Generating Hypotheses of Trends in High-Dimensional Data Skeletons

Chandan K. Reddy[*]
Department of Computer Science
Wayne State University

Snehal Pokharkar[†]
Department of Computer Science
Wayne State University

Tin Kam Ho[‡]
Bell Labs
Alcatel-Lucent

Abstract

We seek an information-revealing representation for high-dimensional data distributions that may contain local trends in certain subspaces. Examples are data that have continuous support in simple shapes with identifiable branches. Such data can be represented by a graph that consists of segments of locally fit principal curves or surfaces summarizing each identifiable branch. We describe a new algorithm to find the optimal paths through such a principal graph. The paths are optimal in the sense that they represent the longest smooth trends through the data set, and jointly they cover the data set entirely with minimum overlap. The algorithm is suitable for hypothesizing trends in high-dimensional data, and can assist exploratory data analysis and visualization.

Index Terms: G.4.1 [Mathematics of Computing]: Mathematical Software—Algorithm design and analysis; I.5.3 [Computing Methodologies]: Pattern Recognition—Clustering

1 Introduction

A fundamental concern in data analysis is to find correlations hidden in large, high-dimensional data sets. To analyze correlations the data must be ordered in a certain way in each notion of variability. With such ordering one can proceed to study how variability in one aspect is associated with those in others. The ordering itself would then describe the general course or direction the distributions extended along i.e a certain trend in the data set.

Ordering data points in a high-dimensional space is a highly non-trivial problem. Simply arranging the points along each coordinate does not serve the purpose, because the intrinsic variability in the data may not necessarily align with a specific observation dimension. An example is that in image analysis the samples may be represented as intensity per pixel, and the pixels are raster-scanned as a feature vector, which is then considered as a point in a multi-dimensional space. Yet, the trends of variability in the image, such as the movement of an object, are not necessarily revealed if one follows the intensity changes in a particular pixel. They are observable only as certain associated changes in multiple feature components (data dimensions). Furthermore, the changes may not necessarily involve all of the components. For example, the images may contain background pixels that do not have intensity changes as the object moves. The trends do not necessarily span the entire image collection either, because the changes may occur only in those samples collected in the time window containing the duration of the object's movement.

In this work, we seek a way to uncover trends in datasets that may be local (i.e., not necessarily global), or may be visible in only certain subspaces. We start with an unsupervised learning procedure to obtain a graph structure that describes the local proximity relationship between data points. We then propose an algorithm to navigate the proximity graph, where each navigation path gives a hypothesis of a possible trend in the dataset. An example is given in Fig. 1 for data in a two-dimensional space. The focus of our work is a novel graph-transformation-based framework for identifying a critical set of independent patterns (or trends) with the least amount of overlap. A set of continuous models for the trends is then built by fitting a principal curve using the subset of data points that belong to each of these trends.

It may appear that our procedure would discover a trend on any arbitrary data set even if there is none. In fact, what we are looking for is simply a dominant (or weakly dominant) ordering of the points that may be present in certain subspaces. We conjecture that such an ordering exists in most of the data sets of practical concern. Ideally, this conjecture should be validated by a statistical testing procedure, which is beyond the scope of our current work. In this work, we treat this as an exploratory analysis tool, and caution that one must validate the discovered trend by trying to make sense of the ordering of the associated raw data, or by developing a correlation model with other relevant parameters.

The trends resulting from this procedure can then be analyzed with standard curve properties such as length, curvature, overlap, etc. Their explicit, continuous models can be used to estimate functional relationships between the trends. Analyzing such trends in data can yield more insights for comparing different variability in the data, such as the effects of observational parameters on the outcome, or correlations between different groups of features. We describe the use of this trend finding algorithm in a visualization task [8], which can be a useful starting point for data analysis.

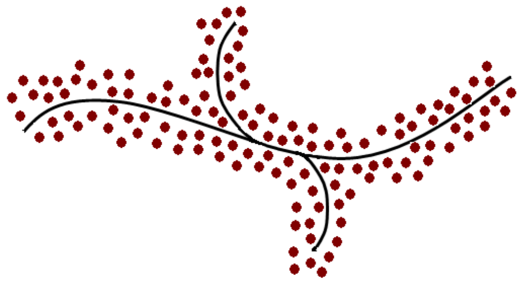
The main contributions in the proposed method are:

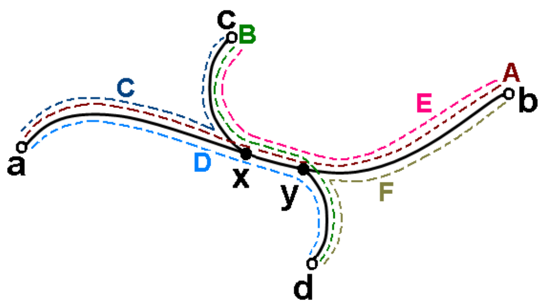
- Dealing with intersecting principal curves by smoothing the data and increasing the curve detection ability of Principal Curves.
- Identifying trends in the data using a continuous representation.
- Obtaining the minimum number of most independent paths that cover the entire dataset.
- Untangling intersecting paths in high-dimensional spaces.
- Visualizing trends that summarize the data in continuous form.
- Hypothesizing correlations amongst various feature trends in the dataset.

The rest of this paper is organized as follows: Section 2 gives information about the relevant background on the various concepts used in the paper. Section 3 describes the problem formulation and explains the key concepts needed to comprehend our algorithm. Section 4 provides the implementation details. Section 5 gives the experimental results of the proposed algorithm on various synthetic and real-world datasets. Finally, Section 6 concludes our discussion and gives the future research directions.

[*]e-mail:reddy@cs.wayne.edu
[†]e-mail:snehalp@wayne.edu
[‡]e-mail:tkh@research.bell-labs.com

IEEE Symposium on Visual Analytics Science and Technology
October 21 - 23, Columbus, Ohio, USA


(a) Original data with principal curves



(b) Possible Paths in the principal curves

Figure 1: Sample data and the possible trends in the data (a) The original data is being fitted using principal curves which go through the "medial-axis" of the data (b) With four different end nodes (a,b,c and d) and two intersection nodes (x,y). There are six possible end-to-end paths (A,B,C,D,E and F). A = (a,x,y,b), B = (c,x,y,d), C=(a,x,c), D=(a,x,y,d), E=(c,x,y,b), F=(d,y,b).

2 Inference of Proximity Structures and Hypothesizing Trends in Data

Fitting continuous models to data is a much explored area in statistics. A popular choice is a spline [7], which is a special function defined by piecewise polynomial (parametric) curves [3], and is often used in applications requiring data interpolation and smoothing of multi-dimensional data. Spline functions for interpolation are normally determined as the minimizers of suitable measures of roughness subject to the interpolation constraints. Smoothing splines may be viewed as generalizations of interpolation splines where the functions are determined to minimize a weighted combination of the average squared approximation error over observed data and the roughness measure. For a number of meaningful definitions of the roughness measure, the spline functions are found to be finite dimensional in nature, which is the primary reason for their utility in computation and representation.

Principal curve [6] fitting is another popular technique that has been successfully used for various applications in pattern recognition and machine learning [4, 9]. They give a representation for low-dimensional manifolds in the data that can be exploited further to form the trends. However, there is a difficulty in applying these models to a new dataset without prior knowledge about the embedded data geometry. Because not all the data points may participate in each possible trend, forcing a fit using all the points may degrade the overall goodness of the fit and damage the models' validity. An example case is that if there are two separate trends that are intersecting, a single application of the principal curve algorithm may not be able to directly ascribe the precise number of trends and then find them. One way to avoid this problem is to build the models in small neighborhoods, and then seek to connect and extend the local models.

To assign points to local groups in close proximity, one may employ various clustering algorithms. Popular choices include the k-means procedure and Gaussian mixture estimation [10]. However, studies on clustering often stop at assigning the data points to clusters. We argue that an important next step is to understand the relationship at the next level, i.e., the relative positioning and potential relationships between the clusters. Therefore, we introduce the cluster relationship graph, as a representation of the proximity structure in the data. A useful choice for this is a minimum spanning tree that connects the cluster centroids. We then seek meaningful ways to traverse this graph that can best reveal underlying trends. Data in the clusters that are considered to be in a trend are then taken to fit a continuous model.

Additional procedures that can be of use in this context include data shrinking [12] methods. These are analogous to thinning and skeletonization in a 2D context that are popular in the computer vision community and have been applied to problems like character recognition [9]. These techniques can reduce the noise and remove any outliers that usually do not contribute to the main trend, and hence avoid the interference of these points to model fitting.

A note of caution related to dimensionality reduction. Dimensionality reduction is an active area of research in exploratory analysis with high-dimensional data [1]. Proper methods can preserve essential proximity structure in the data. But simplistic schemes do not always yield desirable results. For example, many attempts for visualization of high-dimensional data involve systematically presenting different low-dimensional projections of the data onto a restricted number of coordinates. We caution that such operations may not allow a good view into the true variability in the data. An example is given in Fig. 2 shows a spiral in 3-D space along with the data generating curve. Two projections of the data (onto x-y plane and onto x-z plane) are also shown. It is almost impossible to obtain the original structure of the data and the parametric form of the data generating curve from these two projections. This demonstrates the significance of fitting the trend models in the original dataset and not in the projected space, unless it can be assured that the proximity structure is preserved in the projection.

In the discussion that follows, we assume that the k-means procedure is used for clustering. The k-means procedure is known to have problems, such as dependence on initialization and employing a strong assumption on the cluster shapes. For only some of these problems there are remedies. However, we find the simplicity of k-means attractive as a starting procedure in exploratory analysis. Furthermore, in our context it serves mainly as a data compression step. As long as a large enough k is used, the shape assumption has only a local effect. The global shape of the manifold is preserved in the resulting *data skeleton*, i.e., the *Principal Tree* that we will describe below. Nevertheless, we expect that our method can work with another more robust clustering procedure, or with the assistance of procedures that give an estimation of the intrinsic dimensionality of the data.

3 Optimal Traversal of Principal Graph for Generating Trend Hypotheses

In this section, we will mathematically formulate the task of generating trend hypotheses as a graph traversal problem. We first give definitions for the notion of *trends* and a number of graph-theoretic entities needed in the description of the algorithm. We then give the problem statement. Table 1 gives the notations followed in this paper.

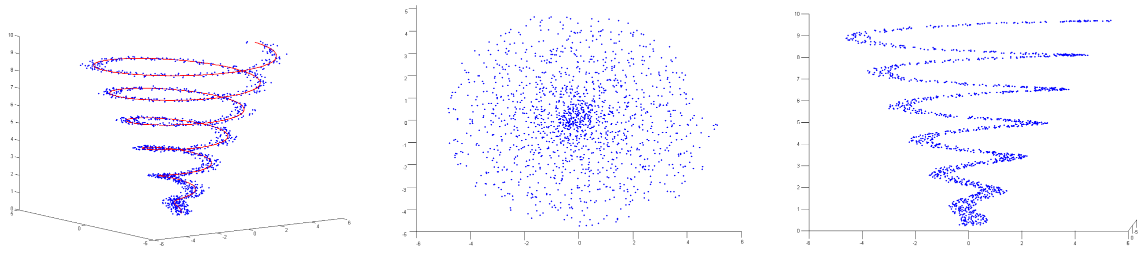

(a) Original data in 3-D (b) Projection of the data onto x-y plane (c) Projection of the data onto x-z plane

Figure 2: The original spiral data (a) the generating curve is represented by a solid line. (b) projection of the data onto a 2D x-y plane. (c) Projection of the data onto a 2D x-z plane.

Table 1: Notations used in this paper.

Notation	Description
$\mathscr{X}$	Input dataset
$\mathscr{D}$	Noise-free dataset
k	Number of clusters
C_i	Cluster centroids
σ	Set of $C_i...C_j$
en_i	End Nodes of the MST
ρ	Set of en_i
in_i	Internal Nodes of the MST
μ	Set of in_i
is_i	Intersection Nodes of the MST
γ	Set of in_i
L_{ij}	Set of edges connecting the Centroids C_i and C_j
G_p	Principal Tree = $(\{C_i\},\{L_{ij}\})$; MST connecting C_is
P_I	End-To-End Paths of G_p = $(en_i,\ldots,in_*,\ldots,en_j)$
A	Adjacency matrix for MST
l_I	Length of P_I
O_{IJ}	Overlap between P_I and P_J = length($P_I \cap P_J$)
ς_I	Curvature factor of paths
w_o	Weight for overlap
w_c	Weight for curvature
w_l	Weight for length
G_t	Traversal Graph =$(\{P_I\},\{O_{IJ}\})$
Φ_I	Weight at the Vertex P_I = l_I
η_{IJ}	Edge Weight for O_{IJ}
τ	Set of Independent P_Is
Γ_{IJ}	Input weight to *Find_Ind_Path* algorithm
Υ	Trends

3.1 The Notion of Trends

Trends (Υ) are the minimal set of most independent sequence of patterns over the feature space containing the dataset. A pattern is a continuous region in a feature space that is filled with data points to a pre-designated density. The degree of independence of the patterns is inversely proportional to the length of overlap between the pair of patterns. The key terms used in the paper are now defined here -

Definition 1 ***Principal Tree*** *- is a graph $G_p = (\{C_i\}, \{L_{ij}\})$ $(i, j = 1,\ldots,k)$ where the vertex set $\{C_i\}$ is the set of k Cluster Centroids obtained using a k-means procedure, and the set of edges $\{L_{ij}\}$ connecting centroids C_i and C_j is the set of links that jointly form a Minimum Spanning Tree (MST) over the vertex set $\{C_i\}$.*

Definition 2 ***End Nodes*** *(en_i) - are those C_i with degree 1 in G_p.*

Definition 3 ***Internal Nodes*** *(in_i) - are those C_i with degree ≥ 2 in G_p.*

Definition 4 ***End-To-End Paths*** *(P_{ij}) - are those paths in G_p that contain exactly two End Nodes en_i and en_j and the Internal Nodes between them. They are ordered sequences of vertices that are of the form $(en_i, in_1, in_2, \ldots, in_m, en_j)$ (if there are m Internal Nodes). There are $\binom{t}{2}$ paths if there are t End Nodes and all nodes appear just once in each P_{ij}.*

Definition 5 ***Intersection Nodes*** *(is_i) - are those Internal Nodes in_i that are on two or more End-To-End Paths.*

Definition 6 ***Overlap****($O_{ij,i'j'}$) - For two End-to-End Paths P_{ij} and $P_{i'j'}$, the overlap is the set $P_{ij} \cap P_{i'j'}$.*

Definition 7 ***Traversal Graph*** *- is a graph G_t =$(\{P_I\},\{O_{IJ}\})$ $(I,J = 1,\ldots,|P_{ij}|)$, where $\{P_I\}$ is a set of End-To-End Paths P_{ij}s and $\{O_{IJ}\}$ is a set of edges representing the overlap between P_I and P_J. The Vertex weight Φ_I associated with each P_I is a measure of the length of P_I and the Edge weight η_{IJ} associated with each O_{IJ} is a measure of the amount of overlap between P_I and P_J.*

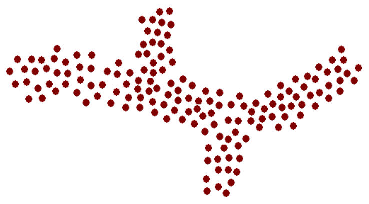
We illustrate these with an example in Fig. 3(a) that contains a two-dimensional dataset. Fig. 3(b) presents the (centroid) C_is in the dataset, one for each cluster identified. C_is are the single point representations of clusters identified by the k-means procedure. Fig. 3(c) shows the Principal Tree with the C_is and their corresponding L_{ij}s. In Fig. 4(a) the end nodes are a,b,c,d as they are incident to only one (link) L_{ij} in the G_p, the internal nodes are w,x,y,z, the intersection nodes are x and y, the path D is a path connecting a,w,x,y,d. The overlap between paths D and E is (x,y). Fig. 4(b) indicates the length weights for each segment of the paths. Every path in Fig. 4(b) is represented as a node in the Traversal Graph given by Fig. 4(c). The overlap between each path is the edge weight in the graph, as shown.

One can see that each possible End-To-End path in the original feature space will be represented as a node in the Traversal Graph. There are a total of $\binom{t}{2}$ nodes in G_t for t End Nodes in the Principal tree. In this paper, we propose an algorithm to extract the *most informative* End-To-End paths that cover the entire data set. To this end, we will now formalize the notion of the most informative End-To-End paths, and define the optimal graph traversal problem that finds the minimal set of such informative paths contained in the data set.

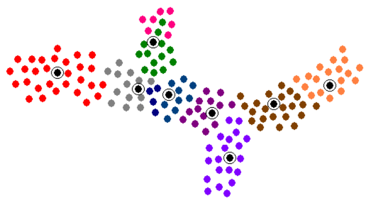
3.2 The Optimal Traversal Problem

Given a Traversal Graph G_t =(P_I, O_{IJ}), a set of weights $\{\Phi_I\}$, $\{\eta_{IJ}\}$, find the minimal set of P_Is such that:
1) the P_Is in the set have the smallest amount of overlap with each other;
2) the P_Is cover all the C_is in the Principal Graph G_p.

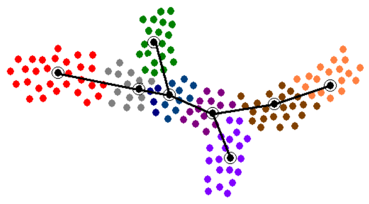
To evaluate the pairs of P_Is and obtain the minimal set, a measure Γ given by Eq.(2) is introduced. Kruskal's algorithm [13] is



(a) Sample Data

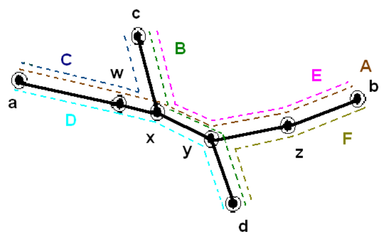


(b) Various Clusters and their Centroids

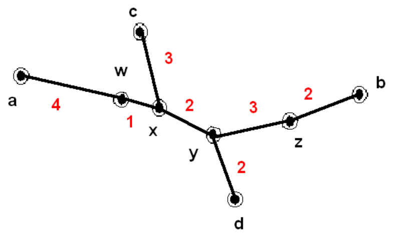


(c) Minimum Spanning Tree on data

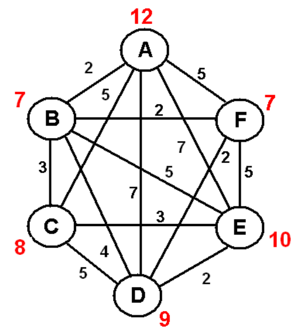
Figure 3: The original data , the cluster identification and the Minimum Spanning Tree of the Cluster Centroids.



(a) Possible End-To-End Paths from the MST skeleton.



(b) Weights for each segments of the End-To-End Paths



(c) Traversal Graph of the End-To-End Paths

Figure 4: Graph Transformation: Vertices are End-To-End Paths with weights as the lengths of the End-To-End Paths. Edges are the overlaps between the End-To-End Paths.

modified and used here. On the identification of the set of optimal End-To-End Paths, a principal curve is fitted to the data points belonging to each P_I to obtain the Υ in the dataset.

3.3 Principal Curves

Definition 8 *(**Principal Curves**) are defined as "self-consistent" smooth curves which pass through the "middle" of a d-dimensional probability distribution or data cloud [6].*

Principal curves are non linear summarizations of multidimensional data points represented by a smooth, one dimensional curve. The curve passes through the densest regions of the dataset, or the 'middle' of the dataset, taking the shape according to the distributions of the dataset.

Let X be a random vector in $\Re^d$ and assume that n samples of X are available. A principal curve $f \subset \Re^d$ is a smooth (C^∞) unit-speed curve explicitly ordered by $\lambda \in \Lambda \subset \Re^1$, that passes through the middle of the d-dimensional data described by the probability distribution of X.

$$f(\lambda) = \left[f_{(1)}(\lambda)...f_{(d)}(\lambda)\right]^T = E[X|\lambda_f(X) = \lambda] \qquad (1)$$

where, the projection index $\lambda_f(x)$ of x is the value of λ for which *f(x)* is closest to x. If a number of such points are present, the point with the largest value of λ is chosen.

3.4 Algorithm

We will now propose the following algorithm to generate trend hypotheses in a dataset:

1. Noise Elimination - implementing a density based noise removal algorithm to reduce the noise and outliers from the dataset.
2. Clustering and Computing Centroids - implementing the k-means algorithm for clustering.
3. Principal Tree Generation - implementing Kruskal's Algorithm to generate an MST connecting the cluster centroids.
4. Identifying Independent Paths - implementing *Find_Ind_Path* algorithm
5. Modeling and Visualizing Trends - fitting principal curves to each identified paths; isolating and ordering data points along each trend for inspection.

4 Implementation Details

The top level algorithm is given as Algorithm 1. In this section, we will elaborate on each of the five main steps in the proposed algorithm.

Algorithm 1 *Finding_Trends*

```
1: Input: Dataset $\mathscr{X}$
2: Output: Trends $\Upsilon$
3: Algorithm:
4: $\mathscr{D} \leftarrow Noise\_Removal(\mathscr{X}, Eps, M)$
5: $\sigma \leftarrow k\text{-}means(\mathscr{D})$
6: $A \leftarrow Kruskal(\sigma)$
7: $\rho, \mu, \gamma, Pends_{i,j} \leftarrow Get\_Nodes\ (A)$
8: $P_I, l_I, o_{I,J}, \varsigma_I \leftarrow Path\_Values\,(Pends, A)$
9: $\tau \leftarrow Find\_Ind\_Path\,(PV, w_o, w_c, w_l)$
10: $\Upsilon \leftarrow P\text{-}Curve(\tau)$
```

Noise Elimination - Most real world datasets contain noise and outliers. Standard clustering algorithms like k-means, Hierarchical, etc., are sensitive to these factors. The density-based algorithm (*Noise_Removal*) is used as a data preprocessing step to eliminate the noise and outliers. Outliers are points in the dataset which are characteristically low in density and can hence be identified using density based approaches. Two parameters are used to determine whether a point is in a sparse region: a neighborhood radius *Eps* and a count *M* of neighboring data points. Every data point which is not within the neighborhood radius *Eps* of at least *M* other data points is classified as a noise point. All such points are eliminated to obtain the output $\mathscr{D}$. Algorithm 2 gives the pseudocode.

Algorithm 2 *Noise_Removal*

```
1:  Input: Dataset $\mathscr{X}$, Threshold value Eps, Min count M
2:  Output: Noise-free dataset $\mathscr{D}$
3:  Pseudocode:
4:  $\mathscr{D} \leftarrow \mathscr{X}$
5:  for i$\leftarrow$ 1 : size($\mathscr{X}$) do
6:      Ind = $\emptyset$
7:      for r $\leftarrow$ 1 : size($\mathscr{X}$), r$\neq$i do
8:          Dist$\leftarrow$ dist[$\mathscr{X}$(i),$\mathscr{X}$(r)]
9:          Ind = [Ind ; (Dist $\leq$ Eps ? $\mathscr{X}$(r) : $\emptyset$)
10:     end for
11:     if size(Ind) $\leq$ M then
12:         $\mathscr{D} \leftarrow \mathscr{D}$ - $\mathscr{X}$(i)
13:     end if
14: end for
```

Computing Centroids - The k-means partitional clustering methodology is chosen here for its better run time efficiency. For a given input of dataset $\mathscr{D}$ and a pre-designated number k, the algorithm partitions the dataset into k clusters as the output. We represent the clusters by their centroid C_i. Membership for each cluster is determined by maximizing the inter-cluster distances and minimizing the intra-cluster distances between feature vectors (the sum of squared error (SSE)).

Principal Tree (MST) Generation - The main purpose of this step is to compute a compact representation of the relationship between the clusters. Here we chose the representation to be the minimum spanning tree connecting the cluster centroids. The connection weights to be minimized are the Euclidean distance between the centroids. To compute the MST, we use Kruskal's algorithm which is an efficient greedy algorithm that runs in O($n$log$n$) time. The output is the Adjacency matrix(A) for the MST or the Principal Tree. All the permutations of pairs of en_{ids} are computed in the matrix *Pends* by the function *Get_Nodes*, *Pends* = [en_{id1},en_{id2};en_{id1},en_{id3};...].

Identifying Independent Paths - This is the core component of our algorithm. The Principal Tree is the skeleton from which the End-To-End Paths P_Is are identified. To obtain the mappings of the P_Is, their l_Is, the pairwise O_{IJ} and the curvature ς_I, the algorithm *Path_Value* (incorporating Dijkstra's Algorithm [13]) is implemented on the input *Pends* and *A* . Some of the P_Is are completely independent of each other while some intersect with others. Some others overlap. The End-To-End Paths are transformed to a G_t with these considerations.

The curvature factor of a given path is a measure of the cumulative cosine function values at the neighborhood of the intersection points. See Fig. 5 where path P1 is a curved path compared to path P2 which is straight. The curvature value for a path is a measure of the summation of the cosine values of the angles made by the line segments at the intersection points. In order to compensate for any uncharacteristic deviations in the location of the centroids, we take the cosine measure of angles between two segments on either side of the intersection point. Here, P1 and P2 intersect at point d.

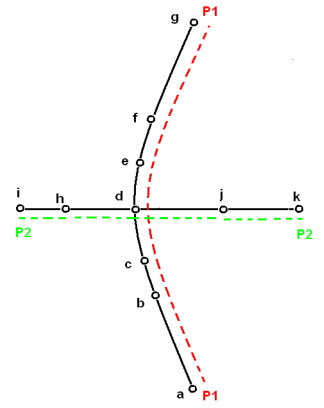


Figure 5: Curvature of paths

Thus, curvature(P1) = cos(bcd)+ cos(cde)+ cos(def) = 0.75. Similarly, curvature(P2) = cos(ihd)+ cos(hdj)+ cos(djk) = 0. Path P1 has a higher curvature value as it has more curvature compared to the straightness of path P2. To obtain the set of P_Is for the Υ, the search is performed for the set of pair-wise combination of P_Is which has in them all the C_is covered and are most independent. The main algorithm here is *Find_Ind_Path* which selects the P_Is using a pair-wise weight Γ. All the P_I,P_J pairs and the corresponding Γ_{IJ} between them are identified as the Path matrix *PV* =[P_I,P_J,Γ_{IJ}]. This is the input to this algorithm along with the normalized weight parameters w_o, w_c and w_l which sum to unity (w_o + w_c + w_l = 1). The weight parameters control the relative importance of overlap, curvature, and length in the selection of the final paths. These are user input parameters whose value depends on the nature of trends desired and the type of dataset. Ind is the set of Indicator variables for C_Is. For every P_I selected, the C_Is $\in$ P_Is are eliminated from Ind until Ind = ϕ.

Algorithm 3 *Find_Ind_Path*

```
1:  Input: Path Matrix PV, Weights $w_o$, $w_c$ and $w_l$
2:  Output: Set of Independent Paths $\tau$
3:  Pseudocode:
4:  sort (PV, $\Gamma$)
5:  Ind = $\sigma$ // Set of Indicator variables for $C_{is}$
6:  for i=1:size(PV) AND Ind $\neq \phi$ do
7:      $\tau$,Ind $\leftarrow$ Find_Path ($P_I$, $\tau$, Ind)
8:      $\tau$,Ind $\leftarrow$ Find_Path ($P_J$, $\tau$, Ind)
9:  end for
10: return ($\tau$)
```

The main goal of computing the Traversal Graph (G_t) is to obtain the most independent set of paths (τ_i) that will cover all the data points. Our goal is to identify the minimal set of paths that cover all the cluster centroids and have the minimum edge weight (ensuring minimum overlap) between them. The algorithm *Find_Ind_Path* generates this minimal set τ. This problem of finding the optimal set of paths is formulated as a min-max optimization problem where the aim is to maximize the vertex weights (hence the inclusion of maximum number of Cluster Centroids) and minimizing the edge weights (thus minimizing the overlap). For a given set of vertices (P_I) and (P_J), the input weight (Γ_{IJ})to the *Find_Ind_Path* algorithm is formulated by the following objective function:

$$\Gamma_{IJ} = w_o * \eta_{IJ} + w_c * (\varsigma_I + \varsigma_J) - (w_l * (\Phi_I + \Phi_J)) \qquad (2)$$

The overlap parameter is ($w_o * \eta_{IJ}$) is minimized here by subtracting the parameter for length (($w_l * (\Phi_I + \Phi_J)$)). Parameters for

Algorithm 4 *Find_Path*

```
1: Input: Vertex $P_I$, set of independent paths $\tau$, indicator matrix for $C_i$s Ind
2: Output: $\tau$, Ind
3: Pseudocode:
4: $P_i \leftarrow Get\_Centroids(P_I)$
5: for j=1:size($P_i$) do
6:     for k=1:size(Ind) do
7:         if $P_i$(j) =Ind(k) then
8:             Ind(k)←0
9:             $s \leftarrow size(\tau)$
10:            $\tau(s+1) \leftarrow P_i$
11:        end if
12:    end for
13: end for
14: return $(\tau, Ind)$
```

curvature ($w_c * (\varsigma_I + \varsigma_J)$) is added as well and the weights w_o,w_l and w_c determine the importance of each parameter in the 'trend' selection process. These weights are user input values and can be varied depending upon the type of trends desired and the nature of the dataset. For example, w_l can be dominant in value amongst the weights if lengthy trends are desired. The End-to-End Paths corresponding to this minimal set of nodes are the trends in the data. As mentioned earlier, all the P_I, P_J pairs and the corresponding Γ_{IJ} between them are described by the Path matrix PV=[P_I,P_J,Γ_{IJ}]. This and the weights w_o,w_c,w_l, are the input to the algorithm whose output is τ.

Algorithm 3 gives the implementation of *Find_Ind_Path*. All pairs (P_I,P_J) are ranked in ascending order as per their Γ. The function *Get_Centroids* obtains the corresponding set of nodes for each P_I. The ranked pairs of paths are checked for the C_is that they include. Those C_is are eliminated from the *Ind* set which is a set of Indicator variables for C_is. The pairs of paths are sequentially checked for any new C_i that they may add and the process terminates once all C_is are eliminated, that is, $(C_i \in \tau) \cap \sigma = \phi$. This task is performed by the function *Find_Path* (See Algorithm 4).

Visualizing Trends - Each P_I includes a sequence of C_is that represent the clusters of the data. Hence each P_I is associated with a set of data points included in those clusters. These data points are identified and a principal curve can be fitted on this subset of the dataset [9]. This is the representation of a *trend* in that data set.

A Note about the Parameters - The procedure as described contains several parameters that require user input. Thus it is not a fully automated procedure yet. Until a systematic way to determine the values of the parameters can be obtained, the procedure is best considered as a tool for exploratory analysis. For example, it can be embedded into a data visualization tool (such as **M**irage[8]), so that a user can iterate on trials of different parameter values.

5 Experimental Results

All programs were written in MATLAB Version 6.5 and run on pentium Dual Core 2.8 GHz machines. Experiments were performed using both synthetic and high-dimensional real-world datasets.

5.1 Synthetic Data sets

Our algorithm was tested successfully on various synthetic data sets that inherently contain *trends*. Several data sets were created with intersecting and overlapping paths in them. Our algorithm identified the data points along these paths and was able to separate out different trends in these data sets. Fig. 6(a)-(d) shows one such synthetic data set where a *sine* wave intersects with a *cos* wave in a four-dimensional space. The data set was created by pseudo-randomly generating data points using these intersecting curves. The *k*-means clustering algorithm identified 18 Cluster centroids which were then used to build the Principal Tree using the Kruskal's algorithm. The weight parameters used in our objective function (see Eq. 2) were: w_l = 0.4, w_o = 0.4 and w_c = 0.2. There were six nodes in the Traversal Graph (G_t) using all the six possible paths constructed from the four end nodes of the MST. For this dataset, our algorithm was able to identify two independent *trends*, as shown in Fig. 6(f). Finally, principal curves were fitted to the data points belonging to each trend individually to obtain a continuous model for the trend.

5.2 Real-world Data sets

Real-world data sets were obtained from image sequences in the CMU Vision and Autonomous Systems Center's Image Database[1]. The motion images were of particular importance because they have numerous motion sequences represented by frames numbered in order of the motion's progression. The movement in the images generated by either the change in the camera placement or its angle, or the actual physical movement of the subject provided a high-dimensional data set with a definite ordered change in the data values as the motion sequence progressed. The entirety of the dataset contains all the images came from the same subject. The feature vectors obtained from the original images were of very high dimensions. A lower number of dimensions were obtained by rescaling the images. Table 2 provides details about various data sets used in our experiments. Most of these motion data sets inherently contain some form of a '*trend*' in a very high-dimensional feature space representing the intensities at the pixels.

In order to obtain data sets that are more suitable for testing our algorithm, we increased the complexity of these sequences by adding more image sequences which were obtained by shifting images at a particular location. In other words, we generated more trends in the data by obtaining sequences using simulated camera panning that provided a new image sequence. The image at a certain location in the original sequence was used as the basis for subsequently generating more sequences by shifting the axis, thus generating either overlapping or intersecting image sequences of transfigured or shifted images. Fig. 7 shows the image data sets containing intersecting and overlapping image sequences overlaid on the original image sequence.

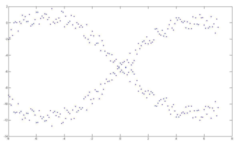
Artichoke data set contains 100 images of a scene which has a toy dog, a cup and a road sign. By shifting the image 50 of the original sequence, an additional set of 121 images were obtained corresponding to a new intersecting trend in the data. Hence, there are a total of 221 images, each of size 60x64, containing trends that intersect at image number 50. This is shown in Fig 7(a) using the images in each sequence/trend.

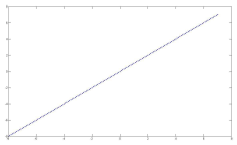
Hand data set contains 481 images of a hand holding a rice bowl. We included only the first 200 images in our original sequence. We obtained overlapping sequences by shifting images at two different locations (100 and 150) in the sequence. A total of 120 (60 + 60) images were added to the original sequence and hence a set of overlapping trends was obtained. The image size is 60x64 and the sequences overlap between images 100 and 150, as shown in Fig 7(b).

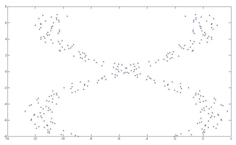
House data set contains a total of 111 images of a toy house. We obtained a set of overlapping sequences by shifting images 25 and 75. At each location, 50 images were generated. The image size is 72x48 and the sequences overlap between images 25 and 75.

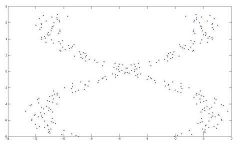
SRI data set contains 125 images of a lab scene. A sequence of 49 images were obtained from Image 60 using a similar transformation used in the Artichoke dataset. The image size is 60x64 and the data set contains two trends intersecting at image 60.

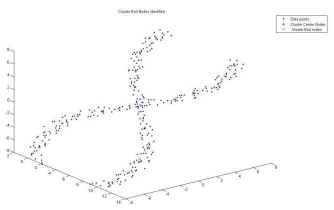
[1]http://vasc.ri.cmu.edu/idb/html/motion/


(a) Dimension 1 Vs 2

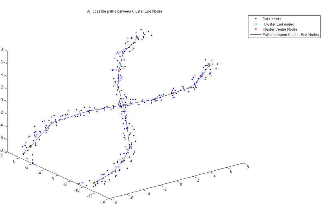

(b) Dimension 1 Vs 3


(c) Dimension 2 Vs 3

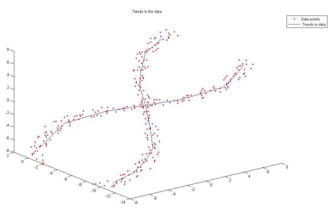

(d) Dimension 2 Vs 4



(e) Cluster Centroids

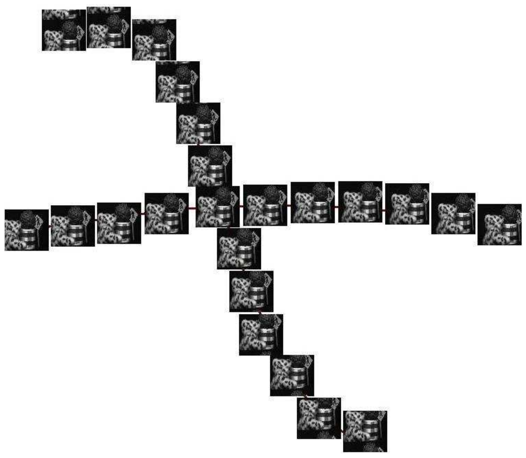


(f) Principal Tree



(g) Two Trends

Figure 6: Example of synthetic data set used to test our algorithm. The first row shows various views of the data across different dimensions. The second row shows the results of various steps in our algorithm. (e) Original data and the cluster centroids obtained using the *k*-means procedure. (f) Principal Tree which is the MST of the cluster centroids. (g) Two most independent End-to-End Paths identified by our algorithm.


(a) Intersecting Trends

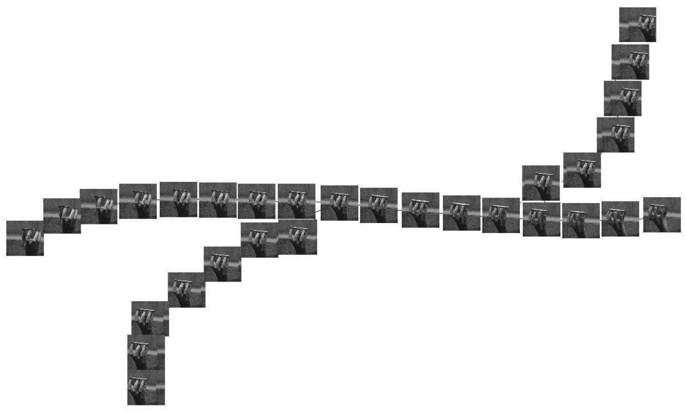

(b) Overlapping Trends

Figure 7: Examples of Intersecting and Overlapping trends in Image datasets. Additional image sequences are obtained by simulating the camera panning which is done by shifting the coordinates. (a) Intersecting trends in the *Artichoke* data set. (b) Overlapping trends containing two intersecting nodes in the *Hands* image sequence data set.

Table 2: Description of various real-world image sequence data sets used in our experiments.

Dataset	Original Size (No. of Images)	Image Width (Pixels)	Image Height (Pixels)	No. of Dimensions	Nature of trends
Artichoke	100	60	64	3840	Intersecting
Hand	200	60	64	3840	Significant overlapping
House	111	72	48	3456	Significant overlapping
SRI	125	60	64	3840	Intersecting

Table 3: Results of our algorithm on real-world data sets.

Dataset	Augmented Data Size (with trends) No. of Images	No. of Clusters	Size of Trend 1	Size of Trend 2	Total Size	w_o	w_c	w_l	Runtime CPU (sec)
Artichoke	221	50	105	125	230	0.4	0.2	0.4	100
Hand	320	60	166	202	368	0.4	0.2	0.4	235
House	211	55	139	128	267	0.3	0.2	0.5	89
SRI	174	40	87	94	181	0.4	0.2	0.4	65

The original image sequence and the transfigured image sequence represent two different trends in the data set. It is important to note that the trends were extracted in the original space and are visualized in the reduced 2D space in Fig. 7. The original high-dimensional feature vectors can be visualized in either 2D or 3D space using traditional manifold learning algorithms [2, 5]. However, one cannot visualize more complicated datasets using such algorithms. Our algorithm, on the other hand, can extract the trends (and data corresponding to such trends) individually and then lay out the data ordered by each trend in a low-dimensional space. This gives users a way to inspect the trend-specific variability in the data that could be difficult to un-entangle by full manifold projections.

In all these data sets, we were able to identify the trends in the data using our algorithm. Table 3 gives the results after applying the algorithm to the datasets. Depending on the form of the data and number of data points, we initialized the *k*-means algorithm with a certain number of centroids between 40 and 60. The only other input parameters to our algorithm are the set of weights which will have to be chosen so that the output trends will have minimum overlap, maximum length and maximum smoothness. The standard parameter values used in our testing were w_l = 0.4, w_o = 0.4 and w_c = 0.2. In some cases (such as the House data set) a minor tweaking of the parameters was needed in order to obtain the original trends. Table 3 also gives the number of data points belonging to each of the trends. Since the MST and the data assignment are done based on the centroid, a data point (especially in the neighborhood of the intersecting centroids) may belong to more than one trend.

6 Conclusion and Future Research

In spite of the vast literature in clustering and curve fitting, some real-world problems pose interesting challenges that require development of novel algorithms that can advance one step further from the basic clustering or curve fitting solutions. Our algorithm proposes to find interesting, continuous *trends* in the datasets. Analyzing these trends can provide useful insight into the nature of the data distributions and existing correlations.

Future work in this direction is to use the uncovered trends for real world datasets and find interesting associations between trends from different descriptions of the data, such as different subsets of features. Clustering procedures other than *k*-means can be explored for potential improvements, like cluster sculptor which allows the user to tune the clustering parameters so that the domain knowledge of the user can assist in creating better clusters [11]. For very high-dimensional datasets, simple Euclidean distances may not be the best metric for clustering. Additional possibilities include an analysis of local intrinsic dimensionality of the data, and using it to guide a weighting on the metric to minimize the effect of noise. Another possible extension is to cut the principal tree (MST) at the longest edges to break the data down to more natural clusters, in case there exists such an edge (known as the inconsistent edge in graph-theoretic methods for clustering). This can help avoid forcing a global trend on data that contain natural clusters that are far apart.

Acknowledgments

This work is partially funded by the Wayne State University Faculty research award and the DHS Dynamic Data Analysis Center grant.

References

[1] A. O. Artero, M. F. de Oliveira, and H. Levkowitz. Enhanced high dimensional data visualization through dimension reduction and attribute arrangement. In *Proceedings of the conference on Information Visualization*, pages 707–712, 2006.

[2] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

[3] C. D. Boor. *A Practical Guide to Splines*. Springer-Verlag, 1978.

[4] K. Chang and J. Ghosh. A unified model for probabilistic principal surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(1):22–41, 2001.

[5] D. L. Donoho and C. Grimes. Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. *Proceedings of the National Academy of Sciences*, 100(10):5591–5596, 2003.

[6] T. Hastie and W. Stuetzle. Principal curves. *Journal of American Statistical Association*, 84:502–516, 1989.

[7] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning. Data Mining, Inference, and Prediction*, chapter Boosting and Additive Trees. Springer-Verlag New York, 2001.

[8] T. Ho. Interactive tools for pattern discovery. *Proceedings of the 17th International Conference on Pattern Recognition, Cambridge, U.K.*, 2:509–512, 2004.

[9] B. Kegl and A. Krzyzak. Piecewise linear skeletonization using principal curves. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 24(1):59–74, 2002.

[10] G. J. McLachlan and K. E. Basford. *Mixture models: Inference and applications to clustering*. Marcel Dekker, New York, 1988.

[11] E. Nam, Y. Han, K. Mueller, A. Zelenyuk, and D. Imre. Clustersculptor: A visual analytics tool for high-dimensional data. In *IEEE Symposium on Visual Analytics Science and Technology (VAST)*, pages 75–82, 2007.

[12] Y. Shi, Y. Song, and A. Zhang. A shrinking-based clustering approach for multidimensional data. *IEEE Transactions on Knowledge and Data Engineering*, 17(10):1389–1403, 2005.

[13] R. L. R. Thomas H. Cormen, Charles E. Leiserson and C. Stein. *Introduction to Algorithms,Second Edition*, chapter Minimum Spanning Trees. MIT Press and McGraw-Hill, 2001.