Too many secants: a hierarchical approach to secant-based dimensionality reduction on large data sets

Henry Kvinge, Elin Farnell, Michael Kirby, and Chris Peterson
Department of Mathematics
Colorado State University
Fort Collins, CO 80523-1874

***Abstract*—A fundamental question in many data analysis settings is the problem of discerning the "natural" dimension of a data set. That is, when a data set is drawn from a manifold (possibly with noise), a meaningful aspect of the data is the dimension of that manifold. Various approaches exist for estimating this dimension, such as the method of Secant-Avoidance Projection (SAP). Intuitively, the SAP algorithm seeks to determine a projection which best preserves the lengths of all secants between points in a data set; by applying the algorithm to find the best projections to vector spaces of various dimensions, one may infer the dimension of the manifold of origination. That is, one may learn the dimension at which it is possible to construct a diffeomorphic copy of the data in a lower-dimensional Euclidean space. Using Whitney's embedding theorem, we can relate this information to the natural dimension of the data. A drawback of the SAP algorithm is that a data set with T points has $O(T^2)$ secants, making the computation and storage of all secants infeasible for very large data sets. In this paper, we propose a novel algorithm that generalizes the SAP algorithm with an emphasis on addressing this issue. That is, we propose a hierarchical secant-based dimensionality-reduction method, which can be employed for data sets where explicitly calculating all secants is not feasible.**

***Index Terms*—Secant sets, dimensionality reduction, big data**

I. Introduction

Determining the dimension of a data set is a basic first step toward a meaningful understanding of the data as well as a foundational part of any related data analysis. This is especially true for high-dimensional data sets where calculating the "intrinsic" dimension can point toward huge efficiency gains via dimensionality reduction. The determination of dimension is a central question in the area of geometric data analysis [1] and the related field of manifold learning [2], [3]. This paper presents a subspace-secant algorithm for computing projections of data which preserves both the topological and Hausdorff dimension of a data set. A byproduct of the computation is an estimate of the dimension and an estimate of the smoothness, and hence stability, of the nonlinear mapping that reconstructs the data.

In [4] the authors described the SAP (Secant-Avoidance Projection) algorithm. This algorithm is based in part on the commonsense notion that a good dimensionality-reduction algorithm should strive to be distance preserving. That is, if two data points are initially far apart, then when we map them into a reduced space they should remain far apart. This can be rephrased to say that a dimensionality-reduction algorithm should preserve the lengths of the secants between points in the data set. The SAP algorithm takes as input the secant set S of a data set in $\mathbb{R}^n$ and an integer $0 < k < n$ and produces a matrix $P : \mathbb{R}^n \to \mathbb{R}^k$ corresponding to a projection PP^T maximizing the value $\min_{s \in S} ||PP^T s||_2$. Furthermore, by studying the quality of projections produced by SAP over a range of projection dimensions we can make a reasonable estimate of the dimension of our data.

One limitation of the SAP algorithm is that it does not scale well to large data sets because the number of secants corresponding to a data set of size T is $T(T-1)/2$. The motivation for this paper is to propose a generalization of the SAP algorithm, which we call the HSAP (Hierarchical Secant-Avoidance Projection) algorithm, which addresses this limitation. The underlying idea of the HSAP algorithm is to use the hierarchy of structure present in a data set in order to reduce the number of secants required to obtain a good secant preserving projection. In this case this means first clustering the data set and then either using a linear approximation of each cluster or a sample of secants from each cluster to capture the secant structure at the local level. At the same time we also sample a fairly small number of secants between clusters to capture the spatial relations between clusters. Roughly then, the HSAP algorithm generates a projection that neither maps the points of a single cluster onto each other nor maps one cluster onto another.

The outline of the paper is as follows: in Section II we review some of the mathematical framework that underlies this paper including the geometry of Grassmann manifolds and dimension estimation. In Section III we describe two versions of the HSAP algorithm and remark on various aspects of it. Finally in Section IV we describe the result of running the HSAP algorithm on both a small-dimensional synthetic data set and also a hyperspectral data set.

This paper is based on research partially supported by the National Science Foundation under Grants No. DMS-1513633, DMS-1322508, as well as DARPA awards N66001-17-2-4020 and D17AP00004.

978-1-5386-5989-2/18/$31.00 ©2018 IEEE

II. Background

Finding compact representations for complicated objects, such as data clouds or high-dimensional arrays, has been an indispensable tool for knowledge discovery within massive data sets. For instance, if a cloud of points cluster along a k-dimensional linear space within a vector space, then for many purposes it is natural to represent the cloud with its linear approximation. If a more refined representation is desired then one approach is to consider the linear space that captures a pre-specified percentage of the energy in the cluster together with a sparse sampling of the points within the cluster which captures some essence of the distribution of the data. The *Grassmannian* $Gr(k,n)$ is a manifold whose points parametrize the k-dimensional subspaces of a fixed n-dimensional vector space. Using this manifold, the cluster can be represented by a point on $Gr(k,n)$ together with the sparse sampling representing the distribution. Suppose now that one would like to condense the information within a very large data cloud residing in a high-dimensional vector space. In many applications, such a data cloud can be hierarchically partitioned into a collection of smaller clusters with each representing some feature of interest. By compactly representing the points in each cluster and the relationships between the clusters, one can hope to better understand the data cloud as a whole. In the sections that follow, we utilize Grassmann manifolds as organizing structures to condense and capture much of the information in a very large data cloud and use this compact representation of the data cloud to drive algorithms towards locally optimal, dimensionality-reducing, structure-preserving projections.

An important feature of Grassmannians, in the context of knowledge discovery in data, is that they can be given the structure of a differentiable manifold. One would like to determine the proximity of various points on a Grassmannian and this is typically carried out by first determining principal angles between the corresponding vector spaces. This derives from the fact that every orthogonally invariant metric on a Grassmann manifold can be described in terms of principal angles. Furthermore, principal angles between vector spaces are readily computable through a singular value computation. In order to understand this statement, we first describe principal angles in the context of an optimization procedure.

Consider the subspaces U and V of a vector space $\mathbb{R}^n$ and let $q = \min\{\dim U, \dim V\}$. The principal angles between U and V are the angles $\theta_1, \theta_2, \dots \theta_q \in [0, \frac{\pi}{2}]$ between pairs of principal vectors $\{u_k, v_k\}$ with $u_1, \dots, u_q$ a distinguished orthonormal set of vector in U and $v_1, \dots, v_q$ a distinguished set of orthonormal vectors in V. These vectors are obtained recursively, for each $1 \leq k \leq q$, by defining

$$\cos\theta_k = \max_{u \in U, v \in V} u^T v = u_k^T v_k$$

subject to

- $||u||_2 = ||v||_2 = 1$
- $u^T u_i = 0$ and $v^T v_i = 0$ for $i = 1, 2, \dots, k-1$.

The key point is that any orthogonally invariant measure of similarity between U and V can be determined as a function of the principal angles.

The principal angles and principal vectors between U and V can be determined from orthonormal bases for U and V as follows. Suppose A (respectively B) are matrices whose columns form orthonormal bases for U (respectively V). From the singular value decomposition we have a factorization $A^T B = Y \Sigma Z^T$. If y_i (respectively z_i) denotes the i^{th} column of Y (respectively Z) then the i^{th} singular vector pair can be computed as $u_i = A y_i$ and $v_i = B z_i$. Furthermore, the singular values of $A^T B$ are equal to $\cos\theta_1, \cos\theta_2, \dots, \cos\theta_q$, where the sequence is assumed to be monotonically decreasing. See [5].

A. Dimension Estimation

The estimation of dimension from data has been addressed by numerous authors including, e.g., [6]–[14].

There is a useful theoretical result for characterizing dimension-preserving transformations. It revolves around the definition of a *bi-Lipschitz* function. A function $f(x)$ is said to be *bi-Lipschitz* on X if for all $x, y \in X$ it holds that

$$a\|x - y\|_{\ell_2} \leq \|f(x) - f(y)\|_{\ell_2} \leq b\|x - y\|_{\ell_2}.$$

The constant a restricts pairs of points from collapsing on top of each other while b restricts pairs of points from blowing apart. In the context of projection, as we consider in the algorithm of the next section, we can restrict to the case $b = 1$. A key feature of bi-Lipshitz functions is:

$$\text{if } f : X \to Z \text{ is bi-Lipschitz, then } \dim(X) = \dim(Z)$$

where the dimension can be taken as the topological dimension, or the Hausdorff dimension; see [15] for details. Thus we see a link between dimension preservation and projections that avoid collapsing secants. Projection-based algorithms that maximally avoid decreasing the length of secants are, in some sense, optimally dimension preserving and form the theoretical motivation for the algorithm presented here. An additional argument for this approach, based on invoking Whitney's easy embedding theorem, is made in [16].

III. The algorithm

We begin this section by noting two different methods of representing the secant set of a cluster: either by a linear approximation (Section III-A) or by a sampling of secants (Secant III-B). We then go on to describe the HSAP algorithm proper (Section III-C).

A. Approximation by linear subspaces

Let D be a set of points in $\mathbb{R}^n$. Using a clustering algorithm one can partition D into N disjoint subsets $D_1, D_2, \dots, D_N$ (see Section III-D for a discussion of clustering methods). To each D_j for $1 \leq j \leq N$ we construct a k_j-dimensional linear approximation. That is, to each of these N linear approximations, we associate an $n \times k_j$ matrix $V_j = [v_1^{(j)}, v_2^{(j)}, \dots, v_{k_j}^{(j)}]$ whose columns form an orthonormal basis for the linear

978-1-5386-5989-2/18/$31.00 ©2018 IEEE

approximation subspace. In particular each V_j also approximates the secant set for points in D_j. We suggest mean-centering each cluster D_j and using Principal Component Analysis ([17], [18]) to determine a good V_j, and we note that this approach is appropriate precisely when the cluster and consequently the secant set are well approximated as linear spaces.

In Algorithm 1, we present this version of the HSAP algorithm, which is the more involved of the two. The modifications to Algorithm 1 required for the version described below in Section III-B should be clear from the context.

B. Approximation by sampled secants

In the case where clusters are expected to be highly non-linear in structure, it makes sense to take an approach which takes this into account. Therefore in the second version of our algorithm, instead of approximating our clusters $D_1, \ldots, D_N$ by linear spaces, we instead approximate the secant set of each of these clusters by a subset of its secant set. For cluster D_j let S_j be the corresponding secant subset, and include each S_j in the set of secants $\tilde{S}$ defined below.

C. The HSAP Algorithm

In order to encode the relations between different clusters, we also sample a small selection of points A_j from D_j for $1 \leq j \leq N$. There are different strategies for doing this, e.g. one might collect a random sample from D_j or one might select extremal points of D_j. We calculate all secants between points in A_i and points in A_j for $1 \leq i < j \leq N$. Define $\tilde{S}$ to be the subset of the full secant set that consists of all secants between points in each pair $(A_i, A_j), i \neq j$. Finally, we choose an initial k-dimensional projection with a corresponding $n \times k$ matrix $P^{(0)}$ whose orthonormal columns span the projection subspace. We propose an initialization $P^{(0)}$ defined as the first k columns of Y, where the data matrix has been decomposed via the singular value decomposition as $Y\Sigma Z^T$.

To obtain a matrix $P^{(i+1)}$ such that the projection $P^{(i+1)}(P^{(i+1)})^T$ better preserves secants or their approximations, at the ith iteration we shift our current matrix $P^{(i)}$ toward the secant (or corresponding approximation) which is currently the worst preserved by $P^{(i)}$. We call this vector the "shortest representative vector" and denote it by w_i (note that it can come from either a linear approximation of a secant set or a genuine secant sampled between clusters). We will also use the projection (which we denote by $w_i^{(p)}$) of this shortest representative vector onto the subspace corresponding to $P^{(i)}$. In order to find which representative vector is worst preserved,

- calculate the singular values $\sigma_1^{(j)}, \ldots, \sigma_{k_j}^{(j)}$ of $(P^{(i)})^T V_j$ for each $1 \leq j \leq N$,
- calculate the length $||(P^{(i)})^T s||_{\ell_2}$ for each $s \in \tilde{S}$.

Note that $\sigma_1^{(j)}, \ldots, \sigma_{k_j}^{(j)}$ correspond to the cosine function applied to the principal angles between the subspaces corresponding to $P^{(i)}$ and V_j. This is a natural higher-dimensional generalization of the process of measuring the length of a unit vector projected onto a subspace (this is one sense in which the HSAP algorithm is a generalization of the SAP algorithm). We now calculate the minimum element of the set R defined below and define the shortest representative vector w_i and its projection accordingly:

$$R = \Big(\bigcup_{j=1}^{N} \{\sigma_1^{(j)}, \ldots, \sigma_{k_j}^{(j)}\} \Big) \bigcup \Big(\bigcup_{s \in \tilde{S}} \{||(P^{(i)})^T s||_{\ell_2}\} \Big).$$

- **Case 1:** If the smallest element is $\sigma_{k_j}^{(j)}$, let $y_{k_j}^{(j)}$ and $z_{k_j}^{(j)}$ be the corresponding left and right singular vectors, respectively, in the SVD of $(P^{(i)})^T V_j$. Then $w_i^{(p)} = P^{(i)} y_{k_j}^{(j)}$ and $w_i = V_j z_{k_j}^{(j)}$. Note that we assume that the singular values of $(P^{(i)})^T V_j$ are ordered from largest to smallest as in the standard singular value decomposition. In this case we only have to calculate the last singular value for the comparison step.
- **Case 2:** If $||(P^{(i)})^T s||_{\ell_2}$ is the smallest element, then $w_i = s$ and $w_i^{(p)} = P^{(j)}(P^{(j)})^T s$.

Finally, we construct $P^{(i+1)}$ from $P^{(i)}$ by first finding the column $P_t^{(i)}$ of $P^{(i)}$ such that $|(P_q^{(i)})^T s^*|$ is maximized over all columns $P_q^{(i)}$. Assume that $\max_{P_q^{(i)}} |(P_q^{(i)})^T s^*| > 0$ (we will treat the special case where $||(P^{(i)})^T w_i|| = 0$ below). We then remove the t-th column of $P^{(i)}$, shift all columns with index strictly less than t forward and add $w_i^{(p)}$ as the first column. We run the Gram-Schmidt algorithm on this new matrix to obtain a matrix $\hat{P}^{(i)}$ whose columns are orthonormal. Note that by construction $P^{(i)}$ and $\hat{P}^{(i)}$ project to the same subspace. $P^{(i+1)}$ is then the matrix obtained by replacing the first column $\hat{P}_1^{(1)}$ by the normalization of $(1-\alpha)\hat{P}_1^{(1)} + \alpha(w_i - \hat{P}_1^{(1)})$ where $\alpha \in [0, 1]$ is small.

In the case where $||(P^{(i)})^T w_i|| = 0$, we replace the first column $P_1^{(i)}$ of $P^{(i)}$ with $(1 - \alpha)P_1^{(i)} + \alpha w_i$ and run the Gram-Schmidt algorithm on the resulting matrix. The result is $P^{(i+1)}$.

D. Remarks on the HSAP algorithm

There are a number of decisions which must be made when applying the HSAP algorithm to a data set.

A choice of parameter α must be made, which controls the extent to which the projection shifts at each step. In practice we have found that when the algorithm is run using α values between 0.01 and 0.05, convergence occurs reasonably quickly but still reliably.

The HSAP algorithm takes as input clusters $D_1, D_2, \ldots, D_N$ for a data set D. Thus a fundamental step in the application of the HSAP algorithm is clustering data. In certain examples, one may have knowledge of clusters withing the data a priori; in general, we must expect that it will be necessary to apply an algorithm that attempts to cluster the data in an optimal way. In light of the fact that the HSAP algorithm is designed to succeed in a scenario in which the data set D is very large, we note that there are many relevant clustering algorithms that are designed for the big data setting. For example, in [19], the authors propose extensions to fuzzy and probabilistic clustering for big data settings. Many authors have suggested implementations

978-1-5386-5989-2/18/$31.00 ©2018 IEEE

Algorithm 1 Hierarchical Secant-Avoidance Projection

inputs Given a data set D.
Initialize parameters: ambient dimension n, number of clusters N, max number of steps (Iterations) or alternative stopping criterion, and shift parameter α.
Use a clustering algorithm to find N clusters $D_1, D_2, \ldots, D_N$ that partition D.
Define matrices V_j for $1 \leq j \leq N$, whose orthonormal columns $\left\{v_i^{(j)}\right\}_{i=1}^{k_j}$ form bases for the linear approximations to clusters $D_1, D_2, \ldots, D_N$. Choose small subsets $A_1, A_2, \ldots, A_N$ of $D_1, D_2, \ldots, D_N$, respectively. Choose an initial matrix $P^{(0)}$ in $\mathbb{R}^{n\times k}$. Define $\tilde{S}$ to be the set of secants between all points in each pair of sets (A_i, A_j) with $i \neq j$.
Calculate all secants between points in A_i and A_j for all $1 \leq i < j \leq N$.
for $i \leq$ Iterations **do**
  Calculate the singular values $\sigma_1^{(j)}, \ldots, \sigma_{k_j}^{(j)}$ for $(P^{(i)})^T V_j$ for each $1 \leq j \leq N$.
  Calculate the length of $(P^{(i)})^T s$ for each $s \in \tilde{S}$.
  Choose the smallest value among $\sigma_1^{(j)}, \ldots, \sigma_{k_j}^{(j)}$ for $1 \leq j \leq N$ and $||(P^{(i)})^T s||_{\ell_2}$ for all $s \in \tilde{S}$ in steps 4-5 above.
  Calculate w_i and $w_i^{(p)}$ as in Section III-C.
  if $w_i^{(p)} = 0$ **then**
    Replace the first column $P_1^{(i)}$ of $P^{(i)}$ by $(1-\alpha)P_1^{(i)} + \alpha w_i$, run the Gram-Schmidt algorithm on the resulting matrix and set the result equal to $P^{(i+1)}$.
  else
    Set $t = \arg\max_{1\leq q\leq k} |(P_q^{(i)})^T w_i|$.
    Apply the modified Gram-Schmidt algorithm to $w_i^{(p)}, P_1^{(i)}, \ldots, P_{t-1}^{(i)}, P_{t+1}^{(i)}, \ldots, P_k^{(i)}$ to obtain a new orthonormal projection $\hat{P}^{(i)}$.
    Replace the first column of $\hat{P}^{(i)}$ with the normalization $(1-\alpha)\hat{P}_1^{(i)} + \alpha(w - \hat{P}_1^{(i)})$.
    $i + 1 \leftarrow i$
  end if
end for
return

of k-means clustering that utilize graphics processors for efficiency, e.g. [20]–[25]. See [26], [27] for reviews of clustering methods for big data. We choose to use a k-means algorithm in the work we present in this paper; we leave it to the reader to select an appropriate clustering algorithm for the particular data setting of interest.

There are several options for the initial matrix $P^{(0)}$. An efficient choice would be to define $P^{(0)}$ to be a random matrix with orthonormal columns. In practice, this seems to suffice, though convergence often requires more iterations when compared with our proposed initialization (the truncated PCA basis for the column space of the data matrix).

In Step 6 of the HSAP algorithm, it is necessary to compare the smallest singular values for each V_i against other scalars. We note that it is possible to compute only the smallest singular value as a means of added computational efficiency. There are several articles containing algorithms to do this (e.g. [28]–[30]) and there are implementations in popular programming languages, such as MATLAB® [31], which relies on [32] and [33].

The HSAP algorithm is a polynomial time algorithm. The complexity is dominated by the computation of the projected cluster bases and of the projection of the secants. We note that, while an SVD is often an expensive computation, in the HSAP algorithm it does not dominate because the relevant matrices are comparatively small: the computation of the SVD in Step 4 is $O(\min\left\{k^2 k_j, k k_j^2\right\})$. Meanwhile, the computation of the products $(P^{(i)})^T V_j$ for all $j = 1, \ldots, N$ is $O(knk_jN) \subseteq O(n^3 N)$. The computation of the norm of the shortest projected secant in Step 6 requires computing products $(P^{(i)})^T s$ for all $s \in \tilde{S}$; these products are $O(kn) \subseteq O(n^2)$. Since $|\tilde{S}|$ is $O(N^2(\max|A_i|)^2)$, the computation of all products is $O(n^2 N^2(\max|A_i|)^2)$. Thus, the HSAP algorithm is $O(\max\left\{n^3 N, n^2 N^2(\max|A_i|)^2\right\})$. In practice, we expect that $\max(|A_i|)$ and N will usually be substantially smaller than n.

IV. Examples

A. A synthetic example

We construct a data set D_{syn} in $\mathbb{R}^3$ consisting of the union of 100 points sampled from two different lines and 500 points sampled from a plane. Specifically we sample points from

$$f_1(t) = \left\langle t, -t, 1 \right\rangle$$
$$f_2(t) = \left\langle t, t, 4 \right\rangle$$
$$f_3(t, s) = \left\langle \frac{t}{2} - s, s, t - s - 3 \right\rangle$$

(see Figure 1). Note that in this case D_{syn} is naturally clustered as subspaces of varying dimensions (this is an artificial situation but serves well as a first illustration of the algorithm).

We ran 80 iterations of the linear approximation version of on the HSAP algorithm on D_{syn} to obtain a matrix $P^{(80)}$ that maps from $\mathbb{R}^3$ into $\mathbb{R}^2$. Each sample of points from within a cluster was chosen randomly and had a size of 20 (i.e. $|A_j| = 20$ for $j = 1, 2, 3$). We also set $\alpha = 0.01$.

In Figure 2 we plot the norm of the projection of the shortest representative vector as a function of iteration. As can be seen, the projection improves fairly quickly over the course of approximately 70 iterations but then stalls in what is probably a local minimum.

We see the results of the HSAP projection of D_{syn} into $\mathbb{R}^2$ in Figure 3. Note that the algorithm has successfully projected the natural clusters in the data into distinct locations in $\mathbb{R}^2$ while also preserving the within-cluster spacing to a reasonable extent.

B. The Indian Pines data set

As a real-world example we apply the HSAP algorithm to the Indian Pines hyperspectral data set (some bands covering

978-1-5386-5989-2/18/$31.00 ©2018 IEEE

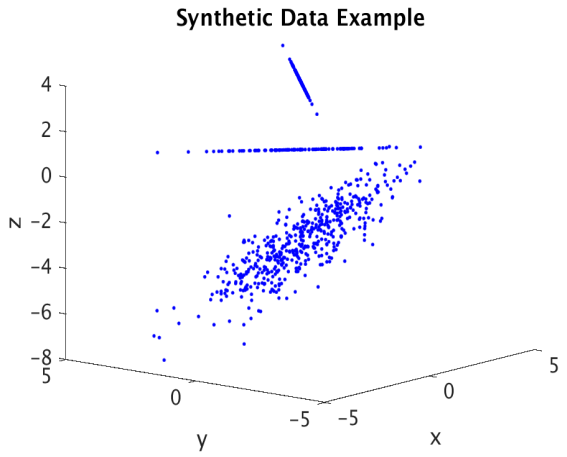

Fig. 1. The synthetic data set D_{syn} in $\mathbb{R}^3$. We construct it to have three natural clusters; the HSAP algorithm should prevent the clusters from collapsing onto each other while simultaneously seeking to preserve the data within each cluster in the projection.

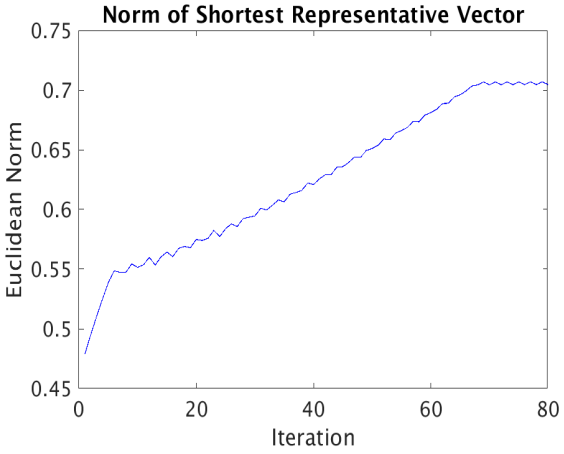

Fig. 2. A plot of the convergence of the linear approximation version of the HSAP algorithm run on the D_{syn} data set. The iteration is given on the x-axis, while the y-axis gives the smallest singular value of $(P^{(i)})^T V_j$ or the smallest representative vector length.

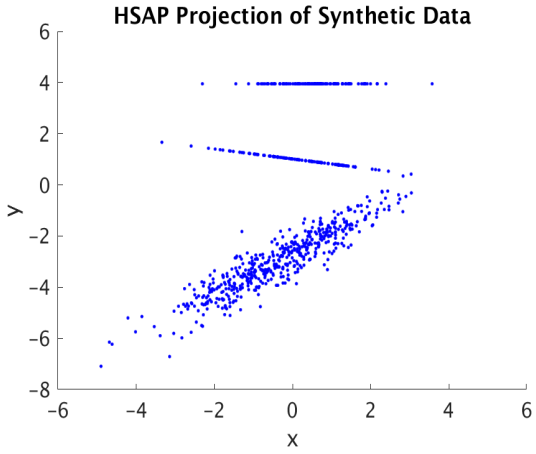

Fig. 3. The projection of the data set D_{syn} using the output $P^{(80)}$ of an application of the linear approximation version of the HSAP algorithm.

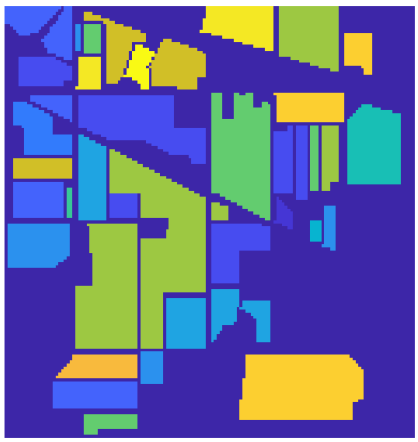

Fig. 4. The ground truth labels for the Indian Pines hyperspectral data set. There are 16 labeled categories, including, e.g. alfalfa, corn, oats, woods, and stone-steel-towers. There is also a 17th category of unclassified pixels.

the region of water absorption are removed) [34]. The data cube is $145 \times 145 \times 200$; that is, there are 200 bands, each with spatial resolution of 145×145. We define a data set $D \subset \mathbb{R}^{200}$ to be the collection of vectors of spectral information taken across all pixel locations. We then have $|D| = 21,025$. There are consequently over 221 million secants for this data set. What is more, one would expect that the Indian Pines data set has a naturally clustered structure, where clusters correspond to materials with different absorbency in the scene. These two qualities make the Indian Pines data an ideal candidate for the HSAP algorithm.

The manually-labeled ground truth available for the Indian Pines data set is displayed in Figure 4. There are 16 labeled categories (e.g. alfalfa, oats, woods, and buildings-grass-trees-drives) and a 17th category of unclassified pixels.

In Figure, 6, we see the result of projecting the Indian Pines data with the result of the HSAP algorithm. In this example, we project into $\mathbb{R}^3$, and we choose to approximate the clusters with linear spaces (see Section III-A). In order to define the input clusters $D_1, D_2, \ldots, D_N$, we apply the k-means clustering algorithm with cosine distance to get approximations of the naturally occurring clusters in the data. Note that cosine distance d is defined to be $d(u, v) = 1 - \cos(\theta)$, where θ is the angle between the two input vectors u and v. See Figure 5 for the visualization of the result of this application of k-means. In Figure 6, the individual clusters are assigned different display colors. Note that the projection appears to do a very good job of preventing these clusters from being collapsed together while also maintaining some spread among the points within clusters.

V. Conclusion

In this paper, we proposed a generalization of the Secant-Avoidance Projection (SAP) algorithm that is particularly

978-1-5386-5989-2/18/$31.00 ©2018 IEEE

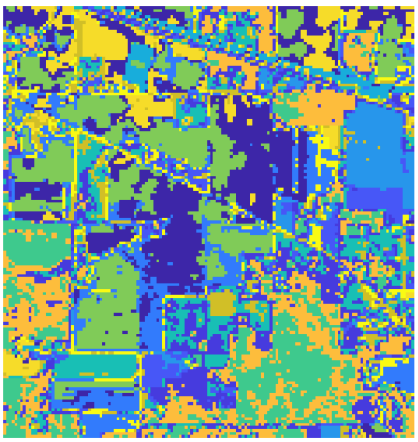

Fig. 5. The 13 clusters determined by an application of the k-means algorithm to the Indian Pines hyperspectral data using cosine distance. We note that this clustering visually appears to have captured some of the relevant structure in the data set as shown in Figure 4.

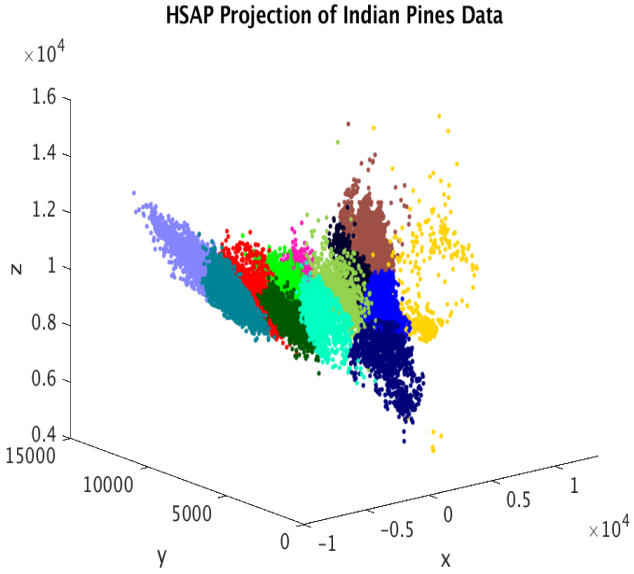

Fig. 6. The projection into $\mathbb{R}^3$ defined by the HSAP algorithm when applied to the 13 data clusters of Indian Pines hyperspectral data shown in Figure 5. Note that the HSAP algorithm appears to have successfully preserved the clusters in the provided projection.

relevant to very large data sets. The Hierarchical Secant-Avoidance Projection (HSAP) algorithm uses a structured approach to selecting appropriate subsets and approximations to the full secant set in order to guide an iterative algorithm. The algorithm returns a projection, which seeks to best preserve the secant set associated to a data set. This projection is useful for both dimensionality reduction and for approximating the dimension of the manifold from which the data was drawn (see [4] for more on this), when such a setting exists. The usefulness and relevance of the algorithm was demonstrated in a synthetic example and in an application to the Indian Pines hyperspectral data set.

Further research can be completed in the following areas.

- Other methods of selecting subsets of secants should be considered.
- We proposed one of many hierarchical structures. There are opportunities to explore alternatives.

References

[1] M. Kirby, *Geometric Data Analysis: An Empirical Approach to Dimensionality Reduction and the Study of Patterns*. Wiley, 2001.

[2] J. B. Tenenbaum, V. d. Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000. [Online]. Available: http://science.sciencemag.org/content/290/5500/2319

[3] S. Roweis and L. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, pp. 2323–2326, 2000.

[4] H. Kvinge, E. Farnell, M. Kirby, and C. Peterson, "A gpu-oriented algorithm design for secant-based dimensionality reduction," in *The 17th IEEE International Symposium on Parallel and Distributed Computing*. IEEE, 2007, to appear.

[5] . Björck and G. H. Golub, "Numerical methods for computing angles between linear subspaces," *Mathematics of computation*, vol. 27, no. 123, pp. 579–594, 1973.

[6] D. S. Broomhead, R. Jones, and G. P. King, "Topological dimension and local coordinates from time series data," *J. Phys. A: Math. Gen*, vol. 20, pp. L563–L569, 1987.

[7] M. Anderle, D. Hundley, and M. Kirby, "The bilipschitz criterion for mapping design in data analysis," *Intelligent Data Analysis*, vol. 6, no. 1, pp. 85–104, 2002.

[8] D. Hundley and M. Kirby, "Estimation of topological dimension," in *Proceedings of the Third SIAM International Conference on Data Mining*, San Fransico, 2001, pp. 194–202.

[9] J. A. Costa and A. O. Hero, "Determining intrinsic dimension and entropy of high-dimensional shape spaces," in *Statistics and Analysis of Shapes*. Springer, 2006, pp. 231–252.

[10] K. Fukunaga and D. R. Olsen, "An algorithm for finding intrinsic dimensionality of data," *IEEE Transactions on Computers*, vol. 100, no. 2, pp. 176–183, 1971.

[11] D. S. Broomhead and M. J. Kirby, "Dimensionality reduction using secant-based projection methods: The induced dynamics in projected systems," *Nonlinear Dynamics*, vol. 41, no. 1, pp. 47–67, Aug 2005. [Online]. Available: https://doi.org/10.1007/s11071-005-2792-1

[12] F. Camastra, "Data dimensionality estimation methods: a survey," *Pattern recognition*, vol. 36, no. 12, pp. 2945–2954, 2003.

[13] J. P. Cunningham and Z. Ghahramani, "Linear dimensionality reduction: Survey, insights, and generalizations," *The Journal of Machine Learning Research*, vol. 16, no. 1, pp. 2859–2900, 2015.

[14] F. Wang and J. Sun, "Survey on distance metric learning and dimensionality reduction in data mining," *Data Mining and Knowledge Discovery*, vol. 29, no. 2, pp. 534–564, 2015.

[15] K. Falconer, *Fractal geometry*, 2nd ed. John Wiley & Sons, Inc., Hoboken, NJ, 2003, mathematical foundations and applications. [Online]. Available: https://doi.org/10.1002/0470013850

[16] D. Broomhead and M. Kirby, "A new approach for dimensionality reduction: Theory and algorithms," *SIAM J. of Applied Mathematics*, vol. 60, no. 6, pp. 2114–2142, 2000.

[17] H. Hotelling, "Analysis of a complex of statistical variables into principal components." *Journal of educational psychology*, vol. 24, no. 6, p. 417, 1933.

[18] I. T. Jolliffe, "Principal component analysis and factor analysis," in *Principal component analysis*. Springer, 1986, pp. 115–128.

[19] R. J. Hathaway and J. C. Bezdek, "Extending fuzzy and probabilistic clustering to very large data sets," *Computational Statistics & Data Analysis*, vol. 51, no. 1, pp. 215–234, 2006.

[20] F. Cao, A. K. Tung, and A. Zhou, "Scalable clustering using graphics processors," in *International Conference on Web-Age Information Management*. Springer, 2006, pp. 372–384.

978-1-5386-5989-2/18/$31.00 ©2018 IEEE

[21] R. Wu, B. Zhang, and M. Hsu, "Clustering billions of data points using GPUs," in *Proceedings of the combined workshops on UnConventional high performance computing workshop plus memory access workshop*. ACM, 2009, pp. 1–6.
[22] Y. Li, K. Zhao, X. Chu, and J. Liu, "Speeding up k-means algorithm by GPUs," *Journal of Computer and System Sciences*, vol. 79, no. 2, pp. 216–229, 2013.
[23] B. Hong-Tao, H. Li-li, O. Dan-tong, L. Zhan-shan, and L. He, "K-means on commodity gpus with cuda," in *Computer Science and Information Engineering, 2009 WRI World Congress on*, vol. 3. IEEE, 2009, pp. 651–655.
[24] S. A. Shalom, M. Dash, and M. Tue, "Efficient k-means clustering using accelerated graphics processors," in *International conference on data warehousing and knowledge discovery*. Springer, 2008, pp. 166–175.
[25] R. Farivar, D. Rebolledo, E. Chan, and R. H. Campbell, "A parallel implementation of k-means clustering on gpus." in *Pdpta*, vol. 13, no. 2, 2008, pp. 212–312.
[26] A. S. Shirkhorshidi, S. Aghabozorgi, T. Y. Wah, and T. Herawan, "Big data clustering: a review," in *International Conference on Computational Science and Its Applications*. Springer, 2014, pp. 707–720.
[27] A. Fahad, N. Alshatri, Z. Tari, A. Alamri, I. Khalil, A. Y. Zomaya, S. Foufou, and A. Bouras, "A survey of clustering algorithms for big data: Taxonomy and empirical analysis," *IEEE transactions on emerging topics in computing*, vol. 2, no. 3, pp. 267–279, 2014.
[28] H. Schwetlick and U. Schnabel, "Iterative computation of the smallest singular value and the corresponding singular vectors of a matrix," *Linear algebra and its applications*, vol. 371, pp. 1–30, 2003.
[29] G. Hongbin, "Irr: An algorithm for computing the smallest singular value of large scale matrices," *International Journal of Computer Mathematics*, vol. 77, no. 1, pp. 89–104, 2001.
[30] N. Lee and A. Cichocki, "Estimating a few extreme singular values and vectors for large-scale matrices in tensor train format," *SIAM Journal on Matrix Analysis and Applications*, vol. 36, no. 3, pp. 994–1014, 2015.
[31] "Matlab and statistics toolbox release," 2018, the MathWorks, Natick, MA, USA.
[32] J. Baglama and L. Reichel, "Augmented implicitly restarted lanczos bidiagonalization methods," *SIAM Journal on Scientific Computing*, vol. 27, no. 1, pp. 19–42, 2005.
[33] R. M. Larsen, "Lanczos bidiagonalization with partial reorthogonalization," *DAIMI Report Series*, vol. 27, no. 537, 1998.
[34] Grupo de Inteligencia Computacional, "Hyperspectral remote sensing scenes," 2014, http://www.ehu.eus/ccwintco/index.php/Hyperspectral_Remote_Sensing_Scenes, Last accessed on 2018-4-30.

978-1-5386-5989-2/18/$31.00 ©2018 IEEE