A Bi-directional Visualization Pipeline that Enables Visual to Parametric Interation (V2PI)

FODAVA Technical Report

Scotland C. Leman, Leanna House, Dipayan Maiti, Alex Endert, Chris North

Abstract

Typical data visualizations result from linear pipelines that include both characterizing data by a model or algorithm to reduce the dimension and summarize structure; and displaying the data in reduced dimensional form. Sensemaking then takes place as users observe, digest, and internalize any information displayed. The problem is that visualisations driven solely by algorithms or models may limit sensemaking because they have the potential to mask expected or known structure in the data. In this paper, we present a framework for creating data displays that rely on both mechanistic data summaries and expert judgement. In order for users to communicate their judgements, we present a new form of human-data interactions to which we refer as "Visual to Parametric Interation" (V2PI). Here, we develop both the theory and methods to create VA tools for users to adjust the parameter space while staying within the visual space. The coupled visual and parametric adjustments defines V2PI. When tools have V2PI capabilities users may not need to leave the visual space to explore data *and* test hypotheses. We demonstrate the benefits of V2PI in three examples.

1 Introduction

Organizing and understanding large data sets are complex tasks for many scientists, engineers, and intelligence analysts. To aid users in such sensemaking endeavors, tools have been developed to display high-dimensional data visually. These tools rely on mathematical models or algorithms that collapse high-dimensional data matrices to much smaller visual spaces (i.e., spaces of only two or three dimensions). For example, a spatialization of textual data (which may include 1000's of dimensions) extends upon a geography metaphor in that it portrays complex observations in a two-dimensional map (Andrews and Fox 2007). For a variety of reasons, a spatial visualization of text data can lack interpretability for users. When this happens, users have limited options to correct any problems, and we say that the "pipeline" or framework in which the visualization was created is broken. In this paper, we propose a new framework for creating data displays that mends the pipeline and fosters sensemaking by enabling a new form of human interaction with data.

Displays of data in two or three dimensions result typically from a visualization pipeline shown in Figure 1, where data (D) are summarized by a mathematical model or algorithm ($M(\theta)$) first and subsequently mapped to a visual display (V). A display is controlled solely by the algorithm that generated it and adheres only to predefined mathematical objectives or constraints denoted by θ. When these constraints contradict expert judgment, visualisations can lose interpretability because they warp or miss useful data features. For example, Principal Component Analysis (PCA) (Jolliffe 2005) and Multidimensional Scaling (MDS) (Kruskal and Wish 1978) are common analytical approaches used to visualize data. PCA projects data sets to two dimensions in the directions with the highest variance, but experts may know in advance that useful features in the data do not correlate with variance. Additionally, MDS places equal weight across the data dimensions when calculating new, two-dimensional coordinates for observations; however, users may know in advance that a subset of the variables deserves more weight than others to explain any expected behavior in the data.

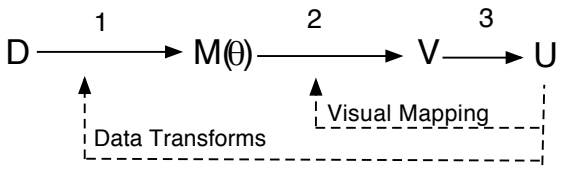

Figure 1: Standard visualization pipeline, where data (D) feeds into a mathematical model ($M(\theta)$), and produces a visualization (V). The users (U) make sense of the visualization to the best other their abilities. To correct any visual inaccuracies, users must either change the data or the mathematical model directly.

It may be difficult (if not, impossible), to develop mathematical models or algorithms that summarize every complexity in a data set and match the extensive domain expertise of users. Yet, as defined by the current pipeline (Figure 1), users do not have an intuitive means to correct visual inaccuracies. The only way for users to correct visual problems is to either transform the data set or adjust the underlying mathematical models or algorithms. This means that users, who may not have the appropriate mathematical training, must have a deep enough understanding of the display-generating models to change them in a way that will result in useful visualizations. When users cannot parameterize their expert judgements, the pipeline is broken and sensemaking ceases.

To assist users in altering quantitative data summaries, some visual analytics (VA) tools have been developed. For example, iPCA (Jeong et al. 2009) and XGvis (Buja et al. 1998) allow users to adjust dials or sliders that either augment or outright change influential parameters in PCA or MDS, respectively. However, iPCA and XGvis still force users to understand the mathematical models or algorithms underlying the visualizations. Without understanding the mathematics, changes to visualizations are blind in that users can only hope (not know) that their slider or dial adjustments will convey their expert judgments appropriately. Additionally, it is

understood that users' mental maps of data sets tend to match visualizations closely, not mathematical characterizations. Therefore, users may never obtain a visualization that reflects their expertise, regardless of how they tweak dials or sliders.

The root of the problems inherent of iPCA and XGvis is that user interactions take place within an abstract parameter space and never where users host their intuition - within the visual space. Tools that enable users to edit displays independent of the underlying models or algorithms still limit usability. Despite the pitfalls of characterizing data quantitatively, rigorous analyses have the potential to reveal new knowledge to users and foster the sensemaking process. Thus, purely editorial changes to displays can diminish the role of visualisations in the sensemaking process as well. Users need a mechanism to balance parametric and editorial or "surface level" adjustments to data displays.

In this paper, we propose a bi-directional, visualization pipeline that parallels the sensemaking process and enables a third form of human interaction with data. We refer as "Visual to Parametric Interation" (V2PI). Our pipeline is an extension of Figure 1 in that users are not simply at the receiving end of the pipeline. Rather, we embed users in the scheme formally so that adjustments to displays are indirect adjustments to underlying model parameters which may, in turn, create new visualizations. The coupled adjustments to the visual and parametric spaces define V2PI.

In order to commit V2PIs to displays, we need smart VA tools that have the capacity to 1) capture user interactions with the visual metaphor, 2) interpret the interactions quantitatively, and 3) reconfigure visualizations based on the user input. Crucially, the tools enable users to remain in a visual metaphor to organize information *and* generate/test hypotheses analytically. In effect, the tools replace the role of an analytical expert (e.g., mathematician, statistician, or computer scientist) in that they parameterize feedback from the users and communicate with the users via the visual metaphor. Users are shielded completely from any underlying mathematical technicalities and are free to focus their attention on the application; i.e., analysts are not distracted from their applied research questions by trying to learn the technical underpinnings of visualizations when they use the VA that we prescribe. In this paper, we develop a general framework for creating these tools, and we provide three examples.

The motivations to develop the bi-directional pipeline and V2PI are grounded within theoretical and practical concepts in VA that are discussed in the next section. We describe in Sections 3 and 4, respectively, a general form for bi-directional visualization pipelines and describe a specific instance when V2PI is possible. Additionally, we make suggestions in Section 4 for creating smart VA tools that parameterize user display interactions and enable V2PI. In sections 5, we exemplify the usefulness of bi-directional visualisations with V2PI in three applications. We conclude with a discussion in Section 6.

2 Visual Analytics (VA)

The key aspect of VA that makes it different from both data analytics and data visualization is its emphasis on user interactions with data and analytical processes that lead to insight. As indicated by Pike et al. (2009), "interaction *is* the insight," and according to Thomas and Cook (2005), VA "is the science of analytical reasoning facilitated by interactive visual interfaces." In this section, we highlight the importance and challenges of modeling high-dimensional, complex data sets and three forms of user interaction. The third form is V2PI.

2.1 Mathematical Models and Algorithms

Before we begin, we note that from this point forward we will use the terms "model" or "algorithm" interchangeably to refer to the quantitative display-generating mechanism or "M" in Figure 1. The differentiation is irrelevant within the context of this paper.

In interactive systems, users rely on automated, mathematical algorithms to create initial data displays. These displays organize data based on both sub-features in data and quantitative constraints and/or assumptions inherent to the algorithms. Although the constraints are necessary to assure mathematical coherence and justifiable inferences, they can inhibit sensemaking as well. Thus, model developers (e.g., mathematicians, statisticians, and computer scientists) undergo a plethora of steps to validate constraints for every application. Among these steps are 1) diagnostically checking that the model assumptions are met by the data, 2) making sure the model answers a relevant question, and 3) the model is interpretable. Step 1 usually relies upon theoretically rigorous methods and/or visual inspections of relevant figures. Visual inspections often suffice for justifying modeling assumptions. Steps 2) and 3) are critical for ascertaining whether we can learn something from the data; i.e., can the model provide any meaningful insights about the data? A mathematical characterization of the data may describe irrelevant aspects of the data, even when the modeling assumptions seem appropriate.

To avoid failures in any of the steps, domain and modeling experts tend to work together. In the case of statistical modeling, a statistician and domain expert may collaborate during all stages of an analysis, including both the data collection and inferential stages. Domain experts can provide guidance regarding expected behavior in the data and modelers can develop procedures which adhere to the guidance. However, verbal communication may be time consuming and error prone. Domain experts often complain that modelers "do not speak their language" in that common or familiar terminologies differ among domain and modeling experts. In turn, VA tools have been developed that enable domain experts to interact with the data and/or models directly. However, domain experts are not relieved from the responsibility of implementing any or all of the aforementioned validation steps.

If modelers could develop a generalizable algorithm that would characterize a variety of data structures successfully, domain experts could be protected from many

technical modeling issues. Needless to say, such an algorithm does not exist, thus domain experts are limited to either the models that they understand or working with quantitative collaborators. In the next section, we develop a form of human-data interaction that empowers users to adjust algorithms so that they are valid and applicable to a variety of data sets.

2.2 Interaction Types

In an effort to allow users to perform an analysis visually, many tools incorporate visual interaction. In VA, various types of interactions have been studied, and Pike et al. (2009) categorize them into two main groups: *lower-* and *higher-* level interactions. The key difference between these groups relates to the goal of the users when they interact with the data. With lower-level interactions, users aim to summarize "low-level structure" in the data including maxima, minima, simple patterns, and linear trends. Examples of such interactions include filtering, sorting, and other specific formal queries. Any interactions that are not considered lower-level are higher-level. The purpose of higher-level interactions is to "understand" the data by uncovering features based on abstract or complex (e.g., nonlinear) data characterizations.

In this section, we refine the interaction groups further and define *surface level*, *parametric*, and *visual to parametric* interactions. We will explain each interaction within the context of Figure 2. Figure 2 was created by a VA tool called IN-SPIRE (Pak Chung et al. 2004) and displays a "Galaxy View" of text data that were collected for an intelligence analysis. In this spatialization, the data points, i.e., documents, are represented by dots and clustered algorithmically by IN-SPIRE. The aim for IN-SPIRE is to assist users in grouping similar documents together and displaying them in an accessible fashion.

2.2.1 Surface level interactions

Surface level interactions are performed purely within the visual domain and are contained in the lower-level class of interactions. Data rotations, reflections, and translations, highlighting or editing observations, and zooming into a portion of the visual space are each examples of surface level interactions. These interactions, while capable of enhancing the understanding of complex data structures, do not necessarily relate coherently to mathematical data structures. Within the context of Figure 2, surface level interactions may include opening, closing, highlighting, and filtering documents or repositioning clusters. For example, users may wish to drag the cluster labeled by, "rain, snow, storm," to the bottom right of the screen because they feel that the cluster is unimportant. This adjustment is independent of the underlying algorithm and committed purely for organizational purposes.

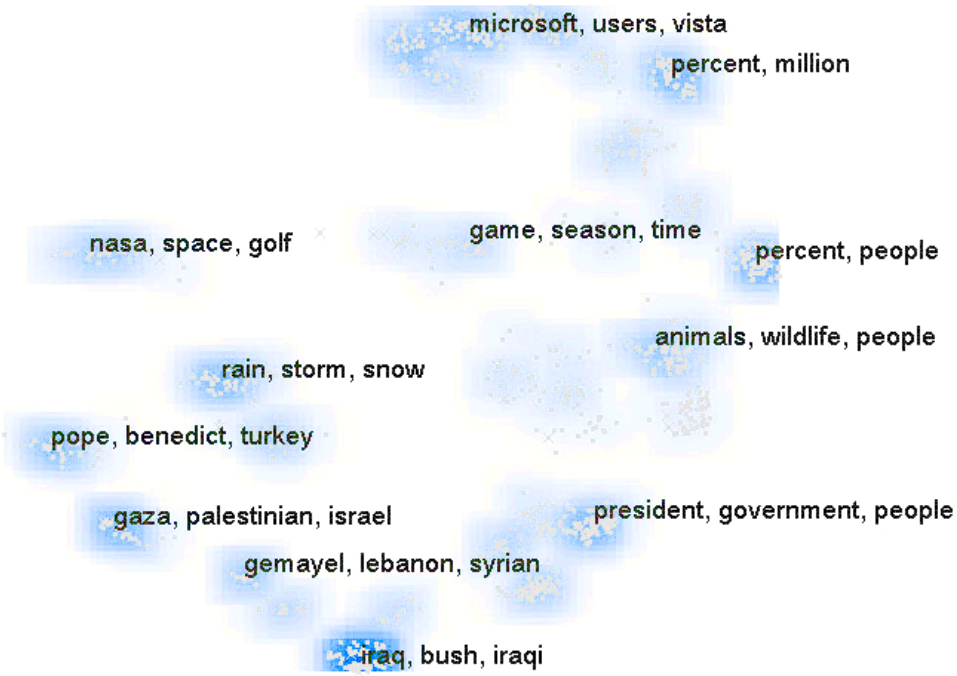

Figure 2: A "galaxy view" of text data from an intelligence analysis that was created by the IN-SPIRE suite of data visualizations. In-SPIRE uses complex mathematical models in order to discern structure (e.g., clusters) in high-dimensional data.

2.2.2 Parametric Interactions

Parametric interactions are performed directly on the mathematical models that control visualizations. iPCA and XGvis are VA tools that permit parametric interactions; iPCA allows users to interact directly with the principle eigenspace of the data, and XGvis enables users to change either the analytical metric scaling method (measure for distance between observations) or the local optimization scheme used to solve for lower dimensional versions of high-dimensional observations. If IN-SPIRE had the capability for a user to specify, say, the number of clusters in Figure 2, it would be an example of a tool that also permits parametric interactions. The following is a non-exhaustive list of other parametric interactions:

- In a cluster-based visualization, a user defines a cluster by specifying the required shape, minimum distance from other clusters, or minimum number of elements.
- In a visualization of a network, a user adjusts the number of nodes and/or edges.
- In a tree-based visualizations, a user adjusts the probabilities that a branch splits.

2.2.3 Visual to Parametric Interactions (V2PI)

Surface level interactions are intuitive to commit to visualizations, but may lack analytical interpretation because they are independent of the mathematical underpinnings of visualizations. Parametric interactions maintain the integrity of mathematical data characterizations, but can be difficult for analysts to implement. To combine the ease of surface level interactions and the mathematical rigor of parametric interactions, we introduce V2PI, Visual to Parametric Interation. Tools that enable V2PI accept and interpret surface level interactions as parametric interactions.

For example, one interpretation of the clustering structure in Figure 2 is that observations within clusters are more correlated than observations between clusters. Suppose a user chose to merge two neighboring clusters together. This surface level action suggests that the algorithm, as parameterized, that drives the IN-SPIRE visualization under estimates the correlation between a subset or all observations. If IN-SPIRE had V2PI capabilities, it would quantify and parametrize the merger to adjust all or a subset of pairwise correlation measurements.

VA tools with V2PI capabilities enable users to make parametric changes to models that control visualizations while remaining in the visual data domain. Thus, the users need not learn about the technicalities of characterizing data with mathematical models. However, developers of V2PI VA tools, must know, in advance, how to interpret, process, and parametrize various surface level interactions. Table 1 provides a set of surface level interactions with parametric interpretations.

In Section 4, we discuss the machinery we use to process and parametrize some surface level interactions. In the next section, however, we show how V2PI fits into a bi-directional visualization pipeline that enhances sensemaking.

Table 1: A non-exhaustive list of V2PI. V2PI requires parametric interpretations of surface level interactions.

	Visualization	**Surface Level Interaction**	**A Parametric Interpretation**
•	Display of data in clusters	Move two points from different clusters to the same cluster	Up weight the current clustering role of the dimensions in which the observations are similar
•	Two dimensional map of data	Change the relative locations of points	Down weight the dimensions that dictate the current map
•	Display a network across nodes/data points	Delete a connection between nodes	Decrease the current correlation between the nodes
•	Classification tree diagram	Delete a classification branch	Reduce the current marginal probability of belong to the corresponding class

3 The Bi-directional Visualization Pipeline

The process of using data to update domain specific knowledge is referred to as sensemaking (Lederberg 1989; Thomas and Cook 2005) and has been represented in the form of a sensemaking process (Pirolli and Card 2005; Card et al. 1999). In this process, analysts (i.e., experts, users, applied researchers, etc.) begin with a knowledge base that they hope to either expand or adjust given the data. The information discernible in data is often unclear to analysts. Thus, learning from data may take place over time or a series iterations during which analysts explore the data and assimilate what they observe with their knowledge bases. The iterative process is the "sensemaking process" and stops when analysts believe that they have "made sense" of the data.

The current method of building visualizations allows analysts to observe structure in data (e.g., clusters, tree diagrams, etc), but the static nature of the end result (the data display) limits or, at best, slows the sensemaking process. The responsibility of understanding and digesting what is revealed in displays lies fully in the hands of users. Analysts are experts in their respective fields and often have mental maps of data which include known or expected data features. When data visualisations and mental data maps do not match, analysts have two choices to complete the sensemaking process. First, they may learn about the underlying visualization-generating algorithm and either adjust the data or the algorithm so that the visualization changes. Second, they may make surface level interactions and *remember* (if they can) how/why the visualisations fail to update their current knowledge base.

In this paper, we propose a bi-directional pipeline that generalizes a framework to create dynamic visualizations that adjust to feedback provided by users. We display our pipeline in Figure 3 and note that it is similar to Figure 1, except users U may receive and distribute information. Specifically, **Steps 1**, **2**, and **3** of the bi-directional pipeline are similar to the original pipeline in Figure 1. Data D are fed into a mathematical model $M(\theta)$ to construct a visualization V that is then assessed by domain experts. In **Step 4** analysts supply feedback F about the model and, in **Step 5**, the feedback is used to update the model. Subsequently, the process repeats to create new visualizations and enable more opportunities for feedback. Steps 2-5 continue until the user decides to stop the sensemaking process.

The mechanisms by which users communicate feedback in step 4 or update the model in step 5 depend upon the resources available to the users and how they may interact with the data. For example, users who understand the model, may internalize their feedback and update the model directly using parametric interactions; or, users who do not understand the model, may commit surface level interactions that, say, a Statistician may interpret to change the model accordingly. Additionally, users who have access to VA tools with V2PI capabilities may make surface level interactions in step 4, that are automatically translated to parametric interactions in step 5 which are worthy of updating the model. In the next section, we return to V2PI and the procedures needed to both parameterize visual adjustments and update display-

generating algorithms.

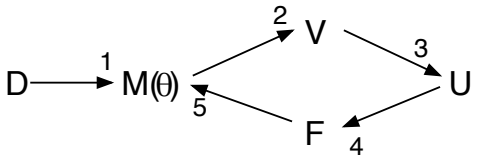

Figure 3: Human interactive pipeline into a mathematical model. Steps: 1) Fit a mathematical model M to the data D, 2) provide data summaries for visualizations V, 3) display visualizations for users U, 4,5) Users adjust the visualization to update model M.

4 Visualizing with V2PI

In the previous section, we suggested a general framework for creating dynamic visualisations. Now, we develop a specific instance of the framework, shown in Figure 4, that relies on VA tools which offer V2PI capabilities. All of the steps are the same in Figure 4 as they are in Figure 3, but we make the forms in which users inject feedback and the mechanisms for updating models explicit. For example, VA tools with V2PI capabilities consider changes to visualizations as feedback for Step 4 of Figure 3 and paramaterize the feedback so that Step 5, updating $M(\theta)$, is straightforward. Thus, in this section, we start by refining our concept of feedback F to reflect the fact that there are two versions in V2PI: a visual or cognitive version and a parametrized version. Additionally, we explain the ease of completing step 5, given parameterized feedback.

4.1 Refining Feedback

When V2PI is an option, analysts use the visual space of the data to communicate their feedback. In effect, they create a new visualization, V', which is an adjustment to V. For this reason, Figure 4 draws an arrow from users U to V' and considers this to be step 4 in the visualization process.

An adjustment to the visual space of data is a representation of a user's cognitive interpretation of the display and his/her judgements. Thus, we define *Cognitive Feedback*, F_c, and show in Figure 4 that the result of step 4 (which is V') in the V2PI bi-directional pipeline relies on the user cognitive feedback F_c.

VA tools with V2PI capabilities have the functionality to recognise the differences between V and V' and interpret them quantitatively. The result of the interpretation is a parameterized version of F_c that we represent as F_p.

.

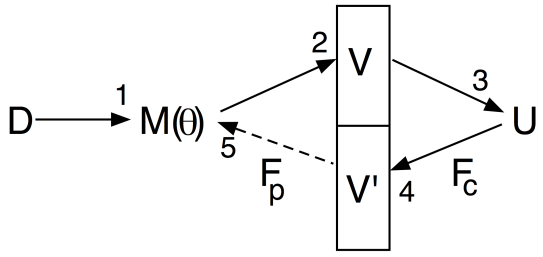

Figure 4: Bi-directional pipeline with V2PI.

4.2 Parameterizing Feedback

We represent the connection between F_p and M in Figure 4 by a dotted line because, to the user, the parametrization of F_c is a "black box" procedure embedded in the VA tools. The black box depends intricately upon both the mathematical model and application at hand. Here, we explain the general procedure and refer readers to Section 5 for examples.

As stated previously, interactive systems rely on mathematical formulations of data which are viewable. These formulations include tunable parameter(s) θ which we set initially by the data to create visualizations V. For example, we may set θ to optimize a predetermined function that depends upon the data. If we were to change θ for a data set, the visualization of the data set would change as well. Thus, provided user adjustments to displays, we have a classic inverse problem. We solve for θ such that the result provides the adjusted display. We denote the solution for θ as θ_F; $F_p = \theta_F$.

We make the distinction between the original value for θ and the parameterized feedback θ_F because we aim to have tunable parameters guided by users, not specified by users. This means that we do not necessarily set θ to θ_F to update underlying models of visualizations, and we allow users to inject feedback sequentially. In the next section we discuss the sequential process for updating models and feedback.

4.3 Update Inferences

An important feature of the bi-directional pipeline is that users may iterate through the visualizations steps multiple times. Suppose that we denote iterations through the bi-directional pipeline in Figure 4 by i. For each iteration, we have $\theta^{(i)}$ from step 1 and $F_p^{(i)} = \theta_F^{(i)}$ from step 4. To derive $\theta^{(i+1)}$, we may consider a variety of methods. We opt to take a weighted average between $\theta^{(i)}$ and $\theta_F^{(i)}$,

$$\theta^{(i+1)} = \frac{\alpha\theta_F^{(i)} + \beta\theta^{(i)}}{\alpha + \beta}, \tag{1}$$

where α and β are measures that support the feedback and previous visualization, respectively, and are chosen either by the user or by default mechanisms. These

measures α and β change from iteration to iteration. Notice that if we normalize α and β, we have $\rho = \alpha/(\alpha+\beta)$ and $1-\rho = \beta/(\alpha+\beta)$ that lay within $[0, 1]$ and reflect the proportion of $\theta^{(i+1)}$ that generates from user feedback and the previous visualization respectively. Users can observe how their feedback changes a visualization by slowly transitioning p between 0 and 1. Selecting a good choice for ρ may require additional expert judgment.

Other methods for updating θ and visualizations rely on formalizing display-generating models probabilistically, where information in data is fully measurable and quantifiable. In fact, House et al. (2010) show by example that the weighted average in Equation (1) is justifiable under a theoretically rigorous Bayesian sequential updating scheme.

Note, that bi-directional pipeline does not have a formal measure of convergence as users iterate and explore data. Rather, users choose to stop iterating when they feel comfortable with their data exploration; e.g., when the data visualizations make sense.

5 Three Examples

In this section, we provide three case studies which outline the steps in the bi-directional pipeline and rely on V2PI. To visualize data in two dimensions, each case study uses one of the following analytical procedures: PCA, MDS, and Isomap (Tenenbaum and V. de Silva 2000). The use of the bi-directional pipeline and V2PI is not limited to these procedures, and, under some conditions, these procedures may provide similar data visualizations. Thus, all three methods may enable users to explore high-dimensional data similarly. Specifically, however, we use PCA in our first example to uncover clusters in data and we use MDS and Isomap in the subsequent examples to assess the relative difference between observations.

The following three subsections begin with a theoretical description of the procedures. Then, we discuss how and/or why a procedure may fail to reveal in a visualization useful data structure. When failures occur, we propose fixing the problems by including expert judgement via V2PI. Thus, we describe how to include the procedures in an interactive, bi-directional pipeline and we detail one method (for each application) that we use to parameterize feedback. We also demonstrate how visualizations may change given user interactions.

The purpose for this section is to demonstrate, by example, a proof of concept for the bi-directional pipeline. We emphasize this point during a short discussion at the conclusion of this section.

5.1 PCA

PCA is a deterministic analytical procedure that relies on an optimal linear projector to reduce the dimension of a data set. Consider a center-shifted, p-dimensional data set $\boldsymbol{x}$ that contains observations $\boldsymbol{x_i}$ where $i \in \{1, ..., n\}$; i.e., $\boldsymbol{x} = \{\boldsymbol{x_1}, ..., \boldsymbol{x_n}\}$ and

$\boldsymbol{x}$ is $p \times n$. PCA relies on the solution for the $q \times p$ transformation matrix $\boldsymbol{W}$ that maximizes the variance of $\boldsymbol{z}$, where

$$\boldsymbol{z} = \boldsymbol{W}'\boldsymbol{x}. \tag{2}$$

To solve for $\boldsymbol{W}$, one option is to take the eigen-decomposition of the sample variance (of $\boldsymbol{x}$), $\boldsymbol{S}$, such that $\boldsymbol{S} = U\Lambda V'$, where U is $p \times p$ and contains the eigenvectors of $\boldsymbol{S}$, $U = V$, and Λ is a diagonal matrix that includes the ordered eigenvalues of $\boldsymbol{S}$ (e.g., the [1,1] element in Λ contains the largest eigenvalue of $\boldsymbol{S}$). Since the eigenvectors that correspond to the two largest eigenvalues mark the top two directions that explain the most amount of variance in $\boldsymbol{x}$, $\boldsymbol{W}$ is assigned to the first two columns of U,

$$\boldsymbol{W} = \boldsymbol{U} \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ \mathbf{0}_{p-2} & \mathbf{0}_{p-2} \end{bmatrix},$$

where $\mathbf{0}_{p-2}$ represents a $(p-2) \times 1$ vector of zeros.

PCA, in its current form, has the potential to miss features in data because of its strict variance criteria and explicit assignment of $\boldsymbol{W}$. Informative data structures may not correlate with variance and we cannot incorporate expert judgement in PCA to guide the specification of $\boldsymbol{W}$. For example, if hidden clusters exist in a high-dimensional data set and the within-cluster variance is larger than the between-cluster variance, the clusters will not appear in the variance-based PCA projection, $\boldsymbol{z}$. Additionally, we cannot adjust $\boldsymbol{W}$ despite knowing the presence and/or characteristics of the clusters.

Now, we transform PCA from a deterministic, dimension reduction algorithm to an expert guided, structure-discovering projection method. Based on the bi-directional pipeline in Section 4, experts may include their judgements or hypotheses about the data in PCA via V2PI. We use the bi-directional pipeline steps described in Sections 3 and 4 to guide our discussion of PCA with V2PI.

5.1.1 PCA with V2PI

We develop the use of V2PI in PCA within the context of a simulated example and assume the roles of both analytical and applied experts. To avoid confusion however, we reserve the word "expert" to reference applied experts only and explain interactive PCA from the point of view of analytical experts, e.g., Mathematicians, Statisticians, or Computer Scientists.

We simulated a $p = 3$ dimensional data set $\boldsymbol{x}$ that contains $n = 300$ observations and three clusters, as shown in Figure 5a. Since we simulated the data, we have access to detailed information concerning the cluster assignments of each observation. Although, for the sake of the exercise, we reveal the cluster assignments to experts for only 20 of the 300 observations; ten observations were selected at random from clusters 1 and 2 each. The goal of this section is to develop an approach that will enable experts to visualize the data using PCA, include what they know about the

selected observations in the visualization, and make sense of the data concerning both the number of clusters and the cluster assignments for the remaining observations.

We start by using PCA for **Steps 1-3** in Figure 4. We derive $\boldsymbol{W}$ in accordance with Equation (2) and display $\boldsymbol{z}$ in Figure 5b. For **Step 4**, experts participate in the data analysis by assessing and/or injecting feedback about the projection. In this case, experts will certainly want to inject feedback because Figure 5b does not display any structure and the 20 observations about which experts know the classifications do not cluster well.

Typical cluster analyses of data aim to group observations together that are similar, and in visualizations, an intuitive measure of observation similarity is distance. Thus, when experts assess a data display, similar observations should appear close in proximity relative to those that differ. However, data projectors are not necessarily distance preserving operators, so projection-based visualizations may fail to depict pairwise relationships, as defined by distance. In such cases (e.g., Figure 5b), experts may adjust the distance between a pair of observations according to their beliefs. In this example, experts may choose to drag two observations from the same cluster that appear far from each each other together (e.g., the observations marked by '$\times$' in Figure 5b). Or, experts may choose to drag two observations from different clusters that seem close in proximity apart (e.g., the observations marked by '+' in Figure 5b). We refer to the result of dragging observations in a display as cognitive feedback, F_c (Section 4).

An important point to make is that F_c is based on the knowledge of pairwise relationships between observations and not on global assessments of the dimensions nor data structures. This means that we do not expect experts to have reliable judgments concerning, say, the dimensions in the data that define clusters; the number of clusters in the data set; nor the size of data clusters. We only expect experts to understand the data and whether visualizations reflect their pariwise, observation-level judgements accurately.

Before we complete step 5, we must parameterize F_c, the separation or consolidation of points. To do so, we keep step 5 in mind in that we recognize that adjustments to a display indicate that we need to update aspects of the data that control the degree to which dimensions are portrayed in the display. In PCA projections, this means that we need to re-weight the variances of the dimensions; dimensions represented well or poorly in PCA projections are those with high and low variances respectively. Thus, we derive a distance matrix as F_p that is both indicative of the observation adjustments and similar in nature to a data covariance matrix; we denote the matrix by $\boldsymbol{S}_F$. The matrix $\boldsymbol{S}_F$ is $p \times p$ and semi-definite. We describe one procedure for deriving $\boldsymbol{S}_F$ from F_c in Appendix A.

5.1.2 Adjusting the Sample Covariance

The final step in V2PI, bi-directional pipeline, **Step 5**, is to update the mathematical procedure the creates the visualization. To do so, we exploit the variance criterion of PCA and adjust the data covariance matrix (used to derive $\boldsymbol{W}$) by $\boldsymbol{S}_F$. Let $\boldsymbol{S}^{(1)}$

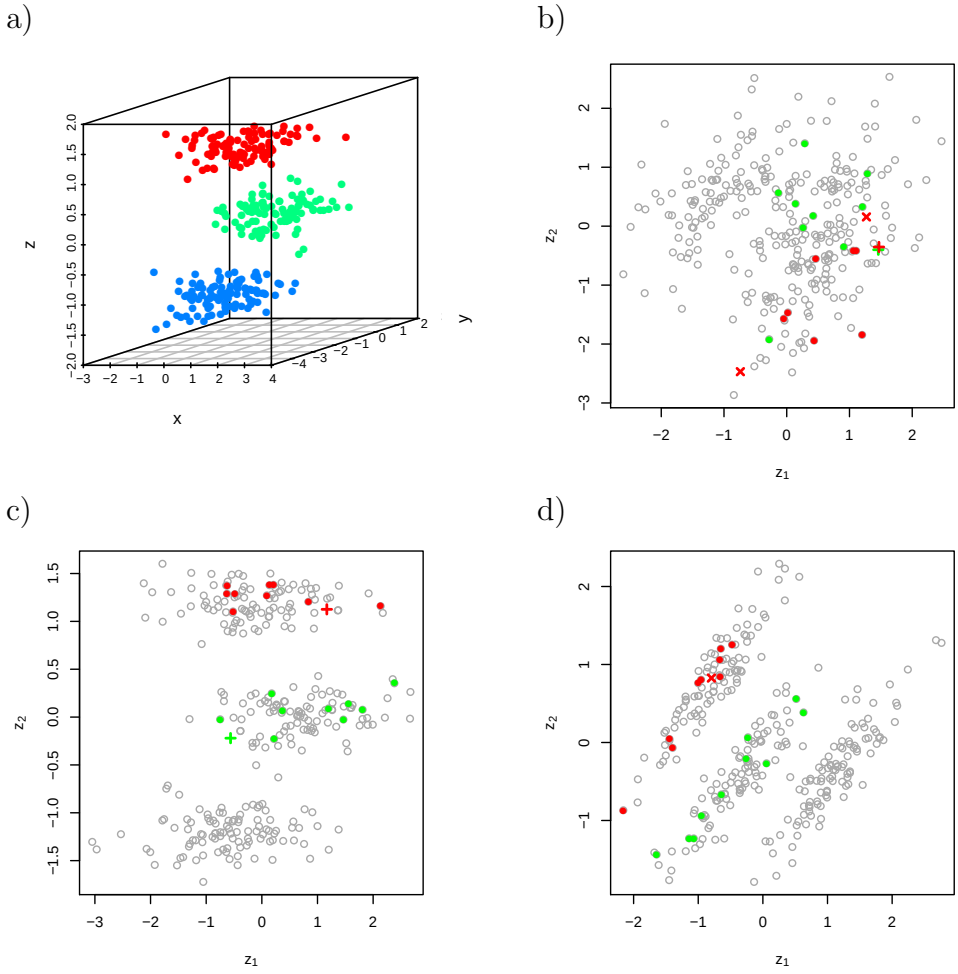

Figure 5: Figure a) displays the simulated data in three dimensions. Observations in red, green, and blue denote groups 1, 2, and 3 respectively. Figure b) displays the PCA projection of the simulated data with the 20 observations that were selected at random to assist experts in adjusting the display. Again, red and green points represent observations in groups 1 and 2 respectively. Figures c) and d) show updated displays after an adjustment to Figure b). Figure c) is the result of moving points marked by '+' in Figure b) apart and Figure d) is the result of moving the points marked by '×' in Figure b) together. Notice that both adjusted visualizations capture the clustering structure.

represent the adjusted variance which we define as

$$\boldsymbol{S}^{(1)} = \rho \boldsymbol{S}_F + (1-\rho)\boldsymbol{S}$$

where ρ ($\rho \in [0,1]$) is provided by the expert and defined in Equation (1). When observations are adjusted in a display, we request that experts provide a measure of their own certainty for the adjustment. This measure tells us the degree to which we should weight their adjustment relative to the current data projection. The weight ρ should be high (close to one) when experts feel strongly about their adjustments and close to zero otherwise. Regardless of the value for ρ, the adjusted matrix $\boldsymbol{S}^{(1)}$ is a semi-definite matrix because both $\boldsymbol{S}_F$ and $\boldsymbol{S}$ are semi-definite. In turn, we can re-apply PCA machinery and update a visualization by deriving a new $\boldsymbol{W}$ based on $\boldsymbol{S}^{(1)}$. The result is a new set of coordinates for $\boldsymbol{z}$ to display.

We provide two adjusted PCA visualizations in Figure 5. Figures 5c and 5d are based on the cognitive feedback that two observations were dragged together and apart in step 4, respectively. Notice that regardless of the action taken for F_c, the adjusted figures display three clusters. From PCA with V2PI, we learned 1) that the data include three, not only two, clusters and 2) the cluster-assignments of every observation simultaneously.

5.2 MDS

In a classical MDS scheme (Torgerson 1958; Kruskal and Wish 1978), the objective is to preserve pairwise distances between observations in low-dimensional representations of high-dimensional data. Thus, we can learn about high-dimensional differences between observations by viewing their relative distances in two dimensional maps.

Given two observations in a high-dimensional space $(\boldsymbol{x_i}, \boldsymbol{x_j})$, the MDS procedure finds points in a lower dimensional latent space $(\boldsymbol{z_i}, \boldsymbol{z_j})$ such that

$$\min_{\boldsymbol{z_1},\ldots,\boldsymbol{z_n}} \sum_{i<j\leq n} \left| \|\boldsymbol{z_i} - \boldsymbol{z_j}\| - \delta_{i,j}^{(\boldsymbol{x})} \right|, \tag{3}$$

where $\boldsymbol{x_i}, \boldsymbol{x_i} \in \mathbb{R}^p$, $\boldsymbol{z_i}, \boldsymbol{z_j} \in \mathbb{R}^q$, $q << p$, and $\delta_{i,j}^{(\boldsymbol{x})} = \|\boldsymbol{x_i} - \boldsymbol{x_j}\|$ is a predefined norm of the distance between $\boldsymbol{x_i}$ and $\boldsymbol{x_j}$. Equation (3) is typically referred to as a *stress* function, and the resolved minimum is called *the stress*. The norms used in MDS will influence the solution to the problem, if the distances themselves are sensitive to the norm under which they are computed. For our purposes, we choose to work in the L_2 norm

$$\delta_{i,j}^{(\boldsymbol{x})} = \sqrt{\sum_d (x_{id} - x_{jd})^2}), \tag{4}$$

where x_{id} and x_{jd} represent the d^{th} element in observations $\boldsymbol{x_i}$ and $\boldsymbol{x_j}$ respectively. This choice is arbitrary and can be adjusted easily to accommodate other norms. The pairwise norms of $\boldsymbol{z}$, e.g., $\|\boldsymbol{z_i} - \boldsymbol{z_j}\|$, are computed similarly to $\delta_{i,j}^{(\boldsymbol{x})}$.

The solution to Equation (3) provides a reasonable q-dimensional representation of $\boldsymbol{x}$ because the low-dimensional, pairwise distances between observations approximates the corresponding pairwise, high-dimensional distances. Albeit, the solution for $\boldsymbol{z}$ is invariant to rotations and reflections, and the scale of $\boldsymbol{z}$ can be arbitrary, but analysts can still learn from MDS about how observations relate to one another by viewing observational proximities. Observations close in proximity relate more to one another than those that are distant.

Weighted Multi Dimensional Scaling (WMDS) (Carroll and Chang 1970; Schiffman et al. 1981) is similar in spirit to MDS, however, the features or dimensions of the high-dimensional space are weighted in order to express their relevance in the q-space. Let $\boldsymbol{w}$ represent a $p-$vector of feature weights, $\boldsymbol{w} = \{w_1, ..., w_p\}$. We compute weighted distances by

$$\delta_{i,j}^{(\boldsymbol{w})} = \sqrt{\sum_{d=1}^{p} w_d (x_{id} - x_{jd})^2},$$

where $\sum_d w_d = 1$. To solve for $\boldsymbol{z}$, we simply replace replace $\delta^{(\boldsymbol{x})}$ with $\delta^{(\boldsymbol{w})}$ in Equation (3). The solutions are identical when $w_i = w_j$ for all $i, j \in \{1, ..., p\}$.

The weights, while not immediately estimable, give users a way to decrease or increase the importance particular features have in deriving $\boldsymbol{z}$ and visualizing useful structure in data. In the next section, we demonstrate how to include feedback from users in WMDS to guide the specification for $\boldsymbol{w}$.

5.2.1 WMDS with V2PI

Consider a case where a user encounters high-dimensional data, where only a small subset of the dimensions contain useful information. We will show by example how the WMDS method with injected human knowledge via V2PI, will be able to identify the useful dimensions and reveal informative visualizations.

Let $\boldsymbol{x}$ represent a six dimensional data set. The first two dimensions represent the longitude and latitude for 19 US cities (Atlanta, Chicago, Denver, Houston, LA, Miami, NYC, San Francisco, Seattle, Washington D.C., Reno, Tucson, Boston, Detroit, Helena, Atlantic City, Charlotte, Knoxville, and Blacksburg). The remaining four dimensions represent random noise; the dimensions consist of four samples of 19 from a Gaussian distribution that is centered at zero and scaled to a degree that is comparable to the variance observed in the longitude and latitude vectors. In this data set $\boldsymbol{x}$, only two of the six features contain real information and the remaining features are essentially junk. Our goal is to visualize the data in a geographically sensible manner.

By applying the WMDS procedure, with $w_i = 1/6$ for $i \in \{1, ..., 6\}$, we obtain a set of two-dimensional latent feature vectors. Since the longitude and latitude are contained in the data set, we would hope that the user would be able to assess a view of the data that is consistent with the US map. Although, even after rescaling the latent features and performing a Mercator projection (Pearson 1990) onto the US

map, the WMDS solution for visualizing the data is less than ideal. We see from Figure 6 that the latent features are inconsistent with the true spatial coordinates of the cities; e.g., Denver appears located in the Gulf of Mexico. The problem is that the four miscellaneous dimensions are masking the two dimensions that are consistent with true spatial proximities of cities.

Figure 6 constitutes **steps 1-3** of the bi-directional pipeline for the six dimensional data set. Without expert judgement, no automated algorithm has the capacity to dif-

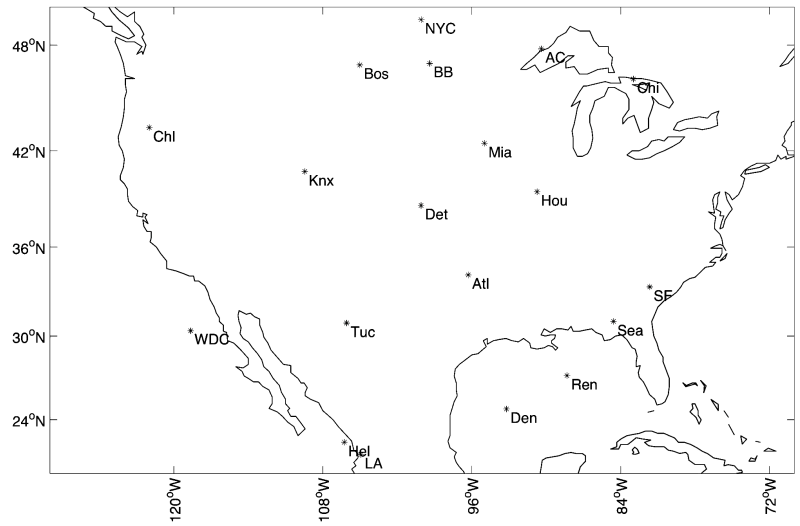

Figure 6: The automated MDS solution for the 6-dimensional data set that includes latitude, longitude, and four noisy dimensions. The solution was rotated and scaled to overlay the US map.

ferentiate the useful from the un-useful features. **Step 4** of the bi-directional pipeline process prompts users for cognitive feedback about visualizations. In this example, users may select a small subset of visualized points and rearrange them so that they match their mental maps of the United States. Figure 7 shows a possible reconfiguration of six cities, Seattle (Sea), San Fransisco (SF), Los Angeles (LA), Houston (Hou), Miami (Mia), and New York City (NYC). We denote the new coordinates for the adjusted cities as $\tilde{\boldsymbol{z}}$. An important point to make is that the bi-directional pipeline does not require users to have comprehensive nor perfect judgements about all of the data. Hence, $\tilde{\boldsymbol{z}}$ need not be of the same dimension of $\boldsymbol{z}$ nor contain the true coordinates of the selected cities.

The subset $\tilde{\boldsymbol{z}}$ represents cognitive feedback for a user and contains information concerning how we should re-weight the importance of the dimensions. We now explain how we parameterize $\tilde{\boldsymbol{z}}$ in the form of a weight vector and complete one iteration of the bi-directional pipeline.

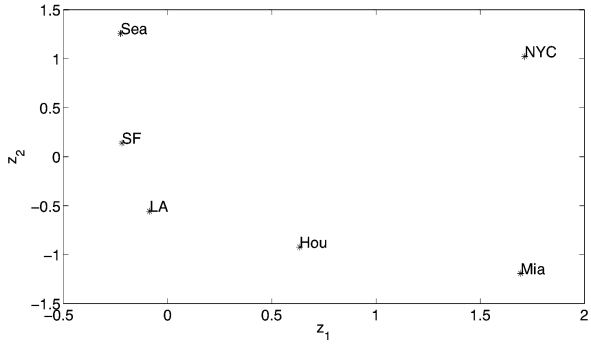

Figure 7: An example of cognitive feedback for WMDS with V2PI.

5.2.2 Updating Weights

To update the weights $\boldsymbol{w} = (w_1, \ldots, w_6)$, we start by calculating a distance matrix for $\tilde{\boldsymbol{z}}$ which we denote by $\delta^{(\tilde{\boldsymbol{z}})}$. If k observations were adjusted, this matrix is $K \times K$, where $K = \binom{k}{2}$. We derive each element in $\delta^{(\tilde{\boldsymbol{z}})}$ using the same L_2 formulation that is provided in Equation (4).

Next, we select the observations that were manipulated in the visualization from the original data matrix $\boldsymbol{x}$ and solve for a new $\boldsymbol{w}$ based on the difference between corresponding pairs of high dimensional weighted norms and elements of $\delta^{(\tilde{\boldsymbol{z}})}$. Explicitly, we solve for $\boldsymbol{w}$ such that $\sum_d w_d = 1$ and

$$\min_{w_1,\ldots,w_6} \sum_{i<j\leq K} \left| \sqrt{\sum_{d=1}^{6} w_d (x_{(i)d} - x_{(j)d})^2} - \delta_{i,j}^{(\tilde{\boldsymbol{z}})} \right|, \tag{5}$$

where $x_{(i)d}$ represents the d^{th} element in the observation of $\boldsymbol{x}$ that maps to $\tilde{\boldsymbol{z}}_{\boldsymbol{i}}$ in the adjusted visualization. The solution to Equation (5) is easily found using a gradient search method (see Mordecai 1976), where the constraint $\sum_d w_d = 1$ is satisfied. We denote the vector of new weights by $\boldsymbol{w}_F$; $F_p = \boldsymbol{w}_F$.

For our example, we found that $\boldsymbol{w}_F = (0.465, 0.5154, 0.0079, 0, 0.0002, 0)$. Since we rescaled the data before computing $\boldsymbol{w}_F$ so that each dimension has a common variance, the elements of $\boldsymbol{w}_F$ are on comparable scales. Thus, the solution for $\boldsymbol{w}_F$ suggests that only the first two features (latitude and longitude) are important and explain the user's mental model or Figure 7.

5.2.3 Updating the visual display

Provided $\boldsymbol{w}_F$, we complete **Step 5** in Figure 4 and update $\boldsymbol{z}$ to reconfigure the data visualization. We find the configuration by minimizing the convex combination of stress functions:

$$\min_{z_1,\ldots,z_n} \sum_{i<j\leq n} \rho \left| \|\boldsymbol{z_i} - \boldsymbol{z_j}\| - \delta_{i,j}^{(\boldsymbol{w}_F)} \right| + (1-\rho) \left| \|\boldsymbol{z_i} - \boldsymbol{z_j}\| - \delta_{i,j}^{(\boldsymbol{w})} \right|, \tag{6}$$

where ρ reflects the degree of certainty the users wish to assign to their feedback. As defined in Section 4.3, $0 \leq \rho \leq 1$.

For our example, we apply Equation (6) (given $\boldsymbol{w}_F$ from Section 5.2.2 and $\rho = 1$) and obtain a new set of latent coordinates $\boldsymbol{z}$. After re-scaling and rotating $\boldsymbol{z}$, we overlay $\boldsymbol{z}$ and the US map in Figure 8. This figure includes both our estimate and the true city coordinates. Notice that the new visualization approximates the true map better than the original visualization in Figure 6. Additionally, the layout of the observations in Figure 7 does not match those in the new visualization. This demonstrates that our procedure is robust to improperly scaled cognitive feedback.

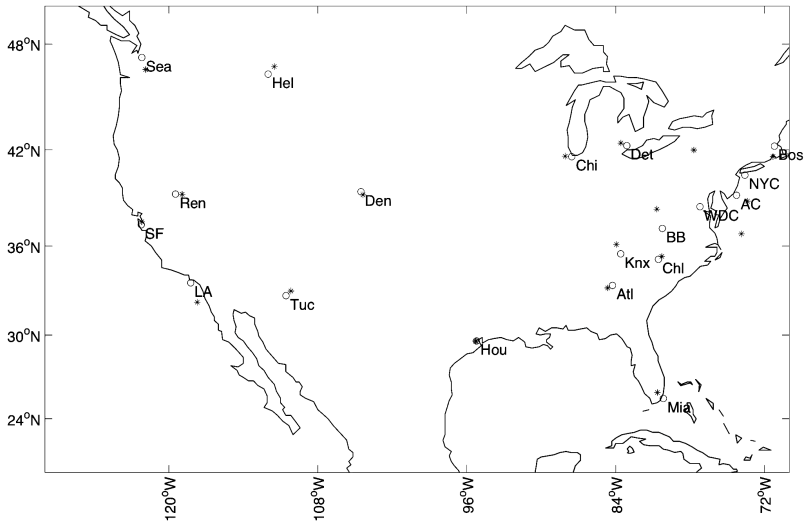

Figure 8: The WMDS solution after V2PI. The solution points are marked by starts ($*$) and the true locations of the cities are marked by open circles ($\circ$). In comparison to Figure 6, the solution using V2PI is more similar to the true map.

Our example shows that while WMDS is useful for summarizing data, it may require human guidance to provide useful spatial visualizations. As the dimension of $\boldsymbol{x}$ increases, we expect the need for guidance from users to increase.

5.3 Isomap

While both PCA and MDS are simple and efficient methods for data visualization, such methods do not approximate nonlinear data structures well. To enable analysts to discover such structures, we consider a data characterization procedure called Isomap (Tenenbaum and V. de Silva 2000). Isomap is used for a variety of applications, including face or pattern recognition and bioinformatics (e.g., Jafri and Arabnia 2009; Nilsson et al. 2004), and estimates low dimensional, nonlinear manifolds that

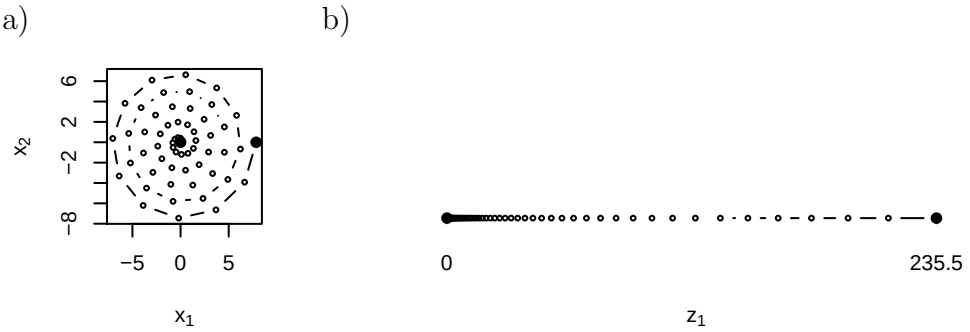

Figure 9: Figure a) represents a high dimension visualization with a low-dimensional manifold. Figure b) display the low-dimensional manifold (i.e., line).

may be embedded in high dimensional data sets. For example, the coil in Figure 9a) contains a one dimensional manifold that is nonlinear in two dimensions. When the manifold is unraveled however, it is a simple line, as shown in Figure 9b). If we were to use Isomap for this example, Isomap would estimate the manifold in Figure 9b). In some contexts, we say that 9b) "is an Isomap of the manifold" within the coil.

Similar to MDS (Section 5.2), the objective for Isomap is to learn about the differences between observations by observing their relative distances on a map of three or fewer dimensions. In the map, Isomap tries to preserve all pairwise, high dimensional geodesic distances between observations. The geodesic distance between two observations is a measure of the distance between observations *along the manifold*. For example, the euclidean distance between the beginning and end of the coil in Figure 9a) is approximately eight units, but the distance between the same points along the manifold, i.e., the geodesic distance, is over 200 units. Figure 9b) preserves the geodesic distance between any pair of observations in the two dimensional space; e.g., between the beginning and end points of the coil.

The challenge is that to measure geodesic distances, an understanding of the manifold - the very structure we are trying to estimate - is needed. Thus, Isomap is a three step procedure that uses estimates for local manifolds within high dimensional spaces to learn global manifolds. The procedure starts by defining localities and subsetting the data. The idea is that if the data are subsetted so that every subset is contained fully on the manifold and approximately linear, the process for estimating all pairwise geodesic distances decomposes to 1) measuring the euclidean distances between pairs within each subset and 2) adding appropriate distances together across subsets. There are a variety of ways to determine subsets in high dimensional data. For our purposes, we use a nearest neighbor approach. For example, let $\boldsymbol{x_i}$ represent an observation in the high dimensional space where $\boldsymbol{x_i}$ is $p \times 1$ and $i \in \{1, ..., n\}$. We define a subset or neighborhood $\mathrm{N}_k(\boldsymbol{x_i})$ by selecting the k nearest observations to $\boldsymbol{x_i}$ in euclidean distance. We segment the entire dataset by the overlapping neighborhoods around each data point. We refer to the boundaries marking the segmented data space as the "graph structure" of the data.

The next step toward estimating an Isomap is to calculate the pairwise euclidean distances within each subset and store a matrix that contains all pairwise geodesic distance approximations. Let $\mathrm{d}_e^h(\boldsymbol{x_i}, \boldsymbol{x_j})$ represent the euclidean distance between observations i and j within neighborhood h. If observations i and j are not within neighborhood h, we set $\mathrm{d}_e^h(\boldsymbol{x_i}, \boldsymbol{x_j}) = \infty$,

$$\mathrm{d}_e^h(\boldsymbol{x_i}, \boldsymbol{x_j}) = \begin{cases} \mathrm{d}_e^h(\boldsymbol{x_i}, \boldsymbol{x_j}), & \text{if } \boldsymbol{x_i}, \boldsymbol{x_j} \in \text{ neighborhood } h \\ \infty, & \text{if } \boldsymbol{x_i}, \boldsymbol{x_j} \notin \text{ neighborhood } h \end{cases}$$

Note, $\mathrm{d}_e^h(\boldsymbol{x_i}, \boldsymbol{x_j}) = \mathrm{d}_e^{h'}(\boldsymbol{x_i}, \boldsymbol{x_j})$ if observations i and j are contained in both neighborhoods h and h'. We store all pairwise, neighborhood euclidean distances in an $n \times n$ matrix D_e, where for all $i, j \in \{1, ..., n\}$ and $h \in \{1, ..., H\}$

$$\mathrm{D}_e[i, j] = \min(\mathrm{d}_e^1(\boldsymbol{x_i}, \boldsymbol{x_j}), ..., \mathrm{d}_e^H(\boldsymbol{x_i}, \boldsymbol{x_j}), \infty).$$

Each pair of observations within the neighborhoods mark an edge that can be traversed to estimate geodesic distances between observations. Let $\hat{\mathrm{d}}_g(\boldsymbol{x_l}, \boldsymbol{x_m})$ represent our estimate for the geodesic distance between observations l and m. The measure $\hat{\mathrm{d}}_g(\boldsymbol{x_l}, \boldsymbol{x_m})$ equals the length of all the edges traversed along the shortest path between observations l and m,

$$\hat{\mathrm{d}}_g(\boldsymbol{x_l}, \boldsymbol{x_m}) = \sum_{h,i,j \in \mathcal{P}_{l,m}} \mathrm{d}_e^h(\boldsymbol{x_i}, \boldsymbol{x_j}),$$

where $\mathcal{P}_{l,m}$ denotes the set of edges and neighborhoods that define the shortest path between observations l and m. We use algorithms including Floyd's algorithm or Dijkstra's algorithm (for sparse neighborhood structures) to learn $\mathcal{P}_{l,m}$ and obtain estimates $\hat{\mathrm{d}}_g(\boldsymbol{x_l}, \boldsymbol{x_m})$ for all pairs (l, m) (Kumar et al. 1994). We refer to the distance traveled on an edge (i.e., between two points on the shortest path) selected by an algorithm as a "hop." Let D_g represent an $n \times n$ distance matrix where $\mathrm{D}_g[l, m] = \hat{\mathrm{d}}_g(\boldsymbol{x_l}, \boldsymbol{x_m})$. Note, $\mathrm{D}_g[l, m] = \mathrm{D}_e[l, m]$ when observations l and m are within one neighborhood.

The final step in the Isomap procedure is to apply classical MDS using the distance matrix D_g. Similar to Equation (3), let $\boldsymbol{z_i}$ represent the lower dimensional analog of observation $\boldsymbol{x_i}$. We solve for each $\boldsymbol{z_i}$ and $\boldsymbol{z_j}$ that preserves the geodesic distance between observations $\boldsymbol{x_i}$ and $\boldsymbol{x_j}$ for all $i, j \in \{1, ..., n\}$.

5.3.1 Tunable Parameters in Isomap

The Isomap is, by definition, a deterministic procedure. Although, depending upon characteristics of the data it can be sensitive to some specifications, including the specification for k and the measure (e.g., euclidean distance) used to assess pairwise observational distances within neighborhoods.

Tenenbaum and V. de Silva (2000) show, that for a uniform and sufficiently high point density on the manifold in the high dimensional spaces, there exists a k such

that the shortest path approaches the true geodesic distance. However, in practice, it can be hard to select the best k which avoids 'short-circuiting' the manifold; e.g., skipping off the manifold and falsely assuming that two points are close in geodesic distance when they are not. The number k is sensitive to both the manifold geometry or shape and the density of the data points on the manifold.

For example, one approach for avoiding the problem of short-circuiting may be to choose an arbitrarily small k. Although, if the data are sparse in the high dimensional space, the local detection of the manifold becomes increasingly difficult, if not, impossible. Estimating a manifold with only a few points is analogous to estimating the radius for the curvature of an arc that is marked by only a few points. Additionally, with sparse data, the k nearest neighbors subroutine can create disjoint graph structures. This means that two observations in disjoint neighborhoods may not have a continuous path between them.

When the high dimensional data are noisy, an Isomap will likely short-circuit the manifold, regardless of the specification for k. To correct the Isomap, it may help to adjust the measure used to assemble D_e that stores all pairwise, neighborhood distances between observations. Since algorithms, such as Floyd's or Dijkstra's, select edges to traverse between two observations based on D_e, a change in D_e may steer the algorithms to select new paths. In particular, if a new measure increases the magnitude for the length of edges that short circuit the manifold, the algorithms will likely avoid these edges when defining the shortest path between observations. In the next section we discuss how to use feedback from experts to scale the lengths of short-circuiting edges.

5.3.2 Isomap with V2PI

Similar to the other methods, we use an example to explain Isomap with V2PI. We simulated $n = 1000$ observations in the form of a 'Swiss roll' as shown in Figure 10a). Our goal is to learn the planar manifold embedded in the Swiss roll using Isomap with V2PI. Note that the observations in Figure 10a) are color-coded so that they change along the manifold according to the rainbow (ROYGBIV). A successful estimate of the manifold will maintain the same sequence of colors.

To implement **Steps 1-3** as defined in Section 3, we set the parameter k to 15 and create the graph structure of the Swiss roll in Figure 10b) to produce a data visualization in Figure 11a). We set $k = 15$ because it has the potential to be small enough ($k = 15$ is less than 2% of $n = 1000$) to avoid short-circuiting the manifold and large enough to avoid gaps in the graph structure. We determine the shortest path between observations by the Dijkstra's algorithm. We can see in Figure 11a) that the Isomap procedure failed to estimate a simple, planar structure of the data.

Similar to PCA and WMDS with V2PI, users have the option to express their cognitive feedback by sliding observation(s) together or apart with a measure ρ (Section 4.3) of their confidence for the move. Moving observations together suggests that analysts believe that the observations are "well connected" or adjacent on the manifold, and moving observations apart suggests otherwise.

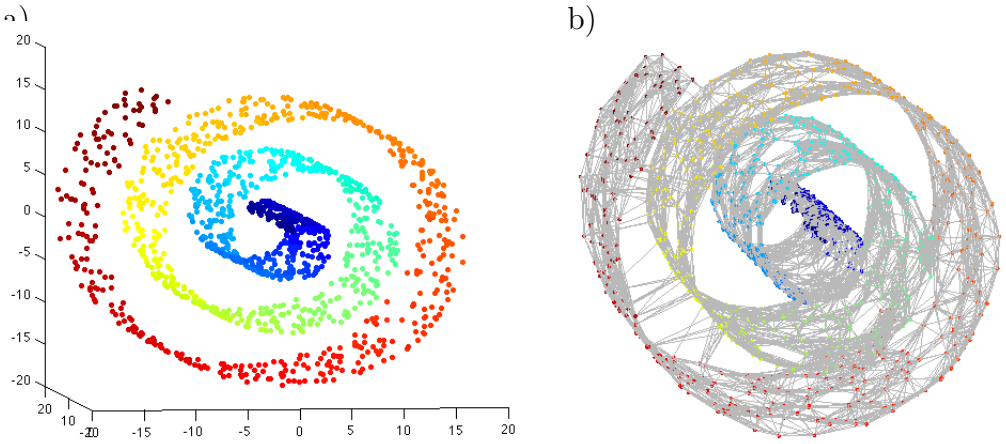

Figure 10: Figure a) displays the simulated Swiss roll data in three dimensions. Figure b) displays the graph structure based on $k = 15$ nearest neighbors. Note the short circuited edges.

For explanatory reasons, suppose a user has the domain expertise to believe strongly that observations l and m which appear close in proximity in Figure 11a) are actually different in the high dimensional space and distant from one another on the manifold. This user completes **Step 4** in the bi-directional visualization pipeline by moving observations l and m apart, as depicted in Figure 11a), and setting $\rho = 0.90$. For this case, we define F_c to represent the specification for ρ and the choice to move observations apart.

Before moving to Step 5, we parameterize F_c. As mentioned in Section 5.3.1, k and D_e are tunable components in the Isomap procedure. Thus, we could interpret F_c as a specification for k or as an adjustment to D_e. We choose the latter. To paramaterize F_c, we derive a new, neighborhood distance matrix which we denote by D_F; $F_p = \mathrm{D}_F$. When users suggest that observations should appear farther apart in the visualizations than they appear, the Isomap procedure has underestimated the geodesic distance between the observations. This means that either the Floyd's or Dijkstra's (in this case, Dijkstra's) algorithm has selected a path $\mathcal{P}_{l,m}$ based on D_e which includes one or more edges that short circuit the manifold.

The first step in deriving D_F is to identify the short circuiting edge(s) in $\mathcal{P}_{l,m}$. For these edges and all those in the structure graph with comparable lengths, we scale the euclidean distances by α where $\alpha \geq 1$,

$$\mathrm{D}_F[i,j] = \begin{cases} \alpha \mathrm{D}_e[i,j] & \text{if edge } \boldsymbol{x_i}, \boldsymbol{x_j} \nsubseteq \text{ manifold} \\ \mathrm{D}_e[i,j] & \text{if edge } \boldsymbol{x_i}, \boldsymbol{x_j} \subseteq \text{ manifold} \end{cases}$$

In Appendix B, we describe how we identity edges to scale and assign α. A notable feature in our development of α is that if there is evidence in the data that suggests a user is incorrect (e.g., a user has dragged two observations apart that are, in deed, adjacent on the manifold), we set $\alpha \approx 1$. Thus, Isomap with V2PI enables experts to

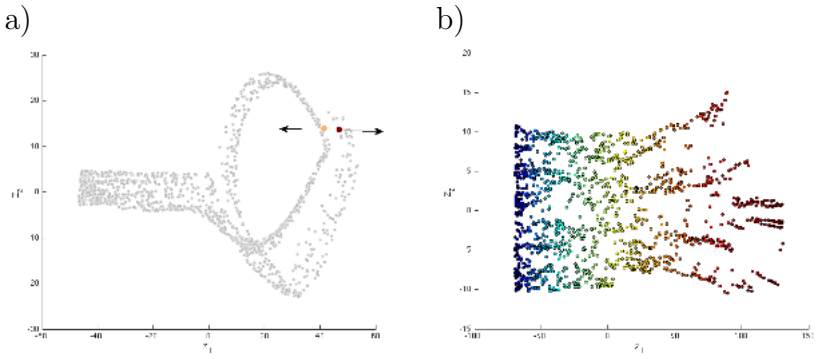

Figure 11: Figure a) displays the Isomap based on $k = 15$ nearest neighbors as seen by the user. User selected data points in color with arrows denoting the cognitive feedback that the colored points should be farther apart from each other. Figure b) displays the Isomap based on $k = 15$ nearest neighbors and after the user feedback is incorporated in the algorithm.

guide the procedure to reveal structure while controlling experts from making poor judgements.

5.3.3 Updating Neighborhood Distance Matrix

As described in Section 4.3 and exemplified in Sections 5.1.2 and 5.2.2, we update the neighborhood distance matrix that underlies Figure 11 by taking a weighted average of D_e and D_F based on ρ. Let ${\mathrm{D}_e}^{(1)}$ represent an adjusted version of D_e. For **Step 5**, we define ${\mathrm{D}_e}^{(1)}$ as

$$ {\mathrm{D}_e}^{(1)} = (1 - \rho)\mathrm{D}_e + \rho \mathrm{D}_F. \tag{7} $$

Figure 11b) displays a new visualization based on ${\mathrm{D}_e}^{(1)}$. Although there is a slight "fraying" of the observations, the Isomap appears generally as a simple plane. Additionally, we validate the use of V2PI by revealing the observation colors in Figure 11b). The colors on the Isomap progress in the order of ROYGBIV.

5.4 Discussion for Examples

In Sections 5.1–5.3, we discussed three common methods to assess and visualize data. In each example, constraints in the mathematical data characterization procedures limited the usefulness of initial data displays because they did not reveal expected nor meaningful structure. In turn, we incorporated the mathematical procedures into the bi-directional visualization pipeline to receive guidance from users via V2PI. The V2PI mechanisms that we presented may be adjusted or altered for different applications. For instance, in each example we prescribed specifically that users move either two (Sections 5.1 and 5.3) or six (Section 5.2) observations to communicate

expert judgement. Different VA tool developers might enable users to move any number of observations. For complex applications, we suspect that an optimal number of observations to move may not exist because the process of exploring and learning from data is expert specific. Different users may benefit more from moving, say, two observations than moving several; and visa verse for other users.

Additionally, VA tool developers may enable users to inject cognitive feedback in forms other than what we present; i.e., moving observations. For example, users may highlight or filter data in displays as forms of injecting cognitive feedback. The only requirement for forms of cognitive feedback is that developers have the means or software in place to parameterize it accurately. This means that in order for developers to be prepared, they must predict how users may naturally include their judgments in visualizations. Future work in V2PI will entail extensive human studies to assess intuitive ways in which users may adjust visualizations.

We conclude this section, by reminding readers that bi-directional pipelines do not include formal measures for convergence. Users may accept and distribute information about data until they feel that they have completed the sensemaking process. One concern is that users may inject judgements that will warp real data structures and mislead users. For some mathematical procedures, the concern is valid and VA tool developers must be aware. Although, for other procedures, the concern is unwarranted. For example, feedback from users for PCA via V2PI does not change the data. Rather, the feedback changes the direction in which data are projected for visualization.

6 Conclusion

In this paper, we introduced two fundamental concepts: the bi-directional visualization pipeline and V2PI. When we combine the two, we have a visualization scheme that enables experts to visualize data and either correct visual inaccuracies or explore potential inaccuracies without understanding the display-generating models. Since users do not need to understand the mathematical underpinnings of visualizations, they are free to build upon their knowledge base and merge their expertise with the information in data instantly. That is, they have an opportunity to learn and interact with the data directly in the visual domain - the domain in which experts host their expertise and intuition.

An important feature of the bi-directional pipeline is that users receive and distribute information; expert judgement and standard data sets are considered to be valid components in quantitative analyses of complex applications. In Bayesian terms, this means that we combine subjective information (communicated via the visualization) with the likelihood to formulate inference. In layman terms, this means that we use both "soft" and "hard" data to update visualisations and construct knowledge. Since we consider feedback to be reliable, useful information worthy of analysis, it is effectively a form of "data" that are collected from experts to learn about the application at hand. Since these data are subject to human biases, we add the qualifier

"soft". Hard data, on the other hand, consists of observations collected formally in a traditional manner for an analysis. Although, hard data sets are still susceptible to biases, even when assumed to contain objective, application specific information. Steps 1 and 2 of the first iteration in the bi-directional visualization scheme rely on hard data.

The successful combination of hard and soft data relies upon the VA tool and mechanisms that support V2PI. The best VA tools are intuitive and accessible to users with varying levels of expertise. In this paper, we did not mention aesthetic aspects of VA tools that need to be considered for human cognitive purposes. Rather, we discussed the mechanics needed to enable tools to allow V2PI. Additionally, VA tool developers must predict the forms in which users may inject feedback so that the machinery is in place to parameterize the feedback and update the visualization. This is a complex task upon which we did not elaborate thoroughly. The act of dragging observations together or apart is only one of many forms in which users may communicate their judgements about visualizations and the underlying models. The best tools will enable multiple forms of feedback injection that are natural and intuitive to users.

Finally, from the perspective of a model or algorithm developer, the methods to update models in this paper may appear ad-hoc. For example, the choice to take the weighted average between data-driven parameter values and feedback might seem arbitrary. Although, we show in House et al. (2010) that if we were to take a fully probabilistic approach to summarize data for visualizations, the weighted average can be justified theoretically for some applications. For other applications, the probabilistic approach may prove that other forms of incorporating feedback into analyses is adequate.

A Paramaterizing Feedback for PCA

Since we can either move observations together or apart, we define distances matrices for each move-type. Let Σ_a and Σ_t represent the distance matrices for the apart and together move-types respectively. We then take a weighted average of each to calculate the F_p,

$$F_p = \nu\Sigma_a + (1-\nu)\Sigma_t,$$

where $\nu \in [0,1]$.

The derivations of ν, Σ_a, and Σ_t are deterministic functions of numerical summaries from the cognitive feedback.

> *Derivation of* ν: This parameter reflects the degree to which experts move observations together or apart. Thus, to determine ν we calculate the ratio of the distances between observations j and k in the two dimensional display before and after the injection of cognitive feedback:
>
> $$\tilde{F} = \frac{||\tilde{\boldsymbol{z}}_{\boldsymbol{j}} - \tilde{\boldsymbol{z}}_{\boldsymbol{k}}||_2}{||\boldsymbol{z}_{\boldsymbol{j}} - \boldsymbol{z}_{\boldsymbol{k}}||_2},$$

where $\boldsymbol{z}_\cdot$ and $\tilde{\boldsymbol{z}}_\cdot$ reference respectively the before and after coordinates of observations j and k. When observations are consolidated, $\tilde{F}$ is less than one and greater than one otherwise. Since $\nu \in [0, 1]$, we use $\tan^{-1}(\cdot)$ to transform $\tilde{F}$ accordingly; $\nu = 2\tan^{-1}(\tilde{F})/\pi$.

Derivation of Σ_a: When observations are separated, we learn that the dimensions reflected poorly in the display need to be up-weighted. Thus, we start by quantifying the degree to which dimensions are unexplained in the visualization. Let $\boldsymbol{d}$ represent the $p \times 1$ raw discrepancy vector between observations j and k,

$$\boldsymbol{d} = |\boldsymbol{x_j} - \boldsymbol{x_k}| \tag{8}$$

and d_l refer to the observation discrepancy in dimension l. We define the projected discrepancy as

$$\boldsymbol{d}^{(p)} = |\boldsymbol{W}(e_l d_l)|$$

where e_l is the l^{th} unit vector. The only nonzero element in $d^{(p)}$ is the l^{th} element, $d_l^{(p)}$. If the raw and projected discrepancies are similar (or different) in dimension l, the visualization characterizes dimension l well (or poorly). To quantify this relationship, we choose to calculate the percent of *unexplained* discrepancy for each dimension l, $U_l = (1 - d_l^{(p)}/d_l)$, and define $d_l^{(u)}$ as

$$d_l^{(u)} = d_l U_l.$$

Collectively, we define the vector of unexplained discrepancies as $\boldsymbol{d}^{(u)} = [d_1^{(u)}, ..., d_p^{(u)}]$.

To define the projection plane that is reflective of the injected feedback, we need two perpendicular vectors which we denote as $\boldsymbol{\kappa}^{(u)}$ and $\boldsymbol{\kappa}^{(o)}$. We define $\boldsymbol{\kappa}^{(u)}$ based on $\boldsymbol{d}^{(u)}$ where $\boldsymbol{\kappa}^{(u)}$ is the normalized sum of $\boldsymbol{d}$ and $\boldsymbol{d}^{(u)}$,

$$\boldsymbol{\kappa}^{(u)} = \frac{\boldsymbol{d} + \boldsymbol{d}^{(u)}}{||\boldsymbol{d} + \boldsymbol{d}^{(u)}||_2}.$$

This vector, in comparison to the first principal component, has the potential to double the weight of under-represented dimensions in the visualisation. For example, if two dimensions have similar residual variances, this vector will add more weight to directions that were not previously explored in the contested visualization. To define the other vector, $\boldsymbol{\kappa}^{(o)}$, we select the direction that is both perpendicular to $\boldsymbol{\kappa}^{(u)}$ and explains the most amount of variance in $\boldsymbol{x}$.

The last step is to combine vectors $\boldsymbol{\kappa}^{(u)}$ and $\boldsymbol{\kappa}^{(o)}$ and determine Σ_a. To do so, we take the outer-produce of $\boldsymbol{\kappa}^{(u)}$ and $\boldsymbol{\kappa}^{(o)}$,

$$\Sigma_a = [\boldsymbol{\kappa}^{(u)}\ \boldsymbol{\kappa}^{(o)}][\boldsymbol{\kappa}^{(u)}\ \boldsymbol{\kappa}^{(o)}]'.$$

The spectral decomposition of Σ_a includes, by definition, eigenvectors $\boldsymbol{\kappa}^{(u)}$ and $\boldsymbol{\kappa}^{(o)}$ with corresponding eigenvalues of one.

Derivation of Σ_t: When observations are consolidated, we learn explicitly that, in an ideal display of the data, the adjusted observations should appear close together. One way to obtain an ideal display is to project the data in the direction defined $\boldsymbol{d}$ in Equation (8). This projection will map the adjusted observations to the exact same coordinates. To determine the vectors that define the projected plane, we solve for vectors $\boldsymbol{\kappa}^{(1)}$ and $\boldsymbol{\kappa}^{(2)}$ that are both perpendicular to one another and $\boldsymbol{d}$. Thus, the solution to the following system of equations determines a useful projection plane that is reflective of observations j and k together:

$$0 = \boldsymbol{d}'\boldsymbol{\kappa}^{(1)} = \boldsymbol{d}'\boldsymbol{\kappa}^{(2)} = \boldsymbol{\kappa}^{(1)'}\boldsymbol{\kappa}^{(2)}.$$

B Paramaterizing Feedback for Isomap

Tunable components in Isomap include the neighborhood size k and the neighborhood-distance matrix D_e. We opt to do the latter.

Users have the option to provide their cognitive feedback by moving one or more points apart. Suppose a user opted to drag a set of points in a visualization which we denote as S_a away from another set which we denote as S_b. Within Section 5.3.2, the sets S_a and S_b each contain only one observation; i.e., $\mathrm{S}_a = \boldsymbol{x_l}$ and $\mathrm{S}_b = \boldsymbol{x_m}$. Let n_a and n_b represent the cardinality of each set respectively.

When users inject the cognitive feedback that observations should appear farther apart in the visualizations than they appear, the Isomap has underestimated the geodesic distance between the observations. This means that the algorithm used to select the shortest paths between S_a and S_b has selected one or more edges within the paths that short circuit the manifold. Thus, we must interpret the feedback so that we a) identify the short circuiting edge(s) and b) transform (i.e., lengthen) the identified edges. The point is to increase the measure for the lengths of the short circuiting edges so that they seem long relative to all neighboring edges. In turn, any algorithm that identifies the shortest paths between observations will avoid the short circuiting edge(s) (because they are long). We store both the scaled and non-scaled edge lengths in an $n \times n$ matrix which we denote as D_F.

Identify the short circuiting edge(s). We start by identifying the shortest paths $\mathcal{P}_{\mathrm{S}_a,\mathrm{S}_b}$ between S_a and S_b; there are $n_a n_b$ paths in $\mathcal{P}_{\mathrm{S}_a,\mathrm{S}_b}$. Under a sufficiently high and uniform data density on the manifold, the largest hop across all $n_a n_b$ paths corresponds likely to a short circuited edge between set S_a and S_b. Thus, we flag the largest edge and denote it's length by d_e^{max}.

Re-scale the measure of the short circuiting edge(s). For this step, we aim to rescale every edge in D_e that is of a comparable length to d_e^{max}. To determine comparability, we assign a weight to every edge that depends upon it's current length and the average edge length in D_e. Let $\mathrm{w}(\boldsymbol{x_i}, \boldsymbol{x_j})$ represent the weight we assign to the edge the connects $\boldsymbol{x_i}$ and $\boldsymbol{x_j}$, and let d_e^{avg} represent the average hop length in D_e.

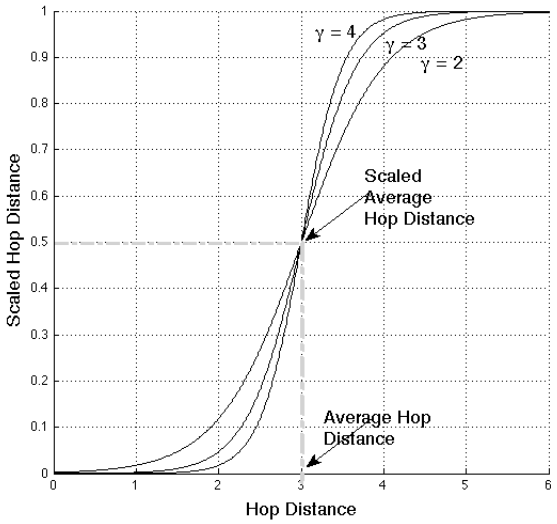

Figure 12: Sigmoid scaling function for hop distances for *average hop distance* equal to 3 and scaling rate, $\gamma = 2, 3$ and 4. *Average hop distance* is always scaled to 0.5.

We use the following sigmoid function to define $\mathrm{w}(\boldsymbol{x_i}, \boldsymbol{x_j})$,

$$\mathrm{w}(\boldsymbol{x_i}, \boldsymbol{x_j}) = 1 - \frac{1}{1 + e^{-\gamma(\mathrm{d}_e^{avg} - \mathrm{d}_e^h(\boldsymbol{x_i}, \boldsymbol{x_j}))}} \tag{9}$$

where γ controls the rate at which we penalize large deviations from d_e^{avg}. This parameter is selected by the VA tool developer and depends upon the sparsity of the data graph structure (e.g., Figure 10b). As a rule of thumb, dense graphs need small values for γ and sparse graphs need high values for γ. Figure 12 shows the impact values for γ may have on $\mathrm{w}(\boldsymbol{x_i}, \boldsymbol{x_j})$. The specific function we chose to define $\mathrm{w}(\boldsymbol{x_i}, \boldsymbol{x_j})$ is arbitrary with the exception that it is has useful properties including, $\mathrm{w}(\boldsymbol{x_i}, \boldsymbol{x_j}) \in [0, 1]$ and $\mathrm{w}(\boldsymbol{x_i}, \boldsymbol{x_j}) = 0.5$ when $\mathrm{d}_e^h(\boldsymbol{x_i}, \boldsymbol{x_j}) = \mathrm{d}_e^{avg}$. Also, small and large edges have weights near zero and one respectively.

Let w^{sd} represent the empirical standard deviation across all weights, and let w^{max} represent the weight that we assign the edge with distance d_e^{max}. We scale the lengths for all edges with weights greater or equal to $\mathrm{w}^{max} - \mathrm{w}^{sd}$. If we denote the scaled distance between observations by $\mathrm{d}_s^h(\boldsymbol{x_i}, \boldsymbol{x_j})$, we have

$$\mathrm{d}_s^h(\boldsymbol{x_i}, \boldsymbol{x_j}) = \begin{cases} \alpha \mathrm{d}_e^h(\boldsymbol{x_i}, \boldsymbol{x_j}) & \text{if } \mathrm{w}(\boldsymbol{x_i}, \boldsymbol{x_j}) \geq \mathrm{w}^{max} - \mathrm{w}^{sd} \\ \mathrm{d}_e^h(\boldsymbol{x_i}, \boldsymbol{x_j}) & \text{otherwise,} \end{cases} \tag{10}$$

where

$$\alpha = \left(1 + \frac{log(\mathcal{M})\mathrm{w}^{max}}{\mathcal{B}}\right)^{\mathcal{B}}, \tag{11}$$

and $\mathcal{M}$ and $\mathcal{B}$ control respectively the magnitude of α and the extent to which α mimics an exponential increase ($e^{\mathrm{W}^{max}} = \lim_{\mathcal{B}} \to_\infty (1 + \frac{\mathrm{W}^{max}}{\mathcal{B}})^{\mathcal{B}}$).

Similar to Equation (9), Equation (11) is also arbitrary with the exception that it controls erroneous cognitive feedback. For example, suppose users provide feedback that is in conflict with the true manifold; e.g., users move observations apart in a visualization when they are actually adjacent on the manifold. Edges of adjacent points on the manifold will be smaller than average so that $\mathrm{w}^{max} \approx 0$. In turn, $\alpha \approx 1$ and $\mathrm{d}_s^h(\boldsymbol{x_i}, \boldsymbol{x_j}) \approx \mathrm{d}_e^h(\boldsymbol{x_i}, \boldsymbol{x_j})$ for all edges.

Define $\mathbf{D}_F$ We represent the parameterized feedback F_p in the form of an $n \times n$ distance matrix, D_F where

$$\mathrm{D}_F[i, j] = \mathrm{d}_s^h(\boldsymbol{x_i}, \boldsymbol{x_j})$$

. This matrix equals D_e for every element excluding those with weights greater than or equal to $\mathrm{w}^{max} - \mathrm{w}^{sd}$.

References

Andrews, N. O. and Fox, E. A. (2007), "Recent developments in document clustering," *Technical report, Computer Science, Virginia Tech*.

Buja, A., Swayne, D. F., Littman, M. L., Dean, N., and Hoffman, H. (1998), "XGvis: Interactive Data Visualization with Multidimensional Scaling," *Journal of Computational and Graphical Statistics*, 5, 78–99.

Card, S., Mackinlay, J., and Shneiderman, B. (1999), *Information Visualization: Using Vision to Think*, Morgan Kaufmann, San Francisco.

Carroll, J. D. and Chang, J. J. (1970), "Analysis of individual differences in multidimensional scaling via an N-way generalization of Eckart-Young decomposition," *Psychometrika*, 35, 238–319.

House, L., Leman, S. C., and Han, C. (2010), "Bayesian Visual Analytics: BaVA," *Technical report, Department of Statistics, Virginia Tech*.

Jafri, R. and Arabnia, H. R. (2009), "A Survey of Face Recognition Techniques," *Journal of Information Processing Systems*, 5, 41–68.

Jeong, D. H., Ziemkiewicz, C., Fisher, B., Ribarsky, W., and Chang, R. (2009), "iPCA: An Interactive System for PCA-based Visual Analytics," *Computer Graphics Forum*, 28, 767–774.

Jolliffe, I. (2005), *Principal Component Analysis*, Springer-Verlag, New York.

Kruskal, J. B. and Wish, M. (1978), "Multidimensional Scaling," *Sage University Paper series on Quantitative Application in the Social Sciences*, 48, 07–011.

Kumar, V., Grama, A., Gupta, A., and Karypis, G. (1994), *Introduction to Parallel Computing: Design and Analysis of Algorithms*, Benjamin Cummings, Redwood City, CA.

Lederberg, J. (1989), *Excitement and Fascination of Science, Twelve-Step Process for Scientific Experiments: Epicycles of Scientific Discovery*, Annual Reviews, Inc., Palo Alto, California.

Mordecai, A. (1976), *Nonlinear Programming: Analysis and Methods*, Prentice-Hall, New Jersey.

Nilsson, J., Fioretos, T., Höglund, M., and Fontes, M. (2004), "Approximate geodesic distances reveal biologically relevant structures in microarray data," *Bioinformatics*, 20, 874–880.

Pak Chung, W., Hetzler, B., Posse, C., Whiting, M., Havre, S., Cramer, N., Anuj, S., Singhal, M., Turner, A., and Thomas, J. (2004), "IN-SPIRE InfoVis 2004 Contest Entry," *Proceedings of the IEEE Symposium on Information Visualization (INFOVIS'04)*.

Pearson, F. (1990), *Map Projections: Theory and Applications*, CRC Press, Boca Raton, FL.

Pike, W. A., Stasko, J., Chang, R., and OConnell, T. A. (2009), "The science of interaction," *Information Visualization*, 5, 78–99.

Pirolli, P. and Card, S. (2005), "Sensemaking Processes of Intelligence Analysts and Possible Leverage Points as Identified Through Cognitive Task Analysis," *Proceddings of the 2005 International Conference on Intelligence Analysis*.

Schiffman, S. S., Reynolds, M. L., and Young, F. W. (1981), *Introduction to Multidimensional Scaling: Theory, Methods, and Applications*, Academic Press, New York.

Tenenbaum, J. B. and V. de Silva, J. C. L. (2000), "A Global Geometric Framework for Nonlinear Dimensionality Reduction," *Science*, 290, 2319–2323.

Thomas, J. and Cook, K. (2005), *Illuminating the Path*, National Visualizations and Analytics Center.

Torgerson, W. S. (1958), *Theory and Methods of Scaling*, John Wiley, New York.