On FastMap and the Convex Hull of Multivariate Data: Toward Fast and Robust Dimension Reduction

George Ostrouchov and Nagiza F. Samatova

Abstract—FastMap is a dimension reduction technique that operates on distances between objects. Although only distances are used, implicitly the technique assumes that the objects are points in a \(p\)-dimensional Euclidean space. It selects a sequence of \(k \leq p\) orthogonal axes defined by distant pairs of points (called pivots) and computes the projection of the points onto the orthogonal axes. We show that FastMap picks all of its pivots from the vertices of the convex hull of the data points in the original implicit Euclidean space. This provides a connection to results in robust statistics, where the convex hull is used as a tool in multivariate outlier detection and in robust estimation methods. The connection sheds a new light on some properties of FastMap, in particular its sensitivity to outliers, and provides an opportunity for a new class of dimension reduction algorithms that retain the speed of FastMap and exploit ideas in robust statistics.

Index Terms — Dimension reduction, convex hull, FastMap, principal components, multidimensional scaling, robust statistics, Euclidean distance.

I. INTRODUCTION

Dimension reduction techniques begin with \(n\) objects as points in a \(p\)-dimensional vector space and map the objects onto \(n\) points in a \(k\)-dimensional vector space, where \(k < p\). A more general situation arises when the point coordinates are not known and only pairwise distances (or a distance function to compute them) are available. This mapping of objects based on their distances only into a \(k\)-dimensional vector space is called finite metric space embedding [8].

FastMap is first introduced in [6] as a fast alternative to Multidimensional Scaling (MDS) [13] and a generalization of Principal Component Analysis (PCA) [9]. MDS is a finite metric space embedding method and PCA is a popular dimension reduction method. The context of [6] is similarity searching in multimedia databases. Given dimension \(k\) and Euclidean distances between \(n\) objects, FastMap maps the objects onto \(n\) points in \(k\)-dimensional Euclidean space. An implicit assumption by FastMap that the objects are points in a \(p\)-dimensional Euclidean space (\(p \geq k\)) is noted in [8].

Because of this assumption, FastMap is usually viewed as a dimension reduction method.

When FastMap begins with Euclidean distances between the \(n\) objects, it has time complexity \(O(n^2)\). If the Euclidean distances must be explicitly computed from a \(p\)-dimensional vector representation, FastMap time complexity is \(O(np)\).

This paper concerns how FastMap operates within the the implicit or explicit \(p\)-dimensional Euclidean space containing the points of a data set. FastMap selects a sequence of \(k \leq p\) orthogonal axes defined by distant pairs of points (called pivots) and computes the projections of the points onto the orthogonal axes. We show that FastMap picks all of its pivots from convex hull vertices of the original data set. This provides a connection to results in robust statistics, where the convex hull is used as a tool in multivariate outlier detection and in robust estimation methods. The connection sheds a new light on some properties of FastMap, in particular its sensitivity to outliers, and provides an opportunity for a new class of dimension reduction algorithms that retain the speed of FastMap and exploit ideas in robust statistics.

We begin in Section II by defining the convex hull and some of its properties. In Section III we describe the FastMap algorithm. The main result, showing that FastMap pivots are pairs of vertices of the convex hull is in Section IV. Finally Section V discusses the implications of this result.

II. CONVEX HULL OF A DATA SET

Let \(S\) be a set of \(n\) points in \(p\)-dimensional Euclidean space. The convex hull of \(S\), denoted by \(C(S)\), is the smallest convex set (a polytope) that contains \(S\) [5], [7]. We can visualize a convex hull in two or three dimensions as a rubber band or an elastic bag stretched around the points. In higher dimensions, we must rely on more formal properties of hyperplanes, and the notion of half-space support. Our definitions below are mostly from [5], [7].

Definition 2.1: A hyperplane is an affine subspace (a translation of a linear subspace) of \(\mathbb{R}^p\) with dimension \(p - 1\).

The set of points

\[
h(u, v) = \{x \in \mathbb{R}^p : (u - v)^T(x - v) = 0\}, \text{ for } u, v \in \mathbb{R}^p,
\]

is a hyperplane perpendicular to the vector \(u - v\) and passing through \(v\). The closed half-space that is defined by this hyperplane and that contains \(u\) is given by

\[
H(u, v) = \{x \in \mathbb{R}^p : (u - v)^T(x - v) \geq 0\}, \text{ for } u, v \in \mathbb{R}^p,
\]

Definition 2.2: If \(S\) intersects \(h(u, v)\) and \(S\) lies in \(H(u, v)\) for some \(u, v \in \mathbb{R}^p\), then \(h(u, v)\) is a supporting hyperplane of \(S\) and \(H(u, v)\) is a supporting half-space of \(S\).

We use Ziegler’s [14, section 2.1] definition of a face of a polytope and state it in terms of a supporting hyperplane.
Definition 2.3: A face of a polytope $C(S)$ is any set of the form

$$C(S) \cap h(u, v),$$

where $h(u, v)$ is a supporting hyperplane of $S$ for some $u, v \in \mathbb{R}^p$. Further, for a $p$-dimensional polytope, facets are $(p - 1)$-dimensional, ridges are $(p - 2)$ dimensional, edges are 1-dimensional, and vertices are 0-dimensional.

The above characterization of a vertex as a single point (a 0-dimensional face) of $C(S)$ that lies in the supporting hyperplane, will be used in Section IV to link FastMap pivots to vertices of the convex hull.

III. FASTMAP OVERVIEW

Given the Euclidean distance between any two points (objects) of $S$, $k$ iterations of FastMap produce a $k$-dimensional $(k \leq p)$ representation of $S$. Each iteration selects from $S$ a pair of points, called pivots, that define an axis and computes coordinates of the $S$ points along this axis. The pairwise distances for $S$ can then be updated to reflect a projection of $S$ onto the subspace (a hyperplane passing through the origin) orthogonal to this axis. The next iteration implicitly operates on the projected $S$ in the subspace. However, these projections are accumulated and jointly performed only for the distances that are needed. In this manner, after $k$ iterations, the $S$ points end up with $k$ coordinates giving their $k$-dimensional representation.

To provide details of the FastMap algorithm, we first introduce some notation. Let $(a_i, h_i)$ be the pair of pivot elements from $S$ at iteration $i$. Let $d_i(x, y)$ be the Euclidean distance between points $x$ and $y$ of $S$ after their $i$th projection onto a pivot-defined hyperplane, so that $d_0(x, y)$ is the initial Euclidean distance. Also, let $x_i$ be the $i$th coordinate of $x$ in the resulting $k$-dimensional representation of $x \in S$.

Pivot elements are chosen by the choose-distant-objects heuristic shown in Fig. 1. Initially, $i = 0$. After selecting a pivot pair $(a_i, h_i)$, the $i$th coordinate of each point $x \in S$ is computed as

$$x_i = \frac{d_{i-1}^2(a, x) + d_{i-1}^2(a, b) - d_{i-1}^2(b, x)}{2d_{i-1}(a, b)}.$$ (3)

This projection is based on the law of cosines and current distances from the two pivot points. The distances are updated whenever needed in Choose-distant-objects or in (3). An update for a single iteration is presented in [6] and we extend this in [1] to a combined update

$$d_i^2(x, y) = d_0^2(x, y) - \sum_{j=1}^{i} (x_j - y_j)^2.$$ (4)

Choose-distant-objects ($S, d_i(.)$)

1) Choose an arbitrary object $s \in S$
2) Let $a_{i+1}$ be the $a \in S$ that maximizes $d_i(a, s)$
3) Let $b_{i+1}$ be the $b \in S$ that maximizes $d_i(b, s)$
4) Report $a_{i+1}$ and $b_{i+1}$ as the distant objects.

Fig. 1. Choose-distant-objects heuristic for iteration $i$

IV. FASTMAP AND VERTICES OF THE CONVEX HULL

Here we prove the main result of this paper, namely that all pivot points are selected from vertices of the convex hull of the data set. We do this in two steps. First we show that the Choose-distant-object heuristic pivot pair is a pair of convex hull vertices within the current working subspace. Then we show that if a point is a vertex in a subspace projection, it is also a vertex in the original $p$-dimensional space.

The Choose-distant-objects heuristic first takes an arbitrary point $b \in S$ and finds $a \in S$, the most distant point from $b$. Because $a$ is the most distant point in $S$ from $b$

$$(s - b)^T(s - b) \leq (a - b)^T(a - b), \forall s \in S.$$ (5)

Now, for any point $s \in S$ distinct from $a$, we have

$$0 < (s - a)^T(s - a) = (s - b + b - a)^T(s - b + b - a) = (s - b)^T(s - b) + 2(s - b)^T(b - a) + (b - a)^T(b - a) \leq 2(s - b)^T(b - a) + 2(b - a)^T(b - a) \text{ by (5)}$$

$$= 2(s - b + b - a)^T(b - a)$$

$$= 2(s - a)^T(b - a).$$ (6)

If we add $s = a$ in (6), we have

$$0 \leq (s - a)^T(b - a), \forall s \in S,$$

which defines a supporting half space $H(a, b)$ for all points in $S$. Since $a$ is the only point in the supporting hyperplane $h(a, b)$ of $S$, it must be a single point face of $C(S)$. This, by Definition 2.3, is a vertex of $C(S)$.

Next, the Choose-distant-objects heuristic finds the point in $S$ most distant from $a$. By the same argument this is again a vertex of $C(S)$. We state this as a lemma.

Lemma 4.1: A single application of the Choose-distant-objects heuristic to a set of points $S$ returns a pivot pair of points that are among the vertices of $C(S)$.

After choosing a pair of vertices, FastMap projects the set $S$ into a subspace orthogonal to the vector defined by the pivot pair $(a, b)$ and repeats the Choose-Distant-Objects heuristic in the subspace of dimension $p - 1$. Pivot pairs and projections are computed until suitably many orthogonal vectors are extracted to be used as the principal axes of the lower dimensional representation of $S$. So far, we have shown that a pivot pair is a pair of convex hull vertices within its current working subspace. Are they all also vertices of $C(S)$ in the original space? The answer is yes, subject to a uniqueness caveat requiring that no pair of points (except the current pivot points) get projected onto the same point. Assuming that the points
S are in sufficiently \textit{general position} \cite{14} takes care of this. Because we have a finite set of points, we can perturb them by an arbitrarily small amount to achieve such a general position. We show that a vertex in a subspace projection is a vertex in the original \( p \) dimensional space.

Let \( P_H \) be a symmetric projection matrix into a subspace \( H \subset \mathbb{R}^p \) and let \( S_H = \{ P_Hu : u \in S \} \) be the set of image points of \( S \) in this subspace. We also need to assume that \( S \) are in sufficiently general position so that all vertices of \( C(S_H) \) are projections of distinct points of \( S \).

\textbf{Lemma 4.2}: If \( P_Hs \) is a vertex in the convex hull of \( S_H \) and \( S \) are in general position, then \( s \) is a vertex in the convex hull of \( S \).

\textbf{Proof}: Since \( P_Hs \) is a vertex of \( C(S_H) \), by Definition 2.3

\[ P_Hs = C(S_H) \cap h(u, v), \]

where \( h(u, v) \) is a supporting hyperplane of \( C(S_H) \) for some \( u, v \in H \). Because \( P_Hs \in h(u, v) \), there is a \( u' \in H \) such that \( h(u, v) = h(u', P_Hs) \). Now, \( P_Hs \) is the only point of \( S_H \) that is in the supporting hyperplane, so that

\[ (u' - P_Hs)^T(P_Hx - P_Hs) > 0, \]

for all \( P_Hx \in S_H \) distinct from \( P_Hs \). Because \( S \) are in general position,

\[ (u' - P_Hs)^T(P_Hx - P_Hs) > 0, \forall x \in S \] distinct from \( s \).

Then,

\[ (u' - P_Hs)^T[x - (I - P_H)x - s + (I - P_H)s] > 0 \]

\[ (u' - P_Hs)^T(x - s) - (u' - P_Hs)^T(I - P_H)(x - s) > 0. \]

Since \( P_H(u' - P_Hs) = (u' - P_Hs) \) (because \( u' \in H \)),

\[ (u' - P_Hs)^T(x - s) > 0, \forall x \in S \] distinct from \( s \).

Equality holds for \( x = s \), so it is the unique point on this supporting hyperplane of \( S \) and thus it is a vertex of the convex hull of \( S \). \( \square \)

Letting \( S_V \subseteq S \) be the vertices of \( C(S) \), Lemmas 4.1 and 4.2 lead to the main result:

\textbf{Theorem 4.3}: FastMap pivot pairs are a subset of the vertices of the convex hull of the data. That is,

\[ a_i, b_i \in S_V, \quad i = 1, \ldots, k. \]

\section{V. Implications}

Convex hull computations in statistics are mostly associated with robust multivariate estimation. Loosely, an estimator of some parameter is said to be robust if it performs well even when the assumed model (implicit or explicit) is not satisfied by the data. For example, when estimating a location parameter, an implicit assumption is that the data are generated by one process that has a location. If more than one process generated the data, a robust estimator would still estimate the location of the dominant process rather than some meaningless location between the processes. The median, for example, is a robust estimator of location while the mean is not. A classic reference on robust estimation is \cite{11}.

The concept of \textit{trimming} extremes is often used in reducing dependence on outliers in data \cite{10}. Tukey is attributed with coining the term \textit{peeling} as the multivariate extension of trimming \cite{10}, where one peels off the vertices of the convex hull before using the remaining points for estimating a location parameter. Here, with the aim of robustness, the very points on which FastMap depends are discarded! Clearly, FastMap is very sensitive to outliers in the data.

In situations where the data generation system is known to work smoothly, such as machine generated data, outliers may not be of concern. For example, we have recently found that in analyzing climate simulation and astrophysics simulation data, methods that are sensitive to extremes often produce the most compelling results. Here, the extremes are not outliers and may be of most interest. On the other hand, massive data sets are often the result of a long run with several checkpoint restarts where anomalies may occur. For example, in \cite{4}, instrument generated Atmospheric Radiation Measurement data \cite{2} contains many instrument restarts that appear as zeros in data with high positive values. Although it is easy to discover these, an automated application of FastMap would be driven by the zero coordinate outliers. Clearly, there are situations where an extremes-sensitive method like FastMap is appropriate or even preferable as well as situations where it will fail.

Outlier sensitivity of FastMap is mentioned in \cite{8} and PCA is presented as more robust. Although PCA is less sensitive to outliers than FastMap, it too is not considered a robust technique. A measure of estimator sensitivity to changes in extreme values of data is the notion of \textit{breakdown point} \cite{3}. Loosely speaking, the breakdown point is the smallest proportion of data that needs to be contaminated to make arbitrarily large changes to the estimator. By this definition, the breakdown point of FastMap is \( \frac{1}{n} \), which is asymptotically zero. Principal Components Analysis, the most popular dimension reduction method, also has a breakdown point of \( \frac{1}{n} \). In both cases, taking one point arbitrarily far in some direction will rotate the first axis in that direction. Some robust PCA methods begin by computing a robust covariance matrix estimate then proceeding with standard PCA as usual. The classical example of a high breakdown estimator is the median with a .5 breakdown point. That is, half of the data must be moved to make an arbitrarily large change in the median. A multivariate extension of the median is proposed in \cite{12}. This extension uses the notion of half-space support to define the \textit{depth} of a data point so that, ignoring ties, the point with maximal depth is the multivariate median.

The main lesson from robust statistics is that the most distant points are often not the best choice for defining a projection axis. The key to new fast and robust methods is the \textit{Choose-distant-object} heuristic by something that considers more than just the maximum distance from a point. One should back-off a little from the maximum, while considering the entire distance distribution. This distribution is already available within the \( O(np) \) complexity. A closer examination, even with more complex algorithms such as clustering, of the distance distribution tail can yield much more robust results, still within the \( O(np) \) complexity. In fact, such methods will be more robust than standard PCA. Clearly there
are many directions that this methodology can be taken and undoubtedly many such algorithms will be proposed.

We would like to note another implication on an algorithm, DFastMap [1], that we recently developed for fast dimension reduction across distributed data sets. Our initial insights that lead to DFastMap produced the main idea for the present paper. Formalizing the convex hull connection to FastMap gives an explanation of why an application of DFastMap to distributed data performs as well as the serial FastMap on a centralized data set. The union of local convex hull vertices necessarily includes all convex hull vertices of the centralized data set. This assertion can be proved using arguments similar to those we used in Section IV. DFastMap centralizes the pivots, arguably a very good subset of the local convex hull vertices (see [1] for more details). This provides a key subset of the combined data convex hull vertices so that little information about extremes is lost when compared to centralizing all the data.

Finally, we also mention an implication on complexity of FastMap and convex hull computations. Because all the FastMap projection axes are computed from points in $S_V$, the convex hull vertices are sufficient for all distant point searches. Clearly FastMap could be faster if $S_V$ were available. Erickson [5] reports that finding $S_V$ by the “gift-wrapping” algorithm takes $O(nf)$ time, where $f = |S_V|$ is the number of vertices. Since FastMap completes in $O(np)$ time, this is not helpful as $f > p$ for any non-degenerate data sets.

REFERENCES


