Dynamic Update of Half-space Depth Contours

M. Burr* E. Rafalin* D. L. Souvaine*

Data depth is an approach to statistical analysis based on the geometry of the data. *Half-space depth*[1] has been studied most frequently by computational geometers. The *half-space depth* of a point x relative to a set of points $\mathcal{S} = \{X_1, ..., X_n\}$ in $\mathbb{R}^d$ is the minimum number of points of $\mathcal{S}$ lying in any closed half-space determined by a line through x [2, 11][2]. *Depth contours*, enclosing regions with increasing depth, help to visualize, quantify and compare data sets. Prior work investigated combinatorial properties and algorithms for computation of depth contours for *static* data sets. We present a *dynamic algorithm* for computing the two-dimensional *rank-based* half-space depth contours of a set of n points in $O(n \log n)$ time per operation and in $O(n^2)$ overall space, an improvement over the static version of $O(n^2)$ time per operation. The same algorithm can compute the half-space depth of a single point relative to a data set dynamically in $O(\log n)$ time and $O(n)$ space. The algorithm does not compute the entire set of contours explicitly but maintains the order (ranking) of points according to their half-space depth. A constant number of contours (e.g. $10\%, \cdots 100\%$) can be constructed in $O(n)$ time from the sorted list of the data points, ranked by depth. Our algorithm uses *generalized dynamic segment trees* to update the depth of every data point and is based on key characterizations of the potential changes in the depth contours upon insertions or deletions[3]. We only consider data sets in general position.

1 Preliminaries

The statistics community produced contradictory definitions for depth contours. The two main approaches were termed *cover* and *rank* [9]. The **cover** approach defines the contour of depth k as the boundary of the set of points in $\mathbb{R}^d$ with depth $\geq k$ (for half-space depth $1 \leq k \leq \lfloor \frac{n}{2} \rfloor$). The *cover-based* half-space depth contour is provably the boundary of the intersection of all closed half-planes containing exactly $n - k + 1$ data points whose bounding line passes through two data points. The **rank** approach defines the αth central region as the convex hull containing the most central fraction of α sample points [5]. The α-central rank-based half-space-depth contour is constructed by sorting all points of the original set according to their half-space depth, yielding $\{X_{[1]}, \cdots X_{[n]}\}$, the *ranking order* of the points and taking the convex hull of data points $X_{[1]}, \cdots X_{[\alpha]}$. Both approaches assign the same depth value to points that are members of the data set $\mathcal{S}$ and create depth contours that are nested. The main visual difference: vertices of the *rank* contours are only data points while vertices of the *cover* contours can be any point from the data set.

Algorithms have existed for some time for constructing depth contours in 2 and higher dimensions under either definition. The best 2-D implementation for computing the ranking order of a set of points or all *cover*-based depth contours runs in $\Theta(n^2)$ [6]. Other implementations (e.g. [10, 3]) compute cover-based contours.

Much prior work exists on *dynamic geometric structures* (e.g. [7, 1]). To the best of our knowledge, we present the *first* dynamic algorithm for computation of half-space contours, addressing prior interest [4].

2 The Algorithm

Rank-based contours do not have the appealing properties of the *cover-based* contours: e.g. a unique structure that is relatively easy to update. Our algorithm utilizes the fact that a data point has equal depth values under the cover and rank approaches, and computes the *rank-based* contour by considering the *cover-based* contours. Thus, the analysis of our algorithm for the *rank-based* contours refers to the complexity of *cover-based* contours.

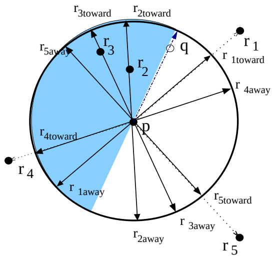
A key idea is, for each data point p, to consider every directed line l passing through p and another data point and the associated closed half-plane H_l to its right. H_l is represented by the unit vector v_{H_l} associated with l, see figure (a). For each point p there is a set of $2(n-1)$ unit vectors, that can be thought of as points on the unit circle, centered at p. Every point $r \in \mathcal{S} \backslash \{p\}$ is assigned to two antipodal vectors, $\hat{r}_{towards}$ and $\hat{r}_{away}$ (where $\hat{r}_{towards}$ is the vector pointing towards r). When a point q is inserted into or deleted from the data set, *exactly the half-planes with associated vectors in the semi-circle counter-clockwise from* $\hat{q}_{toward}$ *to* $\hat{q}_{away}$ have their depth incremented or decremented. These half-planes are updated *simultaneously*, to recompute the depth of p efficiently. To do so we use the concept of *defining lines, half-planes and edges.* If p represents a data point of depth k with respect to set $\mathcal{S}$ of n points, p will appear *exactly* once on the (cover) contour of depth k. If p is on a non-degenerate contour it has exactly two incident edges, the *defining edges of*

*Department of Computer Science, Tufts University, Medford, MA 02155. {mburr,erafalin,dls}@cs.tufts.edu. Partially supported by NSF grant CCF-0431027

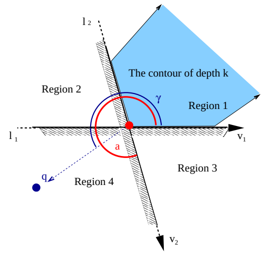
[1] Also called *location depth* or *Tukey depth.*

[2] For the reminder of the paper, every half-plane is considered *closed* unless otherwise mentioned.

[3] A detailed analysis can be found in [8] where we also present an $O(n \log^2 n)$ time and over all $O(n^2)$ space algorithm for dynamically computing *cover-based* half-space depth contours.



(a)



(b)

p, on the contour of depth k. Every edge on any depth contour is a sub-segment of a line created by joining two data points q_1, q_2 of the set $\mathcal{S}$. The *defining lines* l_1, l_2 for p with respect to $\mathcal{S}$ are the lines containing p's *defining edges*. Each defining line l_i, $i \in \{1, 2\}$, bisects the plane into two closed half-planes containing $k+1$ and $n-k+1$ data points where the half-plane containing $k+1$ points does *not* contain the k-th depth contour. The *defining half-planes* H_{l_1}, H_{l_2} for p with respect to $\mathcal{S}$ are the *closed* half-planes bounded by the defining lines of p which contain $k+1$ data points

When point q is inserted into $\mathcal{S}$ the cover-based depth of a point $p \in \mathcal{S}$ remains unchanged if q is inside p's depth contour and can increase only if q is in the region outside p's depth contour. The update of every data point p, when point q is inserted to or deleted from $\mathcal{S}$, depends on the location of q relative to the *defining lines* for p[4]. *Nine cases* **completely** *determine how p's depth changes and how its two defining lines are transformed*[5], see figure (b). These updates are computed in $O(\log n)$ time for each data point p: the number of data points in every half-plane defined by p and each point in $\mathcal{S} \setminus p$ is recomputed; the defining lines of p with respect to the new data set $\mathcal{S} \cup q$ or $\mathcal{S} \setminus q$ are found; the *number of data points in the defining half-plane* is the *updated depth* of p; knowing all half-planes containing exactly $k'+1$ data points and determined by a line through p and another data point makes it possible to determine p's *new defining half-planes* as well. (Note that at least one of the defining half-planes for every data point remains unchanged after a single insertion or deletion, see [8]).

Data Structures: For efficiency the algorithm uses *new generalized dynamic segment trees*. Each tree represents the half-planes passing through a data point $p \in \mathcal{S}$ and is implemented as an augmented dynamic red-black tree.

To re-sort data points, we use a linked list of buckets for depths, from 0 to $n/2$ (the minimum to maximum possible), to hold all data points. Bucket k holds a linked list of data points of depth k. Upon insertion or deletion every point q that changes its depth is moved from its old bucket to a new bucket. Since the depth of q changes by at most 1, the update takes $O(1)$ time.

[4]The data point to be inserted or deleted lies in one of four regions or one of the defining lines, yielding nine cases.

[5]For example, if q is inserted into exactly one defining half-plane H_{l_2}, then p's depth is unchanged, but H_{l_2} is no longer a defining half-plane. It can be shown that the vector for p's new defining half-plane H_{m_2} is the first vector found by traversing the vectors starting from $v_{H_{l_2}}$ towards $v_{H_{l_1}}$ whose associated half-plane contains $k+1$ data points.

3 Open Questions

- The lower bound for computing the half-space depth rank of data points (and thus their rank contours) is $\Omega(n \log n)$, based on reduction to sorting. We are seeking a method to order data points according to their depth in $o(n^2)$.
- To the best of our knowledge, no dynamic algorithm for computing depth contours according to other depth measures exists. We are working on a dynamic scheme to compute regression depth contours (envelopes of the arrangement of lines).
- Most real life experiments are high-dimensional. Since existing static algorithm for computing depth contours for most data depth measures are exponential in dimension, *dynamic approximation algorithms* for depth contours of *multivariate data* are needed.

Acknowledgement: The authors wish to thank S. Venkatasubramanian, S. Krishnan and R. Liu.

References

[1] Y. Chiang and R. Tamassia. Dynamic algorithms in computational geometry. *Proc. of the IEEE*, 80(9):1412–1434, 1992.

[2] J. Hodges. A bivariate sign test. *The Annals of Mathematical Statistics*, 26:523–527, 1955.

[3] S. Krishnan, N. H. Mustafa, and S. Venkatasubramanian. Hardware-assisted computation of depth contours. In *13th ACM-SIAM SODA*, 2002.

[4] R. Liu. Private communications, May 2003. Department of Statistics, Rutgers University.

[5] R. Liu, J. Parelius, and K. Singh. Multivariate analysis by data depth: descriptive statistics, graphics and inference. *The Annals of Statistics*, 27:783–858, 1999.

[6] K. Miller, S. Ramaswami, P. Rousseeuw, T. Sellarés, D. Souvaine, I. Streinu, and A. Struyf. Fast implementation of depth contours using topological sweep. In *Proc. 12th SIAM-ACM SODA*, pages 690–699, 2001.

[7] M. H. Overmars and J. van Leeuwen. Maintenance of configurations in the plane. *J. Comput. System Sci.*, 23(2):166–204, 1981.

[8] E. Rafalin, M. Burr, and D. L. Souvaine. Dynamic update of half-space depth contours. *to appear.*

[9] E. Rafalin and D. L. Souvaine. Data depth contours - a computational geometry perspective. Tech. Report 2004-01, Tufts University, CS Department, May 2004.

[10] I. Ruts and P. J. Rousseeuw. Computing depth contours of bivariate point clouds. *Comp. Stat. and Data Analysis*, 23:153–168, 1996.

[11] J. W. Tukey. Mathematics and the picturing of data. In *Proc. of the Int. Cong. of Math., Vol. 2*, pages 523–531. Canad. Math. Congress, Montreal, Que., 1975.