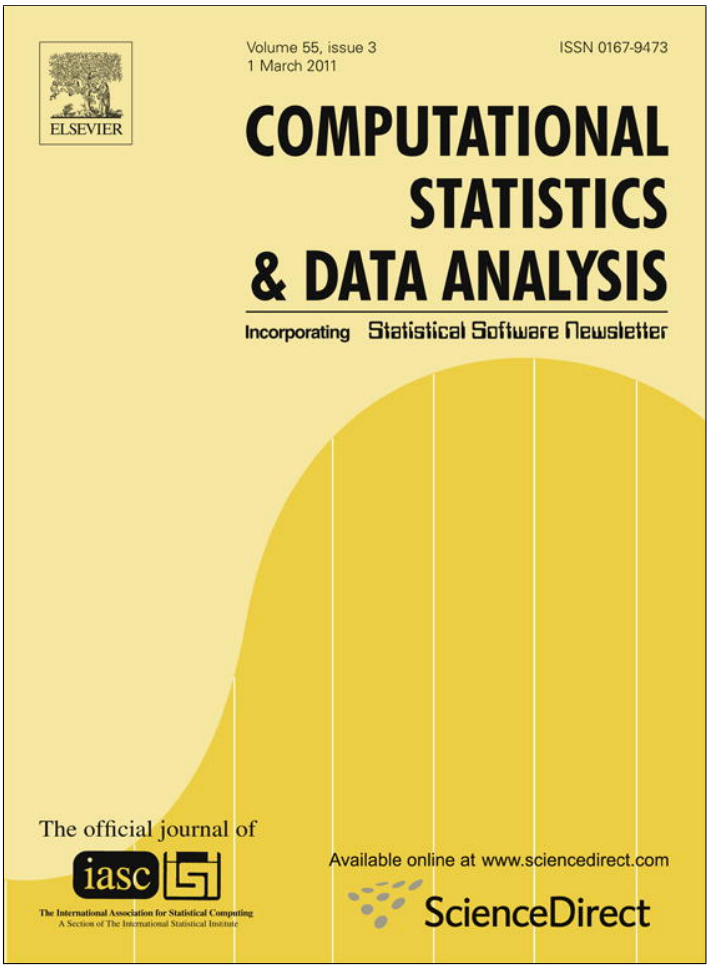

Computational Statistics and Data Analysis 55 (2011) 1173–1179

Contents lists available at ScienceDirect

Computational Statistics and Data Analysis

journal homepage: www.elsevier.com/locate/csda

Exact computation of bivariate projection depth and the Stahel–Donoho estimator☆

Yijun Zuo [a,b,*], Shaoyong Lai [a]

[a] School of Economic Mathematics, Southwestern University of Finance and Economics, Chengdu, Sichuan 610074, China

[b] Department of Statistics and Probability, Michigan State University, East Lansing, MI 48824, USA

ARTICLE INFO

Article history:
Received 12 May 2009
Received in revised form 1 September 2010
Accepted 7 September 2010
Available online 29 September 2010

Keywords:
Projection depth
Exact computation
Stahel–Donoho estimator

ABSTRACT

The idea of data depth provides a new and promising methodology for multivariate nonparametric analysis. Nevertheless, the computation of data depth and the depth function has remained as a very challenging problem which has hindered the methodology from becoming more prevailing in practice. The same is true for the powerful Stahel–Donoho (S–D) estimator. Here, we present an exact algorithm for the computation of the bivariate projection depth (PD) of data points and consequently of the S–D estimator.

1. Introduction

Data depth for multivariate data has triggered the interest of researchers in multivariate nonparametric analysis and robust statistics. Among many proposed notions of data depth, projection depth (Liu, 1992; Zuo and Serfling, 2000a,b; Zuo et al., 2004) is a favorite one among its competitors. Projection depth weighted mean (Zuo et al., 2004) as a very robust alternative to the regular mean also leads to the famous S–D estimator (Stahel, 1981; Donoho, 1982; Tyler, 1994; Maronna and Yohai, 1995; Zuo et al., 2004) as a special case. The latter was the first constructed in high dimension that enjoys very high breakdown point robustness and affine equivalence (see Tyler (1994), Maronna and Yohai (1995) and Zuo et al. (2004)). The computation of data depth and the S–D estimator (seemingly intractable, involving the supremum over infinitely many directions) has remained an open problem in practice for the past two decades. This has hindered the methodology from becoming more prevailing in practice and has prohibited practical inference procedures from developing.

Approximating algorithms for computing projection depth and the S–D estimators have been proposed. Without exact computation results, no one knows how good (accurate) the approximate results are. In this paper we present an algorithm for exact computation of projection depth (subsequently the S–D estimator) for bivariate data.

The paper is organized as follows. Section 2 introduces projection depth and the S–D estimator. Section 3 is devoted to an exact computing algorithm. Section 4 examines the performance of the exact and approximate algorithms in terms of their accuracy and computation times.

2. Projection depth and S–D estimator

2.1. Outlyingness

Let $\mu(F)$ and $\sigma(F)$ be some *robust* location and scale measures of a distribution F of r.v. X in $\mathbb{R}^d$ ($d = 1$), respectively. For simplicity, we consider the most popular robust choice of μ and σ: the median (Med) and the median absolute deviations

☆ This research was partially supported by NSF grants DMS-0234078 and DMS-0501174.

* Corresponding author at: School of Economic Mathematics, Southwestern University of Finance and Economics, Chengdu, Sichuan 610074, China.
E-mail addresses: zuo@msu.edu, yijun.zuo@gmail.com (Y. Zuo), laishaoy@swufe.edu.cn (S. Lai).

0167-9473/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.csda.2010.09.010

(MAD) throughout the paper. Assume that $\sigma(F) > 0$, namely, F is not degenerate. For a given point x, we define in $\mathbb{R}^1$, the one-dimensional *outlyingness* (the standardized absolute deviation of x to center μ of X) of x with respect to (w.r.t.) X as

$$O_1(x, X) = |x - \mu(X)|/\sigma(X) \tag{1}$$

and in $\mathbb{R}^d$, $d > 1$, the high-dimensional *outlyingness* of x w.r.t. X is (see Stahel (1981) and Donoho (1982))

$$O_d(x, X) = \sup_{\|u\|=1} O_1(u'x, u'X) \tag{2}$$

i.e., the worst case projected one-dimensional outlyingness, where $u'x$ and $u'X$ are projections of point x and random variable X to unit direction u. In the sample case, replacing X by the sample data set, we can define the sample versions of (1) and (2) by using the sample versions of μ and σ, respectively.

2.2. Projection depth

With the outlyingness function defined above, we can then define any non-increasing function of it, as follows, which is called *projection depth* (PD) of x w.r.t. X in $\mathbb{R}^d$ (see Liu (1992), Zuo and Serfling (2000a,b) and Zuo (2003))

$$PD(x, X) = 1/(1 + O_d(x, X)). \tag{3}$$

See Remark 2.1 of Zuo (2003) for the reasoning and the logic behind this definition. For concrete examples of *PD* and its sample version, see also Zuo (2003).

2.3. Stahel–Donoho estimator

A general depth weighted mean is defined and studied in Zuo et al. (2004) for a general depth function $D(\cdot, \cdot)$ (see Zuo and Serfling (2000a,b)), a distribution F in $\mathbb{R}^d$, and a weight function w on $[0, 1]$

$$L(F) = \frac{\int x w(D(x, F))F(\mathrm{d}x)}{\int w(D(x, F))F(\mathrm{d}x)}. \tag{4}$$

To ensure a well-defined $L(F)$, we imposed the following conditions:

$$\int w(D(x, F))F(\mathrm{d}x) > 0; \qquad \int \|x\| w(D(x, F))F(\mathrm{d}x) < \infty.$$

With a strategically selected w, the first part holds trivially. The second part holds straightforwardly if $E\|X\| < \infty$ or if $w(D(X, F))$ is 0 outside some bounded set, which is true for typical depth functions.

For a given random sample $X^n = \{X_1, \ldots, X_n\}$ from F or X, let F_n be the empirical distribution that assigns a mass $1/n$ to each point X_i, $i = 1, \ldots, n$. If we replace F in the definition above with F_n, we obtain the sample versions.

If the depth function in the general depth weighted mean (4) is the projection function *PD*, then the weighted mean is called the Stahel–Donoho (S–D) estimator (see Stahel (1981), Donoho (1982) and Zuo et al. (2004)).

3. Exact algorithm for computing outlyingness

Given a sample $X^n = \{x_1, \ldots, x_n\}$, the task is to compute the outlyingness. So we first confine our attention to this computing problem. First recall:

$$O_1(x, X^n) = |x - Med(X^n)|\big/MAD(X^n), \qquad O_d(x, X^n) = \sup_{\|u\|=1} O_1(u'x, u'X^n)$$

where $u'x$ is the projection of x to unit direction u and $u'X^n = \{u'X_1, \ldots, u'X_n\}$. Let $Y_{(1)} \le Y_{(2)} \le \cdots Y_{(n)}$ be order statistics based on $Y^n = \{Y_1, \ldots, Y_n\}$ in R^1, then we have (where $\lfloor\cdot\rfloor$ is the floor function)

$$Med(Y^n) = \frac{Y_{(\lfloor (n+1)/2 \rfloor)} + Y_{(\lfloor (n+2)/2 \rfloor)}}{2},$$
$$MAD(Y^n) = Med\{|Y_i - Med(Y^n)|, i = 1, \ldots, n\}.$$

To facilitate our discussion of the algorithm, we first discuss the idea of a circular sequence (see, e.g. Edelsbrunner (1987)), which has already been employed in the calculation of data depth, depth contours, and trimmed regions by several authors (e.g., Rousseeuw and Ruts (1996), Dyckerhoff (2000) and Cascos and Molchanov (2007)). Here it also plays a key role in our algorithm.

Let a bivariate data set be *in general position* (such that any two straight lines, each of which connects two data points of the given data set, are not parallel). Here it implies that there are no more than two points of the data set on any line. A data set from an absolutely continuous distribution will be in general position with probability 1.

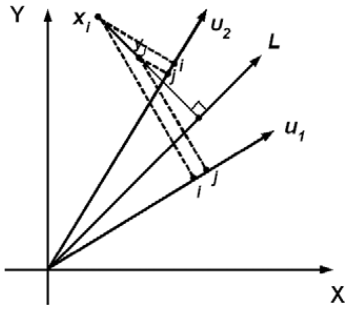

Fig. 1. L is the direction perpendicular to the line segment connecting X_i and Y_j. Project the two points to u_1 and u_2. Then on u_1, i precedes j; this is reversed on u_2.

Now consider an arbitrary direction u that is not perpendicular to any line segment connecting two data points of X^n. Then the projection of X^n on u ($u'X^n$) constitutes a permutation of $\{1, 2, \ldots, n\}$ in terms of their subscripts from left to right (assume that $u'X_{i_1} \le u'X_{i_2} \le \cdots \le u'X_{i_n}$, then $\{i_1, \ldots, i_n\}$ forms a permutation of $\{1, 2, \ldots, n\}$).

We can obtain a sequence of permutations by rotating u counterclockwise. This periodic sequence of permutations is called a *circular sequence*. We note that the permutation obtained from the projection of X^n on u is exactly the reverse of the permutation obtained from the projection of X^n on $-u$. Also observe that two successive permutations of a circular sequence differ only by switching two integers in the sequence. The permutation changes whenever the rotation of u passes through a direction perpendicular to a line segment connecting two data points in a given data set X^n; see Fig. 1.

The idea of the algorithm is to *divide and conquer*. We utilize a circular sequence to partition just the halfplane containing the origin on its boundary (it is sufficient). Given a bivariate data set, we connect any two data points out of the n data points, and get $f(n) = n(n-1)/2$ directions. Take the directions perpendicular to these directions, and determine their polar coordinate angles. We can use these angles (plus $-\pi/2$ and $\pi/2$) to partition the halfplane into $f(n) + 1$ angular regions, and within each region we will have a fixed permutation (circular sequence) when projecting the n data points. This implies that the median of projected data in this region will be the projection of a fixed data point (when n is odd) or the middle point of a line segment connecting two data points (when n is even).

Write $O(u; x) = O_1(u'x; u'X^n)$. It is easy to evaluate $O(u; X_j)$, $j = 1, \ldots, n$ along the $f(n)$ directions that we use to partition the halfplane into angular regions. If we can find the local maxima of $O(u; X_j)$ in each of these regions, then we can find the global maximum of $O(u; X_j)$ in the halfplane (hence in the entire plane). Now the immediate question is when could $O(u; X_j)$ achieve a local maximum? Let us restrict our attention to one of the regions in the following discussion. We can actually show that we do not need to consider all the $f(n) + 1$ but at most $O(n)$ angular regions. Now call the directions that divide the halfplane into $O(n)$ angular regions *Med sequence*.

Here is the description which leads to an algorithm for finding these directions: Let $\alpha(i, j)$ be the (polar coordinate) angle of the direction perpendicular to the line segment connecting X_i and X_j, and let α_k, $k = 1, \ldots, f(n)$ be the sorted sequence of these angles, and $\alpha_0 = 0$. Let u_k be the corresponding directions. Let r_k be the permutation that sorts the projections of the data onto u for $\alpha_{k-1} < u < \alpha_k$. Each angular region is now characterized by its lower limit α_{k-1}, its upper limit α_k, and its permutation r_k. Let m_k be the middle index in permutation r_k if n is odd, while if n is even, let m_k and m'_k be the two middle indices. The algorithm works as follows:

Initial: Let $\ell = 0$, $k(0) = 1$ and determine the middle index m_1 (or pair (m_1, m'_1)) of the permutation r_1. Set $m(0) = m_{k(0)}$(or $= (m_{k(0)}, m'_{k(0)})$).

Loop: For $\ell = 1, 2, \ldots, f(n)$ do the following: Find the first $k > k(\ell - 1)$ for which m_k changes if n is odd, or for which the pair $(m_k; m'_k)$ changes if n is even. Set this k as $k(\ell)$ and $m(\ell) = m_{k(\ell)}$(or $(m_{k(\ell)}, m'_{k(\ell)})$). Find the MAD sequence (see below) within the angular region from $\alpha_{k(\ell)-1}$ to $\alpha_{k(\ell)}$ (see below). Store this sequence and $\alpha_{k(\ell)}$. It will be sufficient to determine the outlyingness values along all the stored directions.

Corresponding to each angular region, we have a fixed point, say Y_κ (either an original data point or the middle point of a line segment connecting two data points) whose projection will be the median of the projected data $u'X^n$ for any u within the region. The $MAD(u'X^n)$ for u within this angular region could change only if the projections of two line segments (both of them with Y_κ as one of the end points) become equal to each other. We have to find and include these directions in our search for the maximum of outlyingness. We call these directions as *MAD sequence*.

Here is the description which will lead to the algorithm to find these directions: Let $Z_i = X_i - Y_\kappa$, $i = 1, 2, \ldots, n$. Then there are at most $f(n)$ direction u's such that $|u'Z_i| = |u'Z_j|$, for $i\ (\neq j), j \in \{1, 2, \ldots, n\}$ (see Fig. 2 as to how to find these directions) (we actually need to consider at most $O(n)$ directions). These directions further divide the angular region $(\alpha_{k(\ell)-1}, \alpha_{k(\ell)})$ into at most $f(n)$ sub-angular regions. Set $\alpha^*_0 = \alpha_{k(\ell)-1}$. Now sort the polar angles of these directions within $(\alpha_{k(\ell)-1}, \alpha_{k(\ell)})$, call them $\alpha^*_p, p = 1, \ldots, f(n)$. Let u^*_p be the corresponding directions and r^*_p be the permutation that sorts the absolute deviations $\{|u'Z_i|, i = 1, \ldots, n\}$ of projected data points to the projected median $(u'Y_\kappa)$ onto u for $\alpha^*_{k-1} < u < \alpha^*_k$. Now each of the (at most) $f(n)$ sub-angular regions is characterized by the lower and upper bounds α^*_{p-1} and α^*_p and the permutation r^*_p. Let n_p be the middle index of r^*_p (or (n_p, n'_p) its middle two indices if n is even). As in the Med sequence case, there are actually at most $O(n)$ sub-angular regions with which we need to be concerned.

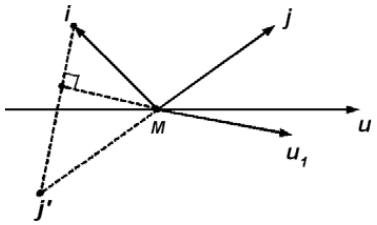

Fig. 2. Point M is the median point of the projected data along u and i and j are two data points. The direction u_1 beyond which the MAD of projected data starts to change.

Initial: Let $\ell = 0$, $p^*(0) = 1$ and determine the middle index n_1 (or middle two indices (n_1, n_1')) of the permutation r_1^*.
Loop: For $\ell = 1, 2, \ldots, f(n)$ do the following: Find the first $p > p^*(\ell - 1)$ for which n_p changes if n is odd, or for which the pair $(n_p; n_p')$ changes if n is even. Set this p as $p^*(\ell)$ and $n(\ell) = n_{p^*(\ell)}$ (or $= (n_{p^*(\ell)}; n'_{p^*(\ell)})$ if n is even). Store all directions $u^*_{p^*(\ell)}$.

Next we present an example to demonstrate the way to find directions u_p^* $(p = 1, \ldots, f(n))$ on a practical way (of course one could find analytical expressions for these directions). See Fig. 2. Here point $M(Y_\kappa)$ is the fixed point whose projection is the median of the projected data along the direction u; i and j are two other original data points corresponding to X_i and X_j (for convenience we write i and j for them). First we find the image point j' of j w.r.t. point M (i.e. we extend the direction jM to point j' such that M is the middle point of jj'). Now simply connect the points i and j' then the direction u_1 perpendicular to ij' is the direction we sought, i.e. along which line segments Mi and Mj will have the same absolute projected value.

With these directions u_p^*, we further divide each angular region into sub-angular regions such that within each of them the MAD of the projected data $u'X^n$ is the absolute value of the projection of some fixed line segment (connecting two original data points) (in the odd n case) (or the average of the absolute values of the projection of some fixed two line segments (in the even n case)) to this sub-region. Note that within each angular region, *the circular sequence* and *the median point of the projected data* are *fixed*. Therefore the number of these directions will be at most $O(n)$.

With the Med and MAD sequences (essentially directions beyond which median or MAD of the projected data will change immediately), we partition the halfplane into totally $O(n^2)$ sub-angular regions. Now within each of these sub-angular regions, it is easily seen that

$$O_1(u'x; u'X_n) = 2|u'(x - X_j)|/(|u'(X_i - X_j)| + |u'(X_k - X_j)|) \tag{5}$$

for some $i, j \in \{1; \ldots; n\}$ and for any u in the sub-angular region, where j is determined by the Med sequence and i and k by the MAD sequence. i and k are the same as in the odd n case.

Note that (5) is a continuous function of u, reaching its maximum at the boundary of the sub-angular region (one can show that the derivative of O_1 w.r.t. u is always positive or negative within the region). So to get the $\sup_{|u|=1}$ in the definition of (2) we do not have to consider infinitely many u's, we actually only need to consider all the boundary directions of the $O(n^2)$ sub-angular regions which are formed by the Med and MAD sequences.

We announce that there is an R package (called ExPD2D) for the exact computation of projection depth of bivariate data already developed by Zuo and Ye (2009). It is part of CRAN now. You can just download the package in R and get the function to compute the projection depth exactly for any set of bivariate data points and any arbitrary points w.r.t. the given data set. This package is based on Fortran code. But now a direct R code program has also been developed (see http://www.stt.msu.edu/~zuo/ExPDdatanew.txt).

4. Comparison

In this section, we first examine the accuracy of the approximate algorithms and directions they used in the computation, then we compare the time approximate and exact algorithms consumed and the mean squared error w.r.t. the exact results.

4.1. Accuracy and depth induced ranking

We start with a concrete data set (Table 1), which was collected as part of a search for new diagnostic techniques at the University of Wisconsin Medical School. Perspiration from 19 healthy females were measured w.r.t. sweat rate and sodium content and analyzed. The two variables are X_1 (sweat rate) and X_2 (sodium). In Fig. 3, larger size of the dot corresponds to the larger depth of the point.

The projection depth in Fig. 3(b) is the exact depth. Before the development of the exact algorithm in this paper, there were several approximate algorithms. They are sub-sampling (Stahel, 1981), resampling design (Rousseeuw, 1993) and fixed/random direction procedures. Stahel (1981) proposed an algorithm based on sub-sampling for approximate computation of the outlyingness. It sub-samples two data points and determines the perpendicular direction to the line segment connecting the two data points, and using these directions replace infinitely many directions in the definition of the outlyingness. The fixed direction procedure uses fixed directions which cut the upper halfplane into equally spaced pieces (angular regions), while the random direction procedure randomly picks some directions.

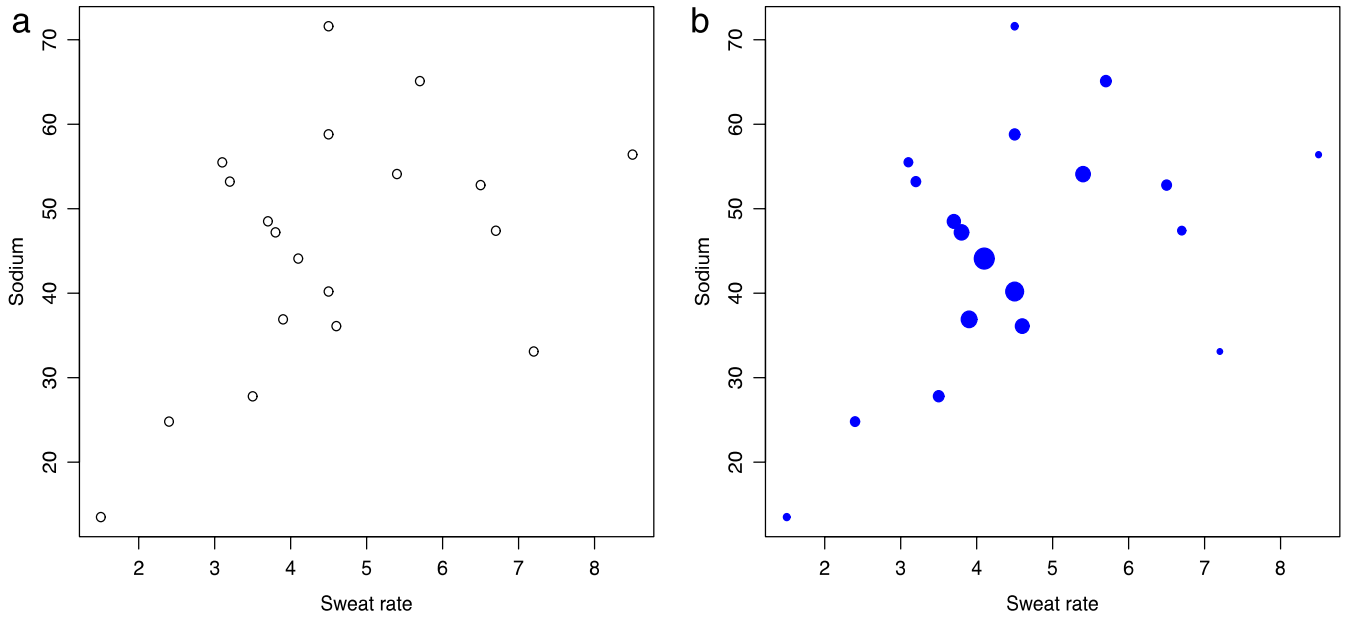

Fig. 3. (a) Left is a scatter plot of sweat rate and sodium of the sweat data. (b) Right is a projection depth-size plot of the sweat data.

Table 1
Sweat data (Johnson and Wichern, 2007, Example 5.1).

Individual	X_1 (sweat rate)	X_2 (sodium)
1	3.7	48.5
2	5.7	65.1
3	3.8	47.2
4	3.2	53.2
5	3.1	55.5
6	4.6	36.1
7	2.4	24.8
8	7.2	33.1
9	6.7	47.4
10	5.4	54.1
11	3.9	36.9
12	4.5	58.8
13	3.5	27.8
14	4.5	40.2
15	1.5	13.5
16	8.5	56.4
17	4.5	40.2
18	6.5	52.8
19	4.1	44.1

Here we compare the exact PD values with approximate ones obtained from the sub-sampling, and fixed direction procedures. We also compare the number of projection directions used by different algorithms and the difference in the depth results and in the ranking induced by the depth of the data points. Note that we can rank multivariate data points based on their depth with large depth corresponding to high rank and small depth corresponding to low rank.

See Table 2 (the last column lists the rank of points induced by the exact depth), where "–" means the same depth as the exact one and the "∗" means that the depth rank of the corresponding point is different from the rank induced by the exact algorithm, and "0000" means that the first 4 digits are the same as the exact one. Note that all the table entries are nonnegative. This is what we expect from the exact computation since we like to get the real supremum. The smaller the PD, the more accurate is the result. The exact algorithm uses 342 u's for projection, while the sub-sampling procedure only uses half the number of u's and gets the results which are very close to the exact ones (only 5 points with slightly different depths), but if one uses some fixed u's, the depth of all points are different even if one uses 100,000 u's (there is still one point with different depth).

4.2. Computation times

Now we examine the average (CPU) time consumed by different algorithms (including the exact (1st), fixed (2nd), and sub-sampling (3rd)) for different sample sizes n based on 3000 replications and simulated normal data sets, we utilize the

Table 2
Comparison of PD (and rank induced) by different algorithms 1st col.: exact PD; the 2nd–6th col.: PD (approximate) − PD (exact); 7th col.: the rank induced by exact PD.

Method # of u's	Exact 342	SubS 171	Fixed	Fixed 171	Fixed 342	Fixed 1400	Rank 10^5
	0.568047337	–	0219	0123	0011	0000	1
	0.568047337	–	0280	0055	0010	0000	2
	0.449288256	–	0002*	0001	0000	0000	3
	0.414516295	–*	0194*	0142*	0013	0000	4
	0.413690236	0303*	0382*	0251*	0002	0001	5
	0.393545029	–	0221	0044	0008	0000	6
	0.375349097	–	0185	0140	0013	0000	7
	0.305747126	0036	0036*	0019	0000	0000	8
	0.303121248	–	0002	0001*	0000	0000	9
	0.30121022	–	0168*	0108*	0010	0000	10
	0.276771606	–	0079*	0083	0001	0000	11
	0.270949533	–	0148*	0121	0011	0000	12
	0.262133297	0073	0004	0006	0000	0000	13
	0.245614035	–	0139	0113	0011	0000	14
	0.234636872	–	0076	0078	0010	0000	15
	0.201177527	–	0129	0071	0008	0000	16
	0.191923191	0061	0003	0004	0000	0000	17
	0.164594729	–	0088	0028	0004	0000	18
	0.154569618	–	0057	0027	0006	0000	19

Table 3
Average CPU time of different algorithms and the empirical mean squared error.

Sample size	Method	Average time (s)	EMSE
30	Exact	2.640693	0
	SubS	0.3868	2.692856
	Fixed	0.6355567	0.316625
50	Exact	19.65004	0
	SubS	1.396973	4.025056
	Fixed	0.8760467	0.5469273
70	Exact	58.57117	0
	SubS	3.150667	5.374985
	Fixed	1.0485	0.794681

system.time() in R to capture the CPU time, but due to the garbage collection time involved, the execution order of the algorithms makes difference. So we start with the 1st algorithm followed by the 2nd algorithm in the first 1000 replications, and the 2nd algorithm first and followed by the 1st algorithm in the second 1000 replications, and the 3rd algorithm first followed by the 1st algorithm in the third 1000 replications; at the same time we compute the empirical mean squared error for different algorithms: $EMSE = \frac{1}{R}\sum_{i=1}^{R} ESE_i$ and $ESE_i = \sum_{j=1}^{n} \|approx(PD)_j - exact(PD)_j\|^2$, where $R = 3000$.

The results in Table 3 were obtained on a Dell Precision M6400 laptop with R 2.9.1. Inspecting the table immediately reveals that the exact algorithm is the slowest (as we expect), followed by the fixed direction algorithm in the small sample size case. The sub-sampling one is the fastest for small sample size, though with the worst accuracy. The fixed direction algorithm keeps a good balance between speed and accuracy, it quite fast, yet quite accurate at all sample sizes.

About the slow exact algorithm, there is a favorable comment from the practitioners. That is, people wait for several months (sometimes years) to gather data. They, of course, are willing to wait for several seconds (or minutes) to get exact results. With that said, we are working on a much faster exact algorithm which should be available any time soon. Our previous version ExPDdatanew has been dramatically improved by cutting the computing time in half at sample size 50.

The sub-sampling algorithm is the fastest one (in the small sample size case) because it only restricts to the directions perpendicular to the line segments connecting two data points, totally $n(n-1)/2$ such directions. Of course its results are the least accurate. Fixed direction algorithm could be faster if one reduces the number of searching directions. One could also improve the accuracy of the fixed direction algorithm by increasing the number of searching directions. In our simulation, we used 1000 directions that equally divide the 0 to π upper halfplanes.

So overall, our suggestion for the approximate computation of PD in two or high dimensions is to use the fixed direction algorithm. The results here, in fact, are consistent with the basic idea of approximate computing of outlyingness in higher dimensions. In higher dimensions, the key idea to compute the outlyingness approximately is to utilize an increasing sequence of outlyingness obtained from different iterations, this sequence theoretically will converge to the exact value with the more and more refined partitions of the upper halfspace (or halfplane in the two dimension case). The idea was first proposed in Dyckerhoff (2004).

Acknowledgements

The author thanks Professor James Stapleton and the referees for their helpful comments and constructive suggestions and the AE and the Editor Stanley Azen for their constant support during the entire process. He is also grateful to Xiangyang Ye for his great assistance in graphs and programming. The research was partially supported by NSF grants DMS-0234078 and DMS-0501174.

References

Cascos, I., Molchanov, I., 2007. Multivariate risks and depth-trimmed regions. Finance Stoch. 11, 373–397.
Donoho, D.L., 1982. Breakdown properties of multivariate location estimators. Ph.D. Qualifying Paper. Dept. Statistics, Harvard University.
Dyckerhoff, R., 2000. Computing zonoid trimmed regions of bivariate data sets. In: Bethlehem, J., van der Heijden, P. (Eds.), COMPSTAT 2000. Proceedings in Computational Statistics. Physica-Verlag, Heidelberg, pp. 295–300.
Dyckerhoff, R., 2004. Data depths satisfying the projection property. Allg. Stat. Arch. 88, 163–190.
Edelsbrunner, H., 1987. Algorithms in Combinatorial Geometry. Springer, Heidelberg.
Johnson, R.A., Wichern, D.W., 2007. Applied Multivariate Statistical Analysis. Prentice Hall.
Liu, R.Y., 1992. Data depth and multivariate rank tests. In: Dodge, Y. (Ed.), L_1-Statistical Analysis and Related Methods. pp. 279–294.
Maronna, R.A., Yohai, V.J., 1995. The behavior of the Stahel–Donoho robust multivariate estimator. J. Amer. Statist. Assoc. 90, 330–341.
Rousseeuw, P.J., 1993. A resampling design for computing high-breakdown point regression. Statist. Probab. Lett. 18, 125–128.
Rousseeuw, P., Ruts, I., 1996. Bivariate location depth. Appl. Statist. 45, 516–526.
Stahel, W.A., 1981. Breakdown of covariance estimators. Research Report 31. Fachgruppe für Statistik. ETH, Zürich.
Tyler, D.E., 1994. Finite sample breakdown points of projection based multivariate location and scatter statistics. Ann. Statist. 22, 1024–1044.
Zuo, Y., 2003. Projection based depth functions and associated medians. Ann. Statist. 31 (5), 1460–1490.
Zuo, Y., Cui, H., He, X., 2004. On the Stahel–Donoho estimators and depth-weighted means for multivariate data. Ann. Statist. 32 (1), 167–188.
Zuo, Y., Serfling, R., 2000a. General notions of statistical depth function. Ann. Statist. 28 (2), 461–482.
Zuo, Y., Serfling, R., 2000b. Structural properties and convergence results for contours of sample statistical depth functions. Ann. Statist. 28 (2), 483–499.
Zuo, Y., Ye, X., 2009. ExPD2D: exact computation of bivariate projection depth based on fortran code. R Package Version 1.0.1. http://CRAN.R-project.org/package=ExPD2D.