Bayesian Visual Analytics: Interactive Visualization for High-Dimensional Data

Chao Han

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Statistics

Scotland C Leman, Co-chair
Leanna L House, Co-chair
Eric P Smith
Chris L North

November 14, 2012
Blacksburg, Virginia

Keywords: Visual Analytics, Bayesian Methods, Dimension Reduction, Human-computer Interaction

(ABSTRACT)

In light of advancements made in data collection techniques over the past two decades, data mining has become common practice to summarize large, high dimensional datasets, in hopes of discovering noteworthy data structures. However, one concern is that most data mining approaches rely upon strict criteria that may mask information in data that analysts may find useful. We propose a new approach called Bayesian Visual Analytics (BaVA) which merges Bayesian Statistics with Visual Analytics to address this concern. The BaVA framework enables experts to interact with the data and the feature discovery tools by modeling the sense-making" process using Bayesian Sequential Updating. In this paper, we use BaVA idea to enhance high dimensional visualization techniques such as Probabilistic PCA (PPCA). However, for real-world datasets, important structures can be arbitrarily complex and a single data projection such as PPCA technique may fail to provide useful insights. One way for visualizing such a dataset is to characterize it by a mixture of local models. For example, Tipping and Bishop [Tipping and Bishop, 1999a] developed an algorithm called Mixture Probabilistic PCA (MPPCA) that extends PCA to visualize data via a mixture of projectors. Based on MPPCA, we developped a new visualization algorithm called Covariance-Guided MPPCA which group similar covariance structured clusters together to provide more meaningful and cleaner visualizations. Another way to visualize a very complex dataset is using nonlinear projection methods such as the Generative Topographic Mapping algorithm(GTM). We developped an interactive version of GTM to discover interesting local data structures. We demonstrate the performance of our approaches using both synthetic and real dataset and compare our algorithms with existing ones.

Acknowledgments

I owe my gratitude to many people who have made this dissertation possible.

My deepest gratitude is to my advisors, Dr. Scotland Leman and Dr. Leanna House. They are always supportive and helped me overcome many difficult times. I had enough freedom to explore the questions by myself and also received a lot of good advise from them. They took good care of me academically and personally. Scotland's long discussion with me greatly helped solve problems. Leanna spent a lot of time teaching me how to write and talk to me just like my mom.

Dr. Smith is very kind and gave me constructive suggestions about my research. Dr. North is also an important contributer to the BaVA project. All the other group members, Alex Endert and Dipanyan Miti were very pleasent to work with and I learned a lot from them.

Most importantly, none of this would have been possible without the love and patience of my family.

Finally, I appreciate the financial support from the National Science Foundation, Computer and Communications Foundations; 0937071.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

With the steady improvement of data storage devices and collection technologies, such as sensors and monitoring systems, massive amounts of data are available daily. It has been estimated by researchers at the University of California, Berkeley that about 5 million terabytes of digital data is produced annually worldwide [Lyman and Varian, 2003]. Typical data sources could be purchases at grocery stores, data from World Wide Web, credit card transactions, phone call records or gene microarray data. Unfortunately, data are being generated much faster than can be processed and digested.

Methods in Data Mining (DM) have shown the most promise and success in processing large quantities of data. DM is a broad analytical approach that can extract implicit, previously unknown, and potentially useful information from large multidimensional datasets. Often DM summaries of data reveal hidden data structures that can be useful for subsequent, formal statistical analysis. However, often, DM summaries are hard to interpret, thus there is a need to generate more intuitive and understandable ways to communicate information in data. Visualizations offer an efficient way to do so. Human brains are good at spotting

patterns in visualization [Wills, 2010]. Thus, good visual presentations of data allow analysts to apply their perceptual abilities and make sense of the data easily.

1.1 Data Visualization

Data visualization has been used and studied in different fields, such as, statistics and computer science. In statistics, the term Exploratory Data Analysis (EDA) was first introduced by Tukey in 1977 [Tukey, 1977], with the goal to gain insight of datasets without formal hypothesis testing. In many cases, graphical forms of standard EDA offer a means to check underlying modeling or hypothesis testing assumptions. For example, residual plots, QQ plots, boxplots and histogram enable analysts to assess independence of observations, normality, and the presence of outliers. In each of these data displays, there is a preconceived idea of what to expect within the data. Not much emphasis is placed on the knowledge discovery potential of EDA [Gelman, 2011a].

EDA, statistical modeling, and visualization are often treated separately [Gelman, 2011b]. For example, many EDA techniques are graphical in nature with only a few quantitative summaries, and "modelers tend to think of graphics as a cute toy for exploring raw data" [Gelman, 2011a]. However, data exploration, statistical modeling and visualization are closely related and they have more potential to uncover hidden structures in data when used together than as separate analytical approaches. As Tukey pointed out (1977) "The greatest value of a picture is when it forces us to notice what we never expected to see" [Tukey, 1977].

The famous Anscombe's quartet dataset [Anscombe, 1973] is a good example for demonstrating the importance of visualization within statistical analysis. Anscombe's quartet is comprised of four datasets that have extremely different structures as shown in Figure 1.1.

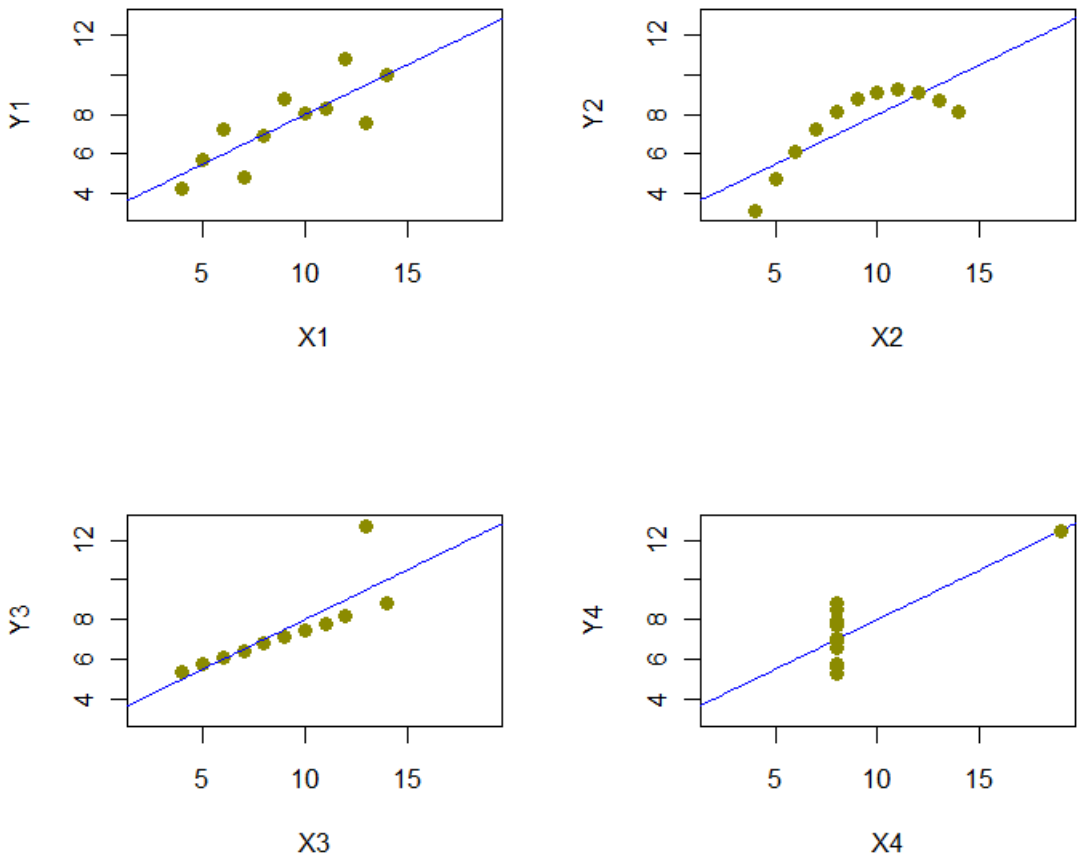

Figure 1.1: Anscombe's quartet dataset.

However, each of the structures have equal statistical measures, including identical means, variances, correlations, and regression summaries (coefficients, coefficient standard errors, regression sum of squares, and residual sum of squares). Without the help of visualization, it would be easy to confuse one group for another based on numerical summaries of the dataset.

DM can be considered, for some applications, EDA for large high-dimensional datasets. Thus, visualization has an equally important role in DM as in EDA. A good representation

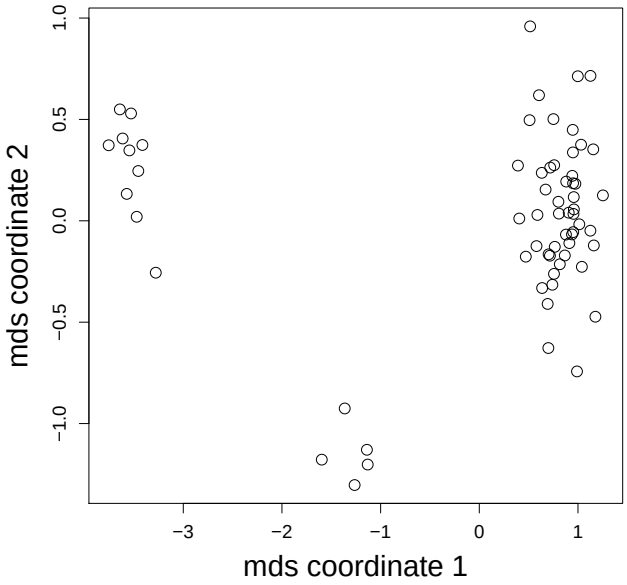

Figure 1.2: The multidimensional scaling (MDS) display for a simulated dataset which has three well separated clusters. Analyst can easily see the cluster number, shape and boundaries from the plot.

of large scale datasets will, undoubtedly, improve the quality and speed of an exploration process. For example, four datasets that consisted of well separated clusters, a visualization of DM summaries would enable analysts to decipher the number, shape and cluster boundaries easily (Figure 1.2).

As seen from the above examples, visualization is a very important tool in the data analysis process. It can help reveal intricate structure in data that cannot be absorbed in any other way [Cleveland, 1993]. However, regardless of the type of analysis, human judgments need to be applied to reach conclusions from a combination of assumptions and evidence. Analysts must interpret and make sense of what they see, and think of subsequent questions to ask. However, doing so can be hard when visualizations are static. Users digest the information

easier through interactive graphs by revealing details progressively [Pirrello, 2010]. Interactive graphs are much more likely than static pictures to generate new hypotheses and results in data discoveries. The reason is that visual representations can become more relevant, focused, and effective for the analysis task through user interactions. For example, JMP [SAS Institute Inc., 2007] has developed many interactive features in its graphs for better data exploration. In Figure 1.3, if we select observations with age 15 or 16 in the histogram for age, other linked graphs (such as the bivariate plot) and spreadsheet show the same selected observations, allowing for finding more interesting patterns.

1.2 Visual Analytics

Data visualization has matured from static graphs to interactive visualizations. In 2001, the US Department of Homeland Security initiated a new field called Visual Analytics (VA) to propose a way of human-computer collaboration through interactive visual interfaces. Visual Analytics combines automated analysis techniques with interactive visualizations to improve learning, reasoning, and decision making based on large and complex datasets [Keim et al., 2008]. The goal of Visual Analytics is to find efficient, effective ways in human-computer interaction for analysts to synthesize massive, noisy datasets, detect unexpected patterns, and obtain insights which may lead to calls for action.

1.2.1 Sensemaking Process

Visual Analytics is not a field devoted to creating displays for data. It is developed to support a whole understanding process that starts from obtaining data and ends with the acquisition of knowledge. The process is often referred to as the "sensemaking process". Sensemaking

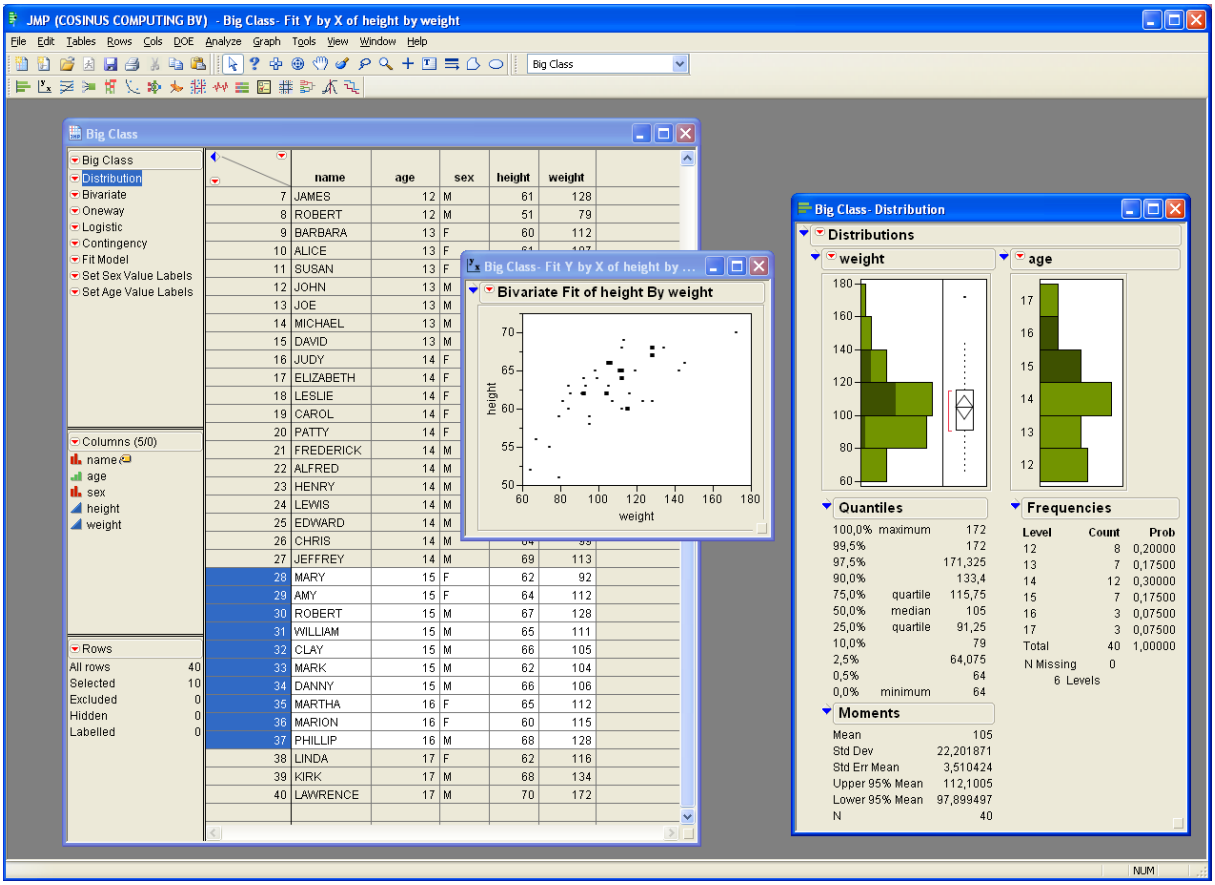

Figure 1.3: In an interactive graph in JMP, selecting an object in one graph results in a selection in a linked graph and spreadsheet [Smit, 2010].

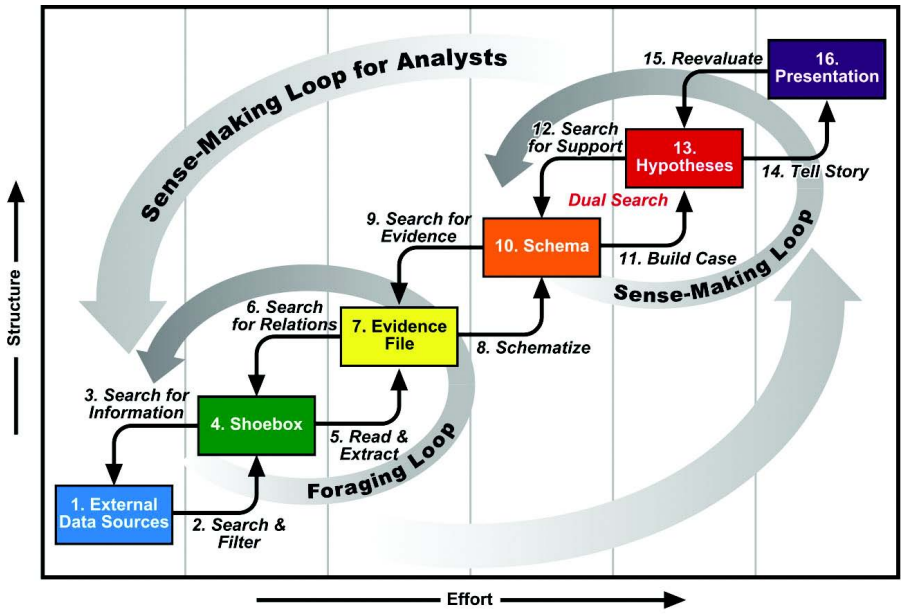

Figure 1.4: Sensemaking Loop from Card et.al 1999.

can be decomposed into small tasks, e.g., formulating a new hypothesis, questioning its validity, elaborating on the hypothesis with detail, comparing alternatives [Dadzie et al., 2009]. Figure 1.4 is an example of a conceptual model that describes sensemaking; we call the model the "sensemaking loop" [Card et al., 1999]. It shows that learning includes a series of tasks, including gathering data, building evidence, creating, and validating hypothesis, representing information to aid analysis, and making decisions. Interacting with a display of data provides a channel to better understand the visualization and quickly test and/or create hypotheses based on previous and/or subsequent iterations.

1.2.2 Interaction Techniques in Visual Analytics

Interaction is a critical component in Visual Analytics. The information visualization community has begun to distinguish between "low-level" interactions and "high-level" interac-

tions [Pike et al., 2009]. The difference in level depends upon the purpose of the interaction. Users apply lower-level interactions, when the goal is to uncover patterns, relationships, and trends in data. In higher-level interactions, the user's goal is to generate understanding and perform analysis [Amar et al., 2005].

Examples of lower-level interaction include: filtering, connecting, projecting, zooming, distorting, and brushing observations in a display. These interaction techniques has been used extensively in the current commercial Visual Analytics systems, such as JIGSAW [Stasko et al., 2008], Wirevis [Chang et al., 2007], Spotfire [Ahlberg, 1996], SeeIT [ADVISOR Solutions Inc., 2007], RESIN [Isenberg and Fisher, 2008] and GreenGrid [Wong et al., 2009]. Low-level interactions raise the problem of "asymmetry in data rates" [Ware, 2000], i.e., the amount of data flowing from systems to users is far greater than from users to systems. The interactions are effectively "photo-shopping" the displays for organizational purposes; they are not encoding human intelligence into the analytical systems creating the display. In turn, users cannot get inside the sensemaking loop via low-level interactions, and it is easy to miss important features in data.

The low-level interactions mostly appear in the systems in which the visualization and analytical algorithms are loosely related. Wong [1999] suggest to tightly couple the two into one data mining tool so that users can rely on higher-level interactions to fulfill the goal of generating understanding. High-level interaction can help guide the analytical system that created the display. For example, many algorithms have a variety of parameters that need to be specified. Human interactions of displays can provide reasonable values for some of these parameters.

There are several Visual Analytics tools that enable interactions in the parameter space. For example, interactive PPCA (iPCA) [Jeong et al., 2009] allows users to manipulate the

data eigen-space or dimensions directly through dials or dialogs of data eigenvalues; and in XGvis [Buja et al., 2008], an interactive MDS visualization, users can change the weight of dimension dissimilarities via sliders. However, direct adjustment of dimension weights or parameters may not be appropriate for very high-dimensional data because of the issue of scalability. More importantly, it is very likely that a deep understanding both of the domain and data analysis tools does not reside in the same user. Parameter changes require users to have clear ideas about the algorithms underlying visualizations. Yet, many analysts do not have the appropriate training. They often understand the data, but not the quantitative methods that created the display.

1.3 Bayesian Updating

In this section, we briefly introduce Bayesian inference and show how Bayesian models may facilitate the sense-making process. The essence of Bayesian inference is a mathematical rule explaining how we can change our existing beliefs in light of new evidence. It allows people to combine new data with their existing knowledge.

Bayes' theorem is fundamental to Bayesian statistics. The application of Bayes' theorem to update beliefs is called Bayesian inference. In the procedure of Bayesian inference, we build a model for data $\boldsymbol{x}$ as $\pi(\boldsymbol{x}|\boldsymbol{\theta})$, where $\boldsymbol{\theta}$ is a set of unknown parameters which characterize the data. The model $\pi(\boldsymbol{x}|\boldsymbol{\theta})$ is also referred to as the likelihood function that reflects the likelihood (probability) of observing data given the parameters $\boldsymbol{\theta}$. Rather than treating $\boldsymbol{\theta}$ as a fixed, unknown constant, a Bayesian statistical model considers a probability distribution on the parameters. We can quantify the uncertainty on the the parameter $\boldsymbol{\theta}$ of a model through a probability distribution, the prior distribution $\pi(\boldsymbol{\theta})$. Then, the inference will be

based on the distribution of parameters $\boldsymbol{\theta}$ conditional on the data $\boldsymbol{x}$, $\pi(\boldsymbol{\theta}|\boldsymbol{x})$, called the posterior distribution. Based on Bayes Theorem, the posterior distribution is defined as:

$$\pi(\boldsymbol{\theta}|\boldsymbol{x}) = \frac{\pi(\boldsymbol{x}|\boldsymbol{\theta}) \times \pi(\boldsymbol{\theta})}{\pi(\boldsymbol{x})}.$$

In the context of statistical learning, the above Bayesian framework describes how a learner could update his or her beliefs in light of data. Often expert knowledge plays a role in selecting a particular type of prior distribution, and incorporate their hypothesis about the parameter $\boldsymbol{\theta}$. After we observe the data, the likelihood serves as a new piece of evidence that the data support the hypothesis. Then the posterior represents the probability of the parameters after taking into account the new piece of information. We can understand this framework as a belief updating procedure by comparison between the prior and posterior distributions. When the prior and posterior are similar, we could infer that the data support the current understanding of $\boldsymbol{\theta}$; and, when they differ, the data suggest a need to change the current degree of belief about our hypothesis to account for evidence.

The posterior probability can be continuously revised as additional information is obtained. Bayesian Sequential Updating provides a theoretical support for the cases when we are dealing with sequential events, whereby new additional information is obtained for a subsequent event, and that new information is used to revise the probability of the initial event. Suppose data arrive sequentially $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(t)}$ at different time points. If the data are independent conditional on $\boldsymbol{\theta}$, the posterior distribution for $\boldsymbol{\theta}$ when $j+1^{st}$ data is available will be:

$$p(\boldsymbol{\theta}|\boldsymbol{x}^{(j+1)}, \boldsymbol{x}^{(j)}) = \frac{p(\boldsymbol{x}^{(j+1)}|\boldsymbol{\theta}, \boldsymbol{x}^{(j)})p(\boldsymbol{\theta}|\boldsymbol{x}^{(j)})}{p(\boldsymbol{x}^{(j+1)}|\boldsymbol{x}^{(j)})} \propto p(\boldsymbol{x}^{(j+1)}|\boldsymbol{\theta})p(\boldsymbol{\theta}|\boldsymbol{x}^{(j)}).$$

Note that each dataset $\boldsymbol{x}^{(j)}$ may be of unique types or forms, which means the user input or feedback about a visualization could also be treated as new data. In this way, experts can assess aspects of the posterior distribution before and after feedback to test new hypotheses

until the inference is make sense. We will show the detail of how we utilize this Bayesian Sequential Updating procedure to incorporate user feedback into our system in Section 2.

1.4 Overview of Dissertation

The remainder of the dissertation is organized as follows. In Section 2, we introduce our Bayesian Visual Analytics (BaVA) framework. In Section 2.3, we show how BaVA can be applied to the Probabilistic PCA (PPCA) model. For complex structured dataset, we rely on Mixture PPCA models Section 3 to seperate the data into mixtures and visualize the data per mixture. With the concerns of the drawbacks of MPPCA, we motivate and explain a new approach called Covariance-Guided MPPCA in Section 4 which can group similar covariance clusters together. In Section 5,a nonlinear projection approach called Generative Topographic Mapping has been investigated and further modified under the framework of Visual to Parametric Interaction, which is a more general framework of BaVA.

Chapter 2

Bayesian Visual Analytics

2.1 Motivation

As seen from Chapter 1, traditional static visualizations tend to display inflexible, deterministic transformations of data that inherently separate data visualization from the visual synthesis process. This means that when a data transformation masks known or intuitive data structure, sense-making may stop. Analysts cannot manipulate displays to inject their domain-specific knowledge into the image nor can they assess the merger of their expert judgment with the data formally. Although tools exist which make the changing of the underlying algorithm parameters possible, they can only summarize the data in a form that is unintuitive to experts (whom may or may not have DM training).

To minimize the human-computer communication gap so that the expert knowledge can be integrated directly into visualizations, we come up with a new idea: Bayesian Visual Analytics (BaVA). In BaVA, interactions with *parameters* take place in the *visual* domain. The scheme can quantitatively interpret user's intent based on user's low-level interactions

and adjust the underlying model parameters. Based on the adjustments, a new data display is generated. BaVA visualizations shield domain experts from the display-generating algorithms.

2.2 The BaVA Process

BaVA combines Visual Analytics with Bayesian statistics to foster learning from data. It is inspired by the sensemaking process which is mentioned in Section 1.2.1. We find that the sensemaking loop parallels the Bayesian operating system quite well. Thus, we re-develop the process into Bayesian statistical sensemaking process (Figure 2.1) which has four interactive phases: 1) Collecting data $\boldsymbol{x}$, 2) Formulating a statistical model, $\mathrm{P}(\boldsymbol{x}|\boldsymbol{\theta})$, where $\boldsymbol{\theta}$ denotes the parameters, 3) Deriving a distribution for hypotheses concerning $\boldsymbol{\theta}$, $\mathrm{P}(\boldsymbol{\theta}|\boldsymbol{x})$, which is the posterior distribution of parameter $\boldsymbol{\theta}$ given data $\boldsymbol{x}$, 4) Visualizing the inference, 5) Injecting expert feedback, if needed. In phase 2, the model form $\mathrm{P}(\boldsymbol{x}|\boldsymbol{\theta})$ depends on what dimension reduction algorithm we use and, usually $\boldsymbol{\theta}$ is closely related to the visualization as shown in Section 2.3.1 and Section 5.2. If we denote the visual display as v, phase 4) creates v which will reflect features in the data given estimates for $\boldsymbol{\theta}$. Since we characterize the problem under a Bayesian paradigm, we suggest using the Maximum A Posteriori (MAP) estimate $\hat{\boldsymbol{\theta}}$; i.e., set $\boldsymbol{\theta} = \hat{\boldsymbol{\theta}}$ and define $v = g(\hat{\boldsymbol{\theta}})$. If v contradicts expert judgment, information can be injected into the system during phase 5) as feedback f. In Section 2.3.2, we show how to parameterize f and derive the distribution $P(f|\boldsymbol{\theta})$. In turn, phases 3, 4, 5 may update and repeat given new information f.

Most visual perception research is directed to understanding and using information in displays when the displays are static [Thomas and Cook, 2005]. This means that most visual-

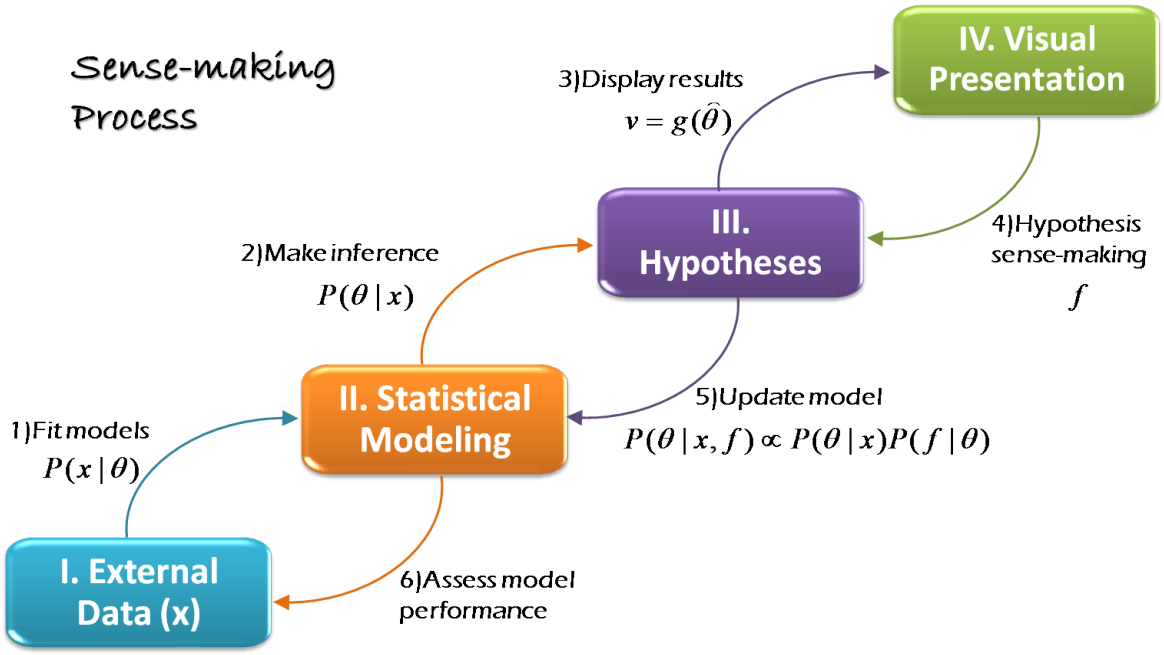

Figure 2.1: Bayesian statistical sensemaking process.

ization methods will stop at phase 4, and cannot adjust when they do not comply with user expertise. Figure 2.2 illustrates this case for a simplified example. Consider a hypothetical dataset that consists of measures on features of several famous movies. The features include year, amounts of action, technology, comedy and drama respectively, and orientation to children. Figure 2.2 shows the PCA projection of the dataset. Apparently, the first principle component mainly accounts for year. Recent movies (e.g. Valentine's Day, 2010) fall on the right side, while on the other side, old movies (e.g., Star Wars V, 1980) fall. Additionally, on the top part of the plot are action movies (e.g., The Terminator, 1984) and the bottom part are those with little action (e.g., Forrest Gump, 1994). Suppose if a user of the data is a mom who cares about whether movies are appropriate for children. She knows that Kung Fu Panda (2008) is okay, so she refers to the plot for comparable movies. However, based on proximity, the plot suggests that Kill Bill 2 (2004) is comparable to Kung Fu Panda (2008).

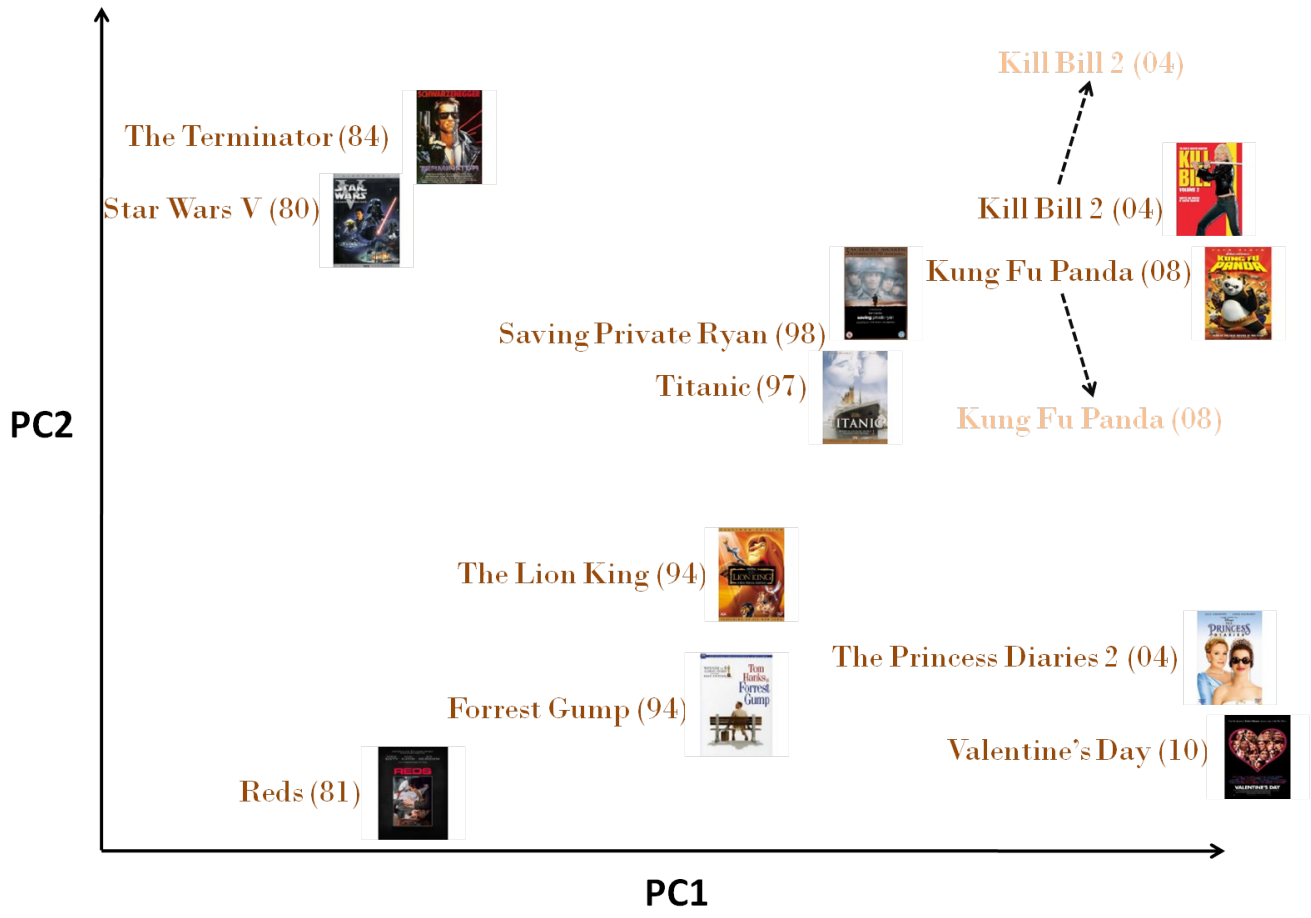

Figure 2.2: PCA display for the movie dataset.

The plot is displaying the films based on features (year and action) that are not of interest to the mom.

In the BaVA paradigm, we allow analysts (e.g., moms) to incorporate their expert feedback by adjusting the data points directly in the display. In the movie data example, the mom can drag Kung Fu Panda and Kill Bill 2 apart directly in the plot. There is no need for her to know the math behind to explore the data structure. Our algorithm will treat her actions as data that can update the model in phase 3), and subsequently adjust the visualization in phase 4). Figure 2.3 shows an adjusted display based on the mom's action of moving movies

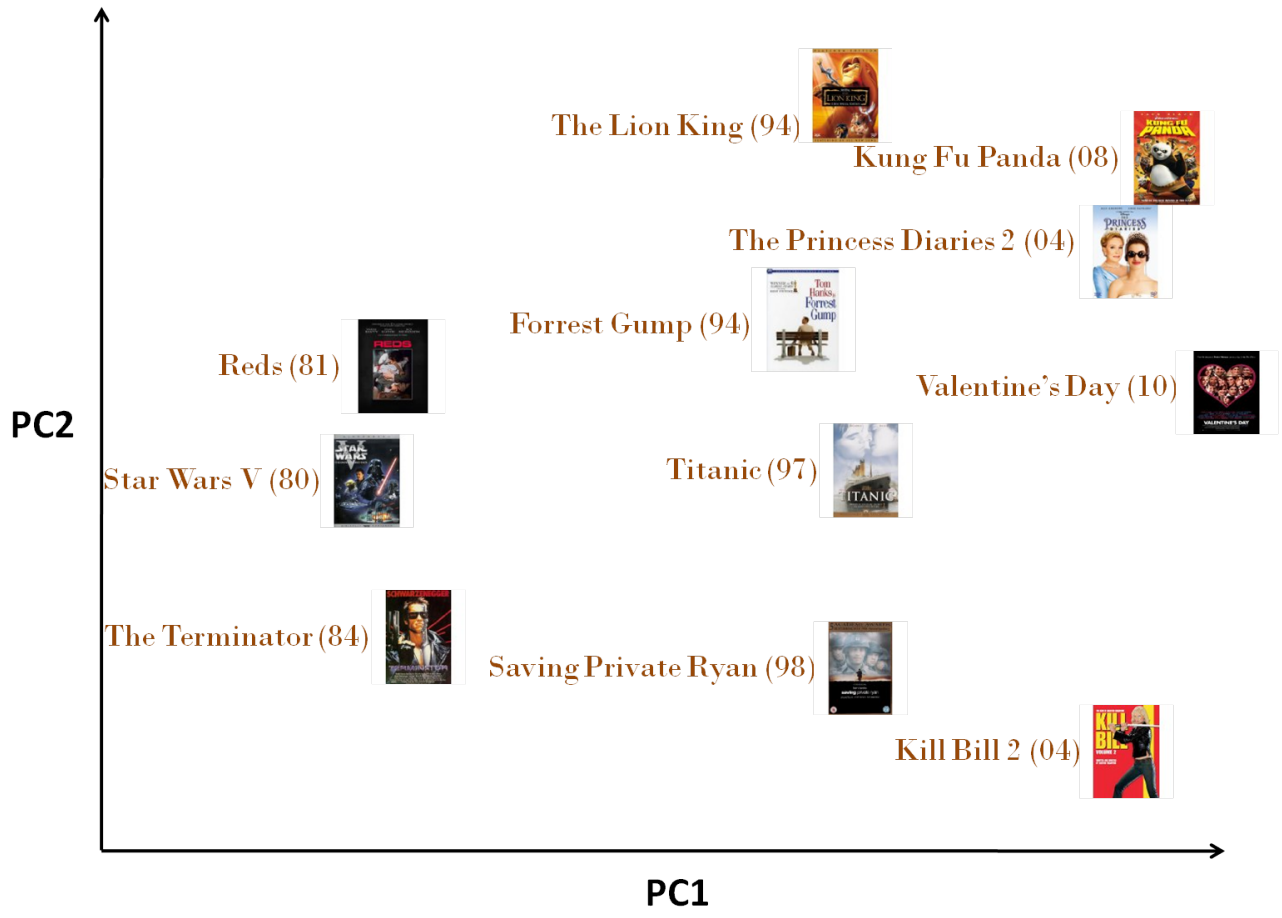

Figure 2.3: Updated BaVA display for the movie dataset.

Kung Fu Panda and Kill Bill 2 apart. In this display, the second principal component mainly takes care of the orientation to children and separates the two movies. Also, Kung Fu Panda and The Lion King are close to each other.

Mathematically, we decompose the user data, i.e., feedback f, into cognitive feedback $f^{(c)}$ and parametric feedback $f^{(p)}$, where $f^{(c)}$ reflects user's expert judgements in the form of manual display adjustments and $f^{(p)}$ is $f^{(c)}$ in a parameterized form. The feedback $f^{(c)}$ is a random variable since there are infinite number of possible movements in the display. Different possibilities rely on the different points choice, moving apart or closer, where to put

the chosen points, to name a few. To incorporate $f^{(c)}$ into the probabilistic model, we need to parameterize $f^{(c)}$ and interpret the adjustment as a function of $\boldsymbol{\theta}$. The parametrization is challenging and context specific. We provide examples in Section 2.3.2. We assume $f^{(p)}$ has distribution $p(f^{(p)}|f^{(c)}, \boldsymbol{\theta})$. With $f^{(c)}$ and $f^{(p)}$ defined, the feedback distribution is:

$$p(f|v, \boldsymbol{\theta}) = p(f^{(p)}, f^{(c)}|v, \boldsymbol{\theta}) = p(f^{(p)}|f^{(c)}, \boldsymbol{\theta})p(f^{(c)}|v).$$

Note that we do not need to know the mathematical form of $p(f^{(c)}|v)$. The parameterized feedback $f^{(p)}$ is a deterministic transformation of $f^{(c)}$ and $f^{(c)}|v$ is independent of $\boldsymbol{\theta}$ and which will not help with inference about $\boldsymbol{\theta}$.

The inclusion of feedback and model updating in phase 3) is accomplished through Bayesian Sequential Updating as mentioned in Section 1.3; when more data or information becomes available after posterior computation, the posterior becomes the next prior. In Bayesian Sequential Updating, each dataset may be of unique types of forms. This means that data in a form of feedback can be used to update the posterior distribution of $\boldsymbol{\theta}$, even that f and $\boldsymbol{x}$ are from different sources. The updating step can be formulated as follows:

$$\begin{aligned} p(\boldsymbol{\theta}|f, v, \boldsymbol{x}) &= \frac{p(f, v|\boldsymbol{\theta}, \boldsymbol{x})p(\boldsymbol{\theta}|\boldsymbol{x})}{\int p(f, v|\boldsymbol{\theta}, \boldsymbol{x})p(\boldsymbol{\theta}|\boldsymbol{x})\mathrm{d}\boldsymbol{\theta}} \\ &\propto p(f|v, \boldsymbol{\theta})p(v|\boldsymbol{\theta})p(\boldsymbol{\theta}|\boldsymbol{x}) \\ &= p(f^{(p)}|f^{(c)}, \boldsymbol{\theta})p(f^{(c)}|v)p(v|\boldsymbol{\theta})p(\boldsymbol{\theta}|\boldsymbol{x}). \end{aligned}$$

Since $f^{(c)}|v$ is independent of $\boldsymbol{\theta}$ and v is a deterministic transformation of $\boldsymbol{\theta}$,

$$p(\boldsymbol{\theta}|f, v, \boldsymbol{x}) \propto p(f^{(p)}|f^{(c)}, \boldsymbol{\theta})p(\boldsymbol{\theta}|\boldsymbol{x}).$$

After using Bayesian Sequential Updating to adjust the underlying model, the model is updated with user feedback to obtain inferences using both $\boldsymbol{x}$ and f. Subsequently, the visualization is updated and the process is repeated until the user obtain a mature solution.

For example, if the process is repeated k times, the visual inferences are based on x and the sequence of feedback $f_1, \ldots, f_k$. Now, the analysis facilitated by BaVA forms a complete loop, connecting both quantitative and qualitative aspects of data analysis.

2.3 The BaVA Process for Probabilistic PCA

Principal Component Analysis (PCA) is one of the most widely used dimension reduction and data visualization tools. PCA finds a linear transformation that reduces a high-dimensional vector to a low-dimensional vector by exploiting the data covariance. PCA de-correlates the resulting components by finding the first direction of maximum variance of the given high-dimensional vectors. The remaining components are mutually orthogonal and maximize the remaining variances subject to the orthogonal condition. The lower-order components are discarded to achieve dimension reduction. Sirovich and Kirby [1987] showed that PCA is an optimal compression scheme in the sense that for a given set of training vectors and any given level of compression rate, PCA minimizes the mean squared error between the original images and their reconstructions. Because of this low reconstruction error property, PCA has gained a lot of success in the area of image processing, with applications such as face recognition [Gottumukkal and Asari, 2004, Conde et al., 2003, Imran et al., 2005, Du and Fowler, 2008].

From a classification perspective, the goal is to learn low-dimensional structure that contains information which discriminates underlying classes. However, PCA may not yield discriminating features if the variance it tries to maximize is not oriented in a direction which aligns with class discrimination. Consider the following example. We simulate a three dimensional dataset that contains three pancake-shaped clusters denoted by different shapes as shown in

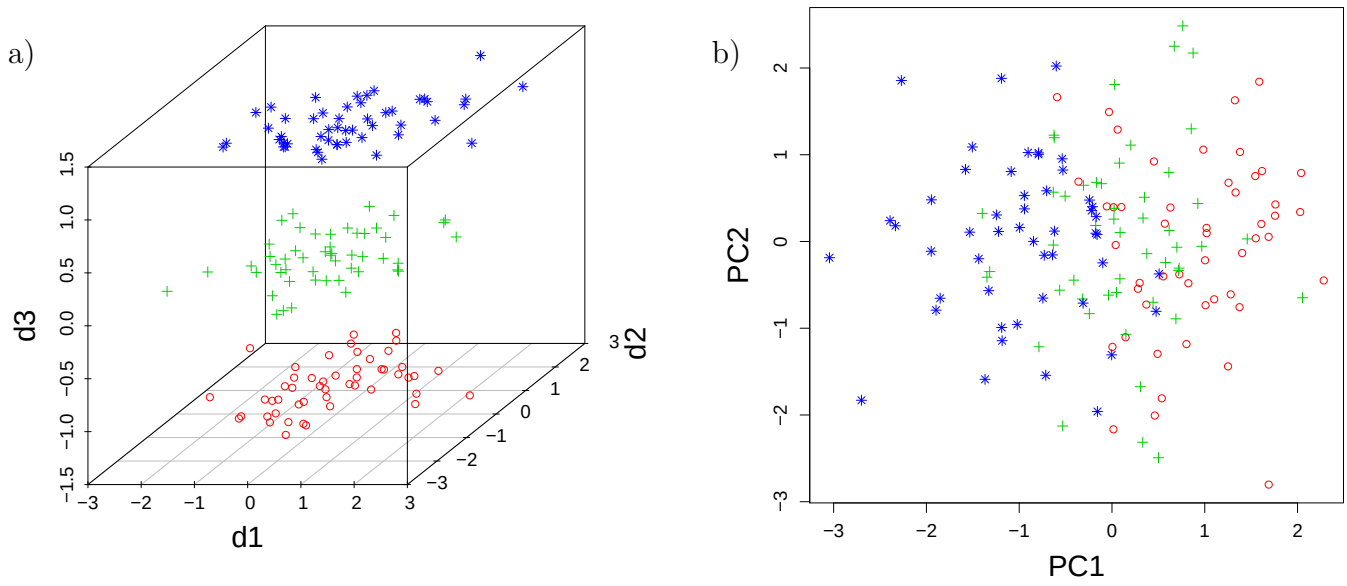

Figure 2.4: a) The 3D simulated data. Different groups are denoted by different marks. b) PCA projection of the simulated data in a).

Figure 2.4a). Figure 2.4a) shows clusters where the within cluster variance is larger than the between cluster variance. Since PCA will project through the direction that maximizes the data variance, all three clusters overlap on each other as shown in Figure 2.4b). Additionally, PCA, in its native forms, has no ability to adjust the display, despite knowing about the presence of the clusters.

In this section, we transform PCA using the BaVA paradigm to enable experts to inject domain knowledge into the image to help guide the projection and find data structures, including the number of clusters and various cluster characteristics (e.g., size, shape). As mentioned in Section 2, in order to use Bayesian sequential updating and parameterize the feedback, we need to model the data and parameters probabilistically, but traditional PCA is a deterministic approach. Thus, rather than the deterministic version of PCA, we employ Probabilistic PCA (PPCA) which was proposed by Tipping and Bishop [1999b].

2.3.1 Probabilistic PCA

Formally, the PPCA model is a factor model [Bishop, 1999] in that data $\boldsymbol{x}_i$ (p dimensional) is conditional on a reduced q dimensional latent factor $\boldsymbol{r}_i$ where $q < p$, and $\boldsymbol{x}_i|\boldsymbol{r}_i$ is independent of $\boldsymbol{x}_{i'}|\boldsymbol{r}_{i'}$ ($i \neq i'$). The PPCA model is based on a linear mapping between $\boldsymbol{r}_i$ and $\boldsymbol{x}_i$ with added Gaussian noise ε_i, such that

$$\boldsymbol{x}_i = \mathsf{W}\boldsymbol{r}_i + \boldsymbol{\mu} + \varepsilon_i, \qquad \varepsilon_i \sim N(0, \sigma^2\mathsf{I}), \tag{2.1}$$

where $\boldsymbol{\mu}$ is a p-dimentional mean vector; W is a $p \times q$ matrix that contains the factor loadings; usually we take $q = 2$ (for visualization); and I is a $p \times p$ identity matrix. The distribution assumption for the error term ε_i makes model (Equation (2.1)) a special case of a standard factor analysis in that PPCA relies on an isotropic variance structure ($\sigma^2\mathsf{I}$) for ε_i.

We model a latent factor $\boldsymbol{r}_i$ using a normal prior $p(\boldsymbol{r}_i) \sim N(0, \mathsf{I})$. The conditional posterior distribution for $\boldsymbol{r}_i$ is

$$p(\boldsymbol{r}_i|\boldsymbol{x}_i, \mathsf{W}, \sigma^2) \sim N((\mathsf{W}'\mathsf{W} + \sigma^2\mathsf{I})^{-1}\mathsf{W}'(\boldsymbol{x}_i - \boldsymbol{\mu}), \Sigma_r). \tag{2.2}$$

Based on Equation (2.2), a natural reduced-dimensional representation of the data $\boldsymbol{x}$ is $\hat{\boldsymbol{r}}_i$, the posterior mean of $\boldsymbol{r}_i$,

$$\hat{\boldsymbol{r}}_i = (\hat{\mathsf{W}}'\hat{\mathsf{W}} + \hat{\sigma}^2\mathsf{I})^{-1}\hat{\mathsf{W}}'(\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}), \tag{2.3}$$

where $\hat{\mathsf{W}}$, $\hat{\sigma}^2$ and $\hat{\boldsymbol{\mu}}$ represent the Maximum Likelihood Estimators (MLE's) of the log-likelihood $\sum_{i=1}^{N} \ln\{p(\boldsymbol{x}_i)\}$. Notice that Equation (2.3) can also be written as a projection times the centered version of data, $P_r(\boldsymbol{x}_i - \hat{\boldsymbol{\mu}})$, and $P_r = (\hat{\mathsf{W}}'\hat{\mathsf{W}} + \hat{\sigma}^2\mathsf{I})^{-1}\hat{\mathsf{W}}'$.

Although, it seems that this model looks like factor analysis except for isotropic noise, it was shown in Tipping and Bishop [1999b] that the columns of $\hat{\mathsf{W}}$ span the principal subspace of the data. Thus, the covariance properties of the Probabilistic PCA model are more similar to

traditional PCA than to factor analysis; factor analysis is invariant under component-wise rescaling (transformation of the data vectors by multiplication with an diagonal matrix), while PPCA and PCA are invariant under coordinate rotations (transformation of the data vectors by multiplication with an orthogonal matrix) [Tipping and Bishop, 1999a]. This probabilistic version of PCA makes it possible to quantify uncertainties in data, perform model comparisons, handle missing data, and model discrete data (by changing the underlying factor model to a discrete analogy).

2.3.2 BaVA-PPCA

Based on the data model and the prior assignment for the latent factor, we can derive the marginal distribution of $\boldsymbol{x}$ as

$$p(\boldsymbol{x}_i|\mathsf{W},\sigma^2,\boldsymbol{\mu}) = \int N(\boldsymbol{x}_i|\mathsf{W}\boldsymbol{r}_i+\boldsymbol{\mu},\sigma^2\mathsf{I})N(\boldsymbol{r}_i|0,\mathsf{I})\mathrm{d}\boldsymbol{r} = N(\boldsymbol{\mu},\Sigma_x),$$

where $\Sigma_x = \mathsf{W}\mathsf{W}' + \sigma^2\mathsf{I}$. The correspondence between Σ_x and $P_r = (\hat{\mathsf{W}}'\hat{\mathsf{W}} + \hat{\sigma}^2\mathsf{I})^{-1}\hat{\mathsf{W}}'$ is clear since they both depend on the same set of parameters.

In order to infer Σ_x under prior $p(\Sigma_x) \propto 1$, the posterior distribution follows as an Inverse Wishart (IW) distribution

$$p(\Sigma_x|\boldsymbol{x}) \sim \mathrm{IW}(N\mathsf{S}_x, p, N-p-1),$$

where $\mathrm{IW}(a,b,c) = |a|^{-c/2}|\Sigma_x|^{-(b+c+1)/2}\exp\{-\mathrm{tr}(a\Sigma_x^{-1})/2\}/2^{bc/2}\Gamma_b(c/2)$ ($\Gamma_b(\cdot)$ is a multivariate gamma function), N is the total number of observations, p is the data dimension, and $\mathsf{S}_x = \frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{x}_i-\boldsymbol{\mu})(\boldsymbol{x}_i-\boldsymbol{\mu})'$ is the sample covariance matrix. If we let the parametric feedback, $f^{(p)}$, reflect cognitive changes, $f^{(c)}$, concerning data variance, we can model $f^{(p)}$ using a Wishart distribution ($\mathrm{Wi}(a,b,c) = |f^{(p)}|^{(c-b-1)/2}\exp\{-\mathrm{tr}(f^{(p)}a^{-1})/2\}/(2^{bc/2}\Gamma_b(c/2))$)

as $p(f^{(p)}|f^{(c)}, \Sigma_x) \sim \text{Wi}(\Sigma_x/v, p, v)$, so that the posterior distribution for Σ_x turns out to be

$$p(\Sigma_x|\boldsymbol{x}, f^{(p)}) \sim \text{IW}(N\mathsf{S}_x + vf^{(p)}, p, N + v - p - 1).$$

The MAP estimator is the weighted average of $f^{(p)}$ and S_x, which follows as:

$$\text{MAP}(\Sigma_x|\boldsymbol{x}, f^{(p)}) = \hat{\Sigma}_x = \frac{v}{v+N} f^{(p)} + \frac{N}{v+N}\mathsf{S}_x. \tag{2.4}$$

It is reasonable that we construct $f^{(p)}$ as a semi-definite matrix similar in nature as S_x so that Equation (2.4) is possible to compute. We describe the details about how we quantify feedback in the next section (Section 2.3.3). Also, based on Bayesian sequential updating, the next MAP estimator $\hat{\Sigma}_x$ should be the weighted average of the feedback matrix and the $\hat{\Sigma}_x$ which contributed to the current display:

$$\hat{\Sigma}_x^{(t+1)} = \frac{v}{v+N} f^{(p)} + \frac{N}{v+N}\hat{\Sigma}_x^{(t)}. \tag{2.5}$$

Here the weight $\alpha = \frac{v}{v+N}$ reflects the proportion of the updated projection generated from user feedback with respect to the current view.

In practice, experts may specify feedback $f^{(c)}$ through a rearrangement of data displays and specify a measure for their confidence in the rearrangement. For example, if observations that are expected to be different lay close to each other in a projected view or observations that are expected to be similar are distant in the visualization, we allow experts to either drag them apart or together respectively. Based on α, estimates Σ_x will change according to Equation (2.5), and projection change follows.

To sum up briefly, BaVA-PPCA treats the display manipulation as information related to dimension selection and update the data variance estimation as a linear combination of feedback matrix and sample covariance matrix supported by Bayesian sequential updating. At last, we find the new direction to rotate the data and project.

2.3.3 Quantify Feedback

Parametric feedback $f^{(p)}$ is a deterministic transform of the user's cognitive feedback $f^{(c)}$. We understand the user's cognitive feedback as a way to re-weight the data dimensions to correct the information lost due to dimension reduction.

It is reasonable that we construct $f^{(p)}$ as a semi-definite matrix similar in nature as S_x, since covariance is the primary parameter which determines the visualization. There are two types of data manipulation method: the drag-apart move and the push-together move. We have to design two separate matrices to describe these two move-types. Let S_a and S_t represent the constructed covariance matrices for the apart and together move respectively. We take the final assignment of $f^{(p)}$ as an weighted average of S_a and S_t

$$f^{(p)} = \beta \mathsf{S}_a + (1 - \beta)\mathsf{S}_t,$$

where the weight β depends on the ratio of the inter-point distances for the two data points j and k in the low dimensional display before and after the data manipulation $\tilde{f}$,

$$\tilde{f} = \frac{||\tilde{r_j} - \tilde{r_k}||_2}{||r_j - r_k||_2}.$$

A greater than one $\tilde{f}$ value denotes a drag-apart movement, while a less than one $\tilde{f}$ value corresponds to a push-together movement. We map $\tilde{f}$ to β through function $\beta = 2\pi^{-1} \arctan(\tilde{f})$, so that $\beta \in [0, 1]$.

2.3.3.1 Specifying S_a

In terms of dimension weighting, the drag-apart movement suggests that the variance of the dimensions that are poorly portrayed for those two points in the display is under-estimated

by the current variance matrix. Then, we will add extra weights to them to re-emphasis the importance of those dimensions.

We first find which dimensions are least explained by the current visualization for the two moved points j and k. Let Δ_d represent the discrepancies in dimension d between points j and k, $\Delta_d = d_{j,d} - d_{k,d}$. Let $\Delta_d^{(0)}$ and $\Delta_d^{(P)}$ denotes the marginalized and projected Δ_d respectively, with $\Delta_d^{(0)} = (0, \ldots, 0, \Delta_d, 0 \ldots, 0)'$ and $\Delta_d^{(P)} = P_r \Delta_d^{(0)}$. We use $\Delta_d^{(u)}$ to describe the amount of the discrepancy in dimension d that remains unexplained by the current visualization, where

$$\Delta_d^{(u)} = \Delta_d (1 - \frac{||\Delta_d^{(P)}||_2}{||\Delta_d^{(0)}||_2})$$

. We compute Δ and $\Delta^{(u)}$ for each dimension to get vectors $\Delta = \{\Delta_1, ..., \Delta_p\}$ and $\Delta^{(u)} = \{\Delta_1^{(u)}, ..., \Delta_p^{(u)}\}$. Based on these vectors, we select one of the directions to project the data to be $v^{(u)}$ and define it as

$$v^{(u)} = \frac{\Delta + \Delta^{(u)}}{||\Delta + \Delta^{(u)}||_2}$$

. This direction is along the dimensions which are mostly unexplained by the current visualization.

Since we want to project data into $q = 2$ dimensions. Another direction is needed in addition to $v^{(u)}$ to construct the feedback covariance matrix. Here, we suggest a direction which is orthogonal to $v^{(u)}$ and maximize the variation in the data. Mathematically, we want to find a direction $v^{(o)}$ such that

$$v^{(o)} = \underset{v^{(o)}}{\text{argmax}} \{ \text{Var}[v^{(o)\prime} \boldsymbol{x}] \} \qquad v^{(o)\prime} v^{(o)} = 1 \text{ and } v^{(o)\prime} v^{(u)} = 0.$$

It is easy to show that $v^{(o)}$ corresponds to the eigenvector of S_x with the largest eigenvalue. After we get two projection directions $v^{(o)}$ and $v^{(u)}$, we can define S_a as

$$\mathsf{S}_a = [v^{(o)}, v^{(u)}][v^{(o)}, v^{(u)}]'.$$

2.3.3.2 Specifying S_t

When experts move two points j and k together, the variance in dimensions represented largely in the visualization is over-estimated by the current covariance matrix. Geometrically, if we can project the data in the direction of their discrepancy Δ, we can map the two points to the same location in the low dimensional display, since the discrepancy direction runs through the two points. We just need to find a projection plane which is orthogonal to Δ. We solve for two vectors $\{v^{(1)}, v^{(2)}\}$ that are both orthogonal to each other and to Δ:

$$0 = \Delta' v^{(1)} = \Delta' v^{(2)} = v^{(1)'} v^{(2)}.$$

At last, we construct S_t as

$$\mathsf{S}_t = [v^{(1)}, v^{(2)}][v^{(1)}, v^{(2)}]',$$

2.3.4 Illustration with Simulated Data

Let us revisit the simulated data shown in Figure 2.4 on page 19 and perform an expert-guided analysis using BaVA-PPCA facilitated by GGobi [Buja et al., 2003]. The initial visualization (Figure 2.5a) is generated by plotting the first two dimensions of the posterior mean $\hat{\boldsymbol{r}}$ in Equation (2.3). Notice that the clusters are indistinguishable in the display.

2.3.4.1 Variable Loading Plot

To help understand the contribution of each variable to the PPCA view, a variable loading plot is provided in Figure 2.5b). As the classical PCA loading plot, the relative position of a variable arrow is plotted according to the coefficients of the first two eigenvectors (i.e., the loadings) in PCA. The Principal Components (PC) are linear combinations of the data

variables $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_p$ (p is the number of the data dimensions). For example, $PC_1 = a_{11}\boldsymbol{x}_1 + a_{12}\boldsymbol{x}_2 + \ldots + a_{1p}\boldsymbol{x}_p = \boldsymbol{a}_1' X$, $PC_2 = a_{21}\boldsymbol{x}_1 + a_{22}\boldsymbol{x}_2 + \ldots + a_{2p}\boldsymbol{x}_p = \boldsymbol{a}_2' X$. Then loadings ($\boldsymbol{a}_1'$ and $\boldsymbol{a}_2'$)can be understood as the weights for each original variable when calculating the principal components. Variable loading plot is a scatter plot of the loadings $(\boldsymbol{a}_1', \boldsymbol{a}_2')$. The length of the arrow reflects the degree to which data variables influence principal components. Also, the angle between any two variable arrows measures the correlation between these variables. Consider the fact that PPCA is a modified version of PCA with $\hat{\boldsymbol{r}} = P_r(X - \mu)$ (Equation (2.3)), loadings here become $P_r = (\hat{\mathsf{W}}'\hat{\mathsf{W}} + \hat{\sigma}^2\mathsf{I})^{-1}\hat{\mathsf{W}}'$ in PPCA. In the variable loading plot, the top variables that have the longest arrows or in other words the largest absolute loading values were labeled. For the simulated data, based on the loading plot (Figure 2.5b) corresponding to the first PPCA projection in Figure 2.5a), it is no surprise that we cannot see any clusters. The top loading variables are d_1 and d_2, but the cluster structure is determined by the third dimension.

2.3.4.2 The "Brush" Tool

In BaVA, experts need to provide judgement concerning the relationship between two or more observations. There are cases when such judgements are hard to make for experts. For example, consider document classification. If the number of documents is very large, it is time consuming to read text and hard for experts to decide where or how to inject their judgements. For such cases, we provide a tool called the "Brush" that can help identify candidate points to move. We first use the "Brush" tool to select a subset of n^* observations in a small area. Let vectors $\boldsymbol{H} = H_{ij}\ i, j \in 1, \ldots, n^*$ and $\boldsymbol{L} = L_{ij}\ i, j \in 1, \ldots, n^*$ denote respectively the distance matrix for the selected data points in the high-dimensional and low-dimensional spaces. We standardize each vector by dividing its maximum to farcilitate

a) b)

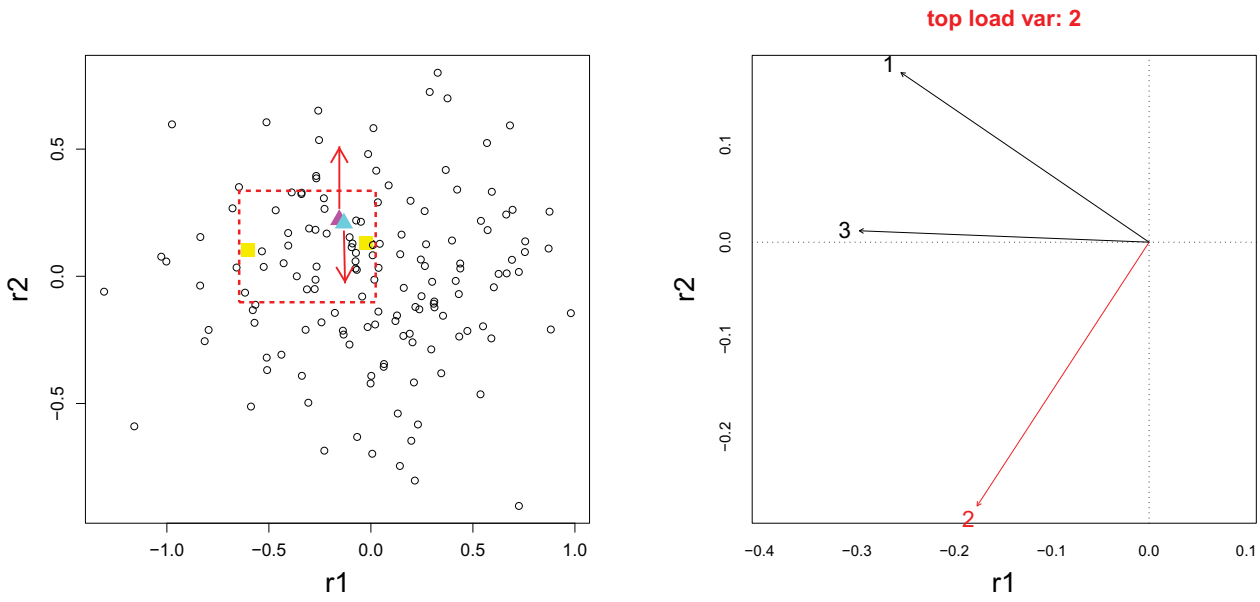

Figure 2.5: a) PPCA projection of the 3D simulated data in Figure 2.4a). The very similar (different) point pair within the brush are marked by '■' ('▲'). b) PPCA variable loading plot for a).

comparison. We then construct a measure $b_{ij} = \boldsymbol{H}_{ij}/\sqrt{\boldsymbol{L}_{ij}}$. A large b_{ij} value means the relative distance between the two points in the high-dimensional space is much larger than what it seems in the low-dimensional display. On the contrary, a small b_{ij} reflect the fact that the large distance between the (i, j) points in the display exaggerates their true distance in the high-dimensional data space. We select a data pair which has $\max b_{ij}$ distance and label them differently to show their difference in the data space. Similarly, we label the data pair which has $\min b_{ij}$ distance in same color to reflect their similarity.

In Figure 2.5a), the square brush selects some observations in a small area and labels two pair of points. Within the brush, the two '■' and '▲' points are respectively the most similar and different pairs in the high dimensional space relative to the low dimensional display. Now experts have the option to either spread the '▲' points or move the '■' points together. Suppose an expert chooses to adjust the '▲' points, (as shown in Figure 2.5a)) and re-project. Figure 2.6 gives the updated displays with weight α ranging from 0 to 1. When $\alpha = 0$, the new display is equivalent to the PPCA projection based on Equation (2.4); when $\alpha = 1$, the new display will purely reflect the user's feedback. In the variable loading plot, for $\alpha = 0.9$ (Figure 2.7), the third dimension shows to be important and a clear clustering structure starts to appear with $\alpha = 0.7$ and higher. Also, the two '▲' points that we dragged are separated in the new display.

2.3.5 Case Study with Education Dataset using BaVA-PPCA

We apply BaVA on an education dataset which is firstly extracted from the 1997 Digest of Education Statistics [Guber, 1999]. This dataset is collected to study the relationship between the academic success (measured by the SAT score) and public school expenditures. To increase the complexity of the dataset, we expanded the data with two additional variables

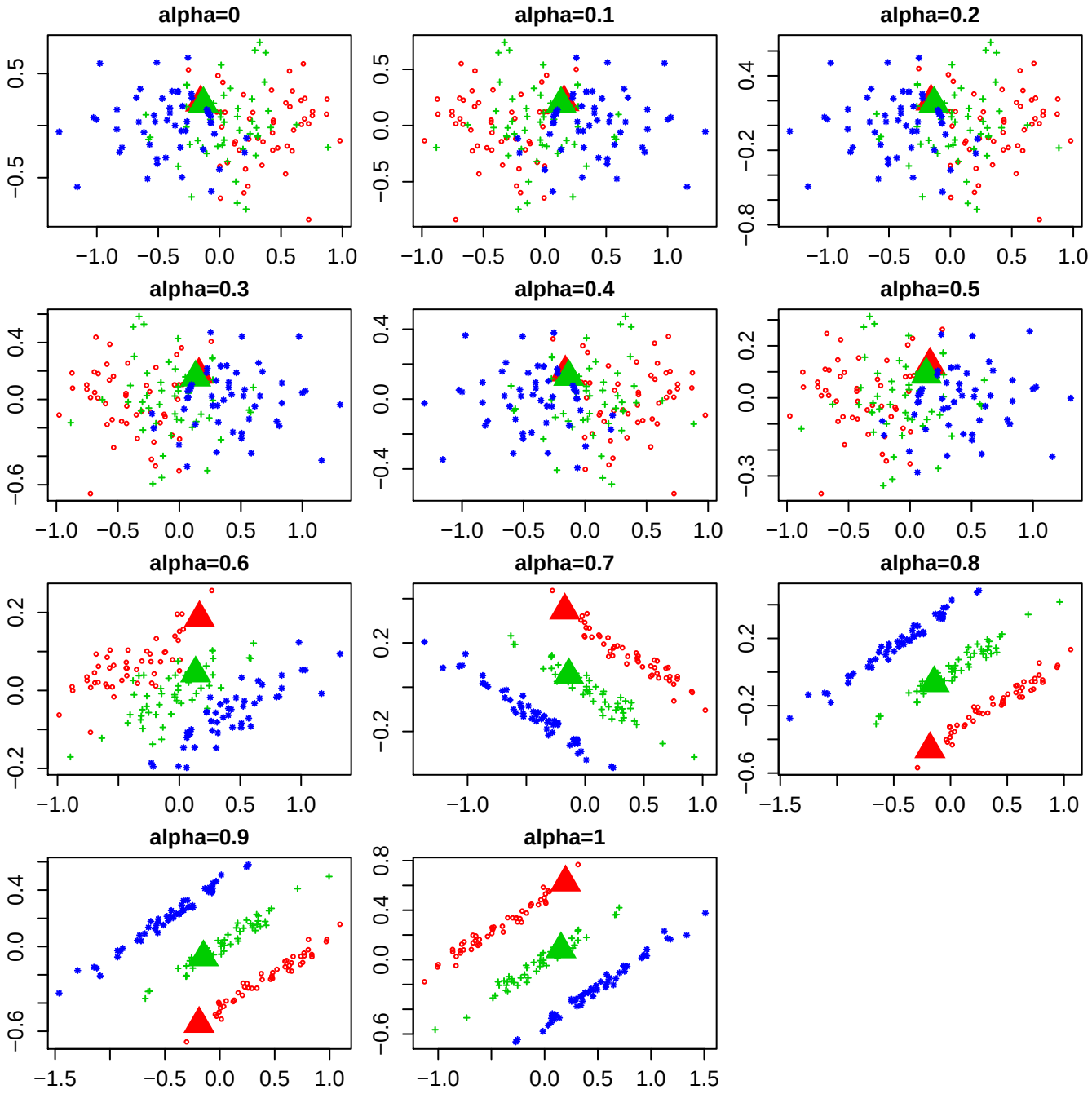

Figure 2.6: Updated plot after moving points apart in Figure 2.5a) with α ranging from 0 to 1. Points are labeled according to the true classification. '▲' denotes the points we moved.

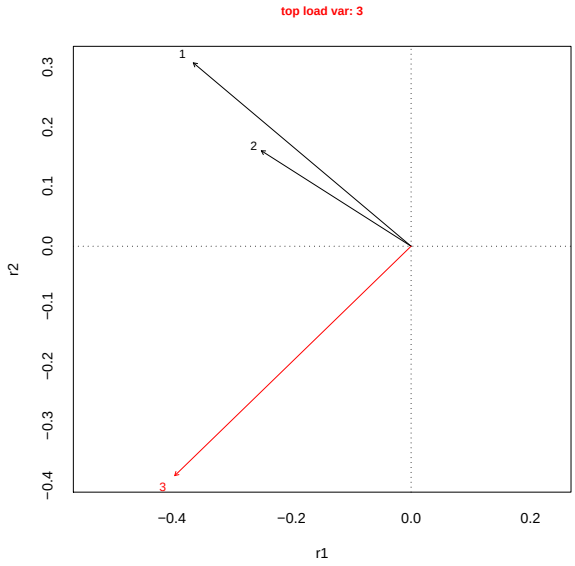

Figure 2.7: Variable loading plot for $\alpha = 0.9$ in Figure 2.6.

which we think might influence the SAT score from the National Center for Education Statistics (www.http:nces.ed.gov). Together, we have seven variables in total for each U.S. state: the average SAT score (SAT); the average annual expenditures per student (EXP); the student/teacher ratio (RAT); the average annual salary for teachers in each state (SAL); the percentage of all eligible students taking the SAT (PER), the number of regular high school graduates (HSG) and the median household income in each state (INC). Our original hypothesis about the data is that the high annual expenditure (EXP) will contribute to high SAT score.

In the initial view of the projected data using PPCA Figure 2.8, we cannot see any clear cluster structure. Similar to the simulated dataset, we apply the brush tool and find two points sit near each other in the PPCA view but actually very different in the high dimensional space. The two observations are Arizona and Minnesota, the values in each dimension are listed below in Table 2.1, and we can see that they are truly different in many variables,

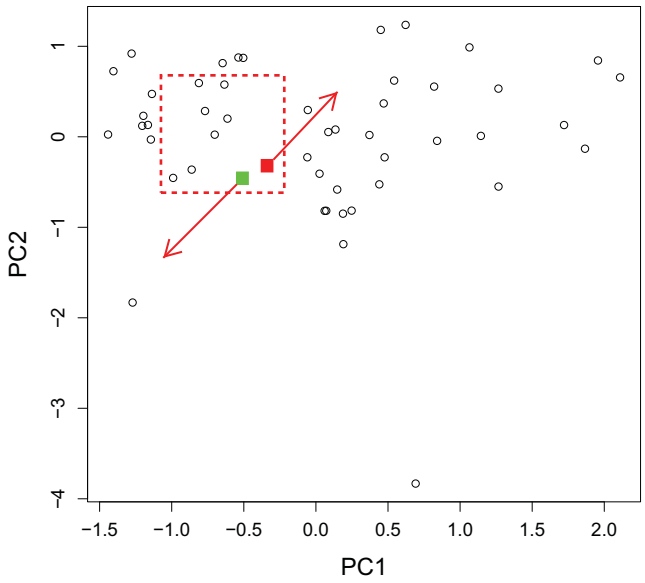

Figure 2.8: The initial PPCA projection of the SAT dataset. Arizona and Minnesota (denoted by squares) are the two dissimilar states selected by the 'brush' tool.

especially in SAT score (SAT), Percent of students taking the SAT (PER) and the number of regular high school graduates (HSG).

We drag the two states apart to get a BaVA-updated view as shown in Figure 2.9. Two clusters can be seen separately at the top and bottom parts of the plot. The two squared points are the ones we dragged. They positioned far away from each other in the new view, which reflects their true relationship in the high-dimensional space.

We rely on a variable loading plot Figure 2.10 to identify the important variables that explain the cluster structure. The top two loading variables are the SAT and PER, and there is a negative relationship between the two variables as reflected by the angle between the two variable arrows. We mark the dataset according to the median SAT score in Figure 2.9 (the

Table 2.1: Record for Arizona and Minnesota.

State	SAT	EXP	RAT	SAL	PER	HSG	INC
Arizona	944	4.788	19.3	32.175	27	29468	30863
Minnesota	1085	6	17.5	35.948	9	49658	37933
All State Average	965.92	5.905	16.6	34.829	35	45695	33875

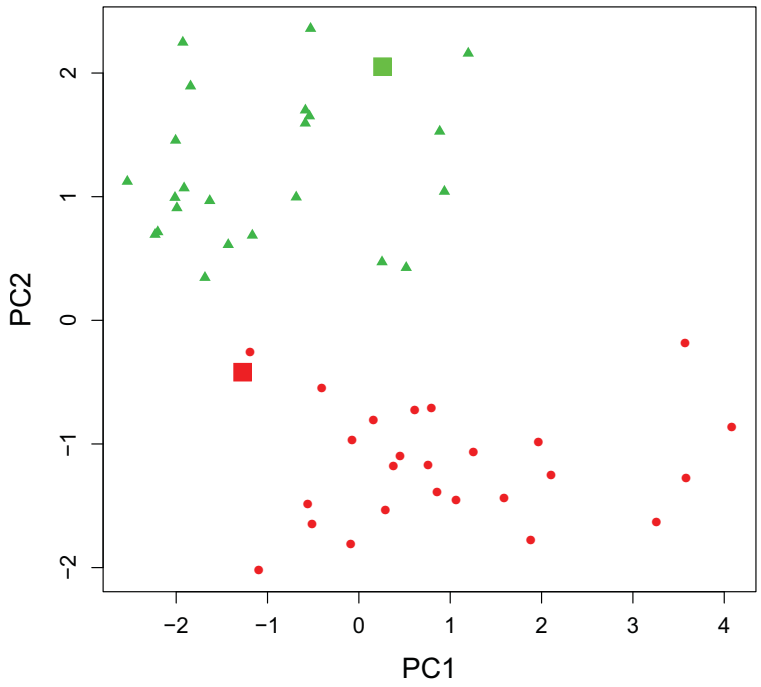

Figure 2.9: The BaVA-updated view of the SAT dataset. Arizona and Minnesota are denoted by squares. Red/Green denotes the state with SAT below/above the country's median.

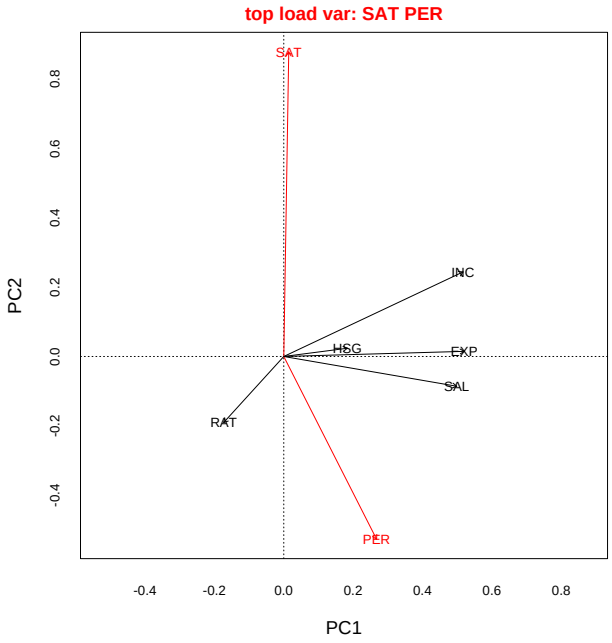

Figure 2.10: The variable loading plot for the updated BaVA view.

red points are the states with SAT below the countrys median level, the green ones are those above the median). We see that the cluster formation is clearly related to SAT, which confirm the fact that SAT has high loading. If we mark the dataset according to the median PER (Figure 2.11) with the same color scheme, the dramatic color shift demonstrates a strong negative relationship between SAT and PER. The reason for this negative relationship is that ACT remains a popular alternative to the SAT among college admissions offices in certain parts of the country. In these states, a select group of bright students intent on attending competitive out-of-state schools are more likely to take and score well on the SAT. Moreover, based on the variable loading plot, there is a weak relationship between public school expenditures and SAT. Thus, further analysis of SAT and EXP should control for the effect of PER.

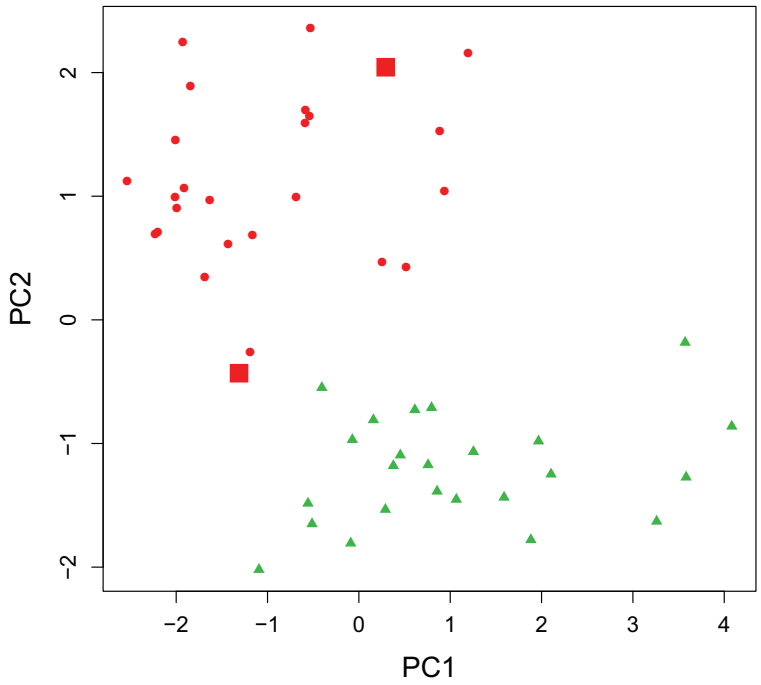

Figure 2.11: The BaVA-updated view of the SAT dataset. Red/Green denotes the state with PER below/above the country's median.

2.4 Discussion

In Section 2.3.4, the feedback was computed based on the movement of two points. If experts would like to move more than two points, the task can be accomplished by assigning those points to two groups, and dragging the groups apart or together. To do so, we developed two possible algorithms: The first way is to calculate the medoids for the first and the second group and apply the procedure defined in Section 2.3 using the medoids. We use the medoids instead of the means to make it more robust to potential outliers. The second way is to compute S_f for every possible pair of points between the two groups, i.e., if there are n_1 points in the first group, n_2 points in the second group, then there are $J = n_1 \times n_2$ possible S_f matrices. As we define $\mathsf{S}_f^{(j)}$ to represent a feedback matrix for pair j ($j \in 1, \ldots, J$), then we consider a final matrix $\hat{\mathsf{S}}_f$ by taking the average:

$$\hat{\mathsf{S}}_f = \frac{\sum_{j=1}^{n_1 \times n_2} \mathsf{S}_f^{(j)}}{n_1 \times n_2}. \tag{2.6}$$

However, if users select a lot of points to move, the second way is computationally expensive when the data are high-dimensional.

One advantage of moving multiple points versus one pair is that one pair of points may not represent the degree to which dimension differ as well as multiple pairs. Moving multiple pairs provides more data concerning user interactions so that the poorly selected pair of observations can be corrected by other pairs, if the remaining are chosen well. However, we found that this way of multiple point movement interaction is more appropriate for another tool called interactive MDS [Endert et al., 2011]. In interacitve MDS, users can move multiple points into new locations. Then the algorithm will invert the optimization, by fixing the locations of the adjusted points and finding an optimal set of weights, which are consistent with the visualization. In this way, we can learn the relative importance of

each data dimention.

It is worth mentioning that a premise of BaVA is that the visualization relies on a probabilistic model which can be incorporated into a Bayesian framework. However, not all visualization algorithms are probabilistic. Furthermore, not all the probabilistic algorithms are appropriate for Bayesian inference. With this limitation in mind, we come up with an algorithmic analogy of BaVA called Visual to Parametric Interaction (V2PI) [Leman et al., 2011]. V2PI is a more general framework than BaVA which does not require incoporating expert judgement through Bayesian Sequantial Updating. We will detail V2PI in Section 5.3.

Chapter 3

Mixture Probabilistic PCA

3.1 Motivation

One assumption of PCA and PPCA is that the data structure is not very complicated in that a single projection can portray all interesting features in the data. However, data clusters can be spatially enclosed within other clusters, such that a single projection cannot reveal all of the structure. A simple example is shown in Figure 3.1. In this example, we simulated a 3D dataset with two clusters, denoted by '◯', that are wrapped in a cube of six clusters, marked by '$*$'. If we use PCA or PPCA as the visualization tool, we cannot find a single projection to discover all the clusters; the inner clusters of the cube are occluded by outer clusters in every direction. If we wanted, we could use BaVA-PPCA to sequentially explore the data space to assess the clusters from different directions in isolated views. However, without extensive knowledge about every clusters, it would be easy to miss features of the complete data. Concerned by this limitation, we develop a BaVA form of Mixture Probabilistic PCA (MPPCA) [Tipping and Bishop, 1999a]. Using MPPCA, we can summarize the data from

multiple projections simultaneously. Thus, we use MPPCA to separate the data into multiple displays (one per mixture) and apply BaVA on the individual mixtures.

3.2 The Mixture PPCA model

Since PPCA is likelihood based, it is readily extendable to a probabilistic mixture framework. MPPCA combines local PPCA through a log-likelihood $\mathcal{L}$ formulation,

$$\mathcal{L} = \sum_{i=1}^{N} \ln\{\sum_{m=1}^{M} \pi_m p(\boldsymbol{x}_i|m)\}, \tag{3.1}$$

where N is the total number of data points; M is the number of mixtures; and π_m represents the probability that an observation comes from mixture m. Conditional distribution $p(\boldsymbol{x}_i|m)$ represents a model for the mixture m,

$$\boldsymbol{x}_i|m = \mathsf{W}_m \boldsymbol{r}_{mi} + \boldsymbol{\mu}_m + \varepsilon_{mi} \qquad m \in 1, \ldots, M \quad (\pi_m > 0, \sum \pi_m = 1)$$

where each parameter has the same meaning as stated for Equation (2.1), except the parameters are mixture-specific. If we assume that the data $\boldsymbol{x}(\boldsymbol{x} = [x_1, \ldots, x_N]')$ are generated from a Gaussian Mixture Model (GMM), as shown in Tipping and Bishop [1999a], all the MPPCA parameters can be estimated through the maximization of the likelihood in Equation (3.1) using the Expectation-Maximization (EM) algorithm. From the EM, we obtain estimates $\hat{\pi}_m, \hat{\mu}_m, \hat{\sigma}_m^2, \hat{\mathsf{W}}_m$, for $m \in 1, \ldots, M$. Each parametric estimate relies on an assessment of the probability that any individual observation $\boldsymbol{x}_i$ belongs to mixture m. Let $R_{im} = p(m|\boldsymbol{x}_i)$ represent the posterior probability that $\boldsymbol{x}_i$ belongs to mixture m, where

$$\begin{aligned} \hat{R}_{im} &= \frac{p(\boldsymbol{x}_i|m)\pi_m}{p(\boldsymbol{x}_i)} \\ &\propto (2\pi)^{(-d/2)} |\sigma_m^2 \mathsf{I} + \mathsf{W}_m \mathsf{W}_m'|^{-1/2} \exp\{-(\boldsymbol{x}_i - \mu_m)'(\sigma_m^2 \mathsf{I} + \mathsf{W}_m \mathsf{W}_m')^{-1}(\boldsymbol{x}_i - \mu_m)/2\}. \end{aligned} \tag{3.2}$$

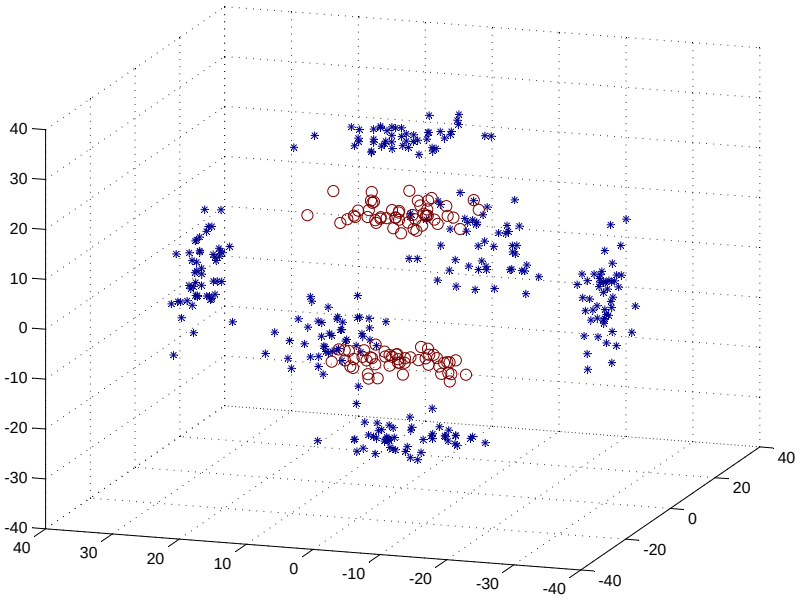

Figure 3.1: A display of the simulated data. Two clusters are denoted by '◯'. The remaining clusters are denoted by '$*$' and form the shape of a cube.

In turn, the parameters W_m and σ_m^2 are determined based on the eigenvalue decomposition of the sample covariance matrix S_m,

$$\mathsf{S}_m = \frac{1}{\hat{\pi}_m N} \sum_{i=1}^{N} R_{im}(\boldsymbol{x}_i - \hat{\mu}_m)(\boldsymbol{x}_i - \hat{\mu}_m)', \tag{3.3}$$

and both $\hat{\pi}_m$ and $\hat{\mu}_m$ are MLE estimates

$$\hat{\pi}_m = \frac{1}{N} \sum_{i=1}^{N} R_{im} \qquad \hat{\mu}_m = \frac{\sum_{i=1}^{N} R_{im}\boldsymbol{x}_i}{\sum_{i=1}^{N} R_{im}}. \tag{3.4}$$

An observation can be assigned to a mixture for which it has the highest probability or "responsibility" of membership; i.e., $\boldsymbol{x}_i$ belongs to mixture m when $max(R_{i1}, \ldots, R_{iM}) = R_{im}$, once the EM algorithm has converged. In turn, MPPCA creates M displays of the data, where each display contains only the assigned mixture observations.

3.3 Comparison with K-means

K-means was first proposed by MacQueen [1967] and is one of the most popular clustering algorithms in DM literature. In the name "K-means", K is the number of clusters and "means" denotes the cluster centroids. A centroid is the mean of the variables in the cluster when Euclidean distance is used to measure the difference between observations. The algorithm minimizes the objective function

$$\sum_{k=1}^{K} \sum_{x_i \in C_k} \|\boldsymbol{x}_i - \boldsymbol{c}_k\|^2,$$

where $\boldsymbol{c}_k$ denotes the k^{th} cluster centroid. Both K-means and MPPCA uncover cluster structures. Both methods have pros and cons that we highlight here.

3.3.1 Relationship between MPPCA and K-means

It has been shown in Celeux and Govaert [1992] that parameter estimates from K-means and the EM algorithm for Gaussian Mixture Models (GMM) would be equivalent given three conditions:

1. In the EM iterations, a classification step is added between the Expectation step and Maximization step. That is, each $\boldsymbol{x}_i$ is assigned to the cluster which provides the Maximum A Posteriori (MAP) probability R_{ik}.

2. The mixture proportions π_k are all equal for every cluster,

$$\pi_k = \frac{1}{K} \qquad k = 1, 2, \ldots, K$$

3. All the clusters share a common variance matrix.

For MPPCA, we have a Gaussian mixture model. When we consider a "winner takes all" strategy, we assign each observation $\boldsymbol{x}_i$ to one cluster $\boldsymbol{C}_k$ that provides the highest posterior probability R_{ik}, and set R_{ik} equals 1. From iteration to iteration in the EM, the cluster assignment may change. The equivalence between MPPCA and K-means can be easily seen through the equations below:

$$\hat{\mu}_k = \frac{\sum_{i=1}^{N} R_{ik}\boldsymbol{x}_i}{\sum_{i=1}^{N} R_{ik}} = \frac{\sum_{x_i \in C_k} \boldsymbol{x}_i}{\|\boldsymbol{C}_k\|},$$

$$\mathsf{S}_k = \frac{1}{\hat{\pi}_k N}\sum_{i=1}^{N} R_{ik}(\boldsymbol{x}_i - \hat{\mu}_k)(\boldsymbol{x}_i - \hat{\mu}_k)' = \frac{1}{\|\boldsymbol{C}_k\|}\sum_{x_i \in C_k} R_{ik}(\boldsymbol{x}_i - \hat{\mu}_k)(\boldsymbol{x}_i - \hat{\mu}_k)'.$$

The weighted observation average of a cluster discovered via MPPCA equals the corresponding centroid from K-means; and the variances per cluster from K-means and the EM

Table 3.1: Basic cluster summary statistics for simulated data.

Measures	Mean (μ)	Variance (S)	Cluster size ($\boldsymbol{C}_k$)
$\boldsymbol{C}_1$	(4,0,5)	(20,30,0.1)	20
$\boldsymbol{C}_2$	(0,0,2)	(10,20,0.1)	50
$\boldsymbol{C}_3$	(0,0,-2)	(20,40,0.1)	30
$\boldsymbol{C}_4$	(-4,0,-5)	(10,10,0.1)	40

algorithm are equal when $\hat{\pi}_k = \hat{\pi}_{k'}$. However, when they are not equal, MPPCA accounts for different mixture proportions and variance matrices. Thus, MPPCA will perform better than K-means for clustering datasets which have unequal cluster sizes and heterogeneous cluster variances.

3.3.2 Advantages of MPPCA illustrated by a numerical experiment

To illustrate the advantages of MPPCA, we simulate a three dimensional data with four mixtures. The basic cluster summary statistics are given in Table 3.1 and the 3D plot for this dataset is shown in Figure 3.2.

We apply MPPCA and K-means on the simulated dataset. To make a fair comparison, we use the same set of random data points to initialize MPPCA and K-means. The outcomes are shown in Figure 3.3 with data points marked according to the classification results of the two clustering methods respectively. Table 3.2 also provides results for several comparison criteria.

In Figure 3.3a), MPPCA separate the data into four disjoint, but locally smooth clusters.

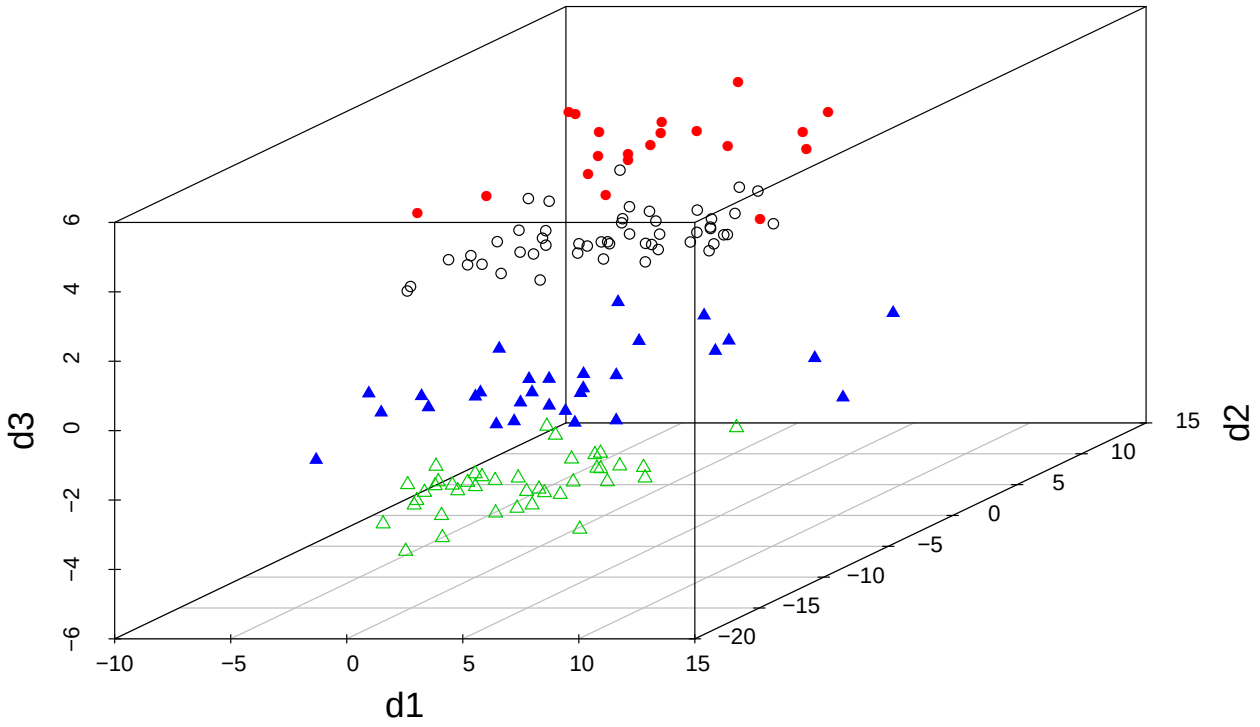

Figure 3.2: Simulated 3D data with various cluster sizes and variances. Different groups are marked by different symbols.

The cluster structure appears at the third dimension, which is similar to the true data structure, as shown in Figure 3.2. K-means (Figure 3.3b)) separated the data into circular shaped clusters that contain nearly equal numbers of data points. We use a Rand Index (RI) and an Adjusted Rand Index (ARI) to measure the classification accuracy. The Rand Index can be expressed as

$$\text{RI} = \frac{\text{number of agreements}}{\text{number of agreements} + \text{number of disagreements}},$$

where "agreements" refers to pairwise agreements between the algorithm's classification and true membership. The Rand index has a value between 0 and 1, with 1 indicating that the data clusters are exactly the same. Two problems with RI are a) the expected value of the RI of two random partitions does not take a constant value and b) RI approaches 1 as the number of clusters increases. The Adjusted Rand Index is the corrected-for-chance version of the RI. It assumes the generalized hypergeometric distribution as the model of randomness to overcome the two problems of RI. Please refer to Hubert and Arabie [1985] for details of ARI. From these two indices, MPPCA does a better classification job than K-means. One possible explanation is that MPPCA addresses both cluster centrality and variability, whereas K-means only consider centrality. MPPCA is a soft partitioning method since it is based on probabilistic memberships which are shared among all of the mixtures. The variance matrices (Equation (3.3)) are local responsibility-weighted. Thus, the classification will be influenced by data points in neighboring clusters.

Given the results from MPPCA, under a converged EM algorithm, we compare likelihood measures to those obtained under a converged K-means algorithm. Based on criteria such as Bayesian Information Criterion (BIC) [Schwarz, 1978] and Integrated Classification Likelihood (ICL) ($ICL \approx BIC - \sum_{i=1}^{N}\sum_{k=1}^{K} \text{MAP}(R_{ik})\log(R_{ik})$) [Biernacki et al., 2000], MPPCA beats K-means as shown in Table 3.2. Since MPPCA does not directly minimize squared

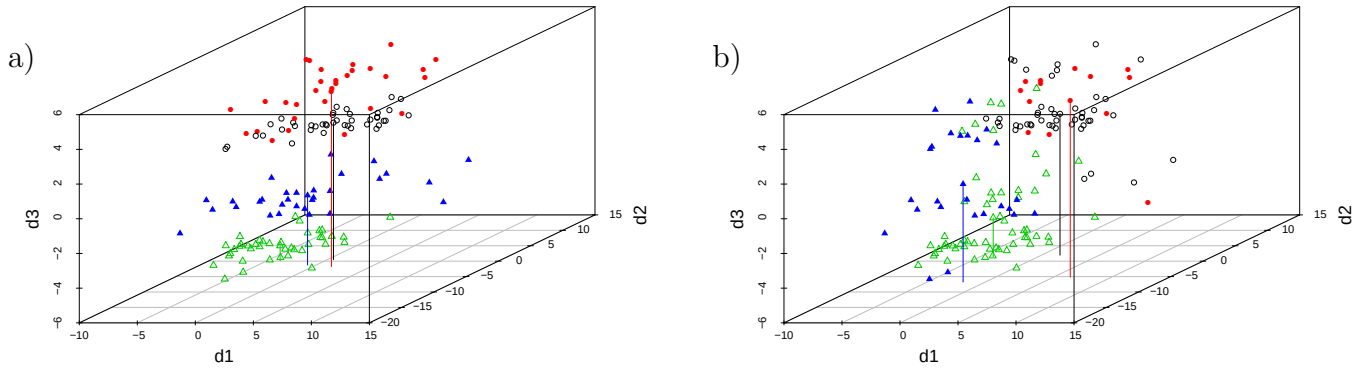

Figure 3.3: a) MPPCA classification result. b) K-means classification result.

Table 3.2: MPPCA and K-means comparison according to different criteria

Algorithm	Rand Index (RI)	Adjusted RI	BIC	ICL	SSW	Calinski's Index
MPPCA	0.93	0.84	-1122.84	-1128.27	5139.102	22.16
K-means	0.68	0.37	-1253.35	-1266.78	3211.432	62.67

reconstruction error as in K-means, K-means is superior according to Within Cluster Sum of Squares (SSW) and Calinski's Index ($\frac{\text{Between Cluster Sum of Squares}/K-1}{\text{Within Cluster Sum of Squares}/N-K}$) [Calinski and Harabasz, 1974]. However, we can see these results are not meaningful in terms of cluster identification as shown in Figure 3.3 b).

Since MPPCA is model based, it has additional flexibility over hard clustering algorithms such as K-means. For example, comparison between different number of mixtures is possible for MPPCA using model selection criteria such as BIC. MPPCA can be applied on different types of data by changing the distribution to something other than Multivariate Normal distribution. The high dimensional data can be modeled with relatively few free parameters that may be controlled through the latent space.

3.4 The Cluster User Interface

To facilitate clustering, we built another cluster GUI interface (Figure 3.4) to help explore the data using R and GGobi. The GUI make multiscale-clustering possible. Users can choose different clustering methods to separate the dataset and put points in different windows. To access the clustering results, users can color points in any chosen window, and resolution per window can be changed by splitting it by clustering into sub-windows. The GUI also gives users enough freedom to specify groups of observations that should form a cluster and put them in a separate window.

One issue with putting data into different windows is that users may lose a sense of the whole dataset and the relationships between cluster windows. Also, the number of clusters chosen for the algorithm can be a hard decision: e.g., too many may exaggerate information in the data. Thus, in our cluster GUI, whenever the number of clusters changes, we provide

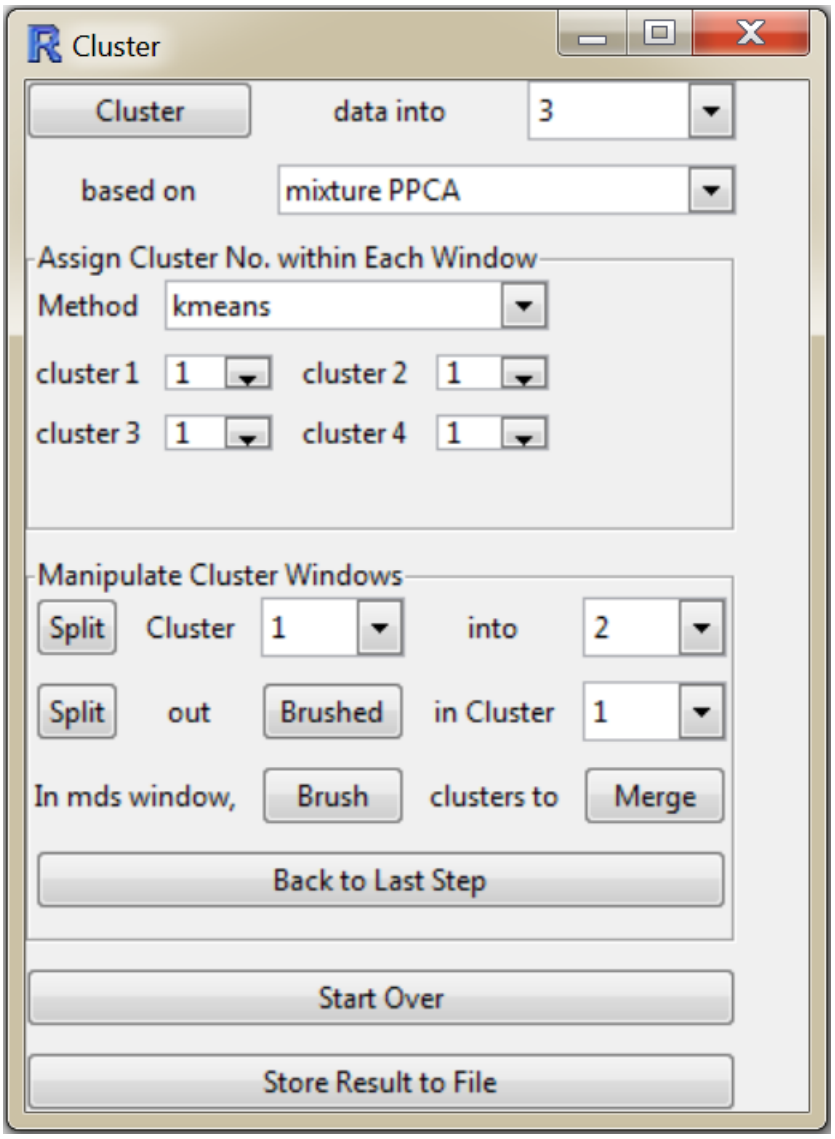

Figure 3.4: The cluster user interface.

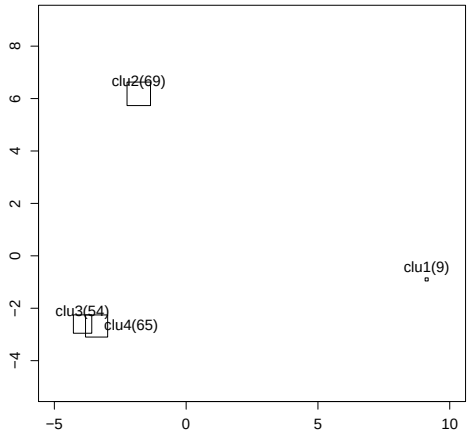

Figure 3.5: Example of a MDS layout of the cluster centroids.

a classical Multi-dimensional Scaling (MDS) [Schiffman et al., 1981] layout of the cluster centroids to help with further clustering decisions. For example, the decision to add or subtract clusters based on the MDS window can be done by merging clusters that are close to each other or splitting a cluster which has too many data points, respectively. Figure 3.5 is an example of the cluster centroids layout. In this MDS plot, squares represent clusters and the square size is proportional to cluster size. The centroids of Clusters 3 and 4 almost overlap. This indicates that it is possibly not necessary to split these two clusters.

3.5 Case Study with Gene Dataset using BaVA-MPPCA

The purpose of this case study is to classify a microarray dataset of 197 (observation number) yeast genes in the Saccharomyces cerevisiae genome based on gene functions. The dataset includes measurements for gene expression from 79 (dimension number) DNA hybridization experiments [Eisen et al., 1998]. Eisen et al. [1998] suggest that genes of similar function yield

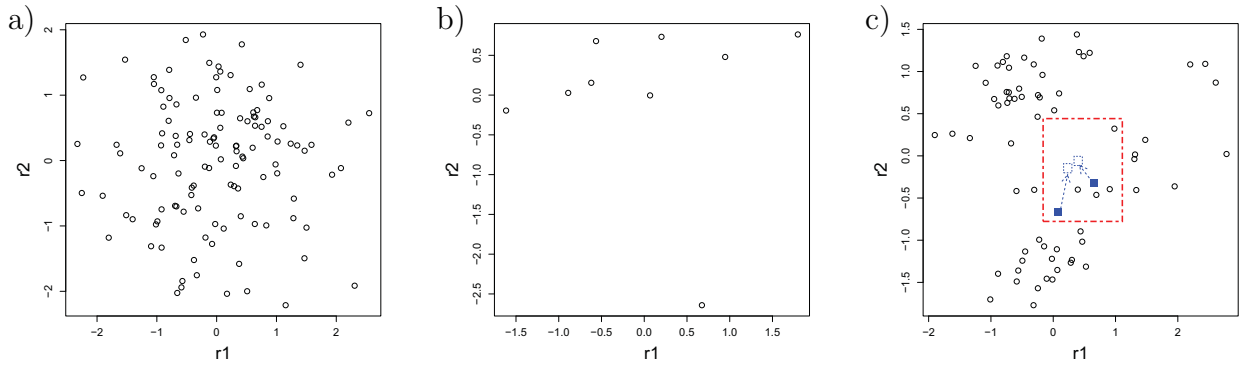

Figure 3.6: Cluster yeast gene data into three groups based on MPPCA. a) Cluster 1. b) Cluster 2. c) Cluster 3.

similar expression patterns in hybridization experiments. Thus, we expect gene of similar function to cluster based on their expression profiles. In particular, we are interested in the characterization of four functional classes: cytoplasmic ribosomes, proteasome, respiration and histone.

We try to group data into three clusters firstly using MPPCA. The resulting cluster displays are shown in Figure 3.6. An interesting cluster pattern appears in cluster 3; multiple clusters could be in cluster 3, but they are not separate well in the display. Thus, we apply our BaVA method on the third cluster; i.e., we select points according to the "brush", adjust the observations, and apply the updating machinery as described in Section 2.3.2. After the re-projection, the cluster structure becomes clearer in the BaVA-tized view (Figure 3.7). Facilitated by the cluster GUI, users may label a set of observations (marked by '△' in Figure 3.7) and split them out from window 3 to form a new cluster. Now, there are four clusters in total as shown in Figure 3.8.

Since the gene function annotation is readily accessible from Munich Information Center

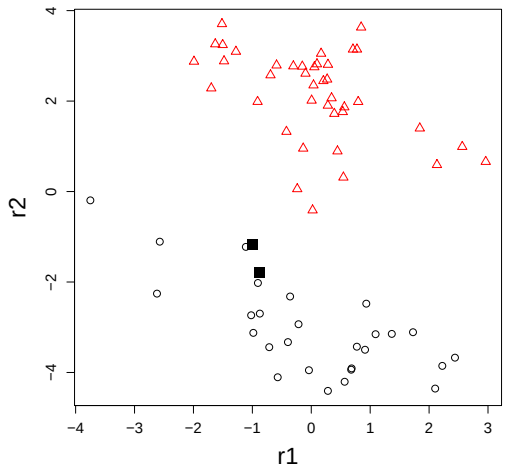

Figure 3.7: BaVA updated view for cluster 3 ('■' points are the ones we moved). Split '△' points out to get another cluster.

for Protein Sequences Yeast Genome Database (MYGD), we compare our classification to the true genetic functions. Data points in the four clusters are labeled according to their ground-truth in Figure 3.8. Only five genes are predicted with error and our true prediction rate is 0.974. Had we used K-means, we would have predicted 17 genes erroneously (a true prediction rate of 0.913).

3.6 Discussion

Using the EM algorithm to estimate the MPPCA parameters has several drawbacks, including: 1) sensitivity to initialization and 2) convergence to local maxima of the likelihood. The success of MPPCA for clustering depends on the appropriate estimation of the covariance for each mixture component. As can be seen in Equation (3.2), the estimation of the posterior probability R_{im} is influenced by the value $(\boldsymbol{x}_i - \mu_m)'(\sigma_m^2 \mathsf{I} + \mathsf{W}_m \mathsf{W}_m')^{-1}(\boldsymbol{x}_i - \mu_m)$. We note

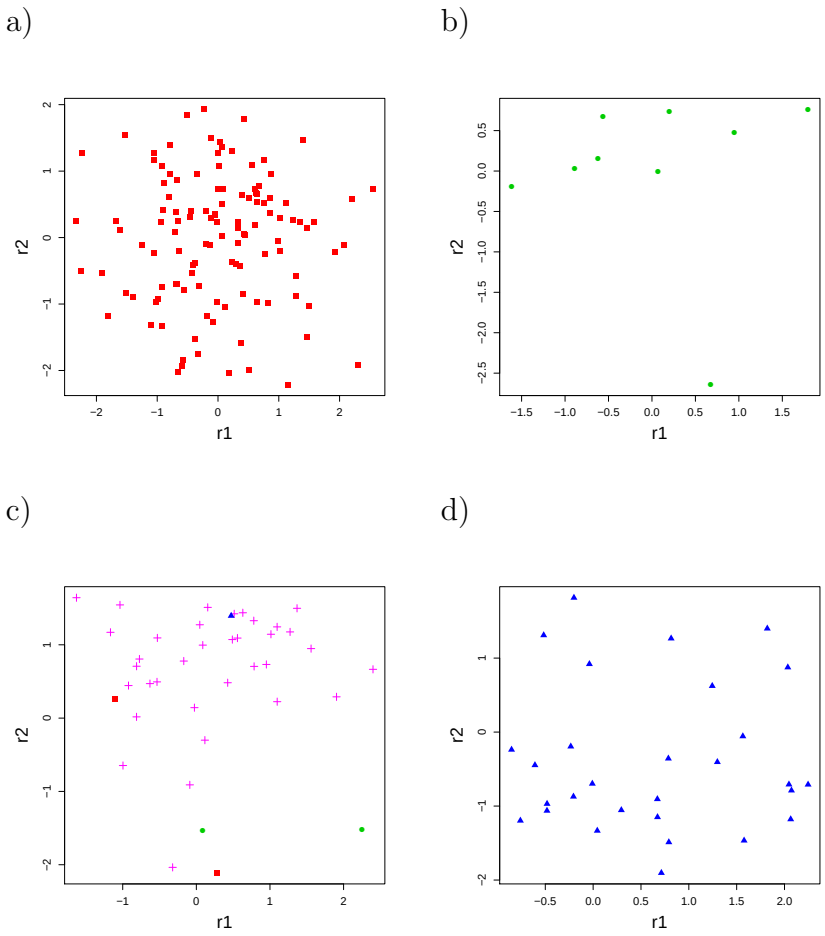

Figure 3.8: Clusters marked by their true gene functions. a) '■' denotes cytoplasmic ribosomes. b) '•' denotes histone. c) '+' denotes proteasome. d) '▲' denotes respiration.

this quantity is the Mahalanobis distance of a data point to the mixture mean. In the intra-class covariance estimate $(\sigma_m^2 \mathsf{I} + \mathsf{W}_m \mathsf{W}'_m)^{-1}$, W_m and σ_m^2 is computed from the eigenvalue decomposition of the local posterior probability-weighted covariance matrix S_m (Equation (3.3)). Since EM is an iterative process, a poor estimate of the covariance matrix will lead to inappropriate cluster assignments through R_{im}. Circuitously, the ill grouped data will provide an inaccurate sample covariance matrix through *Equation*(3.3).

Let us consider a case when it is hard for MPPCA to provide good covariance estimation. Figure 3.9 shows a simulated three dimensional dataset. There are four groups of data which are simulated from multivariate normal distributions. The ∘ and • groups are located in the upper right corner and each group mainly varies along the x axes. While the △ and ▲ groups are located in the lower left corner and each group mainly varies along the z axis. Due to the large gap between the upper right and lower left corner, the dataset as a whole primarily varies along the first principal component (red arrow in Figure 3.9). This is not consistent with either principle direction of the local groups. In the MPPCA procedure, we do not have any group information before hand, so we often initialize the covariance matrix for each mixture using the global sample covariance. This starting covariance estimate is not informative for local covariance estimation, and therefore contributes to clustering along the distance of the global principle direction. This incorrect global direction results in poor mixture assignment. In the next EM iteration, the wrong data grouping reflected in R_{im} will reinforce the non-informative covariance estimation through Equation (3.3). Finally, this loop of bad estimations results in cluster cutting as shown in the projections of the four clusters found by MPPCA in Figure 3.10. The large number of bad results all follow from a poor initial estimate of the local covariances. Also, to initialize observation assignments to clusters and mixtures, MPPCA often uses K-means [MacQueen, 1967]. K-means requires the prior knowledge of the number of clusters K. When K is unknown, Agglomerative model-

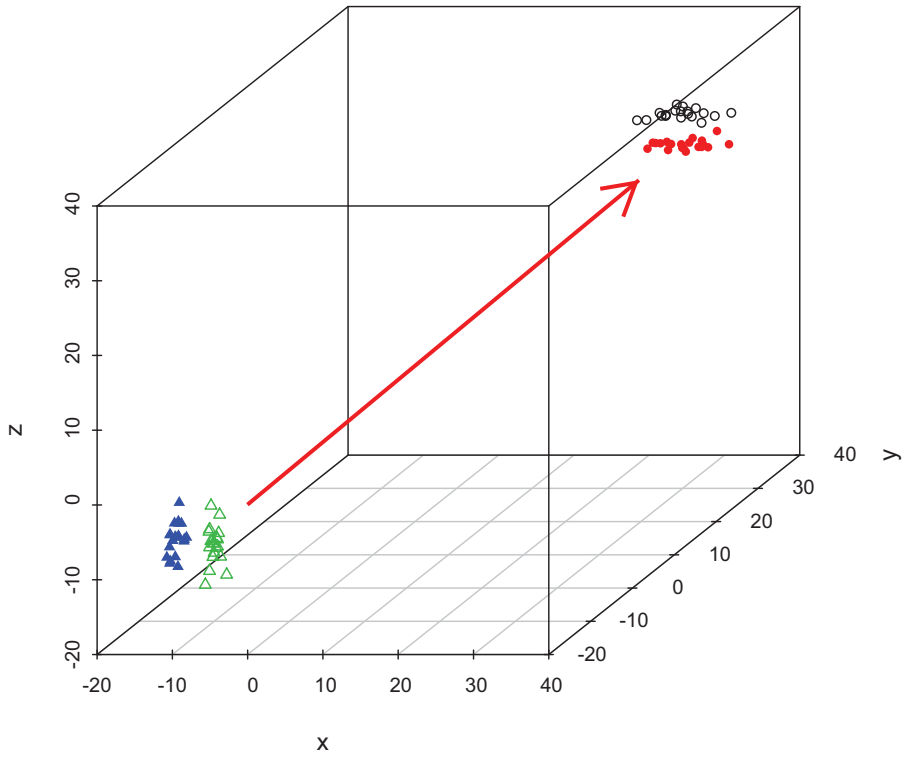

Figure 3.9: Simulated three dimensional dataset. The ∘ and • groups share the same covariance. The △ and ▲ groups share another covariance.

based clustering could be used. The Agglomerative model-based clustering method estimates the number of clusters dynamically (based on BIC) and has been shown to work well for the initialization for MCLUST [Fraley and Raftery, 2002]. However, the agglomerative method may require substantial computation that seems unworthy for the sole task of initialization.

There are also cases when the pre-specified number of mixtures M in a MPPCA is less than the true number of clusters K in a dataset. In this situation, at least two of the clusters will be combined into one mixture. As defined by MPPCA, the estimated covariance for this mixture will be a weighted average of the intra-class covariances, e.g., for a mixture j with 2 clusters, $\mathsf{S}_j = \sum_{k\in(1,2)} w_k \mathsf{S}_\mathsf{k}$, where $\sum_{k\in(1,2)} w_k = 1$ and $w_k \geq 0$. In turn, a visualization

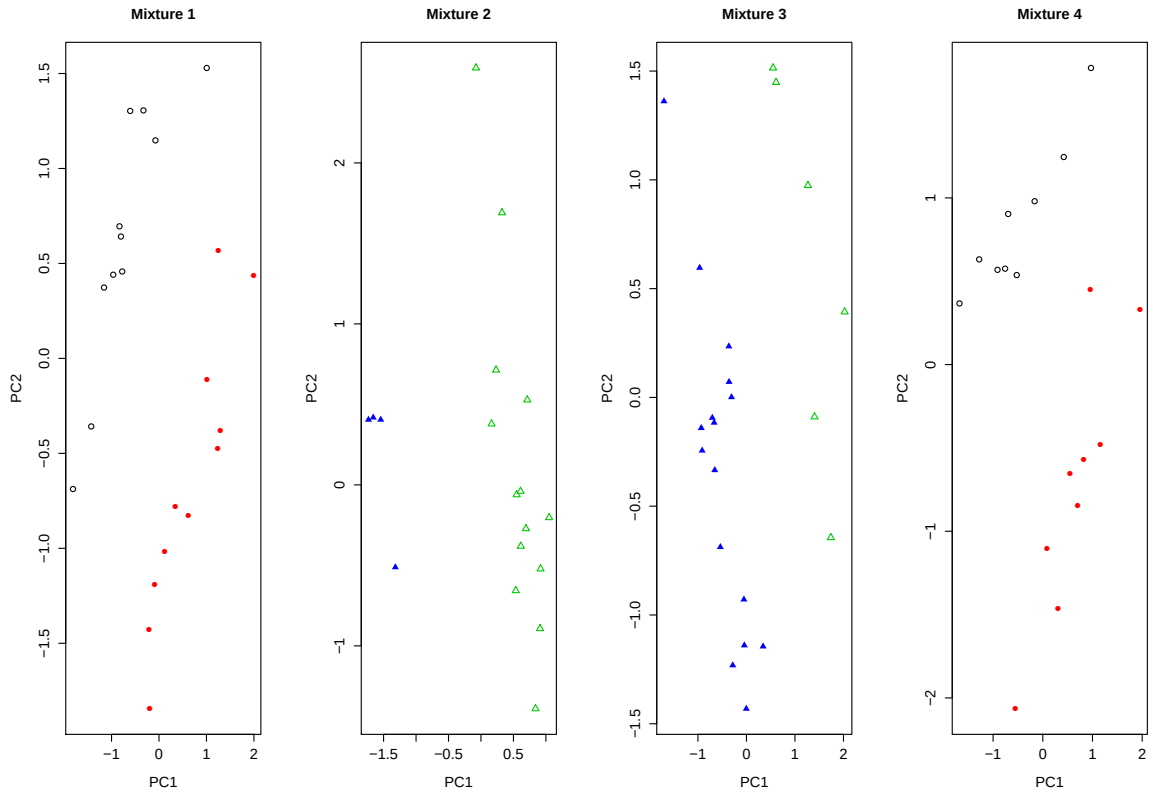

Figure 3.10: Clustering and projection provided by MPPCA for the simulated three dimensional data. Clusters are cut into different mixtures. For example, ∘ observations in mixture 1 and mixture 4 should occur in the same mixture.

of the mixture may not reveal clusters, since the eigen-decomposition of the mixed S_j can only provide an arbitrary projection. Variable loading plots for this mixture will not be meaningful.

Chapter 4

The C-MPPCA Method

4.1 Motivation

Centrality and variability are two important features of data clusters. Most data clustering and visualization algorithms only take centrality into account. For example, when the number of visualizations for one dataset is chosen to be less than the true number of clusters in the dataset, most clustering algorithms (e.g., MPPCA) will group clusters with close centroids. However, in time series, spatial, cross-sectional and panel data applications, data dependency and cluster covariance structure may be more important than centrality. When data have different structures in different parts of the input space, it will be helpful to know the distinct covariance structures in the data and which clusters are associated with each structure. Thus, we might want to group similar covariance structured clusters into mixtures. Additionally, observations in clusters with similar covariance will vary along similar principal directions. Thus, it is easy to find a single direction to project the clusters per mixture component and visualize potential trends. In this way, we overcome the problem of

hybrid covariance estimates, as defined in Section 3.6. Variable loading plots per mixture may identify important variables that differentiate clusters.

To illustrate our motivation, let us develop an example from a simulated dataset. Figure 5.2 a) shows a simulated three dimensional (x,y,z) dataset. There are four clusters of data (labeled by $\circ$, $\bullet$, $\triangle$, $\blacktriangle$ in black, red, green and blue, respectively) which are simulated from multivariate normal distributions. The black ($\circ$) and red ($\bullet$) clusters vary mainly in the x dimension and share the same covariance, Σ_1. While the green ($\triangle$) and blue ($\blacktriangle$) clusters vary mainly in the z dimension and share the same covariance, Σ_2. MPPCA cannot summarizing these data well. It put the red ($\bullet$) and green ($\triangle$) clusters into one mixture and the black ($\circ$) and blue ($\blacktriangle$) clusters into another mixture, as shown in Figure 5.2b). This is a natural result due to the fact that MPPCA grouped the data based on their cluster locations. Also, we can not reveal the structure within each mixture using one projector, since the covariance in each mixture is a combination of two different covariances Σ_1 and Σ_2.

In the next section, we extend MPPCA to group clusters based on similar covariance structures. Had we applied such a method for the example above, we would have obtain a meaningful, clean, and informative visualization, similar to that of Figure 5.2c).

4.2 The C-MPPCA Algorithm

We develop a new method, under the framework of MPPCA, to group and visualize similar covariance clusters together. We call our method Covariance-guided MPPCA (C-MPPCA). Generally, C-MPPCA 1) relies on fast clustering methods to initialize and estimate local clusters, and 2) includes a new way of combining similar covariance structures via likelihood comparisons.

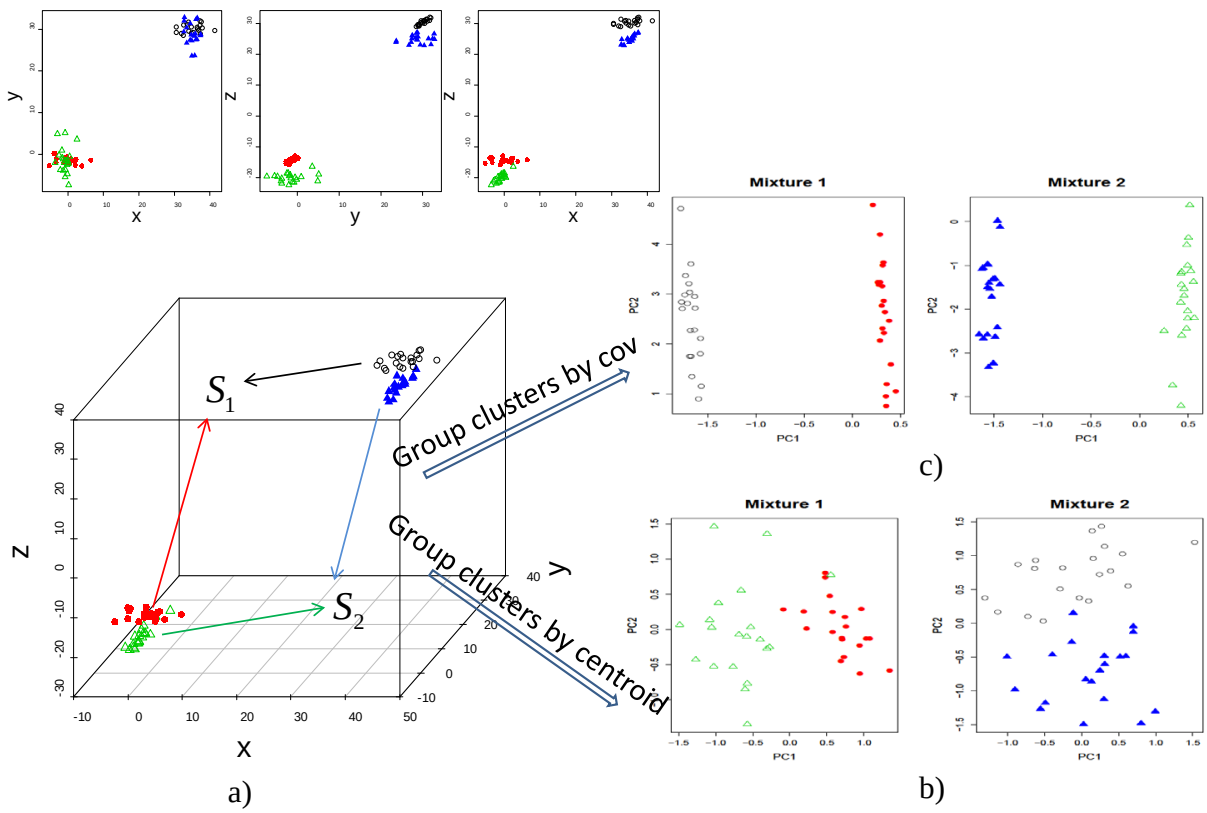

Figure 4.1: a) Simulated three dimensional dataset (plotted in 3d and 2d). The ∘ and • clusters share the same covariance. The △ and ▲ clusters share another covariance. b) Clustering and projection provided by MPPCA for the simulated data. c) The visualization we can expect if using C-MPPCA.

In C-MPPCA, we employ DBSCAN [Ester et al., 1996] for initialization. DBSCAN is a density-based clustering algorithm based on the notion of density reachability. DBSCAN does not require the number of clusters in the data a *priori* and it is able to find arbitrarily shaped clusters. However, DBSCAN requires two parameters: a neighborhood distance ε and a minimum number of points required to declare a cluster ν. A default value for ν is

5. While this number is somewhat arbitrary, we don't wish to consider clusters which only relate to a very few points, so $\nu = 5$ is a reasonable choice. Given ν, we take the average of the ν-nearest neighbors' distance for each data point, and set this as the value of ε. We want to find all of the dense areas of the dataset and take the mean and covariance of each dense area to initialize our algorithm.

In a dataset, the number of distinct covariance structures might be less than the number of clusters. For example, in the simulated dataset (Figure 5.2), the number of distinct covariance structures equals two. Our algorithm requires this number a *priori*. In the applications, such as time series or spatial analysis, this information may be readily available. For cases when we know nothing about the data structure, we suggest an exploratory procedure that we explain by example in Section 4.4.

Given that we know the number of clusters K from DBSCAN and the number of distinct covariance structures M, we start our C-MPPCA algorithm. Our purpose is to get M mixtures, where the clusters within each mixture share similar covariance structures. Since we do not have the information about which clusters to group into mixtures, we try all the possible ways of putting K distinguishable cluster centers into M indistinguishable mixtures, and make sure no mixture is empty. The total number of ways to do this is represented by the Stirling number: $\frac{1}{M!}\sum_{i=0}^{M}(-1)^i\binom{M}{i}(M-i)^K$ [Graham et al., 1988]. Suppose in a trial, several clusters are assigned into a mixture C_m with the assumption that they share a similar covariance. Instead of using Equation (3.3) to estimate the covariance for each mixture respectively, we compute the pooled covariance estimator:

$$\mathsf{S}_m = \frac{1}{(\sum_{k\in C_m}\hat{\pi}_k)N}\sum_{k\in C_m}\sum_{i=1}^{N} R_{ik}(\boldsymbol{x}_i - \hat{\mu}_k)(\boldsymbol{x}_i - \hat{\mu}_k)'. \tag{4.1}$$

We take S_m as the estimation for each mixture covariance in cluster C_m. If we put dissimilar covariance clusters into the same mixture, the pooled covariance matrix will provide a poor

estimate as compared to each cluster's own covariance and decrease the likelihood. On the contrary, if we put similar-covariance clusters into one mixture, the pooled covariance matrix should be similar to each individual cluster's covariance and the data likelihood will increase. Since every combination of clusters corresponds to a different probability model, after C-MPPCA converges, we choose the model with the largest likelihood. For data visualization, we project data based on the eigen-decomposition of the covariance S_m^*:

$$\mathsf{S}_m^* = \frac{1}{(\sum_{k \in C_m} \hat{\pi}_k)N} \sum_{k \in C_m} \sum_{i=1}^{N} R_{ik}(\boldsymbol{x}_i)(\boldsymbol{x}_i)'. \tag{4.2}$$

Note that in Equation (4.2), we do not subtract cluster means from the data points, so that we can distinguish the clusters within each mixture by their cluster centroids. To summarize this section, we provide Figure 4.2 that states the steps of C-MPPCA explicitly.

In summary, we start at the local level through DBSCAN initialization and group the clusters to achieve meaningful mixtures. In this way, we overcome the drawback of global covariance initialization sensitivity as in the original MPPCA. When the number of mixtures is less than the number of clusters, we put similarly structured clusters together to avoid poor weighted-average covariance estimates, as mentioned in Section 3.6. In the next section, we exemplify the advantages of C-MPPCA using simulated and real datasets.

4.3 Case Studies

4.3.1 Simulation Experiment

To illustrate the advantages of C-MPPCA, we simulate a dataset which consists of six clusters within a data space of 15 dimensions. The projection provided by PCA for this dataset is shown in Figure 4.3. Different clusters are marked by different symbols. Some, but not all,

Figure 4.2: The C-MPPCA Steps:

STEP 1 Run DBSCAN on the dataset and use the returned K cluster centers and covariance matrices to initialize MPPCA.

STEP 2 Given the number of distinctive covariances M, try every combination of assigning K cluster centers into M mixtures.

STEP 3 Run MPPCA for each combination with modified S_m estimation: for the clusters in mixture m, estimate their covariance using pooled S_m:

$$\mathsf{S}_m = \frac{1}{(\sum_{k \in C_m} \hat{\pi}_k)N} \sum_{k \in C_m} \sum_{i=1}^{N} R_{ik}(\boldsymbol{x}_i - \hat{\mu}_k)(\boldsymbol{x}_i - \hat{\mu}_k)'.$$

STEP 4 Choose the combination that provides the maximum data likelihood.

STEP 5 Project data based on the eigen-decomposition of the covariance estimator without adjusting the cluster means:

$$\mathsf{S}_m = \frac{1}{(\sum_{k \in C_m} \hat{\pi}_k)N} \sum_{k \in C_m} \sum_{i=1}^{N} R_{ik}(\boldsymbol{x}_i)(\boldsymbol{x}_i)'.$$

of the clusters share a similar covariance structure. The clusters denoted by •,▲,■ share the same covariance structure Σ_1, while clusters denoted by ×,○,□ share the same covariance structure Σ_2; covariances Σ_1 and Σ_2 are different from each other. Because there are different local covariance structures. Figure 4.3 does not separate the clusters well.

We apply traditional MPPCA and C-MPPCA on the simulated dataset. The outcomes are shown in Figure 4.4 and Figure 4.5 respectively. MPPCA put the ○ group into one mixture since this cluster is far away from the other five clusters. While the other five clusters are occluded by each other in the other mixture. C-MPPCA, on the other hand, separates the

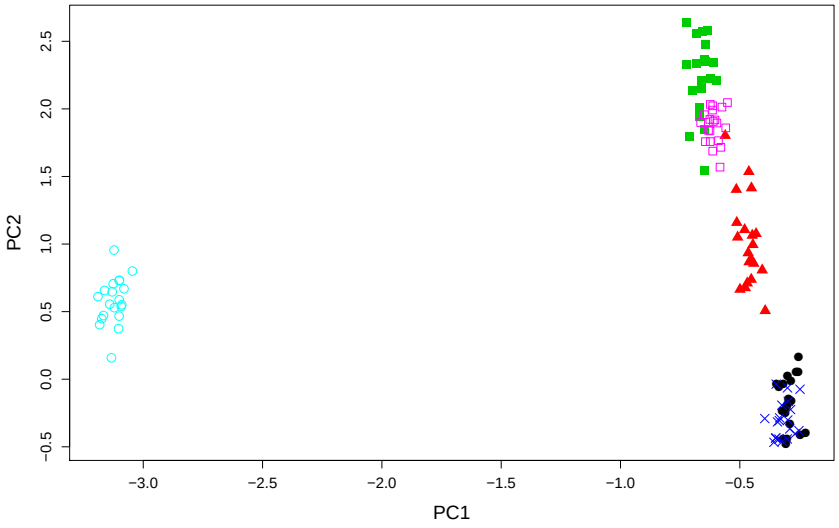

Figure 4.3: PCA projection for the simulated dataset.

two groups of clusters beautifully; the clusters with similar covariances grouped accordingly. Additionally, it is possible to display the disjoint clusters in a single projection per mixture (Figure 4.5).

4.3.2 Real World Application

We now apply our method to a time series dataset which has a distinctive data correlation structure. This dataset contains 600 examples of control charts synthetically generated by the process in Alcock and Manolopoulos [1999]. There are 60 time points in each control chart. Each time point corresponds to a dimension of the dataset. The control charts can be classified into six different classes: Normal, Cyclic, Increasing Trend, Decreasing Trend, Upward Shift, and Downward Shift. Figure 4.6 shows one example from each class. The Increasing and Decreasing Trend groups share comparable covariance structures, but have

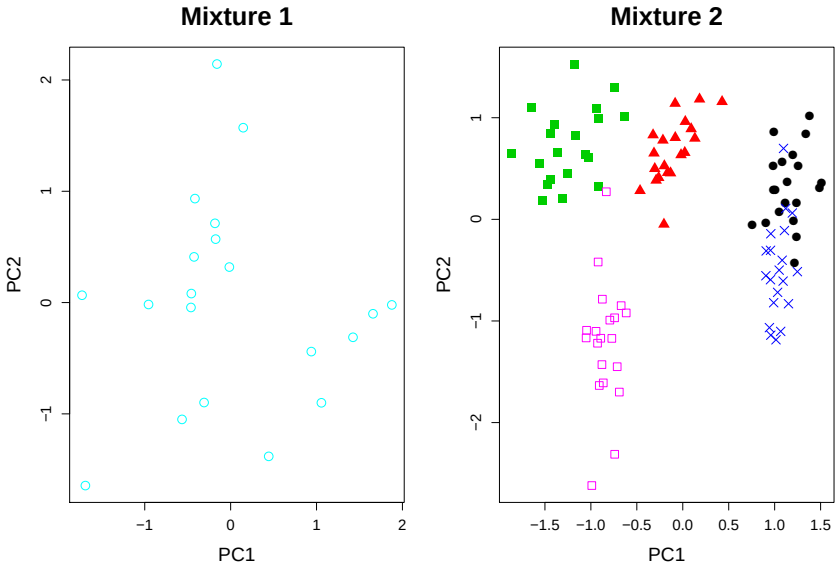

Figure 4.4: Clustering and projection provided by MPPCA for the simulated dataset.

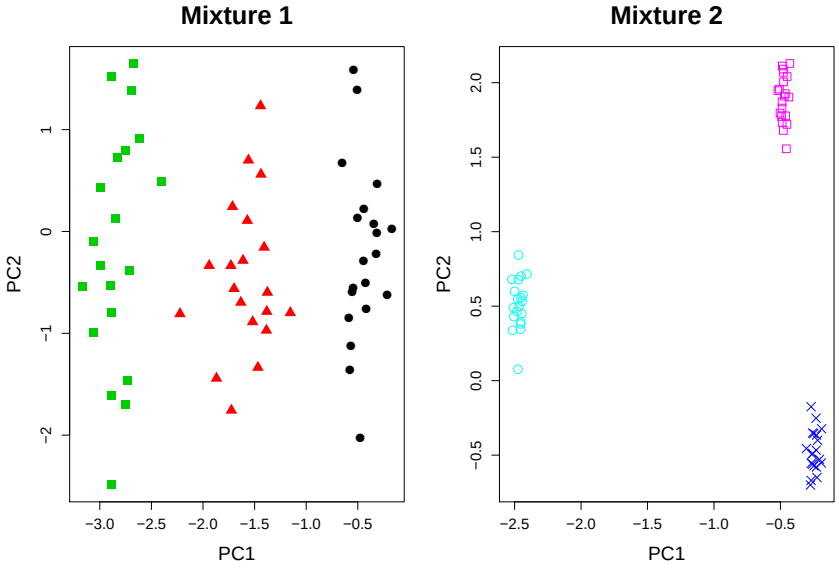

Figure 4.5: Clustering and projection provided by C-MPPCA for the simulated dataset.

different means. The same is for the Upward and Downward Shift groups. The clusters' mean differences can be clearly seen in the PCA projection of the data (Figure 4.7).

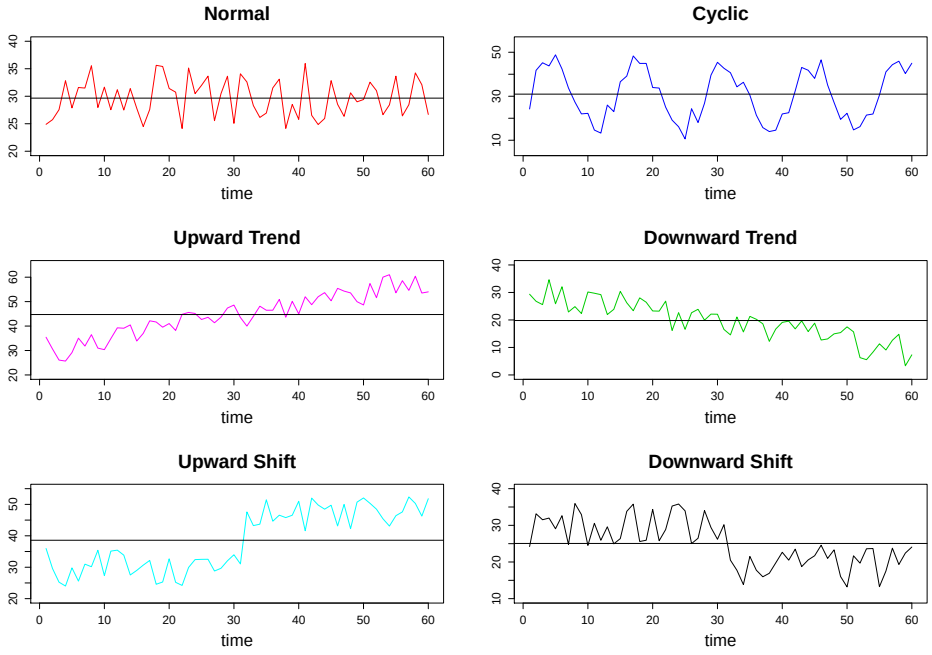

Figure 4.6: Examples of six different time series.

As in the simulated case, the clusters are spread across mixtures when MPPCA is applied (Figure 4.8). The reason is that the intra-class covariance is different from the global data covariance. We do not have any group information before hand, thus, as prescribed by MPPCA, we initialized the covariance matrix for each mixture in the EM to the global sample covariance. This starting covariance estimate is not informative of local structure, and therefore contributes only to clustering in the global principle directions. Additionally, clusters with neighboring means will always be put together without consideration of their different shapes. Ultimately, the clusters are separated into different groups, as seen in Figure 4.8.

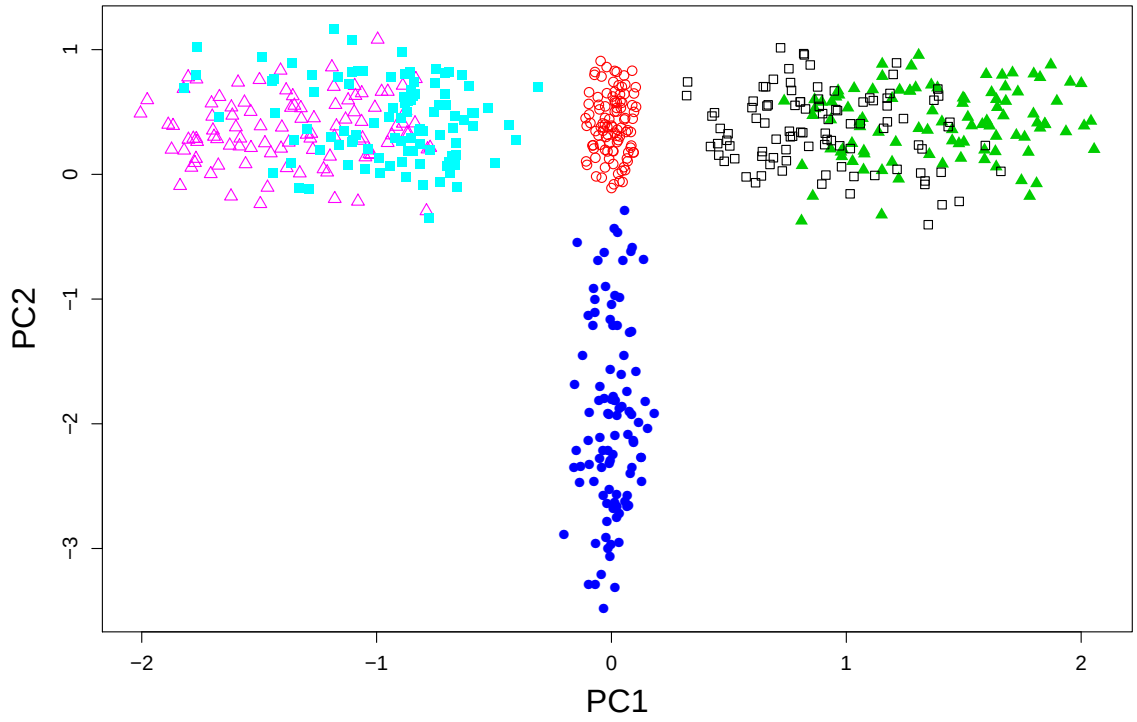

Figure 4.7: PPCA projection for the control chart dataset. ○ denotes Normal; ● denotes Cyclic; △ denotes Increasing trend; ▲ denotes Decreasing trend; ■ denotes Upward shift; □ denotes Downward shift. (The same notation also applies to the following Figures.)

We applied C-MPPCA to the same dataset. There are four covariance structures in the data. Each one is associated with Normal, Cyclic, Increasing or Decreasing trend, Upward or Downward shift respectively. Given this information, we set the number of mixtures to 4. DBSCAN finds 6 dense areas and yields 6 centroids and covariances for initialization. C-MPPCA searches over all possible ways of positioning the clusters within mixtures (65, in this case), and selects the configuration that maximizes the likelihood. Figure 4.9 shows the projected results. The clusters within each mixture window are more distinguishable than

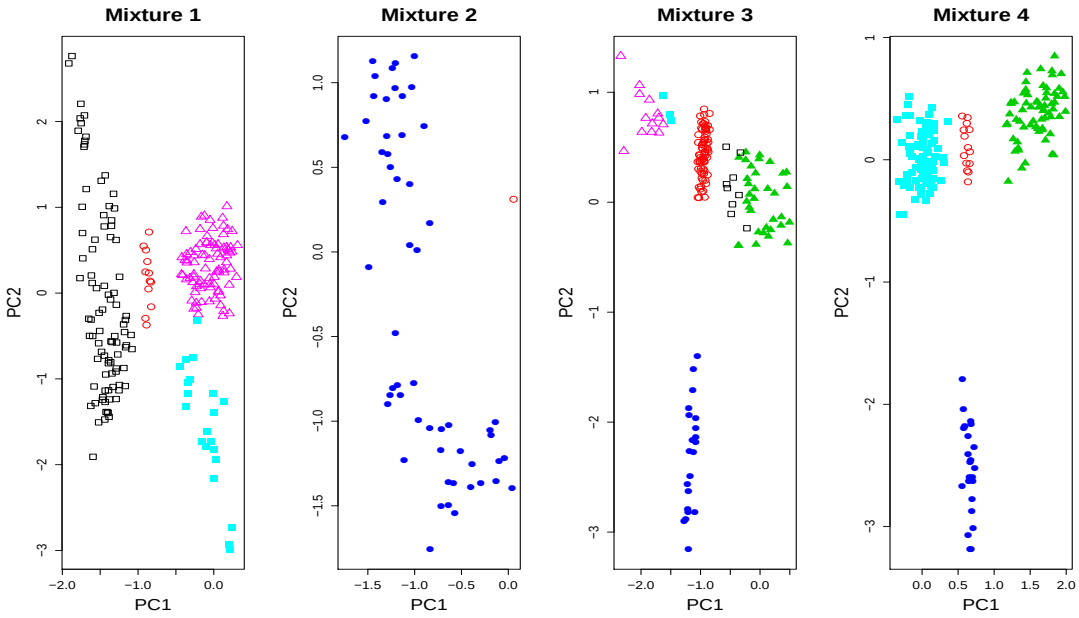

Figure 4.8: Clustering and projection provided by traditional MPPCA for the control chart dataset. ○, ●, △, ▲, ■, □ represent Normal, Cyclic, Increasing trend, Decreasing trend, Upward shift and Downward shift respectively. Note that Normal (○), Upward Shift (■), Cyclic (●), Increasing Trend (△) and Decreasing Trend (▲) are broken into multiple mixtures.

that given by the original MPPCA projection. Also, and more importantly, the Increasing and Decreasing trend groups have been assigned to the same window, while the Upward and Downward shift groups are together in another window. The Normal and Cyclic charts were assigned to separate windows. This allows us to compare spread and shape of observations within each cluster.

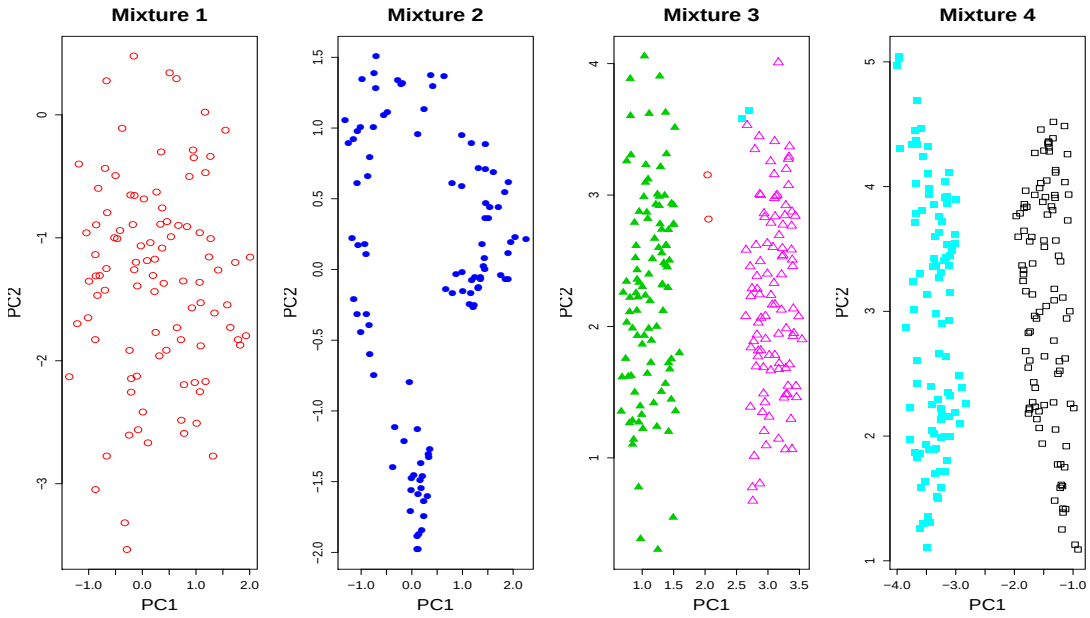

Figure 4.9: Clustering and projection provided by C-MPPCA for the control chart dataset. ○, ●, △, ▲, ■, □ represent Normal, Cyclic, Increasing trend, Decreasing trend, Upward shift and Downward shift respectively. Notice each cluster is assigned, in entirety, to a mixture. In mixture 3 and 4, there are 2 clusters each.

4.4 Semi-supervised Version

A limiting disadvantage of C-MPPCA is that when the number of clusters K is relatively large (e.g., greater than 6), the number of cluster combinations across M mixtures will be high. Thus, the computational time for C-MPPCA may exceed acceptable standards. For cases when users have relevant domain expertise, we propose a semi-supervised version of C-MPPCA that relies more on the clustering information provided by DBSCAN than currently suggested. By using this information, running time decreases dramatically and, ultimately, affords users more flexibility.

Compared to the unsupervised version of C-MPPCA as described in Section 4.2, a semi-supervised version does not need to perform Step 2 (Figure 4.2) to try every combination of clusters to estimate covariance. Instead, in Step 3, we supervise the covariance estimation by incorporating prior knowledge under the Bayesian framework. Let Σ_m denote the true covariance for mixture m. The conjugate prior of covariance Σ_m, when data are modeled by a Multivariate Normal distribution, is the Inverse Wishart distribution. Suppose we have an informed estimate Σ_m^* of Σ_m. We can then parameterize the prior distribution of Σ_m as $p(\Sigma_m|\Sigma_m^*) = \text{IW}(h\Sigma_m^*, p, h)$, where $IW(a,b,c) = |a|^{-c/2}|\Sigma_x|^{-(b+c+1)/2} \exp -\text{tr}(a\Sigma_x^{-1})/2/2^{bc/2}\Gamma_b(c/2)$, and $\Gamma_b(\cdot)$ is a Multivariate Gamma function. Given this Inverse Wishart prior, the posterior distribution of Σ_m is an Inverse Wishart distribution,

$$p(\Sigma_m|\Sigma_m^*, \boldsymbol{x}) \sim \text{IW}\left(\left(\sum_{i=1}^{N} R_{im}\right)\mathsf{S}_m + h\Sigma_m^*, p, \left(\sum_{i=1}^{N} R_{im}\right) + h - p - 1\right),$$

where S_m is the sample covariance matrix for mixture m. The MAP estimator of Σ_m is the weighted average of Σ_m^* and S_m:

$$\text{MAP}(\Sigma_m|\Sigma_m^*, \boldsymbol{x}) = \hat{\Sigma}_m = \frac{h}{h + \left(\sum_{i=1}^{N} R_{im}\right)}\Sigma_m^* + \frac{\left(\sum_{i=1}^{N} R_{ik}\right)}{h + \left(\sum_{i=1}^{N} R_{im}\right)}\mathsf{S}_m. \tag{4.3}$$

In the original C-MPPCA algorithm, we estimate the model parameters by maximizing the log-likelihood:

$$\mathcal{L}(\boldsymbol{x}|\Sigma_m) = \sum_{m=1}^{M} \ln\pi_m p(\Sigma_m|\boldsymbol{x}) \propto -\frac{1}{2}\sum_{m=1}^{M} N\pi_m \ln|\Sigma_m| + \text{tr}(\Sigma_m^{-1}\mathsf{S}_m). \tag{4.4}$$

Under our current Bayesian framework with the prior adjusted estimate of Σ_m, the log-likelihood becomes

$$\mathcal{L}(\boldsymbol{x}|\Sigma_m, \Sigma_m^*) = \sum_{m=1}^{M} \ln\pi_m p(\Sigma_m|\Sigma_m^*, \boldsymbol{x}) \propto -\frac{1}{2}\sum_{m=1}^{M} (N\pi_m + h)\ln|\Sigma_m| + \text{tr}(\Sigma_m^{-1}\hat{\Sigma}_m). \tag{4.5}$$

By comparing Equation (4.5) with Equation (4.4), it is easy to see that the only difference is the change from S_m to $\hat{\Sigma}_m$, with an adjusted degree of freedom term. As in C-MPPCA,

W_m and σ_m^2 are estimated through the eigen-decomposition of the corresponding sample covariance S_m. Now we estimate the parameters, W_m and σ_m^2, through eigen-decomposition of the MAP estimator $\hat{\mathsf{\Sigma}}_m$ instead.

We get the informed covariance estimate $\mathsf{\Sigma}_m^*$ based on DBSCAN. First, we take the clustering result from DBSCAN and calculate the cluster covariance matrices $\mathsf{S}_1, \ldots, \mathsf{S}_K$. Second, we measure all pairwise distances between the cluster covariance matrices using measurements, such as Kullback-Liebler distance or Riemannian distance [Dryden et al., 2009] to get a dissimilarity matrix $\mathsf{D}_{K\times K}$ with $\mathsf{D}_{k,k'} = Dist(\mathsf{S}_k, \mathsf{S}_{k'})$. Finally, we input the pairwise distances D into a hierarchical clustering algorithm to obtain a dendrogram and select the number of distinct covariance matrices accordingly. Our selection is based on our own judgment and interpretation of the dendrogram. For example, Figure 4.10 shows a dendrogram of the simulated dataset mentioned in Section 4.1. The dendrogram supports the truth in that there are two distinctive covariance structures. Thus, $\mathsf{\Sigma}_1^*$ can be estimated by the sample covariance of merged $\circ$ and $\bullet$ groups; and $\mathsf{\Sigma}_2^*$ can be estimated by the sample covariance of merged $\triangle$ and $\blacktriangle$ groups. If we input the two priors into Equation (4.3) and perform C-MPPCA, the final clustering is able to put similar covariance structured mixtures together as displayed in Figure 4.11.

4.5 Discussion

As shown in the above examples, C-MPPCA is able to put similar covariance structured clusters into mixtures and provides clean visualizations. Although we gained such benefits using C-MPPCA, our algorithm has three limitations. Firstly, within each mixture, we project data using PPCA - a method that is sensitive to outliers. For unstructured text

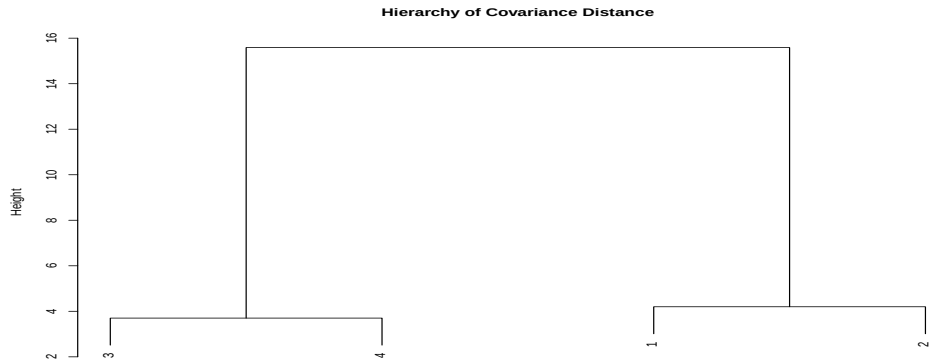

Figure 4.10: Dendrogram generated by Hierarchical clustering based on pairwise distance between cluster covariance, where group 1, 2, 3 and 4 correspond to the •, ∘, △ and ▲ groups in Figure 5.2.

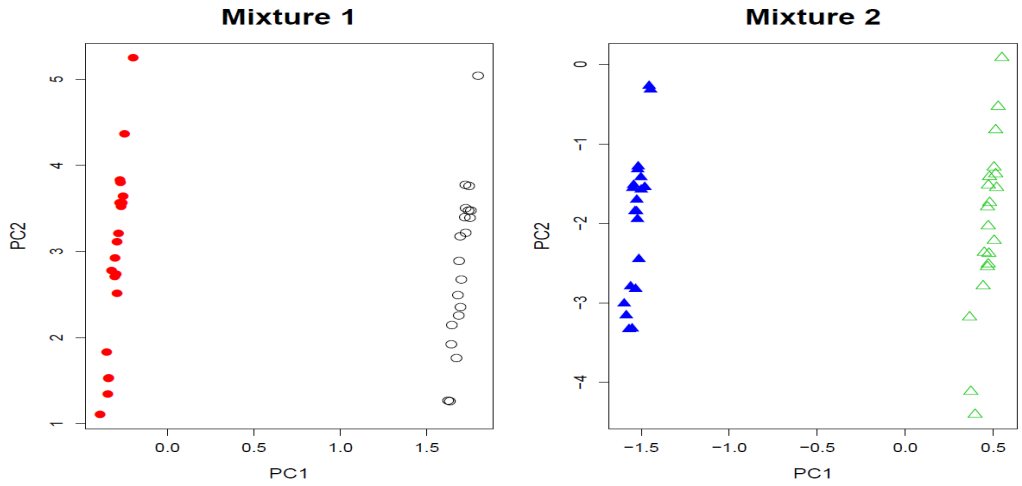

Figure 4.11: Clustering and projection provided by semi-supervised C-MPPCA for the simulated three dimensional data in Figure 5.2.

data with heavy noise, PPCA will likely display several outliers with a cloud of occluded points. Second, we assume that one linear projection per mixture is adaquate to assess data structure. Like MPPCA, C-MPPCA characterizes global nonlinear data structure via

a piecewise combination of linear approximations, i.e., mixtures, to the data manifold. Thus, if a dataset is highly nonlinear, even locally, we need a lot mixtures to assess it adaquately. Yet, as mentioned in Fraley and Raftery [1998], the EM algorithm may be impractical for models with large numbers of mixtures. Additionaly, because PPCA has the tendency to occlude clusters if the within cluster variance is larger than the between cluster variance, we suggest as future work to project mixture components using nonlinear projection methods, such as the Generative Topographical Mapping (GTM) model [Bishop et al., 1998]. Third, in the semi-supervised version of C-MPPCA, user input is constrained. Namely, users can only input their expert knowledge by either specifying the number of distinct covariances and/or deciding which covariances to combine based on a dendrogram. In the future, we plan to add interactive components that draw more from user intuituin and enable users to inject feedback at the observation level; e.g., move observations directly in a display to suggest the presense of clusters or pairwise relationship between observations.

Chapter 5

V2PI-GTM

5.1 Motivation

For noisy or highly nonlinear datasets, we can consider a single nonlinear model. Bishop et al. [1998] proposed a nonlinear manifold learning algorithm called the Generative Topographical Mapping (GTM) model. GTM corresponds to a constrained mixture of Gaussian radial basis functions (nonlinear) in which the components share a common covariance, and the means are constrained to lie on a smooth two-dimensional manifold. GTM is robust to outliers due to a uniform distance separation of the latent lattice points. Additionally, GTM has the ability to visualize the nonlinear manifold in the reduced dimensional latent space which can help reveal the nonlinear local data structure. In the next section, we develop an interactive version of GTM which is capable of dealing with noisy or nonlinear datasets and allows user to tune the model parameters by moving the data points in the display. In this way, the common parameterization assumption in GTM can be relaxed. Moreover, we can achieve more flexible data clustering based on user input without requiring the user to have a good

knowledge about the underlying algorithm.

5.2 The GTM Model

GTM is a non-linear latent variable model, where the latent variables define a manifold that is bent and/or twisted to embed in the high-dimensional data space. The latent space is arbitrary in dimension (so long as it is smaller than the data space), but it is usually two-dimensional for visualization purposes and summarized by a two-dimensional lattice $\boldsymbol{r} = [r_1, \ldots, r_J]$ ($q \times J$ matrix, $q = 2$ usually), as shown on the left hand side of Figure 5.1. Latent points $\boldsymbol{r}$ are nonlinearly mapped to reference points $\boldsymbol{y} = [y_1, \ldots, y_J]$ ($p \times J$ matrix, p is the data dimension) which sit on the manifold in the high-dimensional data space. Given the manifold, we model each observation in the data $\boldsymbol{x} = [x_1, \ldots, x_N]$ ($p \times N$ matrix) by a Multivariate Gaussian distribution, with mean y_j ($j \in \{1, ..., J\}$) and precision matrix $\mathbf{I}_p\beta$ so that

$$p(x_i|y_j, \beta) = (\frac{\beta}{2\pi})^{p/2}\exp(-\frac{\beta}{2}\|x_i - y_j\|^2), \tag{5.1}$$

where $\| \ \|$ denotes the Euclidean norm. The right hand side of Figure 5.1 shows a three-dimensional manifold example, where $\boldsymbol{y}$ (denoted by $\star$) are the centers of the radial symmetric Gaussian distributions (denoted by balls).

The nonlinear mapping in GTM takes the format of a linear regression model, in which y_j is estimated by a linear combination of a set of fixed K basis functions, such that,

$$y_j = \mathsf{W}\boldsymbol{\Phi}(r_j), \tag{5.2}$$

where W is a $p \times J$ transformation matrix. The basis function $\Phi_k(r_j)$ (for $k \in 1, \ldots, K$ and $j \in 1, \ldots, J$) represents a radially symmetric Gaussian kernel,

$$\Phi_k(r_j) = \exp(-\frac{\|r_j - \mu_k\|^2}{2\sigma^2}), \tag{5.3}$$

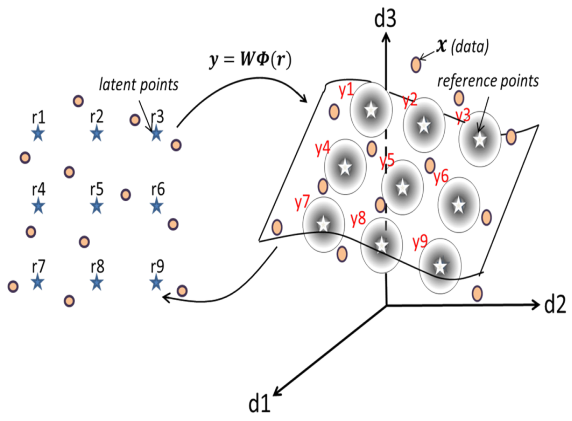

Figure 5.1: This exemplifies how the latent space constructed by $\boldsymbol{r}$ (denoted by $\star$ on the left) and the manifold constructed by $\boldsymbol{y}$ (denoted by $\star$ on the right) in a three-dimensional data space relate. Raw data points $\boldsymbol{x}$ are denoted by yellow $\bullet$.

where σ^2 designates the spread of the radial functions and $\boldsymbol{\mu} = [\mu_1, \ldots, \mu_K]$ ($q \times K$ matrix, K is the number of basis functions) are the Gaussian centers which are typically selected to cover the latent space $\boldsymbol{r}$ uniformly. In this paper, we call the vector functions $\boldsymbol{\Phi}() = [\Phi_1(), \ldots, \Phi_K()]$ "attractors" since they define the degree to which similar points in the high-dimensional space attract toward one another in the low-dimensional visualization.

With this model setting, the objective of GTM is to find the optimal manifold described by $\boldsymbol{y}$ that a) approximates the structure of the observed data points $\boldsymbol{x}$ well (as determined by the data likelihood) and b) when unraveled and flattened by inverting Equation (5.2), creates a reasonable visualization of the high-dimensional data. In the visualization, one reasonable low-dimensional coordinate is selected for each observation x_i based on the posterior multinomial distribution of r given the data x_i, where r can take any value in the lattice $\{r_1, ..., r_J\}$ with probabilities $\{R_{i1}, \ldots, R_{iJ}\}$; i.e., $r|x_i \sim$ multinomial$(R_{i1}, ..., R_{iJ})$. Each posterior probability or "posterior responsibility" (as termed in Bishop et al. [1998])

equals

$$R_{ij} = p(r_j|x_i) = p(x_i|r_j)/\sum_{j'=1}^{J} p(x_i|r_{j'}). \tag{5.4}$$

Given estimates for $R_{i1},\ldots,R_{iJ}$ for $i \in [1,...,N]$, GTM may plot any summary of r, including the posterior expectation of r, the posterior mode of r, or a posterior quantile of r. Typically the posterior expectation is plotted.

In a GTM low-dimensional plot, topological ordering is preserved in that observations close or distant in the high-dimensional space will also appear close or distant, respectively, in the visualization. To see this, we simulated a three-dimensional dataset from five different Multivariate Normal distributions and apply GTM. Figure 5.2 plots the raw data and shows five clusters; there are two groups of clusters that are located in opposite corners of the three-dimensional space. When we apply GTM, with $K = 16$ and $J = 400$, and plot the posterior expectation of r for each observation, we obtain Figure 5.3. Notice that even though GTM is not a formal clustering algorithm, GTM roughly maintains the cluster structure and clearly separates the two groups of clusters. This is because GTM preserves topographical ordering. Observations within a cluster are, by construction, similar to one another and thus appear close to one another in a GTM visualization.

That said, the interpretation of "close" or distance between observations in a GTM visualization requires explanation. Unlike linear methods, such as PCA and MDS, the meaning of one unit in distance is not necessarily uniform across a GTM view. For example, suppose two pairs of low-dimensional points, r_a, $r_{a'}$ and r_b, $r_{b'}$ are equi-distant from one another (dist(r_a, $r_{a'}$) $\approx$ dist(r_b, $r_{b'}$)), but appear in different regions of a GTM visualization. How we infer the degree to which the observations x_a and $x_{a'}$ are similar (or different) to one another, relative to the relationship between x_b and $x_{b'}$, depends upon the high-dimensional manifold. Crudely, if the high-dimensional manifold is flat, dist(x_a,$x_{a'}$)$\approx$dist(x_b,$x_{b'}$). Whereas, if

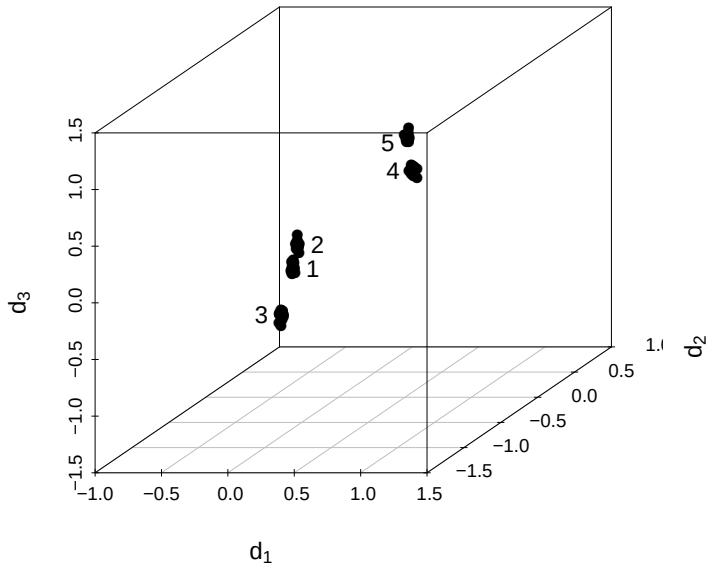

Figure 5.2: A simulated three-dimensional dataset from five different Multivariate Normal distribution is plotted. There are two groups of clusters. The first group includes clusters 1, 2, and 3. The second group includes clusters 4 and 5.

$(x_a, x_{a'})$ are separated by hills, valleys, and/or twists in the manifold and $(x_b, x_{b'})$ are not, dist(x_a,$x_{a'}$)>dist(x_b,$x_{b'}$); observations $(x_b, x_{b'})$ are more similar to one another than $(x_a, x_{a'})$. To help with the interpretation of distance in GTM, Bishop et al. [1998] suggest color-coding a "magnification factor" that informs us about the slope of the high-dimensional manifold at locations in the a GTM display. Formally, the magnification factor is the logarithm of the rate of change between distance or area in the latent space and the corresponding distance or area on the manifold [Svensén, 1998]. In Figure 5.3, we include the magnification factor. Green and yellow represent erratic and flat regions in the manifold, respectively. Notice that

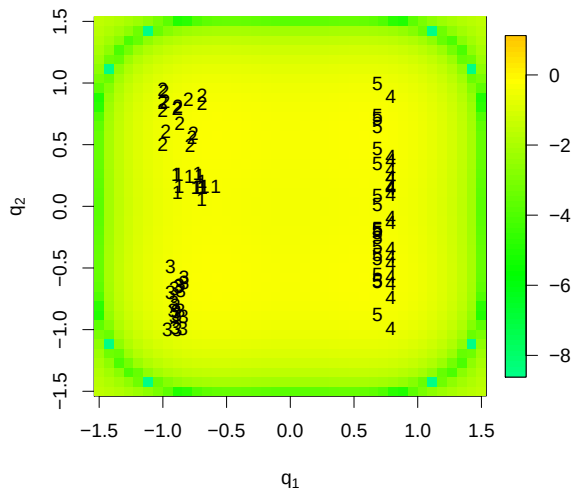

Figure 5.3: This is a GTM display (in latent dimensions q_1, q_2) of the simulated dataset when $K = 16$, $J = 400$. The data points are labeled according to their cluster numbers.

the manifold is fairly flat for the simulated data. Thus, we can rely on the relative distance between observations across the visual space to infer the observed cluster structure in the high-dimensional space.

Also, the current parameterization of GTM could not separate all of the clusters well, and GTM, in its current form, is not flexible. It would be hard for typical users of GTM visualizations to make worth while parametric changes. Thus, in the following sections, we extend GTM to respond to user guidance via the visualization.

5.2.1 Advantages over PCA and SOM

Relative to PCA and SOM, GTM is a complex statistical model with parameters that are hard to interpret. However, with the complexity comes advantages of characterizing unstructured datasets well. In this section, we highlight some of the advantages.

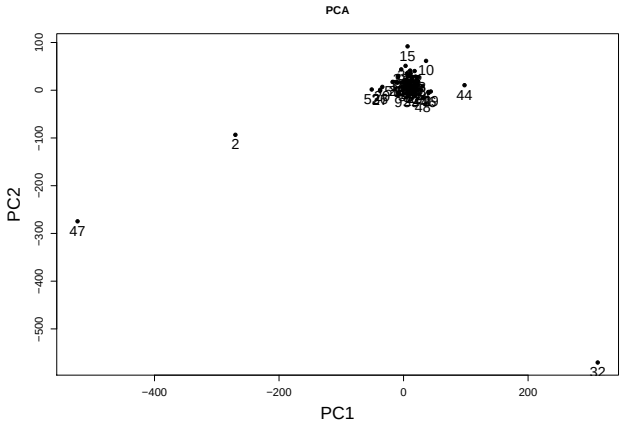

Figure 5.4: PCA projection of the NIH dataset.

GTM is not sensitive to outliers compared to PCA. To show this, we plot a PCA and GTM visualization in Figure 5.4 and Figure 5.5 respectively of a text dataset to which we refer as "the NIH award data". The dataset includes 54 abstracts from the NIH award application documents. Notice the outliers in Figure 5.4 and the natured spread of observations in Figure 5.5. When outliers are present, PCA will display one or two outliers and a cloud of occluded points. However, GTM relies on a latent lattice that covers the low-dimensional space uniformly. Thus, by construction, GTM forces observation to every region of the latent space and reduces the effect of outliers. Thus, the potential effect of the outliers on the visualization is reduced.

GTM is not a projection method, whereas PCA relies on the assumption that one linear projection exists that can reveal useful structure. This assumption may not hold for complex datasets, such as text datasets. A nonlinear methodology, comparable to GTM, is needed to summarize such datasets. From our experience, we find that GTM performs better than standard projection methods in revealing clusters within complex datasets.

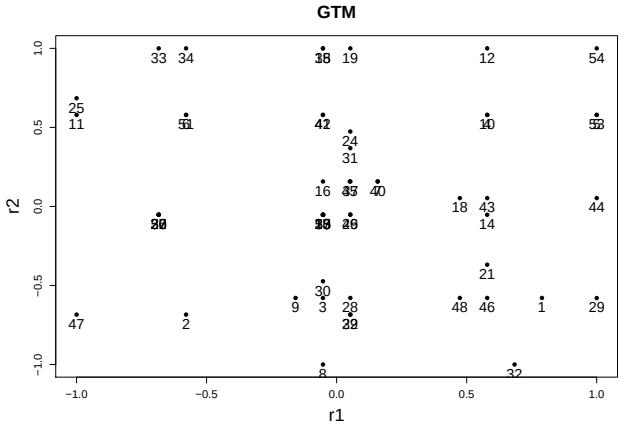

Figure 5.5: GTM mapping of the NIH dataset.

The Self-Organizing Map (SOM) [Kohonen, 1990] is a unsupervised manifold learning algorithm which produces a low-dimensional (typically two-dimensional), discretized representation of the data. The low-dimensional grids are formed by cells known as neurons which have a spatial organization corresponds to reference vectors (prototype vectors) in the high-dimensional data space. During the training stage, the positions of the reference vectors are gradually adjusted in an attempt to preserve neighborhood relationships that exist within the dataset. People have utilized SOM for document visualization and gained certain success, e.g., Kaski et al. [1998] and Chen et al. [1996].

GTM is a probabilistic version of Self-Organizing Map (SOM). The probabilistic nature makes the comparison with other models possible and evaluation of the log likelihood can be used to monitor convergence. It also has the relative advantage of handling the outliers and missing data. As from visualization point of view, SOM use the "winner takes all" strategy, whereas the GTM distributes the responsibility for a data point over a number of latent points. Thus GTM has the flexibility to visualize the posterior mode, posterior mean and various quantiles of latent variable $\boldsymbol{r}$. Additionally GTM makes inference tasks with

assessments of uncertainty possible.

5.2.2 GTM Pitfalls

Although GTM has tremendous advantages, it has two main pitfalls that may lead to unclear visualizations. First, GTM has many tunable parameters that can have big influences on the manifold (hence visualization), including the number of latent variables J; the number of attractors K; the data precision β; and the spread of the basis functions σ^2. Second, there is a limit to which GTM may fold and twist a data manifold [Cruz-Barbosa and Vellido, 2008]. The parameters impact the model fit globally so that it is difficult for GTM to uncover meaningful local structures.

5.3 Visual to Parametric Interaction

To minimize the human-computer communication gap so that the expert knowledge can be integrated directly into visualizations, we consider a form of interaction called Visual to Parametric Interaction (V2PI) [Leman et al., 2011]. In V2PI, users guide display-generating parameters by interacting only with the data. The idea is that, if users change a display to reflect what they know (or conjecture) about the data, the display-generating parameters need to be adjusted as well. Thus, the machinery of V2PI quantifies the user interaction within the display and updates the parameters based on the quantifications. Formally, the steps involved for V2PI are as follows: 1) characterize the data by a model or algorithm to reduce the dimension for visualization, 2) users assess the visualization and communicate their expertise or "provide feedback" about the display by interacting with it (e.g., by adjusting the positions of observations), 3) interpret the feedback quantitatively and tune the

model parameters to reflect the feedback, and 4) reconfigure the visualizations based on the updated parameters. The procedure of V2PI is very similar to BaVA. However, V2PI is a broader framework where Bayesian method is not nessecerily used.

V2PI is most successful when users only participate in step 2; i.e., users only interpret and interact with the data in a visualization. Thus, software is needed to implement steps 1, 3, and 4. To build the software, however, we need the theory and methods to model or summarize the data in a reduced dimensional form, parameterize feedback, and update the display-generating parameters. Since GTM is an insightful approach to visualize data, we develop V2PI theory and methods for GTM. We refer to the new method as V2PI-GTM.

5.4 V2PI-GTM

To overcome the pitfalls and foster making sense of data with human-data interactions, we develop V2PI-GTM. With V2PI-GTM, users can guide the complicated GTM parametrization and assess data from varying perspectives. In this section, we develop V2PI-GTM in three stages. At each stage, we make an improvement to GTM with user interactions, but identify an issue that we address and overcome in the next stage. The third and final stage describes our complete version of V2PI-GTM. We use the simulated data from Section 5.2 to exemplify each stage.

5.4.1 Stage 1: Basic V2PI-GTM Set-up

As we described in Section 5.2, GTM is an nonlinear modeling approach that spatializes data in a visualization so that distance between observations has meaning. Thus, one natural form of interaction is to drag one or more observations so that the spatialization changes; i.e., the

distances between the selected point and the remaining points change. For example, user could drag an observation from location A in Figure 5.6a to location B. Here, we develop V2PI-GTM so that we can quantify an adjusted spatialization and update the parameters of GTM to create a new display of the data.

When users select observations to move, we effectively give users control over two GTM parameters. To explain, suppose a user selects and adjusts the coordinates of one low-dimensional point that represents x^* in the dataset $\boldsymbol{x} = [x_1, \ldots, x_N]$. To interpret the adjustment, we add to the GTM model 1) a low-dimensional coordinate r^* that maps directly to the selected observation x^* and 2) an attractor $\Phi^*()$. This means that we expand sets $\boldsymbol{r}$ and $\boldsymbol{\Phi}$ (as defined in Section 5.2) so that $\boldsymbol{r} = [r_1, \ldots, r_J, r^*]$ and $\boldsymbol{\Phi}() = [\Phi_1(), \ldots, \Phi_K(), \Phi^*()]$, where $\Phi^*() = \exp(-\frac{\|r_j - \mu^*\|^2}{2\sigma^2})$ and $\mu^* = r^*$. We map x^* to r^* by setting the posterior responsibility (Equation (5.4)) that r^* generates x^* via y^* to 1, where y^* is defined by Equation (5.2). By making the additions to GTM, users specify parameters r^* and μ^* by moving the low-dimensional coordinates of the selected observation x^*.

Conditional on the the specifications for r^* and μ^*, the GTM machinery estimates the remaining parameters and plots the data accordingly. For this reason, observations similar to x^* should appear close to r^* as it is moved in the visualization and far, otherwise. For example, with V2PI-GTM in place, we select a point from our simulated data at location A of Figure 5.6a and move the point to location B. As we see in Figure 5.6b, one observation follows. Similarly, we move a point at location C to D. Those most similar follow again.

Ideally, all of the observations within the clusters of the moved points would shift in Figure 5.6b, but they do not. In part, this is due to the drastic change in the manifold, as reflected by the differences in the magnification factors of Figures 5.6a and 5.6b. When K is large ($K = 16$, in this case), there is enough flexibility in the model to add hills and valleys to the

manifold to maintain the original layout of the data. To overcome the new hills and valleys, the attraction provided by the new Φ^* may not be strong enough; it cannot compete with all of the other attractors. This suggests that Stage 1 V2PI-GTM is sensitive to K - the number of attractors.

In the next section we improve V2PI-GTM. We reduce the impact that K has on GTM visualizations.

5.4.2 Stage 2: Scaling

In the second stage, we keep the basic model set-up for V2PI-GTM in that we a) allow users to adjust the location of an observation and b) we expand the GTM model to include new components r^* and Φ^* (and parameters within). Now, however, we change the way by which GTM updates in response to user specifications for r^* and Φ^*.

We develop a method that is similar in spirit to local regression [Loader, 1999] for high-dimensional data. We modify the generative Gaussian distribution (Equation (5.1)) for which y^* is the Gaussian center, so that the distribution is a function of $\|x^* - x_i\|$. That is, we scale $x_i - y^*$ in the likelihood (Equation (5.1)) by the square-root of a scaling function V which depends upon $\Delta_i = \|x^* - x_i\|/h$,

$$p(x_i|y^*, \beta) = (\frac{\beta}{2\pi})^{p/2} \exp(-\frac{\beta}{2} V(\Delta_i) \|x_i - y^*\|^2),$$

where h is user-defined. From our experience, suggested $V(\Delta)$ functions include Δ, Δ^2, Δ^3, $\exp(-\frac{1}{\Delta})$ and $\exp(-\frac{1}{\Delta^2})$.

This local regression scaling works in the following way. When a data point x' is near x^* in the data space, $V(\Delta')$ will be small and, due to the increase of the posterior responsibility that r^* generates x' (Equation (5.4)), the coordinates for x' will gravitate toward x^* in the

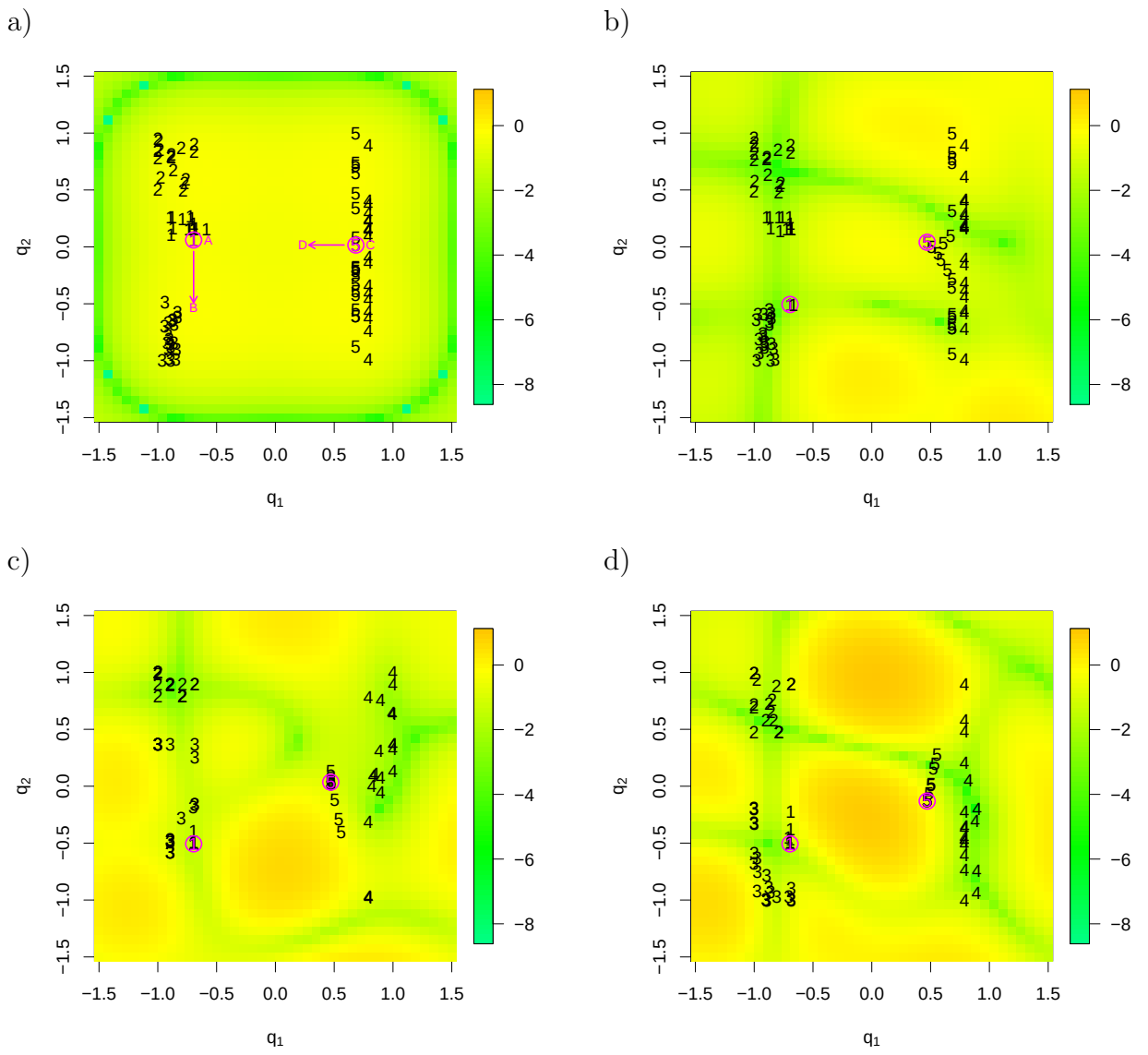

Figure 5.6: Figure a is the same as Figure 5.3, but shows how a user may interact. A user may move one point from location A to location B and another point from location C to location D. Figures b, c, and d show respectively how the observations respond (or do not respond) to the move when stages 1, 2, and 3 of V2PI-GTM are in place.

two-dimensional view. Similarly, points that are far from x^* in the data space will push further away. The degree to which points push and pull depends on h. We found that good

specifications for h (in our definition of Δ) relate to how many observations users expect will follow or are similar to the selected observation x^*. We recommend that h equals a distance such that the number of nearest neighbors within h from x^* equals the expected number. For example, if a user wants to make sure that x^* attracts at least 3 matches, h equals the distance between x^* and the third nearest data point.

To show the effectiveness of including $V(\Delta')$, we apply GTM to the simulated data again. We also move the same point shown from location A to B (Figure 5.6a. With $V(\Delta) = \Delta^3$ and h set to 20, we see how the data respond to the adjusted observation in Figure Figure 5.6c. In particular, notice that, unlike Figure 5.6b, all the points in cluster 1 move. Similarly, when we move a point from location C to D. Again, all of the points in its cluster move.

Unfortunately, there are two drawbacks for this stage of V2PI-GTM. First, the transition from Figure 5.6a to Figure 5.6c is not smooth. There can be abrupt changes in the visualization, no matter what scaling function $V()$ we use. Second, except for the points that are similar to the adjusted point r^*, all of the observations drift away from their original places. With this side effect, data explorations could be become complicated. Users could loose track of the relationship between the new cluster locations and their original locations, so that a sequential data exploration becomes hard to develop.

The drawbacks are due to global changes in the manifold y. The way by which GTM is parameterized currently results in users making global manifold changes when they only want local. In the next section, we improve V2PI-GTM again.

5.4.3 Stage 3: Mixtures of Manifolds

To minimize abrupt changes in the stage 2 V2PI-GTM, we extend GTM to include mixtures of manifolds. Let $y_j^{(c)}$ and $y_j^{(u)}$ represent the current and user-adjusted manifolds, respectively. We define the V2PI-GTM estimate for the manifold, $y_j^{(c+1)}$, by

$$y_j^{(c+1)} = \delta_j y_j^{(c)} + (1 - \delta_j) y_j^{(u)},$$

where $\delta_j = ||r_j - r^*||/b$ and $b = \max\{||r_1 - r^*||, \ldots, ||r_m - r^*||\}$ so that $\delta_j \in [0, 1]$. This definition for $y_j^{(c+1)}$ controls the visualization so that only the regions of interest respond to user interactions; points in the areas that are distant from dragged observations do not change their positions.

We apply one final application of V2PI-GTM to the simulated data. Again, $K = 16$, $V(\Delta) = \Delta^3$ ($h = 20$), and we move the same set of points from location A to B and from location C to D. Now, however, we have mixtures of manifolds in the GTM model. The visualization updates smoothly as r^* moves. In the final view (Figure 5.6d, the circled points are able to attract the data points which belong to their respective clusters. The other clusters stay at their original positions, except for several observations from cluster 3 that are repelled (as they should be).

This final stage of V2PI-GTM describes our complete approach for allowing users to guide the parameterization of GTM via dragging one observation at a time. It might seem, at face value, that V2PI-GTM is similar to a visual analytic method called Dust and Magnets (DnM) that was developed by Yi et al. [2005]. In DnM, users drag or shake nodes that represent variables in the dataset and watch as relevant observations (denoted by particles of iron dusts) follow the nodes. However, V2PI-GTM differs in two fundamental ways. First, the nodes that users adjust in V2PI-GTM represent individual observations, not variables.

Thus, users need only to understand the relationship between two or more observations to inject their expertise or conjectures about the data into the visualization. Users are not expected to know the relative importance of entire dimensions in the dataset. Second, when users drag observations they are effectively comparing all (not two) of the variables in the dataset simultaneously, relative to the observation moved. In this simulated example, the three dimensions work jointly to define the clusters. Hence, as the observations moved across the latent space, the clusters stayed together. However, in stage 2, we changed how the variables were weighted to define the clusters and snapped cluster 1 in two. If we wanted, we could use features of GTM to "tag" the latent space and learn the heavy-weighted variables. In the next section, we take advantage of tagging the latent GTM space and and apply the final stage of V2PI-GTM to a real world dataset.

5.4.4 V2PI-GTM Summary

In summary, Interactive GTM connects user interaction conducted in the low dimensional display with the high dimensional data points through adding extra latent point r^* and attracting function centered at r^*. The newly added latent point r^* represents the low-dimensional coordinate for the moving point x^*. There is a reference point y^* in the data space corresponds to r^*. To make y^* be the high dimensional representation of x^*, we assign the posterior responsibility of y^* generating x^* to be 1.

To propagate the effect of moving r^* to the remaining visualization, we take a local regression approach. We multiply a weight inside the exponential part of the generative Gaussian distribution in Equation (5.1). The weight is a function of the distance between the moving point x^* and all the other raw data points, so that it favors the similarity and penalize the dissimilarity with the moving point in the high d space.

To make sure that the visualization update smoothly without abrupt global manifold change, we estimate the manifold as a mixture of two manifolds: the current and user-adjusted manifold. In this way, only the regions of interest respond to user interactions; areas that are distant from the dragged observations do not change.

5.5 Text Mining Application

Exploring a collection of documents can be a time consuming, complex task. Often analysts use keyword searches or document matching [S. Weiss, 2005] to identify patterns in the dataset. Searching for keywords (and the documents that contain them) is simple and fast, but lacks rigor. For example, keyword searches may identify documents with similar keywords, but used in different contexts; miss documents that contain combinations of the keywords; or prioritize words inappropriately for the purposes of the data exploration. Document matching, on the other hand, groups documents based on many keywords, phrases, and/or query topics. It is an improvement over keyword searches, but can be hard to implement. GTM-V2PI provides a natural, interactive, and visual way to document match.

In this section, we illustrate the application of V2PI-GTM within the context of text mining. We have a collection of 54 abstracts with 2365 entities (words) from proposals funded by the National Institute for Health (NIH). Based on the abstracts, NIH Program Managers would like to assess how they allocate their funds to varying research areas. However, how 'areas' are defined is ambiguous. Goals of proposals can overlap regardless of the fields with which, say, the proposal principal investigators may associate. For this reason, we apply V2PI-GTM to explore the data and learn about the NIH priorities.

Before we apply V2PI-GTM, however, we pre-process the data. First, we apply standard text

mining procedures to remove uninformative entities, such as, stop words or redundant words (e.g., run and running become run). Second, we rank the entities using a new algorithm that we call the Imp-Index (ImpI) and select those that are top ranked. ImpI is based on the Gini coefficient [Gini, 1921] and described in Appendix A. Given the pre-processing, we reduce the data from 54×2365 to 54×1000 and each word per document has a continuous measure of importance - the ImpI.

Based on the ImpI metric, spatial visualization methods, such as GTM, could be used to explore text data. Thus, we apply GTM for $K = 16$ and $J = 400$ to obtain an initial display of the proposals, shown in Figure 5.7a. Notice a manifold with hills and valleys separates the data into four clusters. We label the clusters A, B, C, and D. Suppose that a program manager has particular interest in Abstract 7 (highlighted in pink in Figure 5.7a) from cluster A. Abstract 7 is about developing new brain tumor therapies. The program manager would like to assess how many other proposals were granted that were similar to Abstract 7.

It is possible that cluster A uses an ideal set of key words to group the proposals, thus the program manager is done exploring the data. However, it is also possible that entities from Abstract 7 in which the program manager is not interested are weighted too heavily and determine the clusters. Thus, before the program manager uses V2PI to interact with the data, it would be helpful to describe the current clusters and define or "tag" the latent space so that the program manager can make informed interactions.

5.5.1 Describing the Latent Space

With GTM-V2PI, users may drag observations with undirected and directed intents. In Section 5.4, we simply wanted to see how the data respond to a moved observation - any

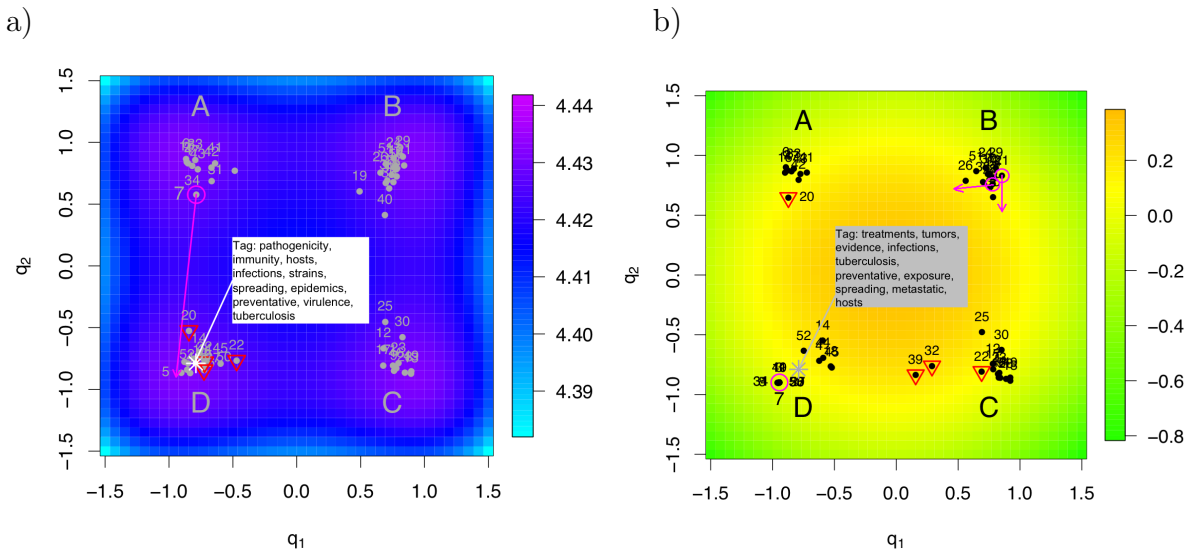

Figure 5.7: We provide a GTM display of the NIH abstracts (labeled by their identification numbers) before and after user interaction in Figures a and b, respectively. The interaction is portrayed by the pink arrow in Figure a; Abstract 7 was moved to a location near cluster D. In addition, to labeling and learning about four clusters in the data (marked by A, B, C, and D), we also tagged the latent GTM space. After the interaction, we see that the clusters grouped differently and the meaning of the latent space changed. Also, the manifold changed dramatically.

move and any observation. Our intent was undirected. For complicated datasets, repeated undirected interactions could inform users about the number of structures in the data, the types of structures in the data, the sensitivity of the data to GTM, etc.. For users with directed intent, e.g., to compare specific documents, it is helpful to understand the space in which GTM plots the data. In this example, a user may benefit from learning about how the clusters differ and what regions of the latent space mean in terms of the variables in the dataset.

To understand how the clusters differ, we determine the words that both overlap the least within each cluster and have the highest ImpI's. We first apply k-means [Hartigan and Wong, 1979] (which is good enough for clustering two dimensional data) to the low-dimensional data coordinates to determine cluster memberships. For each cluster, we sum the ImpI vectors across the documents and rank the entities based on the ImpI sum. Entities ranked highest are those that 1) have importance in the corpus (as determined by the ImpI) and 2) have occurred most frequently in the document cluster. Given top-ranking termlist from each cluster, we delete those shared by all four clusters so that we have unique key words that describe each cluster. We list these words in Table 5.5.1. Cluster A represents proposals that include brain related cancer studies and their clinical applications. Cluster B is related to cell based studies; e.g., stem cells, neuro-degenerative diseases, and neural circuits. Cluster C represents proposals that address genomic and transcriptomic research problems. Cluster D represents proposals about infectious diseases, such as tuberculosis, and immunity.

Knowing how the clusters differ is helpful and correlates strongly with the meaning of the latent space around the clusters. However, we can assess or tag points in the latent space directly using the mechanics of GTM. To do so, we select spot, r^+, in the visualization and use Equation (5.2) to estimate its corresponding location on the manifold, y^+. The estimate

Table 5.1: This table lists the Top 10 keywords that either differentiate clusters A, B, C, and D or are shared among all of the clusters in Figure 5.7.

Cluster A	tumors, brains, stem, treatments, patients, generations, drugs, ordering, controlling, therapeutics
Cluster B	stem, neuronal, brains, proteins, deliveries, regulations, neural, patients, differentiation, expression, treatments
Cluster C	stem, genetically, regulations, drugs, structurally, proteins, genomics, epigenetics, RNAs, complexities
Cluster D	Infections, treatments, tuberculosis, expression, patients, drugs, strains, resistance, vaccination, immunity
Shared	cells, functionalization, diseases, developments, genes, cancerous, studying, researchers, proposing, mechanisms, specification

y^+ will be a 1000×1 vector of ImpI's. We could report all or a subset of the 1000 entities. For this exploration, we tag r^+ by reporting the entities that were top ranked. For example, in Figure 5.7a, we pick a spot r^+ (represented by a pink circle) that locates roughly at the center of cluster D. According to its corresponding location on the manifold, the top 10 keywords include: pathogenicity, immunity, hosts, infections, strains, spreading, epidemics, preventative, virulence, tuberculosis. As expected, several of these words overlap with the words describing cluster D.

Based on the meaning of the clusters and one spot in the latent space, the program manager moves Abstract 7.

Table 5.2: Descriptions of Abstracts 20, 22, 32 and 39 in Figure 5.7.

20	discusses diagnosis of HIV infection in patients who live with limited access to therapeutic treatments
22	discusses expression characteristics of a drug-resistant gene
32	discusses varying yeast strains
39	discusses Lymphocyte Homing

5.5.2 Cluster reorganization of NIH dataset

From the previous section, we know that Abstract 7, shares the following keywords with cluster A: tumors, brains, cancerous, therapeutics and chemotherapy. However, Abstract 7 also shares some keywords with cluster D; e.g., treatments, strategies, patients, drugs, resistance, clinically. The program manager is particularly interested in the latter set of key words. Thus, the manager drags Abstract 7 to the lower left corner of the display (shown by the pink arrow in Figure 5.7a near the location of cluster D and observes how the remaining documents react in Figure 5.7b.

As expected, many documents in cluster D gravitate toward Abstract 7 or stay close to their original locations in Figure 5.7b. However, some repel. Abstracts 20, 22, 32 and 39 that were originally in cluster D, relocate to new regions of the visualization. Table 5.5.2 describes each abstract. Abstracts 20 and 22 repelled from Abstract 7 and shifted respectively to cluster A and C because the redefined-manifold down-weighted their shared entities with the original cluster D and up-weighted their unshared entity tumor. Similarly, Abstracts 32 and 39 separated slightly from cluster D and gravitated toward cluster C because they have a few keywords in common with each cluster, but not enough to place them in either corner.

To further support the re-clustering of the abstracts, we assess changes in the manifold as described by the magnification factor. Overall, the magnification factor is lower in Figure 5.7b than in Figure 5.7a which suggests that the new manifold is flatter then the original. Clusters in Figure 5.7 are located in hilly regions on the manifold, whereas, clusters in Figure 5.7b are in flat, stable regions. Thus, clustered observations in Figure 5.7b are closer to one another in the high-dimensional space than clustered observations in Figure 5.7a. This suggests that the observations surrounding Abstract 7 are more similar to one another after the interaction than before. Had this not been the case, the program manager would have reason to re-consider the original clustering of the data.

The program manager could also continue exploring the data and select other abstracts to contrast and compare. For instance, although the abstracts in group B are all related to cell-based studies, they discuss different types of cells; e.g., Abstract 8 (highlighted with a pink circle in Figure 5.7 b) is about the analysis of functional neural cells, while Abstract 21 is about developing new materials for stem cell niche. The program manager may want to explore group B in more detail to find topics associated with specific cell types. Thus, the manager drags Abstract 8 and Abstract 21 a little bit away from their original locations (shown by the pink arrow in Figure 5.7 b) to separate these two documents.

Interestingly, as a result of point dragging, abstracts in group B separate into three subclusters (as labeled in Figure 5.8). Abstract 8 attracts similar abstracts to form subcluster 1 which relates to studies on neural cells or neurons and their effects on functions of the brain. Subcluster 2 is formed by abstracts similar to Abstract 21 and represents proposals about stem cells and their differentiation. The three leftover proposals (Abstracts 16, 38, 48) form subcluster 3 which is about cellular delivery and nanoscale approaches. We list the top-ranking keywords for each subcluster in Table 5.3. Additionally, Abstract 31 (denoted

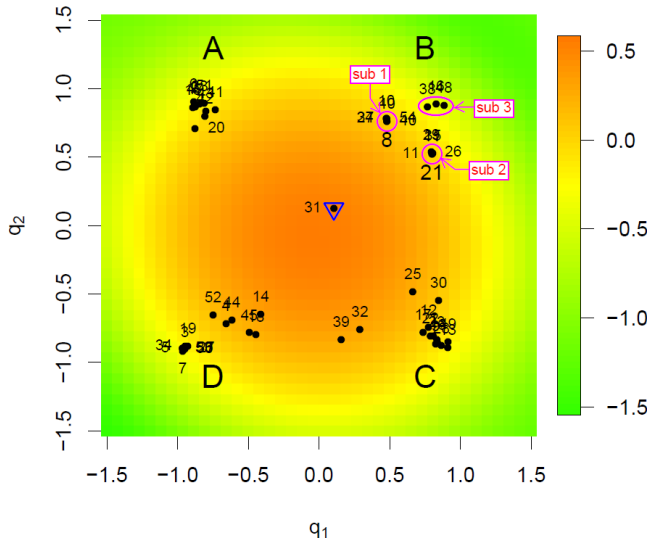

Figure 5.8: GTM display of the NIH abstracts after dragging apart Abstract 8 and 21. After the interaction, we see that group B is split into three subclusters.

by a blue triangle in Figure 5.8) talks about the generation of tumor stem cell lines for directed therapeutics of brain cancer. It was originally attracted to the lower left corner by the movement of Abstract 7 in the last iteration of interaction, because it is related to developing brain tumor therapies. Now, it gravitates toward the upper right corner due to attraction from Abstract 21, since it is also about stem cells.

As can be seen from the current iteration of interaction, V2PI-GTM can support continuous data explorations. Thus, users can built upon what they learn from the data as they explore and refine information in the display. Here, the user found four clusters and subclusters within.

Table 5.3: This table lists the Top 10 keywords that either differentiate subclusters 1, 2 and 3 or are shared among all of the subclusters in Figure 5.8.

Subcluster 1	neurons, function, genes, brains, patients, mechanisms, expression, changing, neural, activation
Subcluster 2	stem, cellular, differentiation, specification, approaches, tissues, cues, regulations, brains, therapies
Subcluster 3	deliveries, cellular, genes, mechanisms, specification, approaches, genetically, targeting, treatments, intracellular
Shared	cells, developments, diseases, useful, functionalization, studying, proposing, researchers, humans, Health

5.6 Discussion

Among current visualization algorithms, GTM has had success in visualizing unstructured data [Kaban, 2005, Olier et al., 2010, Cruz-Barbosa and Vellido, 2008]. It is robust to outliers, and offers more flexibility than standard linear projection methods. However, GTM is complicated by its extensive, confusing parameterization, which prohibits exploration by direct parametric interaction.

In this section, we modified the original GTM to a) take advantage of its strong visualization capability, and b) overcome its drawbacks. We develop our methods within an interaction framework: Visual to Parametric Interaction (V2PI). In V2PI-GTM, analysts can freely organize observations directly in the display and watch how the GTM machinery relocates the remaining observations in response. In a way, V2PI replaces the role of quantitative experts and protects users from the mathematics of strict definitions of the parameters. The communication between users and models is through the visual metaphor. We show the

utility of V2PI-GTM in a text mining application.

Future work in V2PI-GTM would be to enable more interaction, in addition to dragging. For example, it would be interesting to explore the parameterization of interactions, such as, filtering, linking, highlighting, and zooming. We could also parameterize the dragging of multiple data points at one time (rather from sequentially). The more ways we can provide users to interact with GTM, the more expert knowledge we can inject into data explorations. Also, since we maintain a probabilistic framework in GTM, subsequent analyses that require formal inferential statements (e.g., inferences with assessments of uncertainty) are a natural progression following a data exploration with the probabilistic version of our framework.

Chapter 6

Future Work

In this section, we describe potential developments for the work presented in this dissertation. Such developments are related to BaVA and V2PI research topics.

We made great advancements by combining Bayesian statistics with Visual Analytics to explore data. However, we can still make improvements. First, beyond dragging observations in the low-dimensional display, we would like to explore other forms of interactions, such as, zooming, filtering, highlighting and linking. This would entail developing software to allow the interactions and finding ways to parameterize the feedback. Second, when we drag observations to inject feedback, the final locations of the points is arbitrary. We would like to include the point locations in our algorithms. Third, it would be helpful to understand how well BaVA-updated visualizations match users' mental maps. This would foster the quantitative and qualitative aspects of sense making. Finally, the current versions of MPPCA and GTM are formulated for dealing with complete continuous data. We would like to extend these models to a) deal with mixed discrete and continuous data and b) handle missing data.

There are several advancements we would like to make to V2PI-GTM. First, the parameters

in GTM are complex and hard to interpret. An alternative framework for GTM can be considered. For example, Gaussian processes have fewer parameters that can be utilized in place of standard regression models to help define the non-linear manifold. Variants of SOM are also good candidates. Second, we need to speed up the V2PI-GTM process. V2PI-GTM is a relatively slow algorithm for visualization purposes. The time complexity for GTM is $O(MNp)$ (M is the number of latent points, N is the observation number, p is the data dimensionality) and the calculation of all pairwise distances to update parameters is time consuming (especially for large datasets). Also, GTM is based on the EM algorithm which requires several iterations to converge. In the text mining example (Section 5.5), it took 0.3 minutes per iteration of updating. To improve the computational time of V2PI-GTM, we will try online versions of GTM [Bishop and Svensén, 1998] where data points are presented one at a time. When points are presented sequentially, we do not need to wait until all of the data points have been considered to update the model parameters. We will also try to avoid EM by developing a sampling algorithm and program in a fast computer language that enables parallel computing.

Chapter 7

Appendix

Importance Index (ImpI)

ImpI is similar to the commonly used term frequency - inverse document frequency (tf-idf) [Karen, 1972] in the sense that it considers both the frequency and uniqueness of words that are shared across documents. Consider an term e_i that occurs f_{ij} times in document d_j. The ImpI for e_i is given by

$$ImpI_i = \frac{\sum_{j=1}^{N}\sum_{k=1}^{N}|f_{ij} - f_{ik}|}{2N^2\mu_i},$$

where N is the total document number and $\mu_i = \frac{\sum_{j=1}^{N} f_{ij}}{N}$ is the average frequency for term e_i. ImpI ranges between 0 and 1. Entities that occur equally frequently in all the documents have ImpI=0 and entities that occur in only one document has ImpI=1. ImpI can be used to rank and hence filter terms. We selected the 1000 entities with the highest ImpI's and describe each proposal by a 1000×1 term frequency vector with frequency f_{ij} weighted by ImpI. The weighted frequency is defined as $f^* = ImpI_i \times \frac{f_{ij}}{\|d_j\|} \times \frac{\|e_i\|}{F}$, where $\|d_j\| = \sum_{i=1}^{K} f_{ij}$, $\|e_i\| = \sum_{j=1}^{N} f_{ij}$ and $F = \sum_{i=1}^{K}\sum_{j=1}^{N} f_{ij}$. For document d_j, the weighted vector is given by $\boldsymbol{f}_j^* = (f_{1j}^*, \ldots, f_{Kj}^*)$; the vector element equals zero when the proposal does not include the

entity.

Chapter 8

Bibliography

Bibliography

ADVISOR Solutions Inc. SeeIT, 2007. http://www.advizorsolutions.com/.

C. Ahlberg. Spotfire: an information exploration environment. *SIGMOD Record*, 25:25–29, 1996.

R. J. Alcock and Y. Manolopoulos. Time-series similarity queries employing a feature-based approach. In *Proceedings of the 7 th Hellenic Conference on Informatics*, 1999.

R. Amar, J. Eagan, and J. Stasko. Low-level components of analytic activity in information visualization. In *2005 IEEE Symposium on Information Visualization*, pages 111–117. IEEE Computer Society Press, 2005.

F. J. Anscombe. Graphs in statistical analysis. *The American Statistician*, 27(1):17–21, 1973.

C. Biernacki, G. Celeux, and G. Govaert. Assessing a mixture model for clustering with the integrated completed likelihood. *IEEE transactions on pattern analysis and machine intelligence*, 22(7):719–725, 2000.

C. M. Bishop and M. Svensén. Developments of the generative topographic mapping. *Neurocomputing*, 21:203–224, 1998.

C. M. Bishop, M. Svensén, and C. K. I. Williams. GTM: the generative topographic mapping. *Neural Computation*, 10(1):215–234, 1998.

Christopher M. Bishop. *Learning in Graphical Models*. MIT Press, 1999.

A. Buja, D.T. Lang, and D.F. Swayne. GGobi: Evolving from XGobi into an extensible framework for interactive data visualization. *Journal of Computational Statistics and Data Analysis*, 43(4):423–444, 2003.

A. Buja, D. F. Swayne, M. Littman, N. Dean, H. Hofmann, and L. Chen. Interactive data visualization with multidimensional scaling. *Journal of Computational and Graphical Statistics*, 17(2):444–472, 2008.

T. Calinski and J. Harabasz. A dendrite method for cluster analysis. *Commun. Stat.*, 3: 1–27, 1974.

S. K. Card, J. D. Mackinlay, and B. Shneiderman. *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufmann Publishers, San Francisco, 1999.

G. Celeux and G. Govaert. A classification EM algorithm for clustering and two stochastic versions. *Comput. Statist. Data Anal.*, 14:315–332, 1992.

R. Chang, M. Ghoniem, R. Kosara, W. Ribarsky, Jing Yang, E. Suma, C. Ziemkiewicz, D. Kern, and A. Sudjianto. Wirevis: Visualization of categorical, time-varying data from financial transactions. In *IEEE Symposium on Visual Analytics Science and Technology: VAST 07*, pages 155–162, 2007.

H. Chen, C. Schuffels, and R. Orwig. Internet categorization and search: A self- organizing approach. *Journal of Visual Communication and Image Representation, Special Issue on Digital Libraries*, 7(1):88–102, 1996.

W.S. Cleveland. *Visualizing data*. Hobart Press, New Jersey, 1993.

C. Conde, A. Ruiz, and E. Cabello. PCA vs low resolution images in face verification. In *Proceedings of the 12th International Conference on Image Analysis and Processing*, pages 63–67. IEEE Computer Society, 2003.

R. Cruz-Barbosa and A. Vellido. Unfolding the manifold in generative topographic mapping. *Lecture Notes in Computer science, HAIS*, 5271:392–399, 2008.

Abasah Dadzie, Vitaveska Lanfranchi, and Daniela Petrelli. Seeing is believing: Linking data with knowledge. *Information Visualization*, 8(3):197–211, 2009.

Ian L. Dryden, Alexey Koloydenko, and Diwei Zhou. Non-euclidean statistics for covariance matrices, with applications to diffusion tensor imaging. *Annals of Applied Statistics*, 3(3): 1102–1123, 2009.

Q. Du and J. E. Fowler. Low-complexity principal component analysis for hyperspectral image compression. *Int. J. High Perform. Comput. Appl.*, 22(4):438–448, 2008.

M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Sciences*, 95: 14863–14868, 1998.

Alex Endert, Chao Han, Dipayan Maiti, Leanna House, Scotland Leman, and Chris North. Observation-level interaction with statistical models for visual analytics. In *IEEE VAST*, pages 121–130. IEEE, 2011.

M. Ester, H. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, pages 226–231, 1996.

C. Fraley and A. E. Raftery. Mclust: Software for model-based cluster and discriminant analysis, 1998.

C. Fraley and A.E. Raftery. Model-based clustering, discriminant analysis, and density estimation. *Journal of the American Statistical Association*, (97):611–631, 2002.

Andrew Gelman. Why tables are really much better than graphs. *Journal of Computational and Graphical Statistics*, 20(1):3–7, 2011a.

Andrew Gelman. Rejoinder for discussion: Why tables are really much better than graphs. *Journal of Computational and Graphical Statistics*, 20(1):36–40, 2011b.

Corrado Gini. Measurement of inequality of incomes. *The Economic Journal*, 31(121): 124–126, 1921.

R. Gottumukkal and V. K. Asari. An improved face recognition technique based on modular pca approach. *Pattern Recogn. Lett.*, 25(4):429–436, 2004.

R. L. Graham, D. E. Knuth, and O. Patashnik. *Concrete Mathematics*. Addison-Wesley, MA, 1988.

D. Guber. Getting what you pay for: The debate over equity in public school expenditures. *Journal of Statistics Education*, 7(2), 1999.

J. A. Hartigan and M. A. Wong. A K-means clustering algorithm. *Applied Statistics*, 28: 100–108, 1979.

L. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 2:193–218, 1985.

S. Imran, S. Bajwa, and I Hyder. PCA based image classification of single-layered cloud types. *Journal of Market Forces*, 1(2):3–13, 2005.

P. Isenberg and D. Fisher. Collaborative brushing and linking for co-located visual analytics of document collections. *Computer Graphics Forum*, 28(3):1031–1038, 2008.

Dong Hyun Jeong, Caroline Ziemkiewicz, Brian Fisher, William Ribarsky, and Remco Chang. iPCA: An interactive system for PCA-based visual analytics. *Comput. Graph. Forum*, pages 767–774, 2009.

A. Kaban. A scalable generative topographic mapping for sparse data sequences. In *Proceedings of International Conference on Information Technology: Coding and Computing (ITCC'05)*, volume 1, pages 51–56, 2005.

S. Karen. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28(1):11–21, 1972.

S. Kaski, T. Honkela, K. Lagus, and T. Kononen. WEBSOM-self-organizing maps of document collections. *Neurocomputing*, 21:101–117, 1998.

D. Keim, G. Andrienko, J. D. Fekete, C. Görg, J. Kohlhammer, and G. Melançon. Visual analytics: Definition, process, and challenges. *Information Visualization*, pages 154–175, 2008.

Teuvo Kohonen. The self organizing map. In *Proceedings of IEEE*, volume 78, pages 1464–1480, 1990.

Scotland Leman, Leanna House, Dipanyan Miti, Alex Endert, and Chris North. Visual to parametric interaction. 2011.

C. Loader. *Local Regression and Likelihood*. Springer-Verlag, 1999.

P. Lyman and H. R. Varian. How much information, uc berkeley idc digital universe study, 2003.

J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceeding of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, 1967.

I. Olier, A. Vellido, and J. Giraldo. Kernel generative topographic mapping. In *Proceedings of the European Symposium on Artificial Neural Networks Computational Intelligence and Machine Learning*, 2010.

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

Chuck Pirrello. Effective visualization techniques for data discovery and analysis, 2010.

T. Zhang F. Damerau S. Weiss, I. Indurkhya. *Text Mining: Predictive Methods for Analyzing Unstructured Information*. Springer, New York, 2005.

SAS Institute Inc., 2007. Cary, NC.

S. Schiffman, L. Reynolds, and F. Young. *Introduction to Multidimensional Scaling: Theory, Methods, and Applications*. Academic Press, 1981.

G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6:461–464, 1978.

L. Sirovich and M. Kirby. Low-dimensional procedure for the characterization of human faces. *Journal of the Optical Society of America*, 4(3):519–524, 1987.

Jan Smit. Key features jmp 8, 2010. http://www.cosinus.nl.

J. Stasko, C. Görg, and Z. Liu. Jigsaw: Supporting investigative analysis through interactive visualization. *Information Visualization*, 7(2):118–132, 2008.

M. Svensén. *GTM: the generative topographic mapping*. PhD thesis, Aston University, April 1998.

J.J. Thomas and K. Cook, editors. *Illuminiating the Path*. National Visualizations and Analytics Center, 2005.

Michael E. Tipping and Christopher M. Bishop. Mixtures of probabilistic principal component analyzers. *Neural Computation*, 11:443–482, 1999a.

Michael E. Tipping and Christopher M. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society, Series B: Statistical Methodology*, 61:611–622, 1999b.

J. W. Tukey. *Exploratory Data Analysis*. Addison Wesley, 1977.

C. Ware. *Information Visualization: Perception for Design*. San Diego, CA, USA, 2000.

Graham Wills. Charts versus tables: A rematch. *Journal of Computational and Graphical Statistics*, 20(1):28–35, 2010.

P. C. Wong, K. Schneider, P. Mackey, H. Foote, G. Chin, R. Guttromson, and J. Thomas. A novel visualization technique for electric power grid analytics. *IEEE Transactions on Visualization and Computer Graphics*, 15(3):410–423, 2009.

Pak Chung Wong. Visual data mining. *IEEE Computer Graphics and Applications*, 19(5): 20–21, 1999.

J. S. Yi, Rachel Melton, John Stasko, and Julie A. Jacko. Dust and magnet: Multivariate information visualization using a magnet metaphor. *Information Visualization*, pages 1–18, 2005.