J Optim Theory Appl (2009) 141: 429–443
DOI 10.1007/s10957-008-9477-0

Isotonic Regression under Lipschitz Constraint

L. Yeganova · W.J. Wilbur

Published online: 7 January 2009
© The Author(s) 2008. This article is published with open access at Springerlink.com

Abstract The pool adjacent violators (PAV) algorithm is an efficient technique for the class of isotonic regression problems with complete ordering. The algorithm yields a stepwise isotonic estimate which approximates the function and assigns maximum likelihood to the data. However, if one has reasons to believe that the data were generated by a continuous function, a smoother estimate may provide a better approximation to that function.

In this paper, we consider the formulation which assumes that the data were generated by a continuous monotonic function obeying the Lipschitz condition. We propose a new algorithm, the Lipschitz pool adjacent violators (LPAV) algorithm, which approximates that function; we prove the convergence of the algorithm and examine its complexity.

Keywords Isotonic regression · Lipschitz continuous function · PAV algorithm

1 Introduction

Given a set of observations, one wants to describe the data by fitting them to a model that describes accurately the relationship between the dependent and independent

Communicated by P.M. Pardalos.

The authors were supported by the Intramural Research Program of NIH, National Library of Medicine.

L. Yeganova (✉) · W.J. Wilbur
Computational Biology Branch, National Center for Biotechnology Information, National Library of Medicine, National Institutes of Health, Bldg. 38A, 8600 Rockville Pike, Bethesda, MD 20894, USA
e-mail: yeganova@ncbi.nlm.nih.gov

W.J. Wilbur
e-mail: wilbur@ncbi.nlm.nih.gov

variables. The model may be used to predict unknown values of the dependent variable for a given value of the independent variable. Such problem falls into the realm of regression analysis. In parametric regression, the data are assumed to come from an *a priori* known class of functions and one wants to find the parameters of the function that yield the best fit to the data. Assuming that the knowledge is correct, the parametric approach provides a simple solution to the regression problem, but it may not capture the whole functional form or the wrong functional form might be chosen. We are, on the other hand, interested in a nonparametric regression where a regression function is estimated without *a priori* knowledge of the parametric form. In particular, we consider here an isotonic regression problem under complete ordering, which belongs to the class of shape-constrained nonparametric regression problems.

The pool adjacent violators algorithm [1–4] is a simple and efficient algorithm often used for the problems of isotonic regression with complete order, which provides an estimate in linear time. PAV does not yield a smooth curve but rather a collection of *blocks* where the regression function is constant. However, one could have reasons to believe that the data originated from some continuous function. Hence, despite the fact that PAV provides the estimate that minimizes the squared error from the data, one may be able to find a better estimate for the function.

In this paper, we consider the formulation where the regression function is required to be isotonic and Lipschitz continuous, which we refer to as isotonic regression under the Lipschitz constraint. We develop the LPAV algorithm, which is a PAV-based algorithm providing the maximum likelihood solution under the Lipschitz constraint. We claim that the solution to that problem can yield a better approximation to the function that generated the data than does PAV.

Section 2 of the paper reviews the PAV algorithm, deals with theoretical issues pertinent to defining the algorithm, and provides the pseudocode. Section 3 generalizes the PAV algorithm to be applied to the groups of points that we call *knots*, instead of individual data points. Section 4 formulates the isotonic regression problem under the Lipschitz constraint, develops the LPAV algorithm, establishes its complexity as $O(N^2)$, and provides the pseudocode for its efficient implementation. We conclude the paper with some remarks.

2 Pool Adjacent Violators

Let $p(s)$ be an unknown monotonically nondecreasing function of some ordered parameter s that describes a process. Suppose that the data $\{(s_i, w_i, p_i)\}_{i=1}^N$ are randomly sampled from that process, where s_i is a parameter value, w_i is the weight or importance of the data at s_i, and p_i is the observed value associated with s_i. In general, such data will be noisy and the monotonic nature of $p(s)$ may not be apparent. The pool adjacent violators (PAV) algorithm provides a simple and efficient way to derive that monotonically nondecreasing estimate of $p(s)$ which minimizes the squared error of an estimate to the data. Under the appropriate assumptions where $p(s)$ can be interpreted as a probability function, the monotonically nondecreasing estimate derived by PAV assigns maximum likelihood to the data. We may assume that no parameter value s_i is repeated in the set. If a value is repeated we simply replace the points that have the given value s_i with a single new point that has the same

parameter value, for which p_i is the weighted average of the contributing points and w_i is the sum of the weights for the contributing points. We may assume also that the data are in order so that $i < j \Rightarrow s_i < s_j$.

We seek a set of values $\{y_i\}_{i=1}^N$ that solve the following problem:

$$\min \sum_{i=1}^{N} w_i (p_i - y_i)^2, \tag{1}$$

$$\text{s.t.} \quad y_i \le y_{i+1}, \quad \forall i = 1, \ldots, N-1. \tag{2}$$

This is a convex quadratic programming problem and we will refer to the set of constraints (2) as monotonicity constraints.

If the sequence $\{y_i\}_{i=1}^N$ is a constant value y, then by elementary calculus it is easy to show that y is the weighted average of p's and this is the unique optimal solution. If the y's are not constant, they partition the set N into K nonempty subsets, called *solution blocks*, defined by a set of intervals $\{[r(j), t(j)]\}_{j=1}^K$ on each of which y takes a different constant value. That value of y is determined as an optimal solution to the following subproblem of (1):

$$\min \sum_{i=r(j)}^{t(j)} w_i (p_i - y)^2. \tag{3}$$

By setting the derivative of the function (3) to zero, we obtain the solution to the convex optimization problem as

$$y_i = \frac{\sum_{k=r(j)}^{t(j)} w_k p_k}{\sum_{k=r(j)}^{t(j)} w_k}, \quad r(j) \le i \le t(j). \tag{4}$$

Let us now consider the properties of these solution blocks. If we split the solution block $[r(j), t(j)]$ at any $n, r(j) \le n < t(j)$ into two consecutive intervals $[r(j), n]$ and $[n+1, t(j)]$, then the corresponding solutions satisfy the inequality

$$\frac{\sum_{k=r(j)}^{n} w_k p_k}{\sum_{k=r(j)}^{n} w_k} \ge \frac{\sum_{k=r(j)}^{t(j)} w_k p_k}{\sum_{k=r(j)}^{t(j)} w_k} \ge \frac{\sum_{k=n+1}^{t(j)} w_k p_k}{\sum_{k=n+1}^{t(j)} w_k}, \quad r(j) \le n < t(j). \tag{5}$$

This must be true because if one of the $\ge$ in inequality (5) could be replaced by $<$ for some n, then the interval $[r(j), t(j)]$ could be broken at that n into two solution blocks with solutions as given by (4) that would yield a better fit to the problem (1). We call the intervals that satisfy condition (5) *irreducible*. The solution blocks are then irreducible intervals. We will use the terms interval and block to mean the same thing.

Solution blocks are maximal among irreducible intervals. This follows from the observation that the union of two overlapping irreducible intervals is again an irreducible interval. That uniquely characterizes solution blocks as being maximal irreducible intervals. It remains to show that these solution blocks exist and how to find them.

We refer to two blocks $[r(j), t(j)]$ and $[r(j+1), t(j+1)]$, where $r(j+1) = t(j)+1$, as consecutive. Two consecutive blocks are in order if

$$\frac{\sum_{k=r(j)}^{t(j)} w_k p_k}{\sum_{k=r(j)}^{t(j)} w_k} = y_{t(j)} < y_{r(j+1)} = \frac{\sum_{k=r(j+1)}^{t(j+1)} w_k p_k}{\sum_{k=r(j+1)}^{t(j+1)} w_k};$$

otherwise, if $y_{t(j)} \geq y_{r(j+1)}$, we say that they are *adjacent violators*. Given such a pair of irreducible violators, it is elementary to show that the pair can be pooled into the union of two blocks to obtain a larger irreducible block. This is the origin of the PAV algorithm. One begins with all blocks consisting of a single integer (degenerate blocks) from 0 to $N-1$. All such blocks are irreducible by default. Adjacent violators are then pooled until this process of pooling is no longer applicable. The end result is a partition of the space into solution blocks, which provide the solution to the minimization problem (1) and (2).

As an illustration of irreducible blocks and pooling to obtain them, consider the set of points

$$w_0 = 1, \quad \ldots, \quad w_5 = 1,$$
$$p_0 = \frac{1}{4}, \quad p_1 = \frac{1}{3}, \quad p_2 = \frac{1}{5}, \quad p_3 = \frac{1}{4}, \quad p_4 = 1, \quad p_5 = \frac{1}{2}.$$

One pass through this set of points identifies that the pairs [1] and [2] as well as [4] and [5] are adjacent violators. When these are pooled, they each result in single blocks of weight 2 with the values $\frac{4}{15}$ and $\frac{3}{4}$, respectively. A second pass through the resulting set of four blocks shows that [1, 2] and [3] are now adjacent violators. They may be pooled to produce the block [1, 3] and the value $\frac{47}{180}$. There are no further violations so that the solution blocks are [0], [1, 3], [4, 5].

Since this is a convex problem with a unique optimum, the order in which the blocks are pooled does not affect the solution. However, to obtain an efficient implementation of PAV one must pool in a careful way. The scheme that we use is to keep two arrays of pointers of length N. An array of forward pointers F keeps at the beginning of each irreducible interval the position of the beginning of the next interval (or N). Likewise, an array of backward pointers B keeps at the beginning of each interval the position of the beginning of the previous interval (or -1). The pseudocode for the algorithm follows.

2.1 PAV Algorithm

Step 1. Set $F[k] := k+1$ and $B[k] := k-1$, $0 \leq k < N$.
Step 2. Set up two first-in-first-out (FIFO) queues Q_1 and Q_2; populate Q_1 with numbers $0, \ldots, N-1$, in order, while Q_2 remains empty.
Step 3. Define a marker variable m and a *flag*.
Step 4. While Q_1 is nonempty, perform Steps 5–7.
Step 5. Set $m := 0$.
Step 6. While Q_1 is nonempty, execute:
Step 6A. $k :=$ dequeue Q_1.

Step 6B. If $(m \leq k)$ perform steps 6C–6D.
Step 6C. Set *flag* $:= 0$.
Step 6D. While $F[k] \neq N$ and intervals at k and $F[k]$ are adjacent violators execute:
Step 6D.I. Pool intervals beginning at k and $F[k]$.
Step 6D.II. Set $u := F[F[k]]$.
Step 6D.III. Redefine $F[k] := u$.
Step 6D.IV. If $(u < N)$ redefine $B[u] := k$.
Step 6D.V. Update $m := u$.
Step 6D.VI. Update *flag* $:= 1$.
Step 6E. If (*flag* $= 1$ and $B[k] \geq 0$) enqueue $B[k]$ in Q_2.
Step 7. Interchange Q_1 and Q_2.

This is an efficient algorithm because, after the initial filling of Q_1, an element is added to Q_2 only if a pooling took place. Thus, in a single invocation of the algorithm, a total of no more than N elements can ever be added to Q_2 and this, in turn, limits the number of tests for violators to at most $2N$. We may conclude that both the space and time complexity of PAV are $O(N)$. See also [5].

The PAV algorithm may be applied to any totally ordered data set and it assigns the estimates to those parameter values that actually occur in the data set. For general applications, it will be important to interpolate and extrapolate from these assigned values to obtain a function estimate for any parameter value. We do this in perhaps the simplest possible way based on the values of p just defined on the points $\{s_i\}_{i=1}^N$.

2.2 Interpolated PAV

Given the function p defined on the set $\{s_i\}_{i=1}^N$, assume that the points come from a totally ordered space S and are listed in increasing order. Then, for any $s \in S$, define $p(s)$ as follows:

Case 1: If $s = s_i$, set $p(s) = p(s_i)$.
Case 2: If $s < s_1$, set $p(s) = p(s_1)$.
Case 3: If $s_i < s < s_{i+1}$, set $p(s) = \frac{p(s_i)+p(s_{i+1})}{2}$.
Case 4: If $s > s_N$, set $p(s) = p(s_N)$.

While one could desire a smoother interpolation, the above interpolation is general. In cases where the data are sparse and the parameter s belongs to the real numbers, one might replace Case 3 by a linear interpolation. However, there is no optimal solution to the interpolation problem without further assumptions. Therefore Sect. 2.2 simply lets us generalize and apply this learning to new parameter values not seen in the training data.

3 Extended PAV

Suppose that, as in the previous section, we have ordered data $\{(s_i, w_i, p_i)\}_{i=1}^N$ sampled from some process and we wish to derive a monotonically nondecreasing estimate of the function $p(s)$ describing that process. Assume now that the data are

partitioned into fixed segments of one or several contiguous points, and with each of these segments we associate a *knot*. The *knot* K_i defined on a segment $[kr(i), kt(i)]$, is an object that starts at $kr(i)$, ends at $kt(i)$, and is characterized by the tuple of heights $[h_{kr(i)+1}, \ldots, h_{kt(i)}]$, where each h_j, $kr(i)+1 \le j \le kt(i)$, is the difference between two consecutive ordinate values of adjacent points in the knot and is positive.

Define the *height* of the knot K_i as the sum $h[K_i] = \sum_{j=kr(i)+1}^{kt(i)} h_j$. The height of the knot is positive except for the degenerate case of a knot, i.e. the knot consisting of a single point, whose height is zero. The tuple of heights $[h_{kr(i)+1}, \ldots, h_{kt(i)}]$ is assumed fixed for any knot. A knot determines a quadratic distance function,

$$K_i(x) = \sum_{j=kr(i)}^{kt(i)} w_j \left(p_j - \left[x + \sum_{u=kr(i)+1}^{j} h_u \right] \right)^2. \tag{6}$$

The x that minimizes the distance function $K_i(x)$ is called the *knot solution*. It is obtained by setting the derivative of the convex function (6) to zero,

$$x = \frac{\sum_{j=kr(i)}^{kt(i)} w_j p_j - \sum_{j=kr(i)+1}^{kt(i)} h_j \sum_{u=j}^{kt(i)} w_u}{\sum_{j=kr(i)}^{kt(i)} w_j}. \tag{7}$$

The knot solution is referred to as $x(K_i)$ and may be thought of as determining the position of the leftmost point in the knot K_i. Combined with the tuple of heights, $x(K_i)$ determines uniquely the solution at all points within the knot K_i as follows:

$$x_j = x(K_i) + \sum_{u=kr(i)+1}^{j} h_u$$

is the solution at the point j, $kr(i) \le j \le kt(i)$. Note that the solution at the rightmost point of the knot $kt(i)$ is just

$$x_{kt(i)} = x(K_i) + h[K_i].$$

Thus, the fixed tuple of heights represents a restriction on the solution, and once $x(K_i)$ is found, all values within the knot are uniquely defined. That characterizes the knot as a rigid object that can shift up and down moving all the points in the knot at once.

Assume now that knots represent the finest possible partition of the set and we would like to derive a monotonically nondecreasing estimate of $p(s)$ by applying the PAV algorithm to the knots $\{K_i\}_{i=1}^{G}$. We are looking for a set of values $\{y_i\}_{i=1}^{G}$ that solve the problem

$$\min \sum_{i=1}^{G} K_i(y_i), \tag{8}$$

$$\text{s.t.} \quad y_i + h[K_i] \le y_{i+1}, \quad \forall i = 1, \ldots, G-1. \tag{9}$$

This is again a convex optimization problem, as we are minimizing a convex function (a sum of quadratic functions) over a convex set, so the local solution of the problem is also global.

If the set of optimal solutions for the knots $\{x(K_i)\}_{i=1}^{G}$ is feasible, i.e. the constraints $x(K_i)+h[K_i] \le x(K_{i+1})$ are satisfied for all pairs of consecutive knots, then $\{x(K_i)\}_{i=1}^{G}$ is the optimal solution to the problem (8), (9) and partitions the set into G subsets, each consisting of a single knot. In general, $\{y_i\}_{i=1}^{G}$ and $\{x(K_i)\}_{i=1}^{G}$ are not the same.

Define a *block of knots* BK_j to be a set of consecutive knots $\{K_i\}_{i=r(j)}^{t(j)}$. Solution for the block BK_j is the set of values $\{y_i\}_{i=r(j)}^{t(j)}$ that solve the following problem:

$$\min \sum_{i=r(j)}^{t(j)} K_i(y_i), \tag{10}$$

$$\text{s.t.} \quad y_i + h[K_i] \le y_{i+1}, \quad \forall i = r(j), \ldots, t(j)-1. \tag{11}$$

For any q, $0 \le q < t(j)-r(j)$ the block BK_j can be partitioned into two blocks $\{K_i\}_{i=r(j)}^{r(j)+q}$ and $\{K_i\}_{i=r(j)+q+1}^{t(j)}$, whose corresponding solutions are $\{y_i\}_{i=r(j)}^{r(j)+q}$ and $\{y_i\}_{i=r(j)+q+1}^{t(j)}$. Define the block of knots to be *irreducible* if any partition of the block into two blocks produces solutions $\{y_i\}_{i=r(j)}^{r(j)+q}$ and $\{y_i\}_{i=r(j)+q+1}^{t(j)}$ that satisfy the inequality

$$y_{r(j)+q} + h[K_{r(j)+q}] \ge y_{r(j)+q+1}, \quad \forall q, 0 \le q < t(j)-r(j).$$

These solutions are called *violating* solutions.

A block of knots determines a *distance function* $BK_j(y)$

$$BK_j(y) = \sum_{i=r(j)}^{t(j)} K_i\left(y + \sum_{p=r(j)}^{i-1} h[K_p]\right). \tag{12}$$

It is a quadratic function and therefore it has a unique unconstrained minimum. The y that minimizes the distance function $BK_j(y)$ is called the *block solution* and is found by setting the derivative of the convex function (12) to zero. We will refer to it as $y(BK_j)$. It may be thought of as determining the position of the leftmost knot in the block BK_j. Minimizing the distance function for the block BK_j is equivalent to finding the minimum of (10) under the assumption that constraints (11) are active, i.e. solving the problem

$$\min \sum_{i=r(j)}^{t(j)} K_i(y_i),$$

$$\text{s.t.} \quad y_i + h[K_i] = y_{i+1}, \quad \forall i = r(j), \ldots, t(j)-1.$$

Combined with the heights of knots in the block BK_j, $y(BK_j)$ uniquely determines a value at the beginning of every knot K_i in the block BK_j, as $y(BK_j)+$

$\sum_{p=r(j)}^{i-1} h[K_p]$. To distinguish the block solution, we will denote it with a tilde as $\tilde{y}_i = y(BK_j) + \sum_{p=r(j)}^{i-1} h[K_p], \forall i = r(j), \ldots, t(j)$, and use $y(BK_j)$ and $\{\tilde{y}_i\}_{i=r(j)}^{t(j)}$ interchangeably.

Let us now examine the properties of the blocks.

Theorem 3.1 *A block of knots is irreducible if and only if its block solution is the solution.*

Proof Consider the block $BK_j = \{K_i\}_{i=r(j)}^{t(j)}$. The solution for the block BK_j is the set of values $\{y_i\}_{i=r(j)}^{t(j)}$ that solve the problem (10), (11). This solution exists and is unique.

Let us first show that, if the block BK_j is irreducible, then the block solution $y(BK_j)$ is the solution for the problem (10), (11), i.e. $y_{i+1} = y_i + h[K_i]$, $\forall i = r(j), \ldots, t(j) - 1$. Assume, by contradiction, that the block solution is not the solution, and there exists at least one $q, 0 \le q < t(j) - r(j)$, such that $y_{r(j)+q} + h[K_{r(j)+q}] < y_{r(j)+q+1}$. Then, the block BK_j can be partitioned into two blocks $\{K_i\}_{i=r(j)}^{r(j)+q}$ and $\{K_i\}_{i=r(j)+q+1}^{t(j)}$ with nonviolating solutions $\{y_i\}_{i=r(j)}^{r(j)+q}$ and $\{y_i\}_{i=r(j)+q+1}^{t(j)}$. That contradicts the fact that the block BK_j is irreducible. Hence, if the block BK_j is irreducible, the block solution must be the solution.

Now let us show that, if the block solution is the solution, then the block is irreducible. Let $y(BK_j)$ be the block solution and assume, by contradiction, that the block BK_j is not irreducible. Then, there exists a partition of the block BK_j into two blocks $\{K_i\}_{i=r(j)}^{r(j)+q}$ and $\{K_i\}_{i=r(j)+q+1}^{t(j)}$ with solutions $\{y_i\}_{i=r(j)}^{r(j)+q}$ and $\{y_i\}_{i=r(j)+q+1}^{t(j)}$ such that $y_{r(j)+q} + h[K_{r(j)+q}] < y_{r(j)+q+1}$. This contradicts the assumption that the block solution is the solution, i.e., $y_{r(j)+q} + h[K_{r(j)+q}] = y_{r(j)+q+1}$. Then, the block BK_j must be irreducible. □

Let us now take a closer look at the distance function (12), that can be written as

$$\begin{aligned} BK_j(y) &= \sum_{i=r(j)}^{t(j)} K_i\left(y + \sum_{p=r(j)}^{i-1} h[K_p]\right) \\ &= \sum_{i=r(j)}^{t(j)} \sum_{l=kr(i)}^{kt(i)} w_l\left(p_l - \left(y + \sum_{p=r(j)}^{i-1} h[K_p] + \sum_{u=kr(i)+1}^{l} h_u\right)\right)^2. \end{aligned} \tag{13}$$

The derivative of the distance function $BK_j(y)$ is

$$\partial BK_j(y)/\partial y = -2 * \sum_{i=r(j)}^{t(j)} \sum_{l=kr(i)}^{kt(i)} w_l\left(p_l - \left(y + \sum_{p=r(j)}^{i-1} h[K_p] + \sum_{u=kr(i)+1}^{l} h_u\right)\right). \tag{14}$$

That derivative could be thought of as the spring *force*; $\forall l \in [r(j), t(j)]$ the weight w_l is equivalent to the spring constant and the term $(p_l - (y + \sum_{p=r(j)}^{i-1} h[K_p] + \sum_{u=kr(i)+1}^{l} h_u))$ is equivalent to the distance from the equilibrium point p_l at l.

The distance function itself could be thought of as the spring *potential energy*. The spring is in equilibrium when the spring force is zero.

Equivalently, the minimum of the distance function is found by setting its derivative (14) to zero,

$$\sum_{i=r(j)}^{t(j)} \sum_{l=kr(i)}^{kt(i)} w_l \left(p_l - \left(y + \sum_{p=r(j)}^{i-1} h[K_p] + \sum_{u=kr(i)+1}^{l} h_u \right) \right) = 0.$$

At equilibrium, $\forall q, 0 \le q < t(j) - r(j)$ within the block $BK_j(y)$, the force on the left side of q and the force on the right side are equal in value and opposite in sign

$$\sum_{i=r(j)}^{r(j)+q} K_i \left(y + \sum_{p=r(j)}^{i-1} h[K_p] \right) = - \sum_{i=r(j)+q+1}^{t(j)} K_i \left(y + \sum_{p=r(j)}^{i-1} h[K_p] \right). \tag{15}$$

Combining that observation with the definition of irreducible block, i.e. with the fact that any partition of the irreducible block into two blocks produces violating solutions

$$y_{r(j)+q} + h[K_{r(j)+q}] \ge y_{r(j)+q+1},$$

we conclude that the force on the left side of the irreducible block in equilibrium is nonnegative and that the force on the right side is nonpositive. These forces are exerted by the data and point toward the data.

Two consecutive blocks $BK_j = \{K_i\}_{i=r(j)}^{t(j)}$ and $BK_{j+1} = \{K_i\}_{i=r(j+1)}^{t(j+1)}$ with corresponding solutions $\{y_i\}_{i=r(j)}^{t(j)}$ and $\{y_i\}_{i=r(j+1)}^{t(j+1)}$ are in order if $y_{t(j)} + h[K_{t(j)}] < y_{r(j+1)}$; otherwise, we say that they are *adjacent violators*. Given such violators, the pair can be pooled into a larger block.

If the violating blocks BK_j and BK_{j+1} are irreducible, then the following theorem holds.

Theorem 3.2 *Two consecutive violating irreducible blocks of knots, when pooled together, form a larger irreducible block.*

Proof Consider two consecutive violating irreducible blocks $BK_j = \{K_i\}_{i=r(j)}^{t(j)}$ and $BK_{j+1} = \{K_i\}_{i=r(j+1)}^{t(j+1)}$, with corresponding block solutions $\{\tilde{y}_i\}_{i=r(j)}^{t(j)}$ and $\{\tilde{y}_i\}_{i=r(j+1)}^{t(j+1)}$. The blocks are adjacent violators, i.e.

$$\tilde{y}_{t(j)} + h[K_{t(j)}] \ge \tilde{y}_{r(j+1)};$$

hence, their solutions have to be adjusted to satisfy the monotonicity constraint. Let

$$y_{t(j)} = \tilde{y}_{t(j)} - \Delta y_{t(j)}, \quad \Delta y_{t(j)} \ge 0,$$

and

$$y_{r(j+1)} = \tilde{y}_{r(j+1)} + \Delta y_{r(j+1)}, \quad \Delta y_{r(j+1)} \ge 0,$$

be the new values of the solution at $t(j)$ and $r(j+1)$, so that the knots at $t(j)$ and $r(j+1)$ do not violate the monotonicity constraint, i.e.

$$y_{t(j)} + h[K_{t(j)}] = y_{r(j+1)}.$$

Since $\{\tilde{y}_i\}_{i=r(j+1)}^{t(j+1)}$ is the block solution, then

$$\tilde{y}_{i+1} = \tilde{y}_i + h[K_i], \quad \forall i \in [r(j+1), t(j+1)).$$

Now $y_{r(j+1)} = \tilde{y}_{r(j+1)} + \Delta y_{r(j+1)}$ and $\tilde{y}_{r(j+1)+1}$ are adjacent violators. From (15), we know that, $\forall q, 0 \le q < t(j) - r(j)$, the data on the right side of q exert non-positive force pushing down the right side of the irreducible block BK_{j+1}, and hence $\tilde{y}_{r(j+1)+1}$. Then, the minimum energy subject to the monotonicity constraint is achieved at

$$y_{r(j+1)+1} = y_{r(j+1)} + h[K_{r(j+1)}].$$

That argument is true along the whole block BK_{j+1}, as positive shift in the left side of an irreducible block results in a cascade of adjacent violators in the right side of the block. Using the same argument, the negative shift on the right side of the irreducible block BK_j produces a cascade of adjacent violators in the left side of the block. Hence, the block solutions $\{\tilde{y}_i\}_{i=r(j)}^{t(j)}$ and $\{\tilde{y}_i\}_{i=r(j+1)}^{t(j+1)}$ will rigidly shift to $\{\tilde{y}_i - \Delta y_{t(j)}\}_{i=r(j)}^{t(j)}$ and $\{\tilde{y}_i + \Delta y_{r(j+1)}\}_{i=r(j+1)}^{t(j+1)}$, such that

$$y_{r(j+1)+1} = y_{r(j+1)} + h[K_{r(j+1)}].$$

Then, the resulting solution is the block solution, which proves that the union of two irreducible blocks is a larger irreducible block. □

Thus, maximal irreducible blocks characterize the unique optimal solution to the problem (8), (9). These blocks are found by pooling violating irreducible blocks into larger irreducible blocks. One begins with all blocks consisting of a single knot; such blocks are irreducible by default. Adjacent violators are then pooled until this process of pooling is no longer applicable. The end result is a partition of the space into irreducible blocks which provide the solution to the order constrained minimization problem (8), (9).

The pseudocode for the PAV algorithm applied to the knots is similar to the one presented in Sect. 2.1 of the paper, with the only difference that the definition of adjacent violators for blocks of knots takes into consideration the heights of knots.

4 Pool Adjacent Violators under the Lipschitz Condition

Suppose that we wish to derive a monotonically nondecreasing estimate of the function $p(s)$ given the ordered data $\{(s_i, w_i, p_i)\}_{i=1}^{N}$. Additionally, assume that the data were generated by a process described by a continuous monotonically nondecreasing function obeying the Lipschitz condition suggesting that the regression function

should also belong to the class of Lipschitz functions with some nonnegative parameter λ [6], i.e. $p(s) \in L(\lambda)$,

$$L(\lambda) = \{p : |p(r) - p(q)| \leq \lambda |r - q|, \text{ for all } r, q \in R\}. \tag{16}$$

In this section, we consider the problem of the isotonic regression under the Lipschitz constraint, which we formulate as a convex quadratic programming problem. Given the ordered data $\{(s_i, w_i, p_i)\}_{i=1}^N$, find a set of values $\{y_i\}_{i=1}^N$ such that

$$\min \sum_{i=1}^{N} w_i (p_i - y_i)^2, \tag{17}$$

$$\text{s.t.} \quad y_i \leq y_{i+1}, \quad \forall i = 1, \ldots, N-1, \tag{18}$$

$$y_{i+1} - y_i \leq \lambda (s_{i+1} - s_i), \quad \forall i = 1, \ldots, N-1. \tag{19}$$

We will refer to every segment $[i, i+1]$ as a *junction*. Then, the constraints (18) and (19) provide upper and lower bounds on the difference between two consecutive values of y at every junction,

$$0 \leq y_{i+1} - y_i \leq \lambda (s_{i+1} - s_i), \quad \forall i = 1, \ldots, N-1,$$

where the parameter λ determines the upper bound on the slope of the function that we are trying to predict. Actually, we need not restrict the parameter λ to have a constant value; instead, we may suppose that the steepness of the function is governed by a set $\{\lambda_i\}_{i=1}^{N-1}$ of $N-1$ parameters, where for each i, λ_i restricts the slope of the function at the junction $[i, i+1]$. That assumption generalizes the set of constraints (19), which we refer to as Lipschitz constraints, and the problem that we seek to solve becomes

$$\min \sum_{i=1}^{N} w_i (p_i - y_i)^2, \tag{20}$$

$$\text{s.t.} \quad y_i \leq y_{i+1}, \quad \forall i = 1, \ldots, N-1, \tag{21}$$

$$y_{i+1} - y_i \leq \lambda_i (s_{i+1} - s_i), \quad \forall i = 1, \ldots, N-1. \tag{22}$$

Now, for each $i \in [1, N-1]$, the parameter λ_i determines the maximum slope of the function at the junction $[i, i+1]$. We will refer to the junctions that are within a knot as *bound junctions*; otherwise, we will call them *unbound junctions*.

Let us now consider a knot K_j defined along a segment $[kr(j), kt(j)]$. The knot can be partitioned into two knots at any bound junction within the knot. Define a knot to be *tight at a junction* $[i, i+1]$, $kr(j) \leq i < kt(j)$ if the partition of the knot at that junction results in two knots whose knot solutions violate the Lipschitz constraint at that junction. Define a knot to be *tight* if it is tight at every junction in that knot.

The knot solution is found by unconstrained minimization of the corresponding distance function, i.e. by setting the derivative of the distance function to zero, which is equivalent to setting its force to zero. Hence at the equilibrium, the force is zero, which implies that, at any junction within that knot, there are forces acting on both

sides of that junction that are of the same value and different signs. If the knot is tight, the force exerted by the data on the left side is negative and on the right side is positive; otherwise, the Lipschitz condition would not be violated if we partitioned at that junction.

Consider now two consecutive knots K_j and K_{j+1} defined along the segments $[kr(j), kt(j)]$ and $[kr(j+1), kt(j+1)]$, whose corresponding knot solutions violate the Lipschitz constraint

$$x(K_{j+1}) - (x(K_j) + h[K_j]) > \lambda_{kt(j)}(s_{kr(j+1)} - s_{kt(j)})$$

at the junction $[kt(j), kr(j+1)]$. Given such a pair of violators, the two knots can be tied into a union to obtain a larger knot, by restricting the height at the junction $[kt(j), kr(j+1)]$ to

$$h_{kt(j)} = y_{kr(j+1)} - y_{kt(j)} = \lambda_{kt(j)}(s_{kr(j+1)} - s_{kt(j)}). \tag{23}$$

If the violating knots are tight, then the following theorem holds.

Theorem 4.1 *Two consecutive tight knots, whose solutions violate the Lipschitz constraint, when tied together, form a larger tight knot.*

Proof Consider two consecutive tight knots K_j and K_{j+1}, violating the Lipschitz constraint, that have been tied. We want to show that the new knot $K_j \cup K_{j+1}$ is also tight.

The knot solution for $K_j \cup K_{j+1}$ is obtained by minimizing the distance function

$$\sum_{i=kr(j)}^{kt(j+1)} w_i \left(p_i - \left[x + \sum_{u=kr(i)+1}^{i} h_u \right] \right)^2.$$

Let $x(K_j \cup K_{j+1})$ be the knot solution and let $\{y_i\}_{i=kr(j)}^{kt(j+1)}$ be the set of values defining the solution at every point within the knot $K_j \cup K_{j+1}$ as

$$y_i = x(K_j \cup K_{j+1}) + \sum_{u=kr(j)+1}^{i} h_u, \quad \forall i \in [kr(j), kt(j+1)].$$

Because the knots were tied to satisfy the Lipschitz constraint, then

$$y_i \geq x(K_j) + \sum_{u=kr(j)+1}^{i} h_u, \quad \forall i \in [kr(j), kt(j)], \tag{24a}$$

$$y_i \leq x(K_{j+1}) + \sum_{u=kr(j+1)+1}^{i} h_u, \quad \forall i \in [kr(j+1), kt(j+1)]. \tag{24b}$$

The original knot $K_{j+1}(x)$ is tight; hence, if we partition that knot at some bound junction $[i, i+1]$, $kr(j+1) \leq i < kt(j+1)$, the force acting on the left of that

junction is negative, and on the right side of it is positive. Let us now split the union $K_j \cup K_{j+1}$ at the same junction $[i, i+1], kr(j+1) \le i < kt(j+1)$ and consider the forces on the both sides of the junction. The force on the right side of the junction is positive because the new solution is bounded by the original solution as shown in (24a); due to the total force at equilibrium being zero, there must be a negative force acting on the left side of the junction. Since at the solution $\{y_i\}_{i=kr(j)}^{kt(j+1)}$ the Lipschitz constraint is active, then if we split the knot at any junction $[i, i+1], kr(j+1) \le i < kt(j+1)$, the solution will violate the Lipschitz constraint. A similar argument holds for any junction $[i, i+1]$ in the knot K_j. Hence, at any junction, if we split, the solutions violate the Lipschitz constraint. Thus, the knot $K_j \cup K_{j+1}$ is tight. □

Let us now consider irreducible blocks of knots that characterize the solution to the problem (17), (18) obtained by PAV. That *solution* is said to be *tight* at a bound junction if partition of the knot at that junction results in a solution that violates the Lipschitz constraint at that junction. A PAV solution is called a *tight solution*, if it is tight at every bound junction.

Theorem 4.2 *If PAV results in a tight solution violating the Lipschitz constraint at an unbound junction, tying the knots at that junction results in another tight solution.*

Proof Consider two consecutive irreducible blocks $BK_j = \{K_i\}_{i=r(j)}^{t(j)}$ and $BK_{j+1} = \{K_i\}_{i=r(j+1)}^{t(j+1)}$, part of PAV solution, with corresponding block solutions $\{\tilde{y}_i\}_{i=r(j)}^{t(j)}$ and $\{\tilde{y}_i\}_{i=r(j+1)}^{t(j+1)}$ that are tight. Suppose that these blocks violate the Lipschitz constraint at the unbound junction $[r(j+1)-1, r(j+1)]$,

$$\tilde{y}_{r(j+1)} - (\tilde{y}_{t(j)} + h[K_{t(j)}]) > \lambda_{r(j+1)-1}(s_{r(j+1)} - s_{r(j+1)-1}),$$

and that the knots $K_{t(j)}$ and $K_{r(j+1)}$ are tied to satisfy the Lipschitz constraint at that junction. The partition of the irreducible block at any unbound junction results in two blocks with solutions violating the monotonicity constraint. That implies that there is a nonnegative force acting on the left side of the unbound junction and a nonpositive force acting on the right side of it. On the other hand, we assumed that the PAV solution is tight; hence, partition of the block at any bound junction results in blocks with solutions violating the Lipschitz constraint at that junction implying that there is negative force acting on the left side of the bound junction and positive force acting on the right side of it. These suggest that, if we split the block $BK_{j+1} = \{K_i\}_{i=r(j+1)}^{t(j+1)}$ at a bound junction and consider the segment on left side of that junction, excluding the knot $K_{r(j+1)}$ because it has been tied with the knot $K_{t(j)}$, the force acting on that segment will be at least as negative as it was with the knot $K_{r(j+1)}$. Hence, if splitting at a bound junction before produced violating solutions, splitting at that junction will again produce violating solutions. A similar argument holds for all the knots in the solution; therefore, the new PAV solution is tight. □

We mentioned in the proof of the above theorem that, when we tie the consecutive knots $K_{t(j)}$ and $K_{r(j+1)}$ which are part of blocks with Lipschitz violating solutions $\{\tilde{y}_i\}_{i=r(j)}^{t(j)}$ and $\{\tilde{y}_i\}_{i=r(j+1)}^{t(j+1)}$, there arises a nonnegative force acting on the block

$\{K_i\}_{i=r(j)}^{t(j)-1}$ and a nonpositive force acting on the block $\{K_i\}_{i=r(j+1)+1}^{t(j+1)}$. These forces cause changes in the solutions that may propagate to the knot $K_{r(j)}$ on the left side and the knot $K_{t(j+1)}$ on the right, the margins of the initial blocks. Moreover, these changes do not correct a Lipschitz violation at the junction $[r(j)-1, r(j)]$ preceding the block $\{K_i\}_{i=r(j)}^{t(j)}$ and the junction $[t(j+2)-1, t(j+2)]$ following the block $\{K_i\}_{i=r(j+1)}^{t(j+1)}$ because the new solutions $\{y_i\}_{i=r(j)}^{t(j)-1}$ and $\{y_i\}_{i=r(j+1)+1}^{t(j+1)}$ for the blocks $\{K_i\}_{i=r(j)}^{t(j)-1}$ and $\{K_i\}_{i=r(j+1)+1}^{t(j+1)}$ are bounded by the original solutions $\tilde{y}_{t(j)-1} \leq y_{t(j)-1}$ and $y_{r(j+1)+1} \leq \tilde{y}_{r(j+1)+1}$. Hence, if the Lipschitz condition was violated at these junctions before the knots $K_{t(j)}$ and $K_{r(j+1)}$ were tied, it will remain violating after they are tied. So we have the following theorem.

Theorem 4.3 *If PAV results in a tight solution violating the Lipschitz constraint at L unbound junctions, tying the knots at one of these junctions results in a new tight solution that violates the Lipschitz constraint in at least the same remaining $L-1$ unbound junctions.*

Clearly, the Lipschitz constraint is not violated at the unbound junctions within an irreducible block of knots; it can only be violated at the unbound junctions between two consecutive irreducible blocks of knots. If the Lipschitz constraint is violated in more then one such unbound junction of the PAV solution, the pairs of violating irreducible blocks may all be tied at the same time based on Theorem 4.3.

Here, we develop the LPAV algorithm, the pool adjacent violators under the Lipschitz constraint, which derives a monotonically nondecreasing estimate of $p(s)$, that finds the least-squared error estimate while satisfying the Lipschitz constraint. The algorithm starts with all knots consisting of a single point, degenerate knots $\{K_i\}_{i=1}^{N}$, and the partition of the set of N into G nonempty irreducible blocks of knots $\{BK_j\}_{j=1}^{G}$ resulting from PAV. Then, it ties the knots at the unbound junctions violating the Lipschitz constraint, thus redefining the set of knots. At this point, we have built up the knots consisting of up to two points. Now, the old solution is dropped, PAV is applied to the modified set of knots, and the new solution is found.

This describes a single iteration of the algorithm, which continues until there are no more pairs of consecutive irreducible blocks that violate the Lipschitz condition. This is a finite algorithm because, at every iteration, there is at least one less unbound junction where the Lipschitz constraint can be violated. The worst case time complexity of the LPAV algorithm is $O(N^2)$, because there are at most $N-1$ unbound junctions; hence, the PAV algorithm is repeated at most $N-1$ times.

In the implementation of the LPAV algorithm, we use the arrays of pointers from the PAV, and define a new array of pointers K of length N. The array K keeps at the beginning of each knot the position of the beginning of the previous knot.

4.1 LPAV Algorithm

Step 1. Run PAV so as to generate arrays F and B.
Step 2. Set $K[k] := k-1, 1 \leq k < N$.

Step 3. Set up a FIFO queue Q_1 and enqueue $F(j)$ into Q_1 in increasing order of $j, 0 \le j < G$; and set *flag* $= 1$.
Step 4. While (*flag* $= 1$) perform steps 5–8.
Step 5. Set *flag* $:= 0$.
Step 6. While Q_1 is nonempty execute:
  Step 6A. $j :=$ dequeue Q_1.
  Step 6B. If the Lipschitz Constraint is violated at the junction $[j-1, j]$ execute:
    Step 6B.I. Tie the knots beginning at $K[j]$ and j.
    Step 6B.II. Redefine $K[j+1] := K[j]$.
    Step 6B.III. Update *flag* $:= 1$.
Step 7. If (*flag* > 0) run PAV so as to redefine arrays F and B.
Step 8. Reset Q_1 by enqueueing $F(j)$ into Q_1 in increasing order of $j, 0 \le j < G$.

Similar to PAV, the LPAV algorithm assigns an estimate only to those parameter values that actually occur in the data set. For applications, it is important to interpolate and extrapolate from these assigned values to obtain a function estimate for any parameter value. That may be done as described in Sect. 2.2.

5 Concluding Remarks

We have presented the LPAV algorithm such that, given the data, it obtains an accurate approximation of the function that generated the data. We have shown that this is a finite algorithm with the computational complexity of $O(N^2)$.

PAV has been used often for converting a score into a probability. Such probabilities might be used in isolation or could be used as inputs to another classifier. We believe that the LPAV algorithm may be particularly useful in the later cases, as it provides a smoother and more accurate estimate.

Open Access This article is distributed under the terms of the Creative Commons Attribution Noncommercial License which permits any noncommercial use, distribution, and reproduction in any medium, provided the original author(s) and source are credited.

References

1. Ayer, M., Brunk, H.D., Ewing, G.M., Reid, W.T., Silverman, E.: An empirical distribution function for sampling with incomplete information. Ann. Math. Stat. **26**, 641–647 (1954)
2. De Simone, V., Marino, M., Toraldo, G.: Isotonic regression problems. In: Floudas, C.A., Pardalos, P.M. (eds.) Encyclopedia of Optimization, vol. 3, pp. 86–89. Kluwer Academic, Dordrecht (2001)
3. Pardalos, P.M., Xue, G.L., Yong, L.: Efficient computation of an isotonic median regression. Appl. Math. Lett. **8**, 67–70 (1995)
4. Wilbur, J., Yeganova, L., Kim, W.: The Synergy between PAV and AdaBoost. Mach. Learn. **61**, 71–103 (2005)
5. Pardalos, P.M., Xue, G.L.: Algorithms for a class of isotonic regression problems. Algorithmica **23**, 211–222 (1999)
6. Bazaraa, M.S., Sherali, H.D., Shetty, C.M.: Nonlinear Programming Theory and Algorithms. Wiley, New York (1979)