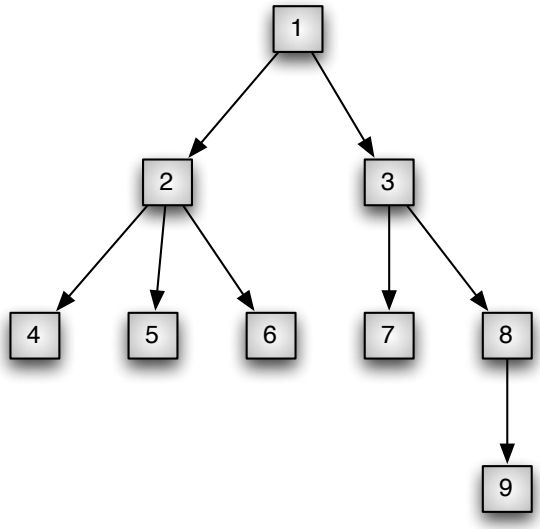
with λ as the *Lagrange multiplier vector*. The associated *Lagrange dual function* can be expressed as

$$g(\lambda) = \inf_x L(x, \lambda) = \inf_x \left(f(x) - \lambda^\top Ax \right). \tag{10}$$

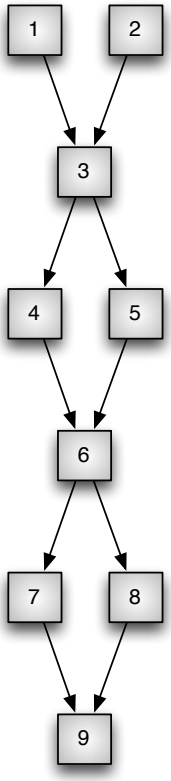
and the corresponding dual optimization problem is

$$\begin{aligned} g(\lambda) &\rightarrow \max! \\ \text{subject to } \lambda_l &\geq 0. \end{aligned} \tag{11}$$

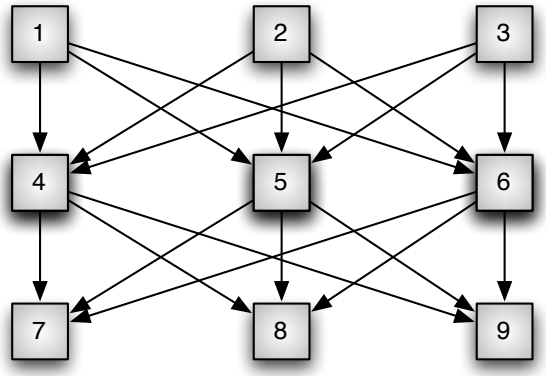
(a) Total order



(b) Tree order



(c) Loop order



(d) Block order

Figure 1: Hasse diagrams for different types of orders.

The Lagrange dual can be reformulated using the *convex conjugate*. Let $x^\star \in \mathbb{R}^n$. The *conjugate function* of $f(x)$ denoted by $f^\star(x^\star)$ is the maximum gap between the linear function $\langle x^\star, x\rangle$ and the target function $f(x)$ and can be expressed as

$$f^\star(x^\star) = \sup_x \left(\langle x^\star, x\rangle - f(x)\right). \tag{12}$$

It can be shown (see Boyd and Vandenberghe 2004, p.˜221) that

$$g(\lambda) = -f^\star(A^\top \lambda) \tag{13}$$

In other words: The Lagrange dual is determined by the convex conjugate with $A^\top\lambda$. This relates the convex primal and the dual optimization problem in the following manner:

$$\min\{f(x) : Ax \geq 0\} \leftrightarrow \min\left\{f^\star(A^\top\lambda) : \lambda \geq 0\right\}$$

As necessary conditions for the minimum we give the *Karush-Kuhn-Tucker* (KKT) conditions for this problem. The convex function f is minimized on a convex set $\{x \mid Ax \geq 0\}$ at $\hat{x}$ iff there exists a vector of Lagrange multipliers $\hat{\lambda}$ (i.e., the KKT vector) such that˜(see Rockafellar 1970, Chapter˜28)

$$A^\top\hat{\lambda} \in \partial f(x), \quad A\hat{x} \geq 0, \quad \hat{\lambda} \geq 0, \quad \hat{\lambda}^\top A\hat{x} = 0.$$

Here $\partial f(\hat{x})$ is the *subdifferential* of f at $\hat{x}$. In general, the subdifferential at x is the set of all *subgradients* of f at x, where y is a subgradient at x if

$$f(z) \geq f(x) + (z - x)^\top y \quad \forall z. \tag{14}$$

The subdifferential is a convex compact set. If f is differentiable at x, there is a unique subgradient, the gradient $\nabla f(x)$. Thus, the necessary and sufficient conditions for a $\hat{x}$ to be a minimizer in the differentiable case are the existence of a KKT vector $\hat{\lambda}$ such that

$$\nabla f(\hat{x}) = A^\top\hat{\lambda}, \quad A\hat{x} \geq 0, \quad \hat{\lambda} \geq 0, \quad \hat{\lambda}^\top A\hat{x} = 0.$$

These conditions are referred to as stationarity, primal feasibility, dual feasibility, and complementary slackness.

2.4. Active set method for convex programming problems

Basically, there are two classes of algorithms to solve convex optimization problems: The first class are *interior point methods* aiming for complementary slackness while maintaining primal and dual feasibility. The second class are active set strategies where we distinguish between two variants:

***Primal active set methods*:** They aim for dual feasibility while maintaining primal feasibility and complementary slackness.

***Dual active set methods*:** They aim for primal feasibility while maintaining dual feasibility and complementary slackness.

Active set methods in general are approaches for solving linear, quadratic, and convex programming problems with inequality constraints and are the most effective for solving small- to medium-scaled problem. In our implementation, by means of the function `activeSet()`, we provide a primal active set strategy which Zangwill (1969, Chapter˜8) denotes as *manifold suboptimization*.

Recall problem $\mathcal{P}$ given in Equation˜8 where the aim is to minimize $f(x)$ over $Ax \geq 0$. The minimum is $\hat{f}(x)$ and the corresponding minimizer is $\hat{x}$. Write I for subsets of the index set $\mathcal{I} = 1, 2, \cdots, m$ for the constraints. Then A_I is the corresponding $\mathbf{card}(I) \times n$ submatrix of A, and $A_{\overline{I}}$ is the $(m - \mathbf{card}(I)) \times n$ complementary submatrix. The *active constraints* at x, which we write as $\mathscr{A}$, are the indices i for which $a_i^\top x = 0$. That is, at a given point x a constraint is called

active if $a_i^\top x = 0$, $i \in \mathscr{A}$, and

inactive if $a_i^\top x > 0$, $i \notin \mathscr{A}$.

Each iteration in the active set algorithm attempts to locate the solution to an equality problem in which only the active constraints occur. There are more ways to state the active set optimization problem for our general convex case with linearity restrictions. Let us first elaborate an auxiliary problem denoted by $\mathcal{P}_I^+$

$$\begin{aligned} f(x) &\rightarrow \min! \\ \text{subject to } A_I x &= 0 \\ A_{\overline{I}} x &> 0 \end{aligned} \tag{15}$$

which partitions the constraints set into equality and strict inequality restrictions. Note that this partitions the constraint set $x \mid Ax \geq 0$ into 2^m faces, some of which may be empty. Solution $\hat{x}_I^+$ is optimal for $\mathcal{P}_I^+$ iff there exists a KKT vector $\hat{\lambda}_I^+$ such that

$$A_I^\top \hat{\lambda}_I^+ \in \partial f(\hat{x}_I^+), \quad A_I \hat{x}_I^+ = 0, \quad A_{\overline{I}} \hat{x}_I^+ > 0.$$

The crucial condition that $\hat{x}_I^+$ is the optimum also for $\mathcal{P}$ is the dual feasibility condition $\hat{\lambda}_I^+ \geq 0$. If this holds then $\hat{x}_I^+$ is optimal for $\mathcal{P}$. Conversely, if $\mathscr{A}^\star$ are the indices of the active constraints at the solution $\hat{x}$ of $\mathcal{P}$, then $\hat{x}$ solves $\mathcal{P}^+$ as well.

A second way to give an active set formulation for our setting is problem $\mathcal{P}_I$ that includes equality restrictions only:

$$\begin{aligned} f(x) &\rightarrow \min! \\ \text{subject to } A_I x &= 0 \end{aligned} \tag{16}$$

Solution $\hat{x}_I$ is optimal for iff there exists a KKT vector $\hat{\lambda}_I$ such that

$$A_I^\top \hat{\lambda}_I \in \partial f(\hat{x}_I), \quad A_I \hat{x}_I = 0.$$

To ensure that $\hat{x}_I$ is optimal for $\mathcal{P}$ as well we have to check whether the primal feasibility $A_{\overline{I}} \hat{x}_I \geq 0$ and, again, the dual feasibility $\hat{\lambda}_I \geq 0$ holds. Conversely, $\hat{x}$ with the corresponding active constraints $\mathscr{A}^\star$ solves $\mathcal{P}$. Thus, if we knew $\mathscr{A}^\star$ we could solve $\mathcal{P}$ by solving $\mathcal{P}_I$.

In order to achieve this we discuss an equivalent formulation of $\mathcal{P}_I$. Basically, the equality constraints $A_I x = 0$ define a relation $\approx_I$ on $1, 2, \cdots, n$, with $i \approx_I k$ if there is a row j of A_I in which both a_{ji} and a_{jk} are non-zero. The reflexive and transitive closure $\overline{\approx}_I$ of $\approx_I$ is an equivalence relation, which can be coded as an *indicator matrix* G_I, i.e., a binary matrix in which all n rows have exactly one element equal to one.

In the package we compute G_I from A_I in two steps. We first make the adjacency matrix of $\approx_I$ and add the identity to make it reflexive. We then apply Warshall's Algorithm (Warshall 1962) to replace the adjacency matrix by that of the transitive closure $\overline{\approx}_I$, which is an equivalence relation. Thus the transitive adjacency matrix has blocks of ones for the equivalence classes. We then select the unique rows of the transitive adjacency matrix (using the `unique()` function), and transpose to get G_I.

Note that G_I is of full column-rank, even if A_I is singular. We write r_I for the number of equivalence classes of $\overline{\approx}_I$. Thus G_I is an $n \times r_I$ matrix satisfying $A_I G_I = 0$, in fact G_I is a basis for the null space of A_I. Moreover $D_I \triangleq G_I^\top G_I$ is diagonal and indicates the number of elements in each of the equivalence classes.

Finally, problem $\mathcal{P}_I$ can be rewritten as unconstrained convex optimization problem

$$f(G_I \xi_I) \rightarrow \min! \tag{17}$$

with $\xi_I \in \mathbb{R}^r_I$. The vector $\hat{\xi}_I$ is a solution iff $0 \in G_I^\top \partial f(G_I \hat{\xi}_I)$. Then $\hat{x}_I = G_I \hat{\xi}_I$ solves $\mathcal{P}_I$. If $0 \in G_I^\top \partial f(G_I \hat{\xi}_I)$ it follows that there is a non-empty intersection of the subgradient $\partial f(G_I \hat{\xi}_I)$ and the row-space of A_I, i.e., there is a KKT vector $A_I^\top \hat{\lambda}_I \in \partial f(G_I \hat{\xi}_I)$.

In R we can simply use the `optim()` function—e.g., with Broyden-Fletcher-Goldfarb-Shanno (BFGS) quasi-Newton—to minimize $f(G_I \xi)$ over ξ, and then set $\hat{x} = G_I \hat{\xi}$. This guarantees (if the optimum is found with sufficient precision) that the gradient at $\hat{x}$ is orthogonal to the indicator matrix G_I, and consequently that Lagrange multipliers can be computed. By making sure that A_I has full row-rank, the Kuhn-Tucker vector is actually unique.

To conclude, in this section we elaborated auxiliary formulations of the active set problem. In the algorithmic implementation described in the next section we tackle optimization of problem $\mathcal{P}$ by defining and solving iteratively subproblems $\mathcal{P}_I$ (including the indicator matrix G_I).

3. PAVA and active set algorithm

3.1. A generalized PAVA approach

Barlow *et˜al.* (1972) present a graphical interpretation of monotonic regression in terms of the *greatest convex minorant* (GCM) and the method of successive approximation to the GCM can be described algebraically as PAVA (see e.g. Robertson *et˜al.* 1988, p.˜9–10). In this paper we focus on the algorithmical description of our PAVA implementation following Barlow *et˜al.* (1972, p.˜15).

PAVA is a very simple iterative algorithm for solving monotonic regression problems. The computation of a non-decreasing sequence of values x_i such that the problem $\mathcal{P}_0$ is optimized. In the simple isotonic regression case we have the measurement pairs (z_i, y_i). Let us assume that these pairs are ordered with respect to the predictors. The initial solution at iteration

$l = 0$ is $x_i^{(0)} := y_i$. The index for the blocks is $r = 1, \ldots, B$ where at step 0 $B := n$, i.e., each observation $x_r^{(0)}$ forms a block.

1. Merge $x^{(l)}$-values into blocks if $x_{r+1}^{(l)} < x_r^{(l)}$ (adjacent pooling).
2. Solve $f(x)$ each block r, i.e., compute the update based on the solver which gives $x_r^{(l+1)} := s(x_r^{(l)})$.
3. If $x_{r+1}^{(l)} \leq x_r^{(l)}$ increase $l := l + 1$ an go back to step 1.

The algorithm stops when the x-blocks are increasing, i.e., $x_{r+1}^{(l)} \geq x_r^{(l)}$ for all r. Finally the block values are expanded with respect to the observations $i = 1, \ldots, n$ such that the final result is the vector x of length n with elements $\hat{x}_i$ of increasing order. In the case of repeated measurements as described in Section˜2.2 the starting values are $x_i^{(0)} := s(y_{ij})$. Then we proceed as described above.

A final notes considers the relation between the generalized PAVA and the active set strategy. Robertson *et˜al.* (1988, Chapter˜6) provide a chapter about duality in monotone regression. For the simple least squares case Best and Chakravarti (1990) show that PAVA can be considered as a dual active set method. In a subsequent paper, Best *et˜al.* (2000) they extend this relation to the general ℓ_p case. Therefore it can be concluded that PAVA for general ℓ_p cases belong to the family of dual active set methods.

3.2. Algorithmic implementation of the active set formulation

Active set descriptions for convex (quadratic) programming problems can be found in Fletcher (1987, Section 10.3) and Nocedal and Wright (1999, Section 16.4). In general, based on a feasibile starting point, the active set strategy works as follows:

1. Solve the equality problem defined by the active set $\mathcal{A}$.
2. Compute the Lagrange multipliers λ for $\mathcal{A}$.
3. Remove a subset of constraints with $\lambda < 0$.
4. Search for infeasible constraints and return to step 1.

These steps are repeated until we find a solution that is "optimal enough". In practice (and in our package), the zero-boundaries for the constraints and the dual feasibility checking are relaxed in favor of a small value ε which leads to $Ax \geq -\varepsilon$, and $\lambda \geq -\varepsilon$.

Again, our goal is to solve problem $\mathcal{P}$ given in (15). Based on the elaborations above, in our active set approach we solve a finite sequence of subproblems $\mathcal{P}_I$, that minimize $f(x)$ over x satisfying $A_I x$. After solving each of the problems we change the active set $\mathscr{A}$, either by adding or by dropping a constraint. The algorithm can be expected be efficient if minimizing $f(x)$ under simple equality constraints, or equivalently minimizing $f(G_I \xi_I)$ over ξ_I, can be done quickly and reliably. Starting with a feasible point $x^{(0)}$ which defines the index sets $I^{(0)} = \mathscr{A}^{(0)}$ and $\overline{I}^{(0)} = \mathcal{I} - I^{(0)}$. For each iteration l

1. Suppose $\tilde{x}^{(l)}$ is a solution of $\mathcal{P}_{I^{(l)}}$. Then $\tilde{x}^{(l)}$ can be either feasible or infeasible for $\mathcal{P}$, depending on if $A_{\bar{I}^{(l)}}\tilde{x}^{(l)} \geq 0$ or not.

2. If $\tilde{x}^{(l)}$ is *infeasible* we choose $x^{(l+1)}$ on the line between $x^{(l)}$ and $\tilde{x}^{(l)}$, where it crosses the boundary of the feasible region (see Equation˜18 below). This defines a new and larger set of active constraints $\mathscr{A}^{(l+1)}$. Go back to step 1.

3. If $\tilde{x}^{(l)}$ is *feasible* we determine the corresponding Lagrange multipliers $\lambda_I^{(l)}$ for $\mathcal{P}_{I^{(l)}}$. If $\lambda^{(l)} \geq 0$ we have solved $\mathcal{P}$ and the algorithm stops. If $\min \lambda_I^{(l)} < 0$, we find the most negative Lagrange multiplier and drop the corresponding equality constraint from $\mathscr{A}^{(l)}$ to define a new and smaller set of active constraints $\mathscr{A}^{(l+1)}$ and go back to step 1.

In step 2 we solve

$$\begin{aligned} &\max_\alpha x^{(l)} + \alpha(\tilde{x}^{(l)} - x^{(l)}) \\ &\text{over} \min_{i \in A_{\bar{I}^{(l)}}} a_i^\top x^{(l)} + \alpha(a_i^\top \tilde{x}^{(l)} - a_i^\top x^{(l)}) \geq 0 \end{aligned} \tag{18}$$

Finding the smallest Lagrange multiplier in step 3 is straightforward to implement in the differentiable case. We have to solve $A_{I^{(l)}}^\top \lambda_I = \nabla f(\tilde{x}^{(l)})$. This is achieved by using the formulation given in (17). Because $G_{I^{(l)}}^\top \nabla f(\tilde{x}^{(l)}) = 0$ and $A_{I^{(l)}}$ is of full row-rank, there is a unique solution $\lambda_{I^{(s)}}$.

4. Special cases for active set optimization

In the convex non-differentiable case, matters are more complicated. We have to deal with the fact that in general $\partial f(x)$ may not be a singleton. It is possible to develop a general theory for active set methods in this case (Panier 1987), but we will just look at two important special cases.

4.1. The weighted Chebyshev norm

The first case we consider is the ℓ_∞ or Chebyshev norm. We have to minimize

$$f(\xi) = \|h(\xi)\|_\infty = \max_{i=1}^n |w_i h_i(\xi)| \tag{19}$$

where $h(\xi) = y - G\xi$ are the *residuals* based on the observed response values y. We assume, without loss of generality, that $w_i > 0$ for all i. The minimization can be done for each of the r columns of the indicator matrix G with elements g_{ij} separately. The solution $\hat{\xi}_j$ is the corresponding *weighted mid-range*. More specifically, let $I_j = \{i \mid g_{ij} = 1\}$. Then

$$f_j(\hat{\xi}_j) = \min_{\xi_j} \max_{i \in I_j} |y_i - \xi_j| = \max_{i,k \in I_j} \frac{w_i w_k}{w_i + w_k} |y_i - y_k|. \tag{20}$$

If the (not necessarily unique) maximum over $(i,k) \in I_j$ is attained at (i_j, k_j), then the minimum of f over ξ is attained at

$$\hat{\xi}_j = \frac{w_{i_j} y_{i_j} + w_{k_j} y_{k_j}}{w_{i_j} + w_{k_j}}, \tag{21}$$

where we choose the order within the pair (i_j, k_j) such that $y_{i_j} \leq \hat{\xi}_j \leq y_{k_j}$. Now

$$\min_{\xi} f(\xi) = \max_{j=1}^{r} f_j(\hat{\xi}_j). \tag{22}$$

These results also apply if I_j is a singleton $\{i\}$, in which case $\hat{\xi}_j = y_i$ and $f_j(\hat{\xi}_j) = 0$. Set $\hat{x} = G\hat{\xi}$.

Next we must compute a subgradient in $\partial f(\hat{x})$ orthogonal to G. Suppose that e_i is a unit weight vectors, i.e., a vector with all elements equal to zero, except element i which is equal to either plus or minus w_i. Consider the set $\mathcal{E}$ of the $2n$ unit weight vectors. Then $f(\xi) = \max_{e_i \in \mathcal{E}} e_i^\top h(\xi)$. Let $\mathcal{E}(\xi) = \{e_i \mid e_i^\top h(\xi) = f(\xi)\}$. Then, by the formula for the subdifferential of the pointwise maximum of a finite number of convex functions (also known as Danskin's Theorem; see Danskin 1966), we have $\partial f(\xi) = \mathbf{conv}(\mathcal{E}(\xi))$ with $\mathbf{conv}()$ as the convex hull.

Choose any j for which $f_j(\hat{\xi}_j)$ is maximal. Such a j may not be unique in general. The index pair (i_j, k_j) corresponds with the two unit weight vectors with non-zero elements $-w_{i(j)}$ and $+w_{k(j)}$. The subgradient we choose is the convex combination which has element -1 at position i_j and element $+1$ at position $k(j)$. It is orthogonal to G, and thus we can find a corresponding Kuhn-Tucker vector.

In the **isotone** package the Chebyshev norm is provided by the solver function `mSolver()`.

4.2. The weighted absolute value norm

For the ℓ_1 or weighted absolute value norm

$$f(\xi) = \|h(\xi)\| = \sum_{i=1}^{n} |w_i h_i(\xi)| \tag{23}$$

we find the optimum $\hat{\xi}$ by computing *weighted medians* instead of weighted mid-ranges. Uniqueness problems, and the subdifferentials, will generally be smaller than in the case of ℓ_∞.

For ℓ_1 we define $\mathcal{E}$ to be the set of 2^n vectors $(\pm w_1, \pm w_2, \cdots, \pm w_n)$. The subdifferential is the convex hull of the vectors $e \in \mathcal{E}$ for which $e_i^\top h(\xi) = \min_\xi f(\xi)$. If $h_i(\xi) \neq 0$ then $e_i = \mathbf{sign}(h_i(\xi))w_i$, but if $h_i(\xi) = 0$ element e_i can be any number in $[-w_i, +w_i]$. Thus the subdifferential is a multidimensional rectangle. If the medians are not equal to the observations the loss function is differentiable. If $h_i(\xi) = 0$ for some i in I_j then we select the corresponding element in the subgradient in such a way that they add up to zero over all $i \in I_j$.

In the package the `dSolver()` function implements the weighted absolute value norm.

4.3. Some additional solvers

The **isotone** package provides some pre-specified loss functions but allows also the user to define his own functions. The corresponding argument in the function `activeSet()` is `mySolver`. This can be either the function name or the equivalent string expression given in brackets below. Each solver has extra arguments which we describe in this section.

A user-defined specification of a convex differentiable function can be achieved by setting `mySolver = fSolver`. The target function itself is passed to the function by means of the

`fobj` argument and the corresponding gradient using `gobj`. Note that it is not at all necessary that the problems are of the regression or projection type, i.e., minimize some norm $\|y - x\|$. In fact, the driver can be easily modified to deal with general convex optimization problems with linear inequality constraints which are not necessarily of the isotone type. We give examples in the next section.

Now let us describe some functions pre-specified in the package. We start with the cases which solve the equivalent formulation of problem $\mathcal{P}_I$ given in (17), that is, the one which minimizes over ξ. Because of the structure of these problems it is more efficient to use this formulation. The solvers passed to the `mySolver` argument are

`lsSolver ("LS")`: Solves the least squares ℓ_2 norm given in (2).

`dSolver ("L1")`: Minimizes the weighted absolute value ℓ_1 norm as given in (23).

`mSolver ("chebyshev")`: Solves the Chebyshev ℓ_∞ norm given in (19).

`pSolver ("quantile")`: Solves the quantile regression problem as given in (7).

`lfSolver ("GLS")`: Tackles a general least squares problem of the form $f(x) = (y - x)^\top W(y - x)$ where W is a not necessarily positive definite matrix of order n.

The extra arguments for all these solvers are `y` for the observed response vector and `weights` for optional observation weights. The functions return a list containing `x` as the fitted values, `lbd` as the Lagrange vector, `f` as the value of the target function, and `gx` as the gradient at point x.

Furthermore, we provide some hybrids which are just wrapper functions and call `fSolver()` internally. It is not needed that `fobj` and `gobj` are provided by the user.

`sSolver ("poisson")`: Solves the negative Poisson log-likelihood with loss function $f(x) = \sum_{i=1}^n x_i - y_i \log(x_i)$.

`oSolver ("Lp")`: Minimizes the ℓ_p norm as given in (4) with $m_i = 1 \ \ \forall i$.

`aSolver ("asyLS")`: Efron (1991) gives an asymmetric least squares formulation based on (2). The weights b_w for $(y_i - x_i) \leq 0$ (extra argument `bw`) and a_w for $(y_i - x_i) > 0$ (extra argument `aw`) are introduced. Consequently, $h_i(y_i, x_i) = (y_i - x_i)^2 b_w$ for $(y_i - x_i) \leq 0$ and $h_i(x_i, y_i) = (y_i - x_i)^2 a_w$ for $(y_i - x_i) > 0$.

`eSolver ("L1eps")`: Minimizes the ℓ_1 approximation for $f(x) = \sum_{i=1}^n w_i \sqrt{(y_i - x_i)^2 + \varepsilon}$. Correspondingly, the additional extra argument is `eps`.

`hSolver ("huber")`: Solves the Huber loss-function (Huber 1981) with extra argument `eps`.

$$h_i(y_i, x_i) = \begin{cases} (y_i - x_i)^2/4\varepsilon & |y_i - x_i| > 2 \\ |y_i - x_i| - \varepsilon & \text{otherwise} \end{cases}$$

`iSolver ("SILF")`: Minimizes the soft insensitive loss function (SILF) defined in Chu *et al.* (2004) with the two extra parameters ε (argument `eps`) and β (argument `beta`). We

have to distinguish as follows:

$$h_i(y_i, x_i) = \begin{cases} -|y_i - x_i| - \varepsilon & |y_i - x_i| \in (-\infty, -(1+\beta)\varepsilon) \\ (|y_i - x_i| + (1-\beta)\varepsilon)^2/4\beta\varepsilon & |y_i - x_i| \in [-(1+\beta)\varepsilon, -(1-\beta)\varepsilon] \\ 0 & |y_i - x_i| \in (-(1-\beta)\varepsilon, (1-\beta)\varepsilon) \\ (|y_i - x_i| - (1-\beta)\varepsilon)^2/4\beta\varepsilon & |y_i - x_i| \in [(1-\beta)\varepsilon, (1+\beta)\varepsilon] \\ |y_i - x_i| - \varepsilon & |y_i - x_i| \in ((1+\beta)\varepsilon, \infty) \end{cases}$$

With a little extra effort various other fashionable support vector machines (SVM) and lasso isotone regressions could be added.

5. Package description and examples

The **isotone** package consists of two main components: The PAVA component with its main function `gpava()` and the active set component with its main function `activeSet()`. In the following sections we describe their basic functionalities and give corresponding examples.

Package	Function	Location	Description	Language
stats (R Development Core Team 2009)	`isoreg()`	external	isotonic regression	C
monreg (Pilz and Titoff 2009)	`monreg()`	external	monotonic regression	C
fdrtool (Strimmer 2008)	`monoreg()`	external	monotonic regression, weights	C
Cir (Oron 2008)	`pava()`, `cir.pava()`	external	isotonic regression, weights	R
Iso (Turner 2009)	`pava()`	external	isotonic regression, weights	Fortran
clue (Hornik 2005)	`pava()`	internal	isotonic regression (mean/median)	R
logcondens (Rufibach and Duembgen 2009)	`isoMean()`	external	isotonic regression, weights	R
sandwich (Zeileis 2004)	`pava.blocks()`	internal	isotonic regression, weights	R
intcox (Henschel *et˜al.* 2009)	`intcox.pavaC()`	internal	isotonic regression, weights	C
SAGx (Broberg 2009)	`pava.fdr()`	external	false discovery rate using isotonic regression	R
smacof (de Leeuw and Mair 2009)	`pavasmacof()`	internal	nonmetric multidimensional scaling, ties	R

Table 1: Existing PAVA implementations in R.

5.1. The PAVA component

Table˜1 gives an overview of available PAVA implementations in R. It quotes the package name, the corresponding function, whether this function is externally accessible, a brief description, and in which programming language the core computations are implemented. The functions that use C or Fortran can be expected to be faster for large observation vectors.

None of the functions in Table˜1 allows for the specifications of general convex functions, repeated measurements and the handling of ties for such structures. Our package which is available on CRAN, allows for the computation of rather general target functions with additional options in terms of repeated measurements, tie handling, and weights.

The main function for this generalized PAVA computation is `gpava()`. The argument `solver` takes the solver function which can be `weighted.mean`, `weighted.median`, `weighted.fractile` or a user-specified function. S3 methods such as `print()` and `plot()` are provided and in the following subsections we present simple two examples.

5.2. Monotone regression with ties for pituitary data

The first example is based on a dataset from Pothoff and Roy (1964) which is also analyzed in Robertson *et˜al.* (1988). The Dental School at the University of North Carolina measured the size of the pituitary fissue (in millimeters) on 11 subjects from 8 to 14 years. The predictor is age and it can be expected that the size of the fissure increases with age. The data are of the following structure:

```
R> require("isotone")
R> data("pituitary")
R> head(pituitary)
```

```
  age size
1   8 21.0
2   8 23.5
3   8 23.0
4  10 24.0
5  10 21.0
6  10 25.0
```

As we see we have a several ties on the predictors. By means of this example we present different results caused by different approaches for handling ties (i.e., primary, secondary, tertiary). We compute the isotonic regression using the function `gpava()`:

```
R> res1 <- with(pituitary, gpava(age, size, ties = "primary"))
R> res2 <- with(pituitary, gpava(age, size, ties = "secondary"))
R> res3 <- with(pituitary, gpava(age, size, ties = "tertiary"))

R> layout(matrix(c(1, 1, 2, 2, 0, 3, 3, 0), 2, 4, byrow = TRUE))
R> plot(res1, main = "PAVA plot (primary)")
R> plot(res2, main = "PAVA plot (secondary)")
R> plot(res3, main = "PAVA plot (tertiary)")
```

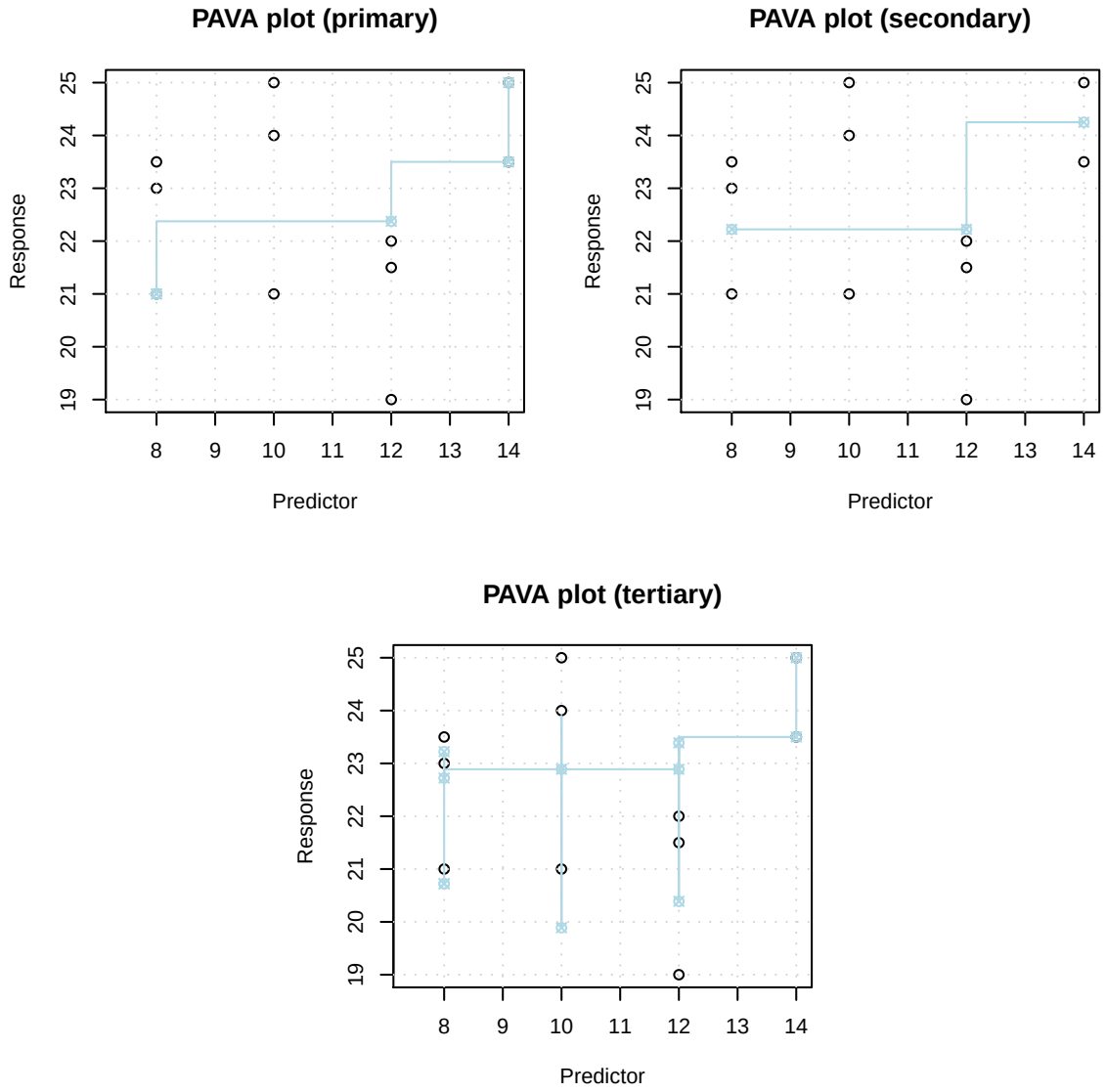


Figure 2: Different tie approaches for pituitary data.

For the primary method we can have different (monotonic) fitted values within tied observations, for the secondary the fitted values are the same within ties, and the tertiary approach only requires monotonicity on the means. This can be examined by doing

```
R> tapply(res3$x, res3$z, mean)
```

```
       8       10       12       14
22.22222 22.22222 22.22222 24.25000
```

Using the tertiary approach the fitted values within ties can even decrease (see, e.g., within

10 year old children). The plots in Figure˜2 show the fitted step functions for each of the approaches.

5.3. Repeated measures using posturographic data

To demonstrate PAVA on longitudinal data we use a subset of the data collected in Leitner *et˜al.* (2009). The sample consists of 50 subjects (healthy controls and patients chronical low back pain). The subjects' task on a sensory organization test (SOT) was to keep the balance on a dynamic 18×18 inches dual force plate. Measurements on various testing conditions were collected and the SOT equilibrium scores computed. They were based on the maximum anterior posterior sway angle during the SOT trials and reflected the overall coordination of visual, proprioceptive and vestibular components of balance to maintain standing posture. These equilibrium scores represent the angular difference between the subject's calculated anterior/posterior center of gravity displacements and the theoretical maximum of 12.5°. A score of 100 theoretically indicates no anterior/posterior excursion. A score of 0 indicates a fall of the subject. Thus, the higher the score, the more able a person is to maintain the balance.

The subset we select for this example is based on the condition where both proprioceptive and visual stimuli are altered by moving a surrounding visual screen and the platform with the subject's anterior/posterior body sway. We examine the relationship between body height and three repeated SOT scores.

```
R> data("posturo")
R> head(posturo)
```

```
   height SOT.1 SOT.2 SOT.3
1    1.64    91    95    91
2    1.80    84    90    90
3    1.63    64    80    82
6    1.59    65    83    83
10   1.75    87    90    86
11   1.69    77    88    84
```

PAVA is performed on the weighted median target function and using the secondary approach for ties.

```
R> res.mean <- with(posturo, gpava(height, cbind(SOT.1, SOT.2, SOT.3),
+    solver = weighted.mean, ties = "secondary"))
R> res.median <- with(posturo, gpava(height, cbind(SOT.1, SOT.2, SOT.3),
+    solver = weighted.median, ties = "secondary"))
```

```
R> plot(res.mean)
R> plot(res.median)
```

The fitted step functions for the two different target functions are given in Figure˜3.

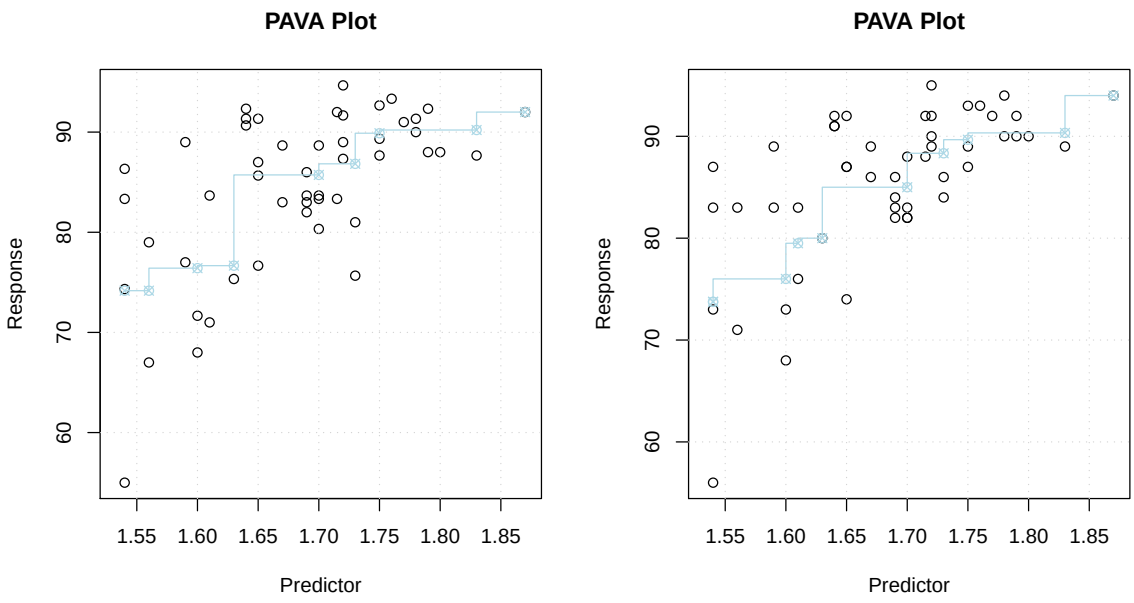


Figure 3: Step function for posturographic data.

5.4. The active set component

As mentioned above a primal active set implementation according to the elaborations in Section˜2.4 is provided by the function `activeSet()` which has the following arguments:

`isomat`: Matrix with 2 columns that contains isotonicity conditions (see examples).

`mySolver`: Either a user-specified solver function or one of the pre-specified solvers as described in Section˜4.3.

`x0`: Optional vector containing starting values. The response values y are passed to the solver function (see below).

`ups`: Upper boundary ε for KKT feasibility checking.

`check`: If `TRUE`, a final KKT feasibility check is performed.

For the solvers described in Section˜4.3, each one requires various extra arguments (see corresponding package help files for detailed description). Each of the solvers needs to have the response vector y as argument.

Now we use some small artificial examples to show the application of different solver routines. The data are

```
R> set.seed(12345)
R> y <- rnorm(9)
R> w1 <- rep(1, 9)
R> Atot <- cbind(1:8, 2:9)
```

The variable `Atot` defines the pairwise isotonicity matrix which needs to have 2 columns and as many rows as unique isotonicity constraints.

```
R> Atot
```

```
     [,1] [,2]
[1,]    1    2
[2,]    2    3
[3,]    3    4
[4,]    4    5
[5,]    5    6
[6,]    6    7
[7,]    7    8
[8,]    8    9
```

We see that this matrix defines a total order. The specification is always as follows: element column 2 $\succeq$ element column 1, e.g., the first row states $x_2 \geq x_1$.

Let us start with a simple least squares model using the `lsSolver()` and the equivalent user-specification of the ℓ_2 norm using `fSolver()` with corresponding target function `fobj` and its gradient `gobj`.

```
R> fit.ls1 <- activeSet(Atot, "LS", y = y, weights = w1)
R> fit.ls2 <- activeSet(Atot, fSolver, y = y, weights = w1,
+    fobj = function(x) sum(w1 * (x - y)^2),
+    gobj = function(x) 2 * drop(w1 * (x - y)))
```

This model has unit weights. We will refit this model with a different weight vector and, subsequently, its generalization using a non-diagonal weight matrix by means of the `lfsolver()`.

```
R> set.seed(12345)
R> wvec <- 1:9
R> wmat <- crossprod(matrix(rnorm(81), 9, 9))/9
R> fit.wls <- activeSet(Atot, "LS", y = y, weights = wvec)
R> fit.gls <- activeSet(Atot, "GLS", y = y, weights = wmat)
```

Quantile regression can be performed by means of the `pSolver()`:

```
R> fit.qua <- activeSet(Atot, "quantile", y = y, weights = wvec,
+      aw = 0.3, bw = 0.7)
```

Now let us generalize the LS problems in terms of other norms using again unit weights. Let us start with the ℓ_1 provided by `dSolver()`.

```
R> fit.abs <- activeSet(Atot, "L1", y = y, weights = w1)
```

This exact absolute value norm can be approximated with either `eSolver()` which requires ε or `oSolver()` which requires the power p.

```
R> fit.eps <- activeSet(Atot, "L1eps", y = y, weights = w1, eps = 1e-04)
R> fit.pow <- activeSet(Atot, "Lp", y = y, weights = w1, p = 1.2)
```

Furthermore, the Chebyshev ℓ_∞ norm can be calculated using `mSolver()`.

```
R> fit.che <- activeSet(Atot, "chebyshev", y = y, weights = w1)
```

Efron's asymmetric LS problem can be solved by means of `aSolver()` with the extra arguments `aw` for the $y_i > x_i$ case and `bw` for the $y_i \leq x_i$ case:

```
R> fit.asy <- activeSet(Atot, "asyLS", y = y, weights = w1, aw = 2,
+      bw = 1)
```

The Huber loss function with the ε parameter and the SILF SVM extension with additionally β as parameter can be computed as follows:

```
R> fit.hub <- activeSet(Atot, "huber", y = y, weights = w1, eps = 1)
R> fit.svm <- activeSet(Atot, "SILF", y = y, weights = w1, beta = 0.8,
+      eps = 0.2)
```

As a final norm let us consider the negative Poisson log-likelihood specified in `sSolver()` using Poisson distribution random numbers with parameter $\lambda = 5$.

```
R> set.seed(12345)
R> yp <- rpois(9, 5)
R> x0 <- 1:9
R> fit.poi <- activeSet(Atot, "poisson", x0 = x0, y = yp)
```

So far we focused on total orders only. Now we back to the LS case and specify different types or orders according to the Hasse diagrams in Figure˜1. The tree order in Figure˜1(b) can be defined as follows:

```
R> Atree <- matrix(c(1, 1, 2, 2, 2, 3, 3, 8, 2, 3, 4, 5, 6, 7, 8, 9), 8, 2)
R> Atree

     [,1] [,2]
[1,]    1    2
[2,]    1    3
[3,]    2    4
[4,]    2    5
[5,]    2    6
[6,]    3    7
[7,]    3    8
[8,]    8    9
```

```
R> fit.tree <- activeSet(Atree, "LS", y = y, weights = w1)
```

The loop order given in Figure˜1(c) and the corresponding LS-fit are

```
R> Aloop <- matrix(c(1, 2, 3, 3, 4, 5, 6, 6, 7, 8, 3, 3, 4, 5, 6, 6,
+    7, 8, 9, 9), 10, 2)
R> Aloop
```

```
      [,1] [,2]
 [1,]    1    3
 [2,]    2    3
 [3,]    3    4
 [4,]    3    5
 [5,]    4    6
 [6,]    5    6
 [7,]    6    7
 [8,]    6    8
 [9,]    7    9
[10,]    8    9
```

```
R> fit.loop <- activeSet(Aloop, "LS", y = y, weights = w1)
```

Finally, using the same data we fit the block order given in Figure˜1(d).

```
R> Ablock <- cbind(c(rep(1, 3), rep(2, 3), rep(3, 3), rep(4, 3), rep(5, 3),
+    rep(6, 3)), c(rep(c(4, 5, 6), 3), rep(c(7, 8, 9), 3)))
R> Ablock
```

```
      [,1] [,2]
 [1,]    1    4
 [2,]    1    5
 [3,]    1    6
 [4,]    2    4
 [5,]    2    5
 [6,]    2    6
 [7,]    3    4
 [8,]    3    5
 [9,]    3    6
[10,]    4    7
[11,]    4    8
[12,]    4    9
[13,]    5    7
[14,]    5    8
[15,]    5    9
[16,]    6    7
[17,]    6    8
[18,]    6    9
```

```
R> fit.block <- activeSet(Ablock, "LS", y = y, weights = w1)
```

Other types of orders can be defined easily by a proper specification of the constraints matrix.

5.5. PAVA computation using active set

In Section˜3.1 we mentioned that PAVA can be considered as a dual active set method. Using the simulated data again, we show that LS PAVA and LS active set lead to the same results. Consequently, the mean squared error becomes almost 0.

```
R> pava.fitted <- gpava(1:9, y)$x
R> aset.fitted <- activeSet(Atot, "LS", weights = w1, y = y)$x
R> mse <- mean((pava.fitted - aset.fitted)^2)
R> mse

[1] 1.324077e-34
```

Obviously, the active set approach is somewhat more general than PAVA in terms of loss functions and different types of orders. But, pertaining to the way we have implemented both functions, in cases in which it applies `gpava()` is more convenient and efficient than `activeSet()`. In the `gpava()` function we provide an easy way to handle multiple measurements. Basically, users can do this in `activeSet()` as well by a corresponding solver specification. Furthermore, `gpava()` offers different approaches to handle ties whereas `activeSet()` does not. Finally, for large data problems, PAVA is computationally more efficient than active set.

6. Discussion

After a historical outline we presented the theory and the algorithmical descriptions of approaches for solving isotone optimization problems: PAVA and active set. We started with the classical representation of the monotone LS regression problem with simple chain constraints and extended it to general convex programming problems with linear inequality constraints. This leads to a general isotone optimization framework that is implemented in the R package **isotone**. Applications on real and simulated data sets were shown.

One option for the extension of the PAVA algorithm is to allow for multiple predictors. This approach is elaborated in Burdakov *et˜al.* (2004).

Of course, the field of convex analysis is rather extensive and our `activeSet()` function covers only a certain fraction (cf. Boyd and Vandenberghe 2004). Since we use `optim()` in the `fSolver()` function this may cause troubles for non-differentiable convex functions. Therefore, a replacement of this optimizer would facilitate the optimization of such non-differentiable problems. In addition, for a target $f(x)$ we allow for one particular type of solver $s(x)$ only, that is, the target function is separable but the components must be of the same type. This could be extended in terms of different solvers for one particular optimization problem.

References

Ahuja RK, Orlin JB (2001). "A Fast Scaling Algorithm for Minimizing Separable Convex Functions Subject to Chain Constraints." *Operations Research*, **49**, 784–789.

Ayer M, Brunk HD, Ewing GM, Reid WT, Silverman E (1955). "An Empirical Distribution Function for Sampling with Incomplete Information." *The Annals of Mathematical Statistics*, **26**, 641–647.

Barlow RE, Bartholomew DJ, Bremner JM, Brunk HD (1972). *Statistical Inference Under Order Restrictions*. John Wiley & Sons, New York.

Barlow RE, Brunk HD (1972). "The Isotonic Regression Problem and Its Dual." *Journal of the American Statistical Association*, **67**, 140–147.

Bartholomew DJ (1959a). "A Test of Homogeneity for Ordered Alternatives." *Biometrika*, **46**, 36–48.

Bartholomew DJ (1959b). "A Test of Homogeneity for Ordered Alternatives II." *Biometrika*, **46**, 328–335.

Best MJ, Chakravarti N (1990). "Active Set Algorithms for Isotonic Regression: A Unifying Framework." *Mathematical Programming*, **47**, 425–439.

Best MJ, Chakravarti N, Ubhaya VA (2000). "Minimizing Separable Convex Functions Subject to Simple Chain Constraints." *SIAM Journal on Optimization*, **10**, 658–672.

Boyd S, Vandenberghe L (2004). *Convex Optimization*. Cambridge University Press, Cambridge, MA.

Broberg P (2009). ***SAGx**: Statistical Analysis of the GeneChip*. R˜package version˜1.18.0, URL http://www.bioconductor.org/packages/2.4/bioc/html/SAGx.html.

Brunk HB (1955). "Maximum Likelihood Estimates of Monotone Parameters." *The Annals of Mathematical Statistics*, **26**, 607–616.

Burdakov O, Grimvall A, Hussian M (2004). "A Generalised PAV Algorithm for Monotonic Regression in Several Variables." In J˜Antoch (ed.), "COMPSTAT, Proceedings of the 16th Symposium in Computational Statistics," pp. 761–767. Springer-Verlag, New York.

Chu W, Keerthi SS, Ong CJ (2004). "Bayesian Support Vector Regression Using a Unified Loss Function." *IEEE Transactions on Neural Networks*, **15**, 29–44.

Danskin JM (1966). "The Theory of Max-Min with Applications." *SIAM Journal on Applied Mathematics*, **14**, 641–664.

de Leeuw J (1977). "Correctness of Kruskal's Algorithms for Monotone Regression with Ties." *Psychometrika*, **42**, 141–144.

de Leeuw J (2005). "Monotonic Regression." In BS˜Everitt, DC˜Howell (eds.), "Encyclopedia of Statistics in Behavioral Science (Vol. 3)," pp. 1260–1261. John Wiley & Sons, New York.

de˜Leeuw J, Hornik K, Mair P (2009). "Isotone Optimization in R: Pool-Adjacent-Violators Algorithm (PAVA) and Active Set Methods." *Journal of Statistical Software*, **32**(5), 1–24. URL http://www.jstatsoft.org/v32/i05/.

de Leeuw J, Mair P (2009). "Multidimensional Scaling Using Majorization: SMACOF in R." *Journal of Statistical Software*, **31**(3), 1–30. URL http://www.jstatsoft.org/v31/i03/.

Dykstra RL (1981). "An Isotonic Regression Algorithm." *Journal of Statistical Planning and Inference*, **5**, 355–363.

Dykstra RL, Robertson T (1982). "An Algorithm for Isotonic Regression for Two or More Independent Variables." *The Annals of Statistics*, **10**, 708–719.

Efron B (1991). "Regression Percentiles Using Asymmetric Squared Error Loss." *Statistica Sinica*, **1**, 93–125.

Fletcher R (1987). *Practical Optimization*. John Wiley & Sons, Chichester.

Hansohm J (2007). "Algorithms and Error Estimations for Monotone Regression on Partially Preordered Sets." *Journal of Multivariate Analysis*, **98**, 1043–1050.

Henschel V, Heiss C, Mansmann U (2009). ***intcox**: Iterated Convex Minorant Algorithm for Interval Censored Event Data*. R˜package version˜0.9.2, URL http://CRAN.R-project.org/package=intcox.

Hornik K (2005). "A CLUE for CLUster Ensembles." *Journal of Statistical Software*, **14**(12), 1–25. URL http://www.jstatsoft.org/v14/i12/.

Huber P (1981). *Robust Statistics*. John Wiley & Sons, New York.

Koenker R (2005). *Quantile Regression*. Cambridge University Press, Cambridge, MA.

Kruskal JB (1964). "Nonmetric Multidimensional Scaling: A Numerical Method." *Psychometrika*, **29**, 115–129.

Leitner C, Mair P, Paul B, Wick F, Mittermaier C, Sycha T, Ebenbichler G (2009). "Reliability of Posturographic Measurements in the Assessment of Impaired Sensorimotor Function in Chronic Low Back Pain." *Journal of Electromyography and Kinesiology*, **19**, 380–390.

Miles RE (1959). "The Complete Almagamation into Blocks, by Weighted Means, of a Finite Set of Real Numbers." *Biometrika*, **46**, 317–327.

Nocedal J, Wright SJ (1999). *Numerical Optimization*. Springer-Verlag, New York.

Oron AP (2008). ***cir**: Nonparametric Estimation of Monotone Functions via Isotonic Regression and Centered Isotonic Regression*. R˜package version˜1.0, URL http://CRAN.R-project.org/package=cir.

Panier ER (1987). "An Active Set Method for Solving Linearly Constrained Nonsmooth Optimization Problems." *Mathematical Programming*, **37**, 269–292.

Pilz K, Titoff S (2009). ***monreg**: Nonparametric Monotone Regression*. R˜package version 0.1.1, URL http://CRAN.R-project.org/package=monreg.

Pothoff RF, Roy SN (1964). "A Generalized Multivariate Analysis of Variance Model Useful Especially for Growth Curve Problems." *Biometrika*, **51**, 313–326.

R Development Core Team (2009). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org.

Robertson T, Wright FT, Dykstra RL (1988). *Order Restricted Statistical Inference*. John Wiley & Sons, New York.

Rockafellar RT (1970). *Convex Analysis*. Princeton University Press, Princeton, NJ.

Rufibach K, Duembgen L (2009). ***logcondens**: Estimate a Log-Concave Probability Density from iid Observations*. R package version 1.3.4, URL http://CRAN.R-project.org/package=logcondens.

Strimmer K (2008). ***fdrtool**: Estimation and Control of (Local) False Discovery Rates*. R˜package version˜1.2.5, URL http://CRAN.R-project.org/package=fdrtool.

Strömberg U (1991). "An Algorithm for Isotonic Regression with Arbitrary Convex Distance Function." *Computational Statistics & Data Analysis*, **11**, 205–219.

Turner R (2009). ***Iso**: Functions to Perform Isotonic Regression*. R˜package version 0.0-7, URL http://CRAN.R-project.org/package=Iso.

van Eeden C (1958). *Testing and Estimating Ordered Parameters of Probability Distributions*. Ph.D. thesis, University of Amsterdam.

Warshall S (1962). "A Theorem for Boolean Matrices." *Journal of the Association of Computer Machinery*, **9**, 11–12.

Zangwill WI (1969). *Nonlinear Programming: A Unified Approach*. Prentice-Hall, Engelwood Cliffs, NJ.

Zeileis A (2004). "Econometric Computing with HC and HAC Covariance Matrix Estimators." *Journal of Statistical Software*, **11**(10), 1–17. URL http://www.jstatsoft.org/v11/i10/.

Affiliation:

Jan de Leeuw
Department of Statistics University of California, Los Angeles
E-mail: deleeuw@stat.ucla.edu
URL: http://www.stat.ucla.edu/˜deleeuw/