Robust Space Transformations for Distance-based Operations

Edwin M. Knorr
Dept. of Computer Science
Univ. of British Columbia
Vancouver, BC Canada
knorr@cs.ubc.ca

Raymond T. Ng
Dept. of Computer Science
Univ. of British Columbia
Vancouver, BC Canada
rng@cs.ubc.ca

Ruben H. Zamar
Dept. of Statistics
Univ. of British Columbia
Vancouver, BC Canada
ruben@stat.ubc.ca

ABSTRACT

For many KDD operations, such as nearest neighbor search, distance-based clustering, and outlier detection, there is an underlying k-D data space in which each tuple/object is represented as a point in the space. In the presence of differing scales, variability, correlation, and/or outliers, we may get unintuitive results if an inappropriate space is used.

The fundamental question that this paper addresses is: "What then is an appropriate space?" We propose using a robust space transformation called the Donoho-Stahel estimator. In the first half of the paper, we show the key properties of the estimator. Of particular importance to KDD applications involving databases is the stability property, which says that in spite of frequent updates, the estimator does not: (a) change much, (b) lose its usefulness, or (c) require re-computation. In the second half, we focus on the computation of the estimator for high-dimensional databases. We develop randomized algorithms and evaluate how well they perform empirically. The novel algorithm we develop called the *Hybrid-random* algorithm is, in most cases, at least an order of magnitude faster than the Fixed-angle and Subsampling algorithms.

Keywords

Space Transformations, Data Mining, Outliers, Distance-based Operations, Robust Statistics, Robust Estimators

1. INTRODUCTION

For many KDD operations, such as nearest neighbor search, distance-based clustering, and outlier detection, there is an underlying k-D data space in which each tuple/object is represented as a point in the space. Often times, the tuple $t = \langle a_1, \ldots, a_k \rangle$ is represented simply as the point $p_t = \langle a_1, \ldots, a_k \rangle$ in the k-D space. More formally, the transformation from the tuple t to the point p_t is the identity matrix. We begin by arguing that the identity transformation is not appropriate for many distance-based operations, particularly in the presence of variability, correlation, outliers, and/or differing scales. Consider a dataset with the following attributes:

- systolic blood pressure (typical range: 100-160 mm of mercury, with mean μ=120)
- body temperature (μ = 37 degrees Celsius, with a very small standard deviation (e.g., 1-2 degrees for sick patients))
- age (range: 20-50 years of age in this example)

Note that different attributes have different scales and units (e.g., mm of Hg vs. degree Celsius), and different variability (e.g., high variability for blood pressure vs. low variability for body temperature). Also, attributes may be correlated (e.g., age and blood pressure), and there may be outliers.

EXAMPLE OPERATION 1 (NEAREST NEIGHBOR SEARCH). Consider a nearest neighbor search using the Euclidean distance function in the original data space, i.e., the identity transformation. The results are likely to be dominated by blood pressure readings, because their variability is much higher than that of the other attributes. Consider the query point (blood pressure = 120, body temperature = 37, age = 35). Using Euclidean distance, the point (120, 40, 35) is nearer to the query point than (130, 37, 35) is. But, in terms of similarity/dissimilarity, this finding is not very meaningful because, intuitively, a body temperature of 40 degrees is far away from a body temperature of 37 degrees; in fact, a person with a body temperature of 40 degrees needs medical attention immediately!

A simple fix to the above problem is to somehow weight the various attributes. One common approach is to apply a "normalization" transformation, such as normalizing each attribute into the range [0,1]. This is usually not a satisfactory solution because a single outlier (e.g., blood pressure = 200) could cause virtually all other values to be contained in a small subrange, again making the nearest neighbor search produce less meaningful results.

Another common fix is to apply a "standardization" transformation, such as subtracting the mean from each attribute and then dividing by its standard deviation. While this transformation is superior to the normalization transformation, outliers may still be too influential in skewing the mean and the standard deviation. Equally importantly, this transformation does not take into account possible correlation between attributes. For example, older people tend to have higher blood pressure than younger people. This means that we could be "double counting" when determining distances.

EXAMPLE OPERATION 2 (DATA MINING OPERATIONS). Data clustering is one of the most studied operations in data mining. As an input to a clustering algorithm, a distance function is specified. Although some algorithms can deal with non-metric distance functions (e.g., CLARANS [20]), most algorithms require metric ones. Among those, a subclass of algorithms that has received a lot of attention recently is the class of density-based algorithms (e.g., DBSCAN [11] and DENCLUE [14]). The density of a region is computed based on the number of points contained in a fixed size neighborhood. Thus, density calculation can be viewed as a fixed radius search. Hence, all the concerns raised above for nearest neighbor search apply just the same for density-based clustering.

Outlier detection is another important operation in data mining, particularly for surveillance applications. Many outlier detection algorithms are distance- or density-based [16, 21, 7]. Again, the issues of differing scale, variability, correlation, and outliers could seriously affect the effectiveness of those algorithms. At first glance, the statement that outliers could impact the effectiveness of outlier detection algorithms may seem odd. But if attention is not paid to outliers, it is possible that the outliers may affect the quantities used to scale the data, effectively masking (hiding) themselves [3].

Contributions of this Paper The fundamental question addressed in this paper is: *"What is an appropriate space in the presence of differing scale, variability, correlation, and outliers?"* So far, we have seen that the spaces associated with the identity, normalization, and standardization transformations are inadequate. In this paper, we focus on robust space transformations, or robust *estimators*, so that for distance computation, all points in the space are treated "fairly". Specifically:

- Among many robust space estimators that have been studied in statistics, we propose using the Donoho-Stahel estimator (DSE). In Section 3, we show two important properties of the DSE. The first is the *Euclidean* property. It says that while inappropriate in the original space, the Euclidean distance function becomes reasonable in the DSE transformed space.
- The second, and arguably the more important, property is the *stability* property. It says that the transformed space is robust against updates. That is, in spite of frequent updates, the transformed space does not lose its usefulness and requires no re-computation. Stability is a particularly meaningful property for KDD applications. If an amount of effort x was spent to set up an index in the transformed space, we certainly would not like to spend another amount x after every single update to the database. In Section 3, we give experimental results showing that the DSE transformed space is so stable that it can easily withstand adding many more tuples to the database (e.g., 50% of the database size).
- Having shown its key properties, in the second half of this paper, we focus on the computation of the DSE for high-dimensional (e.g., 10 attributes) databases. The original DSE algorithm was defined independently by both Donoho and Stahel [25]; we refer to it as the *Fixed-angle* algorithm. In Section 4, we show that the original algorithm does not scale well with dimensionality. Stahel also proposed a version of the algorithm which uses subsampling (i.e., taking samples of samples) [25]. However, the number of subsamples to be used in order to obtain good results is not well known. We follow the work of Rousseeuw on least median of squares [22], and come up with a heuristic that seems to work well, as shown in Section 6. For comparison purposes, we have implemented this algorithm, applied some heuristics (e.g., number of subsamples), and evaluated effectiveness and efficiency.
- Last but not least, in Section 5, we develop a new algorithm, which we refer to as the *Hybrid-random* algorithm, for computing the DSE. Our experimental results show that the Hybrid-random algorithm is at least an order of magnitude more efficient than the Fixed-angle and Subsampling algorithms. Furthermore, to support the broader claim that the DSE transformation should be used for KDD operations, the Hybrid-random algorithm can run very efficiently (e.g., compute the estimator for 100,000 5-D tuples in tens of seconds of total time).

Related Work Space transformations have been studied in the database and KDD literature. However, they are from the class of *distance-preserving* transformations (e.g., [12]), where the objective is to reduce the dimensionality of the space. As far as space transformations go, our focus is not so much on preserving distances, but on providing robustness and stability.

Principal component analysis (PCA) is useful for data reduction, and is well-studied in the statistics literature [17, 15, 10]. The idea is to find linear combinations of the attributes, while either maximizing or minimizing the variability. Unfortunately, PCA is not robust since a few outliers can radically affect the results. Outliers can also be masked (hidden by other points). Moreover, PCA lacks the stability requirements that we desire (cf: Section 3). SVD is not robust either; it, too, may fail to detect outliers due to masking.

Many clustering algorithms have been proposed in recent years, and most are distance-based or density-based [11, 26, 1, 14]. The results presented in this paper will improve the effectiveness of all these algorithms in producing more meaningful clusters.

Outlier detection has received considerable attention in recent years. Designed for large high-dimensional datasets, the notion of DB-outliers introduced in [16] is distance-based. A variation of this notion is considered in [21]. The notion of outliers studied in [7] is density-based. Again, all of these notions and detection algorithms will benefit from the results presented in this paper.

Developing effective multi-dimensional indexing structures is the subject of numerous studies [13, 4, 6]. However, this paper is not about indexing structures. Instead, we focus on determining an appropriate space within which an index is to be created.

In [24], nearest neighbor search based on quadratic form distance functions is considered, that is, distances computed using some matrix A. That study assumes prior knowledge of A. For some applications, A may be data-independent and well-known. For example, for a distance between two color histograms, each entry in A represents the degree of perceptual similarity between two colors [12]. However, for most applications, it is far from clear what a suitable A could be. The focus of this paper is to propose a meaningful way of picking A in a data-dependent fashion.

2. BACKGROUND: DONOHO-STAHEL ESTIMATOR

If two similar attributes are being compared, and those attributes are independent and have the same scale and variability, then all points within distance D of a point P lie within the circle of radius D centered at P. In the presence of differing scales, variability, and correlation, all points within distance D of a point P lie within an ellipse. If there is no correlation, then the major and minor axes of the ellipse lie on the standard coordinate axes; but, if there is correlation, then the ellipse is rotated through some angle θ. (See Fig. 4 later.) This generalizes to higher dimensions. In 3-D, the ellipsoid resembles a football, with the covariance determining the football's size, and the correlation determining its orientation.

An *estimator* A, also called a *scatter matrix*, is a $k \times k$ square matrix, where k is the dimensionality of the original data space. An estimator is related to an ellipsoid as follows. Suppose x and y are k-dimensional column vectors. The Euclidean distance between x and y can be expressed as $\mathrm{d}(x,y) = \|x-y\| = \sqrt{\sum_{i=1}^{k}(x_i - y_i)^2} = \sqrt{(x-y)^T(x-y)}$, where T denotes the transpose operator. A quadratic form distance function can be expressed as $\mathrm{d}_A(x,y) = \|x-y\|_A = \sqrt{(x-y)^T A (x-y)}$. For $x \neq \mathbf{0}$, and $x^T A x > 0$, A is called a *positive definite* matrix, and $x^T A x$ yields an ellipsoid.

Donoho-Stahel Estimator and Fixed-angle Algorithm
The DSE is a robust multivariate estimator of location and scatter. Essentially, it is an "outlyingness-weighted" mean and covariance, which downweights any point that is many robust standard deviations away from the sample in some univariate projection [19, 22]. This estimator also possesses desirable statistical properties such as affine equivariance, which not all transformations (e.g., principal component analysis) possess.

The DSE is our estimator of choice, although there are many estimators to choose from [22]. For example, we chose the DSE over the Minimum Volume Ellipsoid estimator because the DSE is easier to compute, scales better, and has much less bias (especially for dimensions > 2) [18]. In our extended work, we consider other robust estimators, but the one that seems to perform the best (based on numerous simulations and analyses) is the DSE. In the interest of space, we only deal with the DSE in this paper. Although the application we focus on here is outlier detection, we add that the DSE is a general transformation that is useful for many applications, such as those described in Section 1.

Fig. 1 gives a skeleton of the initial algorithm proposed by Stahel [25] for computing the estimator in 2-D. Let us step through the algorithm to understand how the estimator is defined. The input is a dataset containing N 2-D points of the form $y_i = [y_{1i}, y_{2i}]^T$. In step 1, we iterate through the unit circle, to consider a large number of possible angles/directions θ, on which to project. We iterate through 180 degrees rather than 360 degrees since the 180-360 degree range is redundant. Hereafter, we call this algorithm the *Fixed-angle* algorithm.

The Fixed-angle Donoho_Stahel Algorithm

1. For $\theta = 0$ to π^- (i.e., $0 \leq \theta < \pi$) using some small increment (e.g., 1 degree), do:
   (a) For $i = 1$ to N, do: compute $x_i(\theta) = y_i \cdot u = y_{1i}\cos\theta + y_{2i}\sin\theta$ (u is the unit vector)
   (b) Compute $m(\theta) = \mathrm{median}(x_i(\theta))$
   (c) Compute $\mathrm{MAD}(\theta) = \mathrm{MAD}(x_i(\theta))$. (The MAD is defined to be $1.4826 * (\mathrm{median}|x_i(\theta) - m(\theta)|)$).
   (d) For $i = 1$ to N, compute $d_i(\theta) = \left|\frac{x_i(\theta)-m(\theta)}{\mathrm{MAD}(\theta)}\right|$
2. For $i = 1$ to N, compute $d_i = \sup_\theta d_i(\theta)$ (i.e., Compute $d_i = \max_{1\leq j\leq k} d_i(\theta_j)$ over all possible θ_j's)
3. Compute the robust multivariate centre $\hat{\mu}_R = \frac{\sum_{i=1}^{N} y_i w(d_i)}{\sum_{i=1}^{N} w(d_i)}$ where $y_i = [y_{1i}\ y_{2i}]^T$ and the weighting function $w(t)$ is defined as follows:
$$w(t) = \begin{cases} 1 & |t| \leq 2.5 \\ \frac{2.5}{|t|} & |t| > 2.5 \end{cases}$$
4. Compute the robust covariance matrix $\hat{\Sigma}_R = \frac{\sum_{i=1}^{N}(y_i-\hat{\mu}_R)(y_i-\hat{\mu}_R)^T[w(d_i)]^2}{\sum_{i=1}^{N}[w(d_i)]^2}$
5. Return the Donoho-Stahel estimator of location and scatter: $(\hat{\mu}_R, \hat{\Sigma}_R)$.

Figure 1: The DSE Fixed-angle Algorithm for 2-D

For each θ, each point is projected onto the line corresponding to rotating the x-axis by θ, giving the value $x_i(\theta)$. Mathematically, this is given by the dot product between y_i and u, which is the unit vector $u = [\cos\theta, \sin\theta]^T$. We call u the *projection vector*.

In step 1(b), we compute $m(\theta)$, which is the median of all the $x_i(\theta)$ values. MAD is an acronym for median absolute deviation from the median. It is a better estimator of scatter than the standard deviation in the presence of outliers. Finally, step 1(d) yields $d_i(\theta)$, which measures how outlying the projection of y_i is with respect to θ. Note that $d_i(\theta)$ is analogous to classical standardization, where each value $x_i(\theta)$ is standardized to $\frac{x_i(\theta)-\mu}{\sigma}$, with μ and σ being the mean and the standard deviation of $x_i(\theta)$, respectively. By replacing the mean with the median, and the standard deviation with the MAD, $d_i(\theta)$ is more robust against the influence of outliers than the value obtained by classical standardization.

Robustness is achieved by first identifying outlying points, and then downweighting their influence. Step 1 computes for each point and each angle θ, the degree of outlyingness of the point with respect to θ. As a measure of how outlying each point is over all possible angles, step 2 computes, for each point, the maximum degree of outlyingness over all possible θ's. In step 3, if this maximum degree for a point is too high (our threshold is 2.5), the influence of this point is weakened by a decreasing weight function. Finally, with all points weighted accordingly, the location center $\hat{\mu}_R$ and the covariance matrix $\hat{\Sigma}_R$ are computed.

3. KEY PROPERTIES OF THE DSE

In this section, we examine whether the estimator is useful for distance-based operations in KDD applications. In Section 6, we provide experimental results showing the difference the estimator can make. But first, in this section, we conduct a more detailed examination of the properties of the estimator. We show that the estimator possesses the *Euclidean* property and the *stability* property, both of which are essential for database applications.

Euclidean Property In this section, we show that once the DSE transformation has been applied, the Euclidean distance function becomes readily applicable. This is what we call the Euclidean property.

LEMMA 1. *The Donoho-Stahel estimator of scatter, $\hat{\Sigma}_R$, is a positive definite matrix.*

The proof is omitted for brevity. According to standard matrix algebra [2], the key implication of the above lemma is that the matrix $\hat{\Sigma}_R$ can be decomposed into: $\hat{\Sigma}_R = Q^T \Lambda Q$, where Λ is a diagonal matrix whose entries are the eigenvalues, and Q is the matrix containing the eigenvectors of $\hat{\Sigma}_R$. This decomposition is critical to the following lemma. It says that the quadratic form distance wrt $\hat{\Sigma}_R$ between two vectors x and y is the same as the Euclidean distance between the transformed vectors in the transformed space.

LEMMA 2. *Let x, y be two vectors in the original space. Suppose they are transformed into the space described by $\hat{\Sigma}_R$, i.e., $x_R = \Lambda^{1/2} Qx$ and $y_R = \Lambda^{1/2} Qy$, where $\hat{\Sigma}_R = Q^T \Lambda Q$. Then, the quadratic form distance wrt $\hat{\Sigma}_R$ is equal to the Euclidean distance between x_R and y_R.*

Proof: $(x-y)^T \hat{\Sigma}_R (x-y) = (x-y)^T Q^T \Lambda Q (x-y) = (x-y)^T Q^T (\Lambda^{1/2} \Lambda^{1/2}) Q (x-y) = [\Lambda^{1/2} Q (x-y)]^T [\Lambda^{1/2} Q (x-y)] = (x_R - y_R)^T (x_R - y_R)$ □

The proof is rather standard, but we include it to provide a context for these comments:

- For each vector x in the original space (or tuple in the relation), each vector is transformed only once, i.e., $x_R = \Lambda^{1/2} Qx$. Future operations do not require any extra transformations. For example, for indexing, all tuples are transformed once and can be stored in an indexing structure. When a query point z is given, z is similarly transformed to z_R. From that point on, the Euclidean distance function can be used for the transformed vectors (e.g., x_R and z_R).
- Furthermore, many existing distance-based structures are the most efficient or effective when dealing with Euclidean-based calculations. Examples include R-trees and variants for indexing [13, 4], and the outlier detection algorithm studied in [16].

The key message here is that space transformation wrt $\hat{\Sigma}_R$ is by itself not expensive to compute, and can bring further efficiency/effectiveness to subsequent processing.

Stability Property The second property we analyze here for the DSE concerns stability. A transformation is *stable* if the transformed space does not lose its usefulness—even in the presence of frequent updates. This is an important issue for database applications. If an amount of effort x was spent in setting up an index in the transformed space, we certainly would not like to spend another amount x after every single update to the index. In statistics, there is the notion of a *breakdown point* of an estimator, which quantifies the proportion of the dataset that can be contaminated without causing the estimator to become "arbitrarily absurd" [27]. But we do not pursue this formal approach regarding breakdown points; instead, we resort to experimental evaluation.

In our experiments, we used a real dataset D and computed the DSE $\hat{\Sigma}_R(D)$. We then inserted or deleted tuples from D, thereby changing D to D_{new}. To measure stability, we compared matrix $\hat{\Sigma}_R(D)$ with $\hat{\Sigma}_R(D_{new})$. In the numerical computation domain, there are a few heuristics for measuring the difference between matrices; but there is no universally agreed-upon metric [9]. To make our comparison more intuitive, we instead picked a distance-based operation—outlier detection—and compared the results. Section 6 gives the details of our experiments, but in brief, we proceeded as follows: (a) We used the old estimator $\hat{\Sigma}_R(D)$ to transform the space for D_{new} and then found all the outliers in D_{new}; and (b) We used the updated estimator $\hat{\Sigma}_R(D_{new})$ to transform the space for D_{new}, and then found all the outliers in D_{new}.

To measure the difference between the two sets of detected outliers, we use standard precision and recall [23], and we define: (i) the *answer set* as the set of outliers found by a given algorithm, and (ii) the *target set* as the "official" set of outliers that are found using a sufficiently exhaustive search (i.e., using the Fixed-angle algorithm with a relatively small angular increment). Precision is the percentage of the answer set that is actually found in the target set. Recall is the percentage of the target set that is in the answer set. Ideally, we want 100% precision and 100% recall.

Fig. 2 shows the results when there were 25%, 50%, 75% and 100% new tuples added to D, and when 25%, 50% and 75% of the tuples in D were deleted from D. The new tuples were randomly chosen and followed the distribution of the tuples originally in D. The deleted tuples were randomly chosen from D. The second and third columns show the number of outliers found, with the third column giving the "real" answer, and the second column giving the "approximated" answer using the old transformation. The fourth and fifth columns show the precision and recall. They clearly show that the DSE transformation is stable. Even a 50% change in the database does not invalidate the old transformation, and re-computation appears unnecessary.

For the results shown in Fig. 2, the newly added tuples followed the same distribution as the tuples originally in D. For the results shown in Fig. 3, we tried a more drastic scenario: the newly added tuples, called junk tuples, followed a totally different distribution. This is reflected by the relatively higher numbers in the second and third columns of Fig. 3. Nevertheless, despite the presence of tuples from two distributions, the precision and recall figures are still close to 100%. This again shows the stability of the DSE.

% Change in Dataset: $D \to D_{new}$	# of Outliers in D_{new}, using: $\hat{\Sigma}_R(D)$	$\hat{\Sigma}_R(D_{new})$	Precision	Recall
25% Inserts	17	15	88.2%	100%
50% Inserts	17	16	94.1%	100%
75% Inserts	32	24	75%	100%
100% Inserts	37	29	78.4%	100%
25% Deletes	13	13	100%	100%
50% Deletes	16	20	100%	80%
75% Deletes	15	19	100%	78.9%

Figure 2: Precision and Recall: Same Distribution

% Junk Tuples Inserted when: $D \rightarrow D_{new}$	# of Outliers in D_{new}, using: $\hat{\Sigma}_R(D)$	$\hat{\Sigma}_R(D_{new})$	Precision	Recall
12.5%	41	40	97.6%	100%
25%	53	52	94.3%	96.2%
37.5%	74	70	91.9%	97.1%
50%	95	90	92.6%	97.8%
62.5%	108	100	90.7%	98.0%

Figure 3: Precision and Recall: Drastically Different Distribution

4. K-D SUBSAMPLING ALGORITHM

In the previous section, we showed that the DSE possesses the desirable Euclidean and stability properties for KDD applications. The remaining question is whether the associated cost is considerable. Let us consider how to compute $\hat{\Sigma}_R$ efficiently, for $k > 2$ dimensions.

Complexity of the Fixed-angle Algorithm Recall that Fig. 1 gives the 2-D Fixed-angle algorithm proposed by Donoho and Stahel. The extension of this algorithm to 3-D and beyond is straightforward. Instead of using a unit circle, we use a unit sphere in 3-D. Thus, there are two angles—θ_1 and θ_2—through which to iterate. Similarly, in k-D, we deal with a unit hypersphere, and there are $k-1$ angles through which to iterate: $\theta_1, \theta_2, \ldots, \theta_{k-1}$.

To understand the performance of the Fixed-angle algorithm, we conduct a complexity analysis. In step 1 of Fig. 1, each angle θ requires finding the median of N values, where N is the size of the dataset. Finding the median takes $O(N)$ time, which is the time that a selection algorithm can partition an array to find the median entry. (Note that sorting is not needed.) Thus, in 2-D, if there are a increments to iterate through, the complexity of the first step is $O(aN)$. For k-D, there are $k-1$ angles to iterate through. If there are a increments for each of these angles, the total complexity of the first step is $O(a^{k-1}kN)$.

In step 2, in the k-D case, there are a^{k-1} projection vectors to evaluate. Thus, the complexity of this step is $O(a^{k-1}N)$. Step 3 finds a robust center, which can be done in $O(kN)$ time. Step 4 sets up the $k \times k$ robust covariance matrix, which takes $O(k^2N)$ time. Hence, the total complexity of the Fixed-angle algorithm is $O(a^{k-1}kN)$. Suffice it to say that running this basic k-D algorithm is impractical for larger values of a and k.

Intuition behind the Subsampling Algorithm in 2-D The first two steps of the Fixed-angle algorithm compute, for each point y_i, the degree of "outlyingness" d_i. The value of d_i is obtained by taking the maximum value of $d_i(\theta)$ over all θ's, where $d_i(\theta)$ measures how outlying the projection of y_i is wrt θ. In the Fixed-angle algorithm, there is an exhaustive enumeration of θ's. For high dimensions, this approach is infeasible.

Let us see if there is a better way to determine "good" projection vectors. Consider points A, B, and C in Fig. 4(a), which shows a 2-D scenario involving correlated attributes. Fig. 4(b) shows the projection of points onto a line orthogonal to the major axis of the ellipse. (Not all points are projected in the figure.) Note that B's projection appears to belong to the bulk of the points projected down from the ellipse; it does not appear to be outlying at all in this projection. Also, although A is outlying on the projection line, C is not. In Fig. 4(c), A and C are not outlying, but B clearly is. Fig. 4(d) shows yet another projection.

As can be seen, the projection vectors which are chosen greatly influence the values for d_i in the Donoho-Stahel algorithm. In applying subsampling, our goal is to use lines orthogonal to the axes of an ellipse (or ellipsoid, in k-D), to improve our odds of obtaining a good projection line. While there may be better projection vectors in which to identify outliers, these are good choices. There is an increased chance of detecting outliers using these orthogonal lines because many outliers are likely to stand out after the orthogonal projection (see Fig. 4). Non-outlying points, especially those within the ellipsoids, are unlikely to stand out because they project to a common or relatively short interval on the line.

If we knew what the axes of the ellipse were, then there would be no need to do subsampling. However, since: (a) we do not know the parameters of the ellipsoid, and (b) in general, there will be too many points and too many dimensions involved in calculating the parameters of the ellipsoid, we use the following approach called subsampling. In 2-D, the idea is to first pick a random point P from the set of N input points. Then compute a line orthogonal to the line joining P and the origin. Note that, with reasonable probability, we are likely to pick a point P in the ellipse, and the resulting orthogonal line may approximate one of the axes of the ellipse. This is the essence of subsampling.

Algorithm Subsampling

For $i = 1, 2, \ldots m$, where m is the number of iterations chosen, do:

(a) Select $k-1$ random points from the dataset. Together with the origin, these points form a hyperplane through the origin, and a subspace V.
   i. Compute a basis for the orthogonal complement of V.
   ii. Choose a unit vector u_i from the orthogonal complement to use as vector u in the Fixed-angle algorithm.

(b) For $j = 1$ to N, do: compute $x_j(i) = y_j \cdot u_i$

(c) (continue with step 1(b) and beyond of the Fixed-angle algorithm shown in Fig. 1, where i takes the role of θ)

Figure 5: DSE Subsampling Algorithm for k-D

More Details of the Subsampling Algorithm in k-D In k-D, we first find a random sample of $k-1$ points. Together with the origin, they form a subspace V. Next, we need to find a subspace that is orthogonal to V, which is called the *orthogonal complement* of V [2]. From this point on, everything else proceeds as in the Fixed-angle algorithm. Fig. 5 outlines the Subsampling algorithm for k-D DSE computation.

One key detail of Fig. 5 deserves elaboration: how to compute m. To determine m, we begin by analyzing the probability of getting a "bad" subsample. For each subsample, $k-1$ points are randomly chosen. A subsample is likely to be good if all $k-1$ points are within the ellipsoid. Let λ (a user-chosen parameter) be the fraction of points outside the ellipsoid. Typically, λ varies between 0.01 to 0.5; the bigger the value, the more conservative or demanding the user is on the quality of the subsamples.

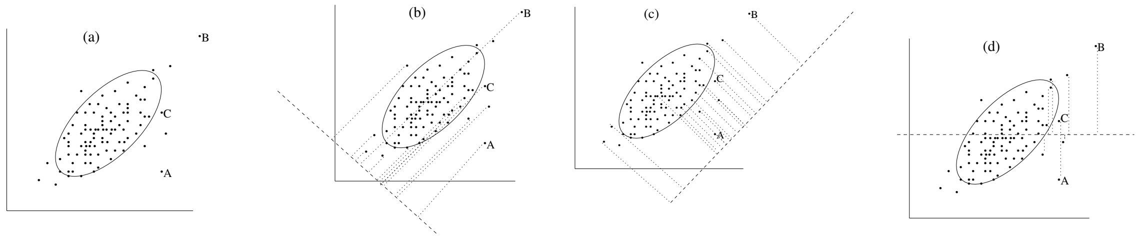


Figure 4: Bivariate Plots Showing the Effect of Different Projection Lines. (a) Data points only. (b) Projection onto a line orthogonal to the major axis of the ellipse. (c) Projection onto a line orthogonal to the minor axis. (d) Projection onto another line.

Let us say that m can be the smallest number of subsamples such that there is at least a 95% probability that we get at least one good subsample out of the m subsamples. Given λ, the probability of getting a "good" subsample is the probability of picking all $k-1$ random points within the ellipsoid, which is $(1-\lambda)^{k-1}$. Conversely, the probability of getting a bad subsample is $1-(1-\lambda)^{k-1}$. Thus, the probability of all m subsamples being bad is $(1-(1-\lambda)^{k-1})^m$. Hence, we can determine a base value of m by solving the following inequality for m: $1-(1-(1-\lambda)^{k-1})^m \geq 0.95$. For example, if $k=5$ and $\lambda=0.50$, then $m=47$. In Section 6, we show how the quality of the estimator varies with m.

Complexity of the Subsampling Algorithm In k-D, we can determine a basis for the orthogonal complement of a hyperplane through the origin and through $k-1$ non-zero points in $O(k^3)$ time, using Gauss-Jordan elimination [2, 9]. Using this basis, we simply pick any unit vector u as our projection vector, and then continue with the basic Fixed-angle algorithm. Recall from Section 4 that the basic algorithm runs in $O(a^{k-1}kN)$ time for step 1 and $O(k^2N)$ time for the remaining steps. For the Subsampling algorithm, however, we perform a total of m iterations, where each iteration consists of $k-1$ randomly selected points, and thus step 1 of the Subsampling algorithm runs in $O(mk^3)$ time. Thus, following the analysis in Section 4, the entire algorithm runs in $O(mk^3+k^2N)$ time.

5. K-D RANDOMIZED ALGORITHMS

The Subsampling algorithm is more scalable with respect to k than the Fixed-angle algorithm is, but the mk^3 complexity factor is still costly when the number of subsamples m is large (i.e., for a high quality estimator). Thus, in this section, we explore how the k-D DSE estimator can be computed more efficiently. First, we implement a simple alternative to the Fixed-angle algorithm, called *Pure-random*. After evaluating its strengths and weaknesses, we develop a new algorithm called *Hybrid-random*, which combines part of the Pure-random algorithm with part of the Subsampling algorithm. In Section 6, we provide experimental results, showing effectiveness and efficiency.

Pure-random Algorithm Recall from Fig. 1 that in the Fixed-angle algorithm, the high complexity is due to the a^{k-1} factor, where a^{k-1} denotes the number of projection unit vectors examined. However, for any given projection unit vector, the complexity of step 1 reduces drastically to $O(kN)$. It is certainly possible for an algorithm to do well if it randomly selects r projections to examine, and if some of those projections happen to be "good" or influential projections. A skeleton of this algorithm called *Pure-random*

Algorithm Pure-random

For $i=1,2,\ldots r$, where r is the number of projection vectors chosen, do:

(a) Select a k-D projection unit vector u_i randomly (i.e., pick $k-1$ random angles)

(b) For $j=1$ to N, do: compute $x_j(i) = y_j \cdot u_i$

(c) (continue with step 1(b) and beyond of the Fixed-angle algorithm, where i takes the role of θ)

Figure 6: DSE Pure-random Algorithm for k-D

is presented in Fig. 6. Following the analysis shown in Section 4, it is easy to see that the complexity of the Pure-random algorithm is $O(rkN+k^2N)=O(rkN)$. Note that randomization is also used in the Subsampling algorithm. But there, each random "draw" is a subspace V formed by $k-1$ points from the dataset, from which the orthogonal complement of V is computed. In the Pure-random case, however, each random draw is a projection vector. In order for the Pure-random algorithm to produce results comparable to that of the Subsampling algorithm, it is very likely that $r \gg m$.

Hybrid-random Algorithm Conceptually, the Pure-random algorithm probes the k-D space blindly. This is the reason why the value of r may need to be high for acceptable quality. The question is whether random draws of projection vectors can be done more intelligently. More specifically, are there areas of the k-D space over which the randomization can skip, or equivalently, are there areas on which the randomization should focus?

In a new algorithm that we develop called *Hybrid-random*, we first apply the Subsampling algorithm for a very small number of subsamples. Consider the orthogonal complement of V that passes through the origin. Imagine rotating this line through a small angle anchored at the origin, thus creating a cone. This rotation yields a "patch" on the surface of a k-D unit hypersphere. From the Fixed-angle algorithm, we know that projection vectors too close to each other do not give markedly different results. So, in the second phase of the Hybrid-random algorithm, we will restrict the random draws of projection vectors to stay clear of previously examined cones/patches.

Using the Euclidean inner product and the Law of Cosines, a collision between two vectors a and b occurs if $\text{dist}^2(a,b) = \|a-b\|^2 = 2(1-|a^Tb|) \leq (2\delta)^2$, where δ is the radius of a patch on the surface of the k-D unit hypersphere. To determine δ, we used the following heuristic. We say that vectors a and b are too close to each other if $\cos\alpha \geq 0.95$, where α is the angle between the vectors. Thus, $(2\delta)^2 = 2(1-\cos\alpha) =$

$2(1-0.95) = 0.1$ radians, and hence, as an upper bound, we use $\delta = \frac{\sqrt{0.1}}{2} \approx 0.1581$. Two observations are in order. First, patches that are too large are counterproductive because many promising projection vectors may be excluded. Second, although increasing the number of patches improves accuracy, favourable results can be obtained with relatively few patches (e.g., 100), as will be shown in Section 6.

Fig. 7 gives a skeleton of the Hybrid-random algorithm. Steps 1 to 3 use the Subsampling algorithm to find some initial projection vectors (including the eigenvectors of the scatter matrix) and keep them in S. In each iteration of step 4, a new random projection vector is generated in such a way that it stays clear of existing projection vectors.

Algorithm Hybrid-random

1. Run the Subsampling algorithm for a small number m of iterations (e.g., $m = 24$).
2. Compute the k eigenvectors of the resulting scatter matrix. This gives us an approximation for the axes of the ellipsoid.
3. Initialize the set S of previously examined projection vectors to consist of the m projection vectors from step 1 and the k eigenvectors from step 2.
4. For $i = 1, 2, \ldots r$, where r is the number of extra random patches desired, do:
   (a) From S, randomly select 2 unique vectors a and b that are at least 2δ radians apart.
   (b) Compute a new vector u_i that is a linear combination of a and b. In particular, $u_i = \gamma a + (1-\gamma) b$, where γ is randomly chosen between [δ, 1-δ].
   (c) If u_i is within δ radians from an existing vector in S, then redo the previous step with a new γ. If these two vectors are still too close during the second attempt, then go back to step 4(a).
   (d) Normalize u_i so that it is a unit vector, and add it to S.
   (e) For $j = 1$ to N, do: compute $x_j(i) = y_j \cdot u_i$.
   (f) (continue with step 1(b) and beyond of the Fixed-angle algorithm, where i takes the role of θ)

Figure 7: DSE Hybrid-random Algorithm for k-D

Recall from our earlier discussion that the complexity of the Subsampling algorithm is $O(m_1 k^3 + k^2 N)$, where m_1 is the number of subsamples taken. As for the Pure-random algorithm, the complexity is $O(r_1 k N)$, where r_1 is the number of random projections probed. It is easy to see that the Hybrid-random algorithm requires a complexity of $O(m_2 k^3 + r_2 k N)$. We expect that $m_2 \ll m_1$, and $r_2 \ll r_1$. Experimental results follow.

6. EXPERIMENTAL EVALUATION

Experimental Setup To evaluate the Donoho-Stahel transformation, we picked the distance-based outlier detection operation described in [16]. As explained in Section 3, we use precision and recall [23], to compare the results.

Our base dataset is an 855-record dataset consisting of 1995-96 National Hockey League (NHL) player performance statistics. These publicly available statistics can be downloaded from sites such as the Professional Hockey Server at http://maxwell.uhh.hawaii.edu/hockey/. Since this real-life dataset is quite small, we created a number of synthetic datasets mirroring the distribution of statistics within the NHL dataset. Specifically, we determined the distribution of each attribute in the original dataset by using a 10-partition histogram. Then, we generated datasets containing up to 100,000 tuples—whose distribution mirrored that of the base dataset. As an optional preprocessing step, we applied the Box and Cox transformation to normality [8] to find appropriate parameters p and D for the distance-based outliers implementation. Unless otherwise stated, we used a 5-D case of 100,000 tuples as our default, where the attributes are goals, assists, penalty minutes, shots on goal, and games played.

Our tests were run on Sun Microsystems Ultra-1 processor, running SunOS 5.7, and having 256 MB of main memory. Of the four DSE algorithms presented, only the Fixed-angle algorithm is deterministic. The other three involve randomization, so we used the median results of several runs. Precision was almost always 100%, but recall often varied.

Usefulness of Donoho-Stahel Transformation In the introduction, we motivated the usefulness of the Donoho-Stahel transformation by arguing that the identity transformation (i.e., raw data), as well as the normalization and standardization transformations, may not give good results. In the experiment reported below, we show a more concrete situation based on outlier detection. Based on the 1995-96 NHL statistics, we conducted an experiment using the two attributes: `penalty-minutes` and `goals-scored`. We note that the range for `penalty-minutes` was [0,335], and the range for `goals-scored` was [0,69].

Fig. 8 compares the top outliers found using the identity, standardization, and Donoho-Stahel transformations. Also shown are the actual `penalty-minutes` and `goals-scored` by the identified players. With the identity transformation (i.e., no transformation), players with the highest `penalty-minutes` dominate. With classical standardization, the dominance shifts to the players with the highest `goals-scored` (with Matthew Barnaby appearing on both lists). However, in both cases, the identified outliers are "trivial", in the sense that they are merely extreme points for some attribute. Barnaby, May, and Simon were all in the top-5 for `penalty-minutes`; Lemieux and Jagr were the top-2 for `goals-scored`.

With the Donoho-Stahel transformation, the identified outliers are a lot more interesting and surprising. Donald

Transformation	Top Outliers Found	`Penalty-mins.` (raw data)	`Goals-scored` (raw data)
Identity	Matthew Barnaby	335	15
	Brad May	295	15
	Chris Simon	250	16
Standardization	Matthew Barnaby	335	15
	Jaromir Jagr	96	62
	Mario Lemieux	54	69
Donoho-Stahel	Matthew Barnaby	335	15
	Donald Brashear	223	0
	Jan Caloun	0	8
	Joe Mullen	0	8

Figure 8: Identified Outliers: Usefulness of Donoho-Stahel Transformation

Brashear was not even in the top-15 as far as `penalty-minutes` goes, and his `goals-scored` performance was unimpressive, that is, `penalty-minutes` = 223 and `goals-scored` = 0. Yet, he has a unique combination. This is because to amass a high number of penalty minutes, a player needs to play a lot, and if he plays a lot, he is likely to score at least some goals. (Incidentally, 0 goals is an extreme univariate point; however, well over 100 players share this value.)

Similar comments apply to Jan Caloun and Joe Mullen; both had 0 `penalty-minutes` but 8 `goals-scored`. While their raw figures look unimpressive, the players were exceptional in their own ways.[1] The point is, without an appropriate space transformation, these outliers would likely be missed.

Internal Parameters of the Algorithms Every algorithm presented here has key internal parameters. In the Fixed-angle case, it is the parameter a, the number of angles tested per dimension. For the randomization algorithms, there are m, the number of subsamples, and r, the number of random projection vectors. Let us now examine how the choices of these parameters affect the quality of the estimator computed. Precision and recall will be used to evaluate quality. However, for the results presented below, precision was always at 100%. Thus, we only report the recall values.

The four graphs in Fig. 9 each contrast: (i) CPU times, (ii) recall values, and (iii) number of iterations (or patches used) for one of the four algorithms. The left hand y-axis defines CPU times (in *minutes* for the top two graphs, and in *seconds* for the bottom two graphs). The right hand y-axis, *in conjunction with the recall curve* (see each figure's legend) defines recall values. Note, however, that the recall range varies from one graph to another. Fig. 9(a) measures CPU time in minutes, and shows that the Fixed-angle algorithm can take a long time to finish, especially as the number of random angles a tested increases. The horizontal axis is in tens of thousands of iterations. Recall that a small decrease in the angle increment for each dimension can cause a very large number of additional iterations to occur. For many of our datasets, it was necessary to use increments as small as 10 degrees (e.g., 75 hours of CPU time, for 100,000 tuples in 5-D), before determining the number of outliers present. We omit these very long runs from our graphs, to allow us to more clearly contrast CPU times and recall values.

Compared to the Fixed-angle algorithm, the Pure-random algorithm achieves a given level of recall more quickly, although, as Fig. 9(b) shows, it can still take a long time to achieve high levels of recall.

Recall that, for the Subsampling algorithm, a key issue was how many subsamples to use. Based on the heuristic presented in Section 4, the base value of m was determined to be 47, and multiples of 47 subsamples were used. From the recall curve in Fig. 9(c), it is clear that below 47 subsamples, the recall value is poor. But even with 3 * 47 = 141 subsamples, the recall value becomes rather acceptable. This is the strength of the Subsampling algorithm, which can give acceptable results in a short time. But, the recall curve has a diminishing rate of return, and it may take a very long time for Subsampling to reach a high level of recall, as confirmed in Fig. 10.

Since the Hybrid-random algorithm uses the Subsampling algorithm in its first phase (with $\lceil \frac{47}{2} \rceil = 24$ iterations), it is expected that the Hybrid-random algorithm behaves about as well as the Subsampling algorithm, at the beginning, for mediocre levels of recall, such as 70-75% (cf: Fig. 10). But, as shown in Fig. 9(d), if the Hybrid-random algorithm is allowed to execute longer, it steadily and quickly improves the quality of its computation. Thus, in terms of CPU time, we start with the Subsampling curve, but quickly switch to the Pure-random curve to reap the benefits of a fast algorithm and pruned randomization.

Achieving a Given Rate of Recall The above experiment shows how each algorithm trades off efficiency with quality. Having picked a reasonable set of parameter values for each algorithm, let us now compare the algorithms head-to-head. Specifically, for fixed recall rates, we compare the time taken for each algorithm to deliver that recall rate. Because the run time of the Fixed-angle algorithm is typically several orders of magnitude above the others (for comparable quality), we omit the Fixed-angle algorithm results from now on.

Fig. 10 compares the Hybrid-random algorithm with both the Pure-random and Subsampling algorithms, for higher rates of recall. In general, the Subsampling algorithm is very effective for quick, consistent results. However, to improve further on the quality, it can take a very long time. In contrast, when the Hybrid-random algorithm is allowed to run just a bit longer, it can deliver steady improvement on quality. As a case in point, to achieve about 90% recall in the current example, it takes the Subsampling algorithm almost 14 hours to achieve the same level of recall produced by the Hybrid-random algorithm in about two minutes. Nevertheless, we must give the Subsampling algorithm credit for giving the Hybrid-random algorithm an excellent base from which to start its computation.

In Fig. 10, the Pure-random algorithm significantly outperforms the Subsampling algorithm, but this is not always the case. We expect the recall rate for Pure-random to be volatile, and there are cases where the Pure-random algorithm returns substantially different outliers for large numbers of iterations. The Hybrid-random algorithm tends to be more focused and consistent.

Scalability in Dimensionality and Dataset Size Fig. 11(a) shows scalability of dimensionality for the Subsampling and Hybrid-random algorithms. We used moderate levels of recall (e.g., 75%) and 60,000 tuples for this analysis. High levels of recall would favor the Hybrid-random algorithm. The results shown here are for 282 iterations for the Subsampling algorithm, and 90 Patches for the Hybrid-random algorithm. Our experience has shown that these numbers of iterations and patches are satisfactory, assuming we are satisfied with conservative levels of recall. Fig. 11(a) shows that both algorithms scale well, and this confirms our complexity analysis of Section 4.

Fig. 11(b) shows how the Subsampling and Hybrid-random algorithms scale with dataset size, in 5-D, for conservative levels of recall. Again, both algorithms seem to scale well, and again the Hybrid-random algorithm outperforms the

[1] Actually, we did not even hear of Jan Caloun before our experiment. During 1995-96, Caloun played a total of 11 games, and scored 8 goals—almost a goal per game, which is a rarity in the NHL. A search of the World Wide Web reveals that Caloun played a grand total of 13 games in the NHL—11 games in 1995-96, and 2 games in 1996-97—before disappearing from the NHL scene. We also learned that he scored on his first four NHL shots to tie an NHL record.

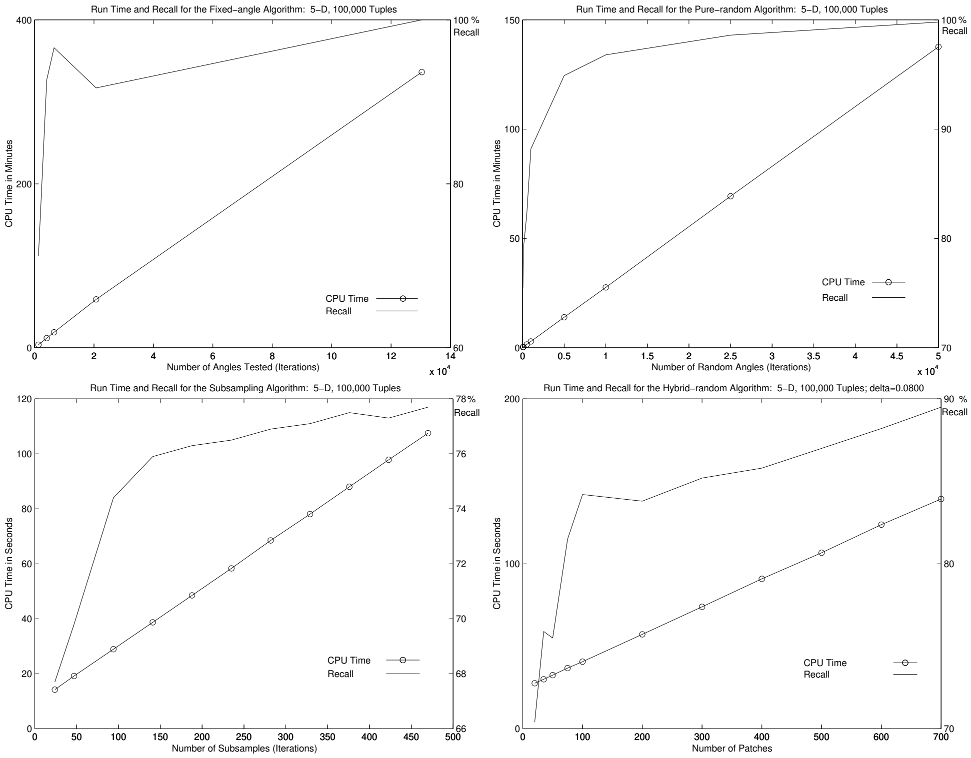


Figure 9: Plots of Run Time and Recall: (a) Top left: Fixed-angle. (b) Top right: Pure-random. (c) Bottom left: Subsampling. (d) Bottom right: Hybrid-random.

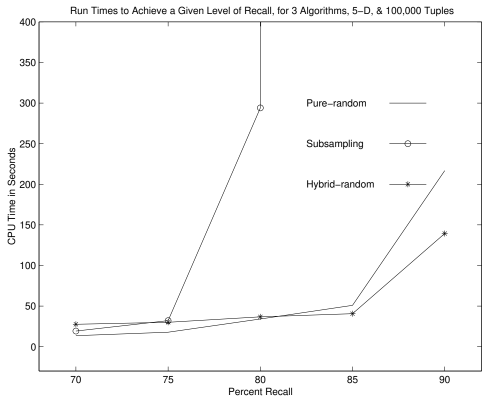


Figure 10: Run Time vs. Recall for Subsampling, Pure-random, and Hybrid-random Algorithms

Subsampling algorithm. High levels of recall would favor the Hybrid-random algorithm, even more so than shown.

7. SUMMARY AND CONCLUSION

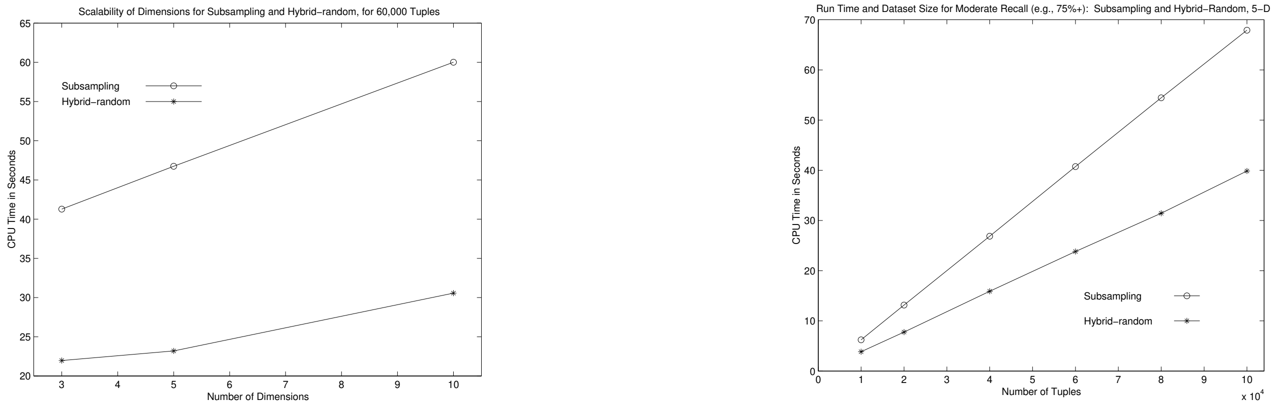
The results returned by many types of distance-based KDD operations/queries tend to be less meaningful when when no attention is paid to scale, variability, correlation, and outliers in the underlying data. In this paper, we presented the case for robust space transformations to support operations such as nearest neighbor search, distance-based clustering, and outlier detection. An appropriate space is one that: (a) preserves the Euclidean property, so that efficient Euclidean distance operations can be performed without sacrificing quality and meaningfulness of results, and (b) is stable in the presence of a non-trivial number of updates. We saw that distance operations which ordinarily would be inappropriate when operating on the raw data (and even on normalized or standardized data), are actually appropriate in the transformed space. Thus, the end user sees results which tend to be more intuitive or meaningful for a given application. We presented a data mining case study on the detection of outliers to support these claims.

After considering issues such as effectiveness (as measured by precision and recall, especially the latter) and efficiency (as measured by scalability both in dimensionality and dataset size), we believe that the Hybrid-random algorithm that we have developed in this paper is an excellent choice among the Donoho-Stahel algorithms. In tens of seconds of CPU time, a robust estimator can be computed which not only accounts for scale, variability, correlation, and outliers, but is also able to withstand a significant number of database updates (e.g., 50% of the tuples) without losing effectiveness or requiring re-computation. For many cases involving high levels of recall, the randomized algorithms, and in particular, the Hybrid-random algorithm can be at least an order of magnitude faster (and sometimes several orders of magnitude faster) than the alternatives. In conclusion, we believe that our results have shown that robust estimation has a place in the KDD community, and can find value in many KDD applications.

8. REFERENCES

[1] R. Agrawal, J. Gehrke, D. Gunopulos and P. Raghavan. Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications. In *Proc. 1998 SIGMOD*, pp. 94-105.

[2] H. Anton and C. Rorres. Elementary Linear Algebra: Applications Version. John Wiley & Sons, 1994.



(a) Scalability of Dimensionality

(b) Scalability of Dataset Size

Figure 11: Scalability of Dimensionality and Dataset Size

[3] V. Barnett and T. Lewis. Outliers in Statistical Data. 3rd edition. John Wiley & Sons, 1994.

[4] N. Beckmann, H.-P. Kriegel, R. Schneider and B. Seeger. The R*-tree: an efficient and robust access method for points and rectangles. In *Proc. 1990 SIGMOD*, pp. 322–331.

[5] K. Bennett, U. Fayyad, and D. Geiger. Density-Based Indexing for Approximate Nearest-Neighbor Queries. In *Proc. 1999 SIGKDD*, pp. 233–243.

[6] T. Bozkaya and M. Ozsoyoglu. Distance-based indexing for high-dimensional metric spaces. In *Proc 1997 SIGMOD*, pp. 357–368.

[7] M. Breunig, H.-P. Kriegel, R. T. Ng, J. Sander. LOF: Identifying Density-Based Local Outliers. In *Proc. 2000 SIGMOD*, pp. 93-104.

[8] G.E.P. Box and D.R. Cox. An Analysis of Transformations (with Discussion). In *Journal of the Royal Statistical Society*, 26, Series B (Methodological), pp. 211–252, 1964.

[9] R. Burden and J. Faires. Numerical Analysis. PWS Publishing, 1993.

[10] C. Croux and A. Ruiz-Gazen. A fast algorithm for robust principal components based on projection pursuit. In Prat, A., editor, *Compstat: Proceedings in Computational Statistics*, Heidelberg: Physica-Verlag, pp. 211–216.

[11] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Proc. 1996 KDD*, pp. 226–231.

[12] C. Faloutsos, R. Barber, M. Flickner, J. Hafner, W. Niblack, D. Petkovic and W. Equitz. Efficient and Effective Querying by Image Content. In: Journal of Intelligent Info. Systems, 3, 4, pp. 231–262, 1994.

[13] A. Guttman. R-trees: a dynamic index structure for spatial searching. In *Proc. 1984 SIGMOD*, pp. 47–57.

[14] A. Hinneburg and D. Keim. An efficient approach to clustering in large multimedia databases with noise. In *Proc. 1998 KDD*, pp. 58–65.

[15] I. Jolliffe. Principal Component Analysis. Springer-Verlag, 1986.

[16] E. Knorr and R. Ng. Algorithms for Mining Distance-Based Outliers in Large Datasets. In *Proc. 1998 VLDB*, pp. 392-403.

[17] G. Li and Z. Chen. Projection-pursuit approach to robust dispersion matrices and principal components: primary theory and Monte Carlo. In *Journal of the American Statistical Association*, 80, pp. 759–766.

[18] R. Martin and R. Zamar. Bias robust estimation of scale. In *The Annals of Statistics*, 21, 2, pp. 991–1017, 1993.

[19] R. Maronna and V. Yohai. The behaviour of the Stahel-Donoho robust multivariate estimator. In *Journal of the American Statistical Association*, 90 (429), pp. 330–341, 1995.

[20] R. Ng and J. Han. Efficient and Effective Clustering Methods for Spatial Searching, In *Proc. 1994 VLDB*, pp. 144–155.

[21] S. Ramaswamy, R. Rastogi and K. Shim. Efficient Algorithms for Mining Outliers from Large Data Sets. In *Proc. 2000 SIGMOD*, pp. 427-438.

[22] P. Rousseeuw and A. Leroy. Robust Regression and Outlier Detection. John Wiley & Sons, 1987.

[23] G. Salton and M. McGill. Introduction to Modern Information Retrieval. McGraw-Hill, 1983.

[24] T. Seidl and H.-P. Kriegel. Efficient User-Adaptable Similarity Search in Large Multimedia Databases. In *Proc. 1997 VLDB*, pp. 506–515.

[25] W. A. Stahel. Breakdown of Covariance Estimators. Research report, 31, Fachgruppe fur Statistik, ETH, Zurich, 1981.

[26] W. Wang, J. Yang and R. Muntz. STING: A statistical information grid approach to spatial data mining. In *Proc. 1997 VLDB*, pp. 186–195.

[27] V. Yohai and R. Zamar. High breakdown point estimates of regression by means of the minimization of an efficient scale. In *Journal of the American Statistical Association*, 83 (402), pp. 406–413, 1988.