Automatic visualized explanations of Multidimensional Projections

Bachelor's Thesis

Rijksuniversiteit Groningen

Author:
Daan Vincent van Driel
s2599252

Supervisors;
Alexandru C. Telea, PhD
Johannes F. Kruiger, MSc

Abstract

Data representations of pictures, texts or sensor observations can range from hundreds to thousands of so-called dimensions or attributes. This makes it hard for end-users to understand these data, as humans are mainly acquainted with understanding 2D or 3D datasets. Multidimensional Projections (MPs) are a family of techniques aimed to help in this context. MPs reduce the data dimensionality, typically to 2 dimensions, while keeping its similarity structure, and visualize the resulting 2D dataset as a scatterplot. This enables users to easily locate related data points, groups of points sharing some common characteristics, and outliers. Displaying a simple scatterplot tells us which data points are similar but not why. Previous methods have focused on a visual approach to describe which dimensions contribute mostly to similarity relationships over the projection. This was achieved by ranking dimensions by variance and distance contributions over each point-neighborhood, and using a visual encoding to show which dimensions best explain the similarity in a neighborhood. This bachelor thesis will seek to expand upon this work, by proposing new metrics and visual encodings that allow us to describe the data in a different, robust and informative way. Using principal component analysis, we analyze point-neighborhoods to deduce and encode dimensionality, according to a fixed parameter which allows us to threshold eigenvalues, which leads us to generate a heatmap of dimensionality in the projection. Using Pearson Correlation Coefficients, we scan point-neighborhoods for linear relationships between attributes, and visually encode the projection according to the strength of different relationships in the data. Both techniques are tested and validated using both synthetic and real-world data.

Contents

Chapter 1

Introduction

1.1 Analysis of Multidimensional data

In research and business endeavors, making the right judgment or decision depends on having a keen understanding of the facts of the topic at hand. If we are presented with a problem or unknown, the examination of related data can have us reassess our position and adjust our viewpoint. Whether we occupy ourselves in the fields of business, statistics or science, the collection and analysis of data persists as a function to increase our understanding. Data, or datasets, represent information about phenomena as sets of measurements, samples, observations or records. What data represent is expressed through their *attributes* or *dimensions*. As our understanding and study of phenomena become more complex, a necessity emerges to increase the scope and complexity of the data in tandem with our increasing knowledge. Such datasets quickly grow to take on forms on large sets of *multidimensional data*, with hundreds of attributes and thousands of samples. And as information becomes more complex, so too does the task of understanding become more difficult. Many different techniques have emerged to deal with this problem, the forms of which can take on many different shapes across different fields. Examples include using *data mining* to identify Key Performance Indicators [1], or using *machine learning* to explore facial beauty analysis [2] . While there are many examples of techniques that allow for configuring complex systems for understanding data, what they lack is an intuitive, visual-impression based method of representing complex data which resolves one of the main problems with multidimensional datasets; that they can grow so large that representing with conventional methods can become impractical. The key analysis tools that try to resolve this problem can collectively fall under the topic of *data visualization*.

The goal of multidimensional data visualization is to transform complex multidimensional data into a more simplified and intuitive format that allows us to more easily identify patterns and structure in the data. Yet data analysis is a generalized process for studying information about many different kinds of phenomena, which leads to many different forms of representation and forms of pattern identification. And as such, many different forms of data visualizations have emerged. These include a family of dimensionality reduction techniques referred to as a *multidimensional projections*. Their aim is to express large multidimensional data on a more limited basis, seeking a transformative compression of the data so it can be represented more compactly while maintaining the similarity structure. Usually, multidimensional projections seek to reduce the di-

mensionality of high-dimensionality data to the point it can be represented in a two-dimensional or three-dimensional format, allowing for it to be visualized in formats most familiar to humans. The "projection" of the data can for example be shown to the user in the form of a simple scatterplot, allowing us to easily identify which points are similar to each other by their proximity to one another. Yet, while multidimensional projections are useful for getting an impression of which points are similar to each other and also lend themselves to estimating the shape of the data, they lack any default means to show *why* the points are similar to each other. This next step in the visualization of projection forms the main subject of this thesis, where we will seek to expand upon the groundwork established on the subject by da Silva et al.[3][4].

In the following paragraph, we will discuss a few of the main findings, work and observations made by da Silva et al. [4], from which they developed techniques to enhance the visualization of multidimensional projections. The focus is on trying to add explanatory forms of color-coding, by indicating which attributes are the most important in describing the similarity of points within different regions of the projection. These techniques were developed to address the shortcomings in typical techniques used to analyze multidimensional data, such as machine learning. More importantly, the techniques seek to address the lack of information in a multidimensional projection. The projection tends to lack direct meaning to the user and it is difficult to relate the information projected to the data. This creates a gap between knowing that there is some structure in the data, and being able to tell what those structure are. To deal with this issue, da Silva et. al described a method of visually encoding projection that allows end-users to easily see which dimensions are responsible for what patterns in the data. The emphasis on explaining points in terms of their *similarity* comes from a recognition that a key element in understanding projections is to understand how observations relate to each other to create patterns in the data. Compared to other multidimensional visualization techniques, this method provides a more intuitive method for the user to explore the data. In Fig. 1.1, a intuitive example is provided of a visualization of a projection using the techniques from da Silva et al. The dataset represents three faces of a 3D cube. In the projection, each of the three faces is colored and labeled by the dimension which is the most dominant in the region of that face. In Fig. 1.2, we apply the explanatory visualization technique

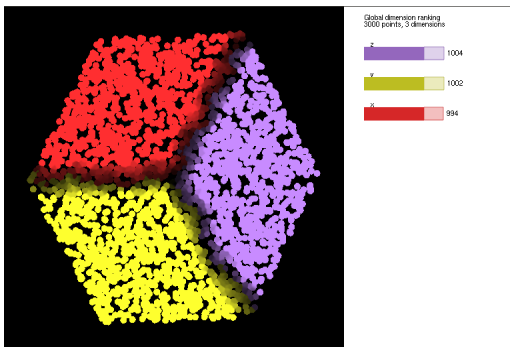
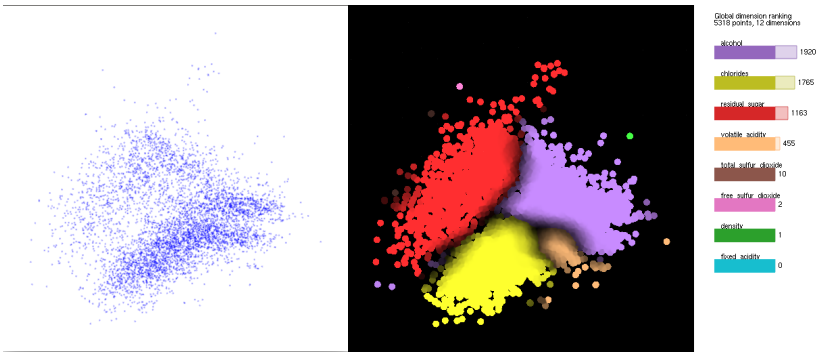


Figure 1.1: A projected set of faces of a 3D-cube classified by variance

to a real-world dataset containing measurements of chemical properties of different wines. If we examine Fig. 1.2(a), we can get a good impression of the shape of the distribution of the data and which points are similar (closer) to each other. However, it is not possible to know why points are

similar without examining the data in detail. Fig. 1.2(b) provides more context to the positions of the points in Fig. 1.2(a). The color-coded regions are a way to visualize which attributes are the most important factor in describing the similarity of points within that region. For example the red region is explained by the *residual sugar* attribute of the wine samples. This tells us that not only is *residual sugar* an important attribute, it also informs us *where* it is important. Similarly, we that the purple *alcohol* and yellow *chloride* regions are also important.



(a) Plain MP (b) MP with color-coding

Figure 1.2: Comparison between plain multidimensional projection of a wine quality dataset and the same projection enhanced with color coding by dominant attribute.

Silva et al. [3][4] combined data visualization with *user interactivity*, allowing the user to freely set parameters and explore the visualization as he sees fit. While their approach presents a large step forward for enhancing the explanatory capabilities of multidimensional projections, it is limited by design. Similarity is one of many options for a metric we can use to label regions in the projection and find patterns in the data. As any research that seeks to expand upon the groundwork laid by da Silva et al. [3][4] should, this thesis will primarily explore and investigate options for such alternative metrics that can expand the range of what information can be identified and visualized in projections.

1.2 Research Question

As explained in the introduction in the previous section on methods on creating explanatory visualizations to explain multidimensional projection, the aim of this project will be to **expand** on these methods; to build on and utilize established techniques that can be used to visualize data in a projection to label and explore it. Here, the emphasis is on finding metrics that can **add** to the current methods by creating visualizations which reveal **new** information in the projection which under the current constraints, cannot be revealed. As such, we can state our research question for this thesis:

> *Can we find new metrics and techniques, that allow users to explore datasets by visualizing multidimensional projection in a way which both complements and differentiates itself from established visualization methods?*

Over the course of the thesis, we deal with two subquestions related to the main question:

1. Is it possible to explore and explain multidimensional projections by visualizing the data by labeling and color-coding points by strengths of different local correlations?

2. Is it possible to explore and explain multidimensional projections by visualizing the data in a way that allows us to indicate different regions of dimensionality?

1.3 Structure and overview

- *Chapter 2: Background and Related work*
  This chapter is dedicated to formally specifying all the techniques, research, projects and software that lay the foundation of the work in this thesis. We will formally define many of the core concepts important to the implementation of the metrics, and summarize the work that forms the basis of this project. Topics include data generation, dimensionality reduction, principal component analysis, correlation coefficients and an overview of the work by da Silva et al.

- *Chapter 3: Implementation*
  This chapter directly follows from chapter. We formally propose our new local explanatory metrics, providing their motivation, formal definitions for computations, and the implementation of the visualization including examples of additional features and applications.

- *Chapter 4: Testing and Results*
  With the necessary background knowledge provided in chapter, and our newly proposed metrics specified in Chapter 3, we can move on to validating and testing our newly proposed metrics by using them to generate explanatory visualizations of datasets. For each metric, we will validate their correctness by applying them to synthetic data, and afterwards test their explanatory capabilities by testing them on real-world data–cross-referencing results from related studies and explanatory visualizations generated from the local distance/variance contribution metrics so that we can draw conclusions about our newly proposed metrics.

- *Chapter 5: Closing*
  This final chapter discusses the limitations of the progress made over the course of this project, what future work should focus on to remedy those limitations, but also reflects on what was achieved and to what degree the aims of the project were attained.

Chapter 2

Background and Related work

This chapter includes a summary of the theoretical work that formed the basis of this thesis, and an overview of all techniques used to achieve the implementation of the newly proposed metrics. In section 2.1, we will specify our formal definitions for *data* and the tools we used to generate and acquire data for this thesis. Section 2.2 focuses on *dimensionality reduction*, its formal definition, use, motivation and provides an overview of methods that can be used to achieve dimensionality reduction. With Section 2.3, we begin introducing a core concept of a global explanatory statistical technique used to implement our local dimensionality metric; *principal component analysis*. Section 2.4 continues along this line, providing an overview of statistical techniques called *correlation coefficients* which we will utilize in the implementation of the correlation metric. With section 2.5, we introduce the concept of *rankings* expressing local explanations based on local variance and distance metrics; core concepts introduced by da Silva et al. which forms the basis of this thesis. This section is closely followed by 2.6, which formally summarizes the way da Silva et al. achieved their local explanatory visualizations.

2.1 Data Generation

Analyzing datasets is a core task in gaining a better understanding of the underlying phenomenon which the data is supposed to represent. To reduce the workload involved in processing complex data frequently, many techniques have been devloped to simplify the process of analyzing data. In this thesis, including statistical software packages and mathematical models. we seek to explore datasets using automatically generated explanatory visuals. Now for any data, we can formally define it as a multidimensional dataset $D = \{p_1, .., p_N\} \in \mathbb{R}^n$ of N nD elements $p_i = \{p_i^1, .., p_i^n\}$ where each of the n dimensions is a real-valued attribute such that $p_i^j \in \mathbb{R}$ for $1 \leq j \leq n$. Each of the dimensions (also called *attributes*, *features* or *variables*) describe some kind of aspect of the data. The descriptions of the attributes together describe what is quantified and measured. If the data were to be visualized in a table, the attributes would form the columns of the table. Attributes can come in many different forms, such as *categorical*, *ordinal*, *integral* and *quantitative*. This thesis will only concern itself with data which only consists of *quantitative* attributes, and the dimensionality metrics and correlation metrics only consider these types of data.

An *observation* (also referred to as a *sample*, *measurement* or *point*) represents a set of n values for each attribute. Observations would form the rows of the table of the data. What constitutes an observation differs per dataset. Some datasets might contain observations of the same phenomena but taken at different moments of time. Others might contain observations that are each samplings of different instances of same physical phenomena. Here, one could think of for example comparing differing instances of chemicals, documents, food or human and animal subjects. Each instances is quantified by the measured observations of certain attributes. A very practical example would be a collection of answers to a population survey, where the responses of each participant would form the set of observations and the subject of the questions forming the attributes.

The datasets we will examine will come in form of both real-world data, measured from physical phenomena, and also synthetic data, created for the purpose of this thesis using software. Examples of real-world data include a dataset measuring chemical content, such as alcohol, sugar or chloride, of different wines [5]; a dataset containing of software quality, quantified by metrics such as lack of cohesion, coupling and structural complexity, of software projects on sourceforge.net [6]. Each of these datasets contain thousands of measurements with about a dozen attributes. This makes them hard to analyze with conventional methods such as tables. By visualizing data, we hope to simplify its representation and make it easier for humans to make sense of the contained information and discover patterns more easily. The dimensionality and correlation metrics at the center of this thesis will try to address this issue by automatically visualizing the data in such a way that patterns and trends are identifiable from a simple visual impression of a user.

Before we can apply our metrics, we first need to validate and explore how they work. The complexity of the wine quality and software quality datasets will lend itself well to explore of how the metrics introduced in this thesis will function when applied to real-world research data, and validate their usefulness to discover patterns in the data. With synthetic data, we can decide for ourselves what shape the data takes. Unlike with real-world data, we would know beforehand what the shape of the data is and therefore can also rationalize about what results we should expect if use metrics to visualize the data. This will allow us to establish so-called "ground-truths" about the correctness of our metrics; in the sense that we can validate that they are working correctly by applying them to these synthetic datasets. There are many tools available for generating synthetic data, and we will briefly cover the ones utilized over the course of this thesis project here.

- Scikit [7]
  A Python library that provides several in-built functions to easily generate n-dimensional datasets such as Gaussian clusters, circles or combinations of shapes. Several different Python scripts were created incorporating this library to generate synthetic data.

- *Distribution Painter* [8]
  A software tool that seeks to avoid the issue of having to create scripts to generate custom multidimensional data, but instead implements a graphical-user-interface that allows the user to "paint" correlations between dimensions and adjust parameters for noise levels.

- FeaturesProjection
  A custom software tool in Java courtesy of Renato da Silva that implements several different methods of projection mappings. This was the main tool used to perform the LAMP projection on datasets.

2.2 Dimensionality Reduction

Datasets can differ qualitatively in what they measure and describe, but can also differ on quantifiable terms in the value of their observations or the shape in which they are represented. Some data may contain a very small amount of attributes, while other might contain very many. As the number of attributes in a dataset increases, so does the difficulty of gaining an intuitive understanding of the shape or structure of the data, performing analysis and drawing robust conclusions about the data. *Projection* techniques or *dimensionality reduction* techniques transform high-dimensional nD datasets, creating a mD dataset with $m < n$ which maintains relationships between points, patterns and trends in the data. For the purpose of making sense of multidimensional data, there are many benefits to using dimensionality reduction techniques. The lower-dimensionality allows us to examine the data more easily, but also allows us better deal with constraints on visualization of high-dimensional data. If we choose m to take on a value of either 2 or 3 then this will allow us to visualize the data much more easily than were n to be of a significantly higher value. Furthermore, dimensionality reduction can allow us to discard attributes which can be considered to be *noise* or redundant. For the purpose of this thesis, dimensionality reduction techniques are a core feature in generating visualizations.

A projection transformation aims to maintain the *shape* or *structure* of the data, ensuring for example the same amount of outliers in both projected and unprojected data, and keeping patterns and trends in the data present. Formally, this *distance preservation* of the projection must ensure that the similarity structure of the data is maintained over the course of the transformation, ensuring that the pair-wise distances between points scale roughly the same between the projected and unprojected data.[4]. *Neighborhood preservation* is another important aspect. Informally, this would mean that if we calculate an nD neighborhood of points v_i^n for a point p_i in nD, and also a mD neighborhood of points v_i^m for a point q_i in mD, the composition of neighborhoods would nearly identical to each other for the observation i.[4].

Recall we can define a dataset $D = \{p_1, .., p_N\} \in \mathbb{R}^n$ of N nD elements $p_i = \{p_i^1, .., p_i^n\}$, with each $p_i^j \in \mathbb{R}$ for all $1 \leq j \leq n$. By applying a dimensionality reduction (or projection) technique, we apply a transformation P: $\mathbb{R}^n \to \mathbb{R}^m$, such that the elements of D are projected onto a lower m-dimensional space resulting in projected elements $D^p = \{ \quad q_i = P(p_i \in D) \quad \}$. These projected elements can then be shown in a mD scatterplot. Usually, the value of m is equal to 2 or 3, 2D or 3D scatterplots being the most intuitive to humans. Now, there are many different kinds of projection techniques available. But for this thesis, we restrict ourselves to applying three different kinds of projection techniques on test datasets, each with its own advantages in terms of respect to distance or neighborhood preservation, efficiency or wide application in the scientific community. The projection techniques are described as follows:

- Principal Component Analysis (PCA)
  As PCA can be used a method for re-expressing a dataset as a linear combination of its original basis, it can be used to achieve dimensionality reduction of high-dimensional data.[9][10] Many software tools that allow for PCA reduction allow the user to specify the number of dimensions necessary to generate the linear combination or target dimensionality (in our case always 2). PCA is a global dimensionality reduction technique, which seeks to find the set of orthonormal principal components or directions which capture most of the variance for a given dataset. Deriving these components allows us to re-express the data on a lower-basis;

thus achieving dimensionality reduction. The general approach to PCA, further detailed in Sec. 2.3, is to derive a covariance matrix and apply eigenvalue decomposition as to define a set of eigenvectors and eigenvalues which can describe the shape of the variation in the data. While PCA is popular, fast and efficient [11] it is has downsides for multidimensional dataset that are not "intrinsically" of a very low-dimensionality. PCA is well-known, fast and easy to implement, but can have problems maintaining the shape of the data over the course of the transformation [10][12]. Because PCA will enforce compression of the nD points onto the 2D plane, points that are very far in nD might become very close in 2D. This leads to inaccuracy, when for example, we calculate the point-neighborhood. Unless we develop a technique to account for this limitation, as we detail in Chapter 3, section 1, using PCA is not the preferred method of projection.

- t-distributed stochastic neighbor embedding (t-sne) [13]
  t-sne is a optimized variation on stochastic Neighbor Embedding. Aiming to retain both the local and global structure of the dataset within a single map, t-sne hopes to take into account high-dimensional datasets that lie or are comprised of several lower-dimensional manifolds. The main aim is to solve a "crowding problem"; which is similar to the compression issue with PCA, namely that the space offered by the plane onto which we want to project the nD dataset to will not always offer enough space to accommodate or preserve the distances in nD. Given a set of points/objects in the high-dimensional space, t-sne will build a probability function based on the similarity of the objects that is used to determine which points are selected as neighbors of each other. T-sne is a well-known method of dimensionality reduction, and is also efficient at neighborhood preservation. [13].

- Local Affine Multidimensional Projection (LAMP) [14]
  This is a relatively novel method of multidimensional projection, one that seeks to address the shortcomings in older projection methods and also the preferred way to analyze real-world, measured data for this project. The method relies on orthogonal mapping theory to address shortcomings of global mapping algorithms, such as PCA, which fail to take into account local changes; while also addressing shortcomings of local mapping algorithms, such as t-sne, which have high computational costs and require a large number of samples. LAMP uses subsets of samples in the data as control points to solve a minimization problem to compute an affine mapping that ensures distances in nD are preserved as much as possible in the projection, thus resolving the issue of compression of points as with PCA projection, while also being more efficient than the computationally expensive t-sne method.

2.3 Data analysis: PCA

As data becomes more complex, with many different dimensions influencing the variation and patterns between the attributes, the need emerges for techniques that allow us to gain an impression of the underlying *dimensionality* or *structure* of the data. An understanding of this property would allow us to make an assessment of the real, inherent complexity of the data, from which can gain a better understanding of how to approach and compare different kinds of datasets. Principal Component Analysis (PCA) is a statistical procedure that can be used to investigate, for a given dataset, whether or not there exists another basis, which is a linear combination of the original basis of the dataset, that best re-expresses the data.[9][10] In multidimensional, complex data with

a large amount of samples and a high probability for a large amount of noise, PCA can be utilized to express the underlying dynamics of the data and reveal its "principal components" or most important dimensions that best characterize the variance in the data. In this thesis, PCA will be used to implement the dimensionality metric, where the idea will be to locally explain regions in the projection by determining the number of relevant principal components within point-neighborhoods. PCA is widely used and popular [11], as it is simple, fast, good for data points which are spread more or less like a Gaussian hyper-ellipsoid [15]. However, because PCA as a method tries to fit a Gaussian hyper-ellipsoid, it tends to have issues for datasets which are not normally distributed, and is also sensitive to small changes such as one vector being changed.[12]. Another limitation of PCA is for when data points do not live on a hyperplane embedded in $\mathbb{R}^n$, which can lead to projection errors being very large. For some datasets, this can lead to dissimilar points in nD falsely appearing as being very similar in 2D.[11]. PCA as a method of projection will therefore encounter problems with key-issues such as neighborhood and distance preservation. Still, the purpose of this thesis is to simply explore the viability of creating explanatory visualizations of multidimensional datasets by locally explaining points by their dimensionality; the purpose is not to reach a definite way to do so. As such, PCA with its simplicity and efficiency is considered an adequate choice for the purpose of this thesis.

PCA can be achieved by a so-called "covariance method" [10] which utilizes a covariance matrix of the dataset, which will be specified in this thesis. With the "covariance method" we calculate a $n \times n$ covariance matrix from X, which is a $N \times n$ matrix that expresses the measurements for the data. In the covariance matrix each index (i, j) the covariance, or joint variability, between dimension i and dimension j is expressed. The covariance *cov* between two variables Q and U, each with N samples, is defined as:

$$cov(Q, U) = \frac{\sum_{i=1}^{N}(Q_i - \bar{Q})(U_i - \bar{U})}{N}$$

In PCA, a preliminary step is to compute a covariance matrix C_X that expresses the covariance between each pair of dimensions (i, j) in X. This is achieved by "centering" the data first by subtracting the mean μ_j of column j (the mean of samples for dimension j) from the j-th column vector in X for each j. The covariance matrix of X can then be calculated as:

$$C_X = \frac{1}{n} X^T X$$

The index $C_X(i, j)$ represents a quantification of the covariance between dimension i and j in X. A large value would imply high correlation while a smaller value would imply a small correlation. If $i == j$, then $C_X(i, j)$ would express the variance of dimension i in X. Note that this must mean that C_X is a square symmetric matrix. As such, properties of eigenvalue decomposition can be used to diagonalize C_X by its eigenvectors to derive a diagonal $n \times n$ matrix A that returns the eigenvalues for C_X. The value of the eigenvalue λ_i at $A(i, i)$ would express the contribution to the total variance in the plane spanned by all eigenvectors by eigenvector i. The matrix A can be calculated as:

$$A = EC_X E^T$$

where E is the column-wise matrix of eigenvectors of C_X. The computation of eigenvalues for the software development of this bachelor's project was achieved through the $C++$ library *Eigen* [16]

which uses Schur decomposition [17] to compute the eigenvalues and eigenvectors of a given matrix.

If an eigenvalue is large compared to the other eigenvalues, then that would indicate that this eigenvector forms an relatively important part of the basis of X. Accumulating a sum of the eigenvalues and normalizing them by the sum will define a new set of eigenvalues, where instead of an absolute value each eigenvalue indicates the ratio of how much it contributes to the total amount of variance in X.

One example of use of eigenvalue decomposition and PCA to understand the underlying structure of data can be found in medical research of diffusion tensor image analysis [18], which concerns itself with quantifiable analysis of visualization of white matter of the human brain created using diffusion tensor magnetic resonance imaging (DTI). The eigenvalues are derived from symmetric, positive-definite diffusion tensor D, a 3×3 symmetric positive matrix which is used to describe the diffusion of water molecules. In this example, the largest eigenvector of the D points in the principal diffusion direction. In other words, the process of eigenvalue decomposition in DTI analysis is part of a process of PCA to derive the most important components which contribute the most to the diffusion in D.

This example application of PCA [18] also provides for a method to characterize the structure or shape of the variance distribution of the analyzed matrix.. Ratios of the eigenvalues, called tensor anisotropy measures, are defined to quantify the shape of the diffusion. The ratios are defined as the fraction between eigenvalues $\lambda_i - \lambda_{i+1}$ and the largest eigenvalue λ_1 for $1 \leq i < n$ to achieve a measure of how much the diffusion/entropy resembles a particular shape. For example, a given ratio D_2 would indicate how well the diffusion fits a linear or 2D shape:

$$D_2 = \frac{\lambda_1 - \lambda_2}{\lambda_1}$$

By computing and comparing the ratios for all different shapes, it's possible to decide which shape characterizes the diffusion the best. In this example the shape associated with the largest ratio would be used to characterize the shape of the diffusion.

2.4 Data analysis: Dimension correlations

Multidimensional data are frequently the subject of analysis that tries to find relationships and dependencies between attributes in the data. Researchers will try to find correlations between pairs of attributes in the data, from which they can infer relationships and trends. A correlation is a measure of the linear association between two continuous variables [19] The correlation value, or correlation coefficient, gives the strength of the association or dependence. This coefficient value takes on a value in the range -1 and 1. A coefficient value of zero indicates no association between the two variables, while either a minimal (-1) or maximal (1) coefficient value indicates a strong relationship. A large positive value indicates a linear relationship between the two variables, while a negative value indicates an **inverse** linear relationship where one decreases where the other increases. Provided a dataset $X = \{x_1, .., x_n\}$ and another dataset $Y = \{y_1, .., y_n\}$, the Pearson

sample correlation coefficient r between X and Y is calculated as [20]:

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

Here $\bar{x}$ and $\bar{y}$ are defined as the means over X and Y respectively. More specifically: $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ A non-parametric variation on the Pearson correlation coefficient, less sensitive to outliers, is the Spearman Rank correlation coefficient. The measurements of x_i and y_i are converted into ranked integer variables x_i^r and y_i^r. The coefficient r is calculated as:

$$r = \frac{\sum_{i=1}^{n}(x_i^r - \bar{x^r})(y_i^r - \bar{y^r})}{\sqrt{\sum_{i=1}^{n}(x_i^r - \bar{x^r})^2}\sqrt{\sum_{i=1}^{n}(y_i^r - \bar{y^r})^2}} \quad (a) \quad \text{or} \quad r = 1 - \frac{6\sum d_i^2}{n(n^2-1)} \quad (b)$$

Here d_i^2 is the sum of the squared differences between x_r and y_r and (b) is only used if x_r and y_r consist of unique values.

Globally, correlation coefficients can be used to generate explanations of the data by indicating the strength of inter-attribute dependencies over the entire dataset. These global explanations however lack the ability to show relationships between attributes in different regions in the data. As we will see in section 3.2, We will use correlation coefficients to locally calculate the strongest correlated pairs within projections, allowing us to visualize the data by color-coding regions by the most frequently top-ranked strongest correlated pairs. For the purpose of this thesis, we will focus exclusively on the application of the Pearson correlation coefficient to investigate the validity and functionality of the correlation metric. Benefits of using the Pearson Correlation coefficients include it being very well known and it providing precise insights over pairs of variables; it's a very 'global' metric i.e. it considers ALL observations when computing the correlation of 2 dimensions; and also it does not tell you much about three or more dimensions. Weakness include its sensitivity to outliers and its assumption of normal distribution in the data. As with PCA and the dimensionality metric, the focus here is exploring the metrics as *options* for generating explanatory visualizations of the data. Optimal implementations are not the goal of this thesis.

2.5 Local Variance, distance contribution and similarity

To develop and implement the metrics we propose in this thesis, we need a basis; paradigms which can use as reliable examples to rationalize our approach. In this respect, this project serves as an extension of research performed by da Silva et al. [3] from 2015. The aim of da Silva et al. was to explain closely projected points in MPs by their similarity to each other by computing a color-and-luminance map from a set of **rankings** for each attribute. In a MP, a dataset $D = \{p_1, .., p_N\} \in R^n$ of N nD elements $p_i = \{p_i^1, .., p_i^n\}$ is projected onto a lower m-dimensional space using dimensionality reduction techniques resulting in projected elements $D^p = \{q_i = P(p_i \in D)\}$. For each point p_i, a ranking vector $\{j, \mu_i^j\}_{1 \leq j \leq n}$ is calculated for all n dimensions, here μ_i^j encodes the rank of dimension j for point i. A lower ranking indicates a better ability for j to explain the similarity between of the points in nD-neighborhood v_i of p_i. da Silva et al. specify two **metrics** to compute rankings: Euclidean Rankings and Variance Rankings.

The Euclidean Rankings [3] are derived by first calculating the contribution of dimension j to

the squared distance $lc^j_{p,r}$ as the nD squared distance between points p and r

$$lc^j_{p,r} = \frac{(p^j - r^j)^2}{||p - r||^2}$$

For each point p_i, the **local** contribution of j is calculated as an average of the contributions between p_i and its neighbors $r \in v_i$:

$$\bar{lc^j_i} = \frac{\sum_{r \in v_i} lc^j_{p_i,r}}{|v_i|}$$

The final ranking for dimension j with p_i is then calculated by taking into account the global contribution gc^j of dimension j for the distance, calculated for the nD centoid point with all points as its neighborhood.

$$\mu^j_i = \frac{\bar{lc^j_i}/gc^j}{\sum_{j=1}^n \bar{lc^j_i}/gc^j}$$

The variance rankings [3] are instead calculated by first calculating the global variance GV for all dimensions over all points

$$GV = (var(p^1), ..., var(p^n))$$

For each point i, the local variance LV_i is computed over its neighborhood v_i. The final ranking between a point i and dimension j is then calculated as:

$$\mu^j_i = \frac{LV^j_i/GV^j}{\sum_{j=1}^n LV^j_i/GV^j}$$

For determining [3] the neighborhood v_i for a point i, a typical method would be to use the k-nearest neighbors [21] algorithm. An issue with using this method is that for a point i which is an outlier in the projection, the neighborhood would include points that have a large distance between them and i. For da Silva et al. it was important that the v_i consisted of points in close proximity to i, as proximity in the projection is also a measure of similarity between the points. Instead, a neighborhood calculation method based on radii is used. For all 2D projected points q_i the definition is given for a 2D neighborhood $v^p_i = \{ \quad q \in D^p \quad || \quad | \quad ||q - q_i|| \leq \rho \quad \}$ given a radius ρ. The nD neighborhood v_i is then defined as $v_i = \{ \quad p \in D \quad | \quad P(p) \in v^P_i \quad \}$. Once the point-neighborhoods are calculated and the metrics defined, the rankings and ranking vectors can be calculated. Allowing us to move on to the next step: Visualization.

2.6 Visual Explanation

There are many different ways to generating explanatory visuals and analytics for multidimensional projection, which could manifest as either *global* explanations or *local* explanations. Sections 2.3 and section 2.4 detailed PCA and correlation coefficients, which are typically used to generate *global* explanations. For the entire global dataset, PCA would allow us to generate a set of eigenvectors/eigenvalues which together can describe the shape of the variance of the data and allow us discard redundant attributes. Correlations compute and display a correlation factor, which is a single number is used to globally explain which the strength of the dependencies between attributes. When different parts of the data behave differently, the global explanations will prove insufficient.

Different pairs might correlate more heavily in different regions of the data, and different neighborhoods in the data might feature a different number of principal components. Imagine if for example we are examining a dataset consisting of several clusters. We have reason to believe that some clusters are in a different class compared to other clusters. The clusters each have a reasonable distance from each other. Local explanations would allow us to indicate separate explanations for each of these clusters, and examine each of them for their local patterns and trends. In comparison, global explanations can only provide us an overview for the entire set of clusters; something which in this example is not very useful as know the clusters are qualitatively different. Generally, the benefit of using local explanations is that they can allow us to indicate smaller, more subtle differences between different regions of the data. In the case of our correlation and dimensionality metric, we would locally investigate the strongest correlated pairs and locally determine the number of relevant dimensions that characterize the shape of the region being examined. In the case of the similarity metrics of distance and variance contribution proposed by da Silva et al. [3][4], we would be looking at generating local explanations based on local contributions to variance and distance, adjusted for the global contribution to distance/variance. The here aim is to generate explanatory visuals which indicate which attributes locally contribute the most to the similarity of points in a local neighborhood. With all of this in mind, how do we move on to generating visualizations?

Basis

Recall from section 2.5 we use the variance/distance metrics to compute, for each point p_i, a ranking vector $\{j, \mu_i^j\}_{1 \leq j \leq n}$ for all n dimensions. Each ranking vector is sorted in ascending order (for distance and variance rankings) or in descending order (if we use a measure of dissimilarity). For each rank, the frequency of that rank being the top rank for a point is calculated for the set of ranking vectors. Using a colormap containing C colors, the C most frequent top-ranked ranks are selected and assigned to one of $C - 1$ colors. If there are more ranks with have a top-ranked frequency higher than zero, their are collectively assigned the same color with index C in the map. For each point, we color that point with the assigned color of its top-ranked dimensions. This ensures that is simple to distinguish between regions explained by the most significant top-ranked attributes.

Confidence

Another part of the visualization [3][4] involves the encoding of the **confidence** of a top-ranked dimension. With the confidence, we wish to encode the contributions of other attributes, other than the top-rank attribute, in explaining the similarity of the neighborhood of a projected point q_i. While one attribute might be the top-rank attribute in this respect, it is rarely the only attribute that contributes to the explanation of similarity in the neighborhood. The encoding of the confidence is expressed in the visualization by the strength of the *luminance* of a point. Points where many attributes have top-ranks similar in value, where the top-rank attribute does not contribute significantly more to the explanation compared to other attributes, will be visualized with darker colors compared to points where there is a single very strong top-ranked attribute. Recall we use a radius ρ to calculate the neighborhood v^p for a point p_i. To calculate the confidence, we use a smaller radius ρ_c such that $\rho_c < \rho$ to define a 2D neighborhood v_c^p centered at q_i. We identify t as the rank *ID* or *number* of the top-ranked attribute for the point p_i such that $t = \arg\max_{1 \leq j \leq n} \mu_i^j$.

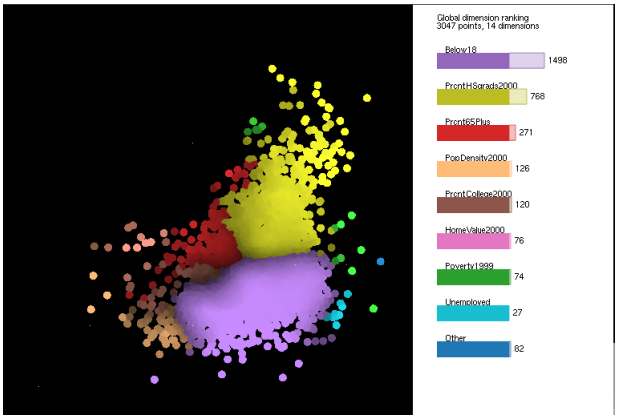


Figure 2.1: Visualization of US Counties data

The confidence c_i^t is then defined as: [4]

$$c_i^t = \frac{\sum_{q_j \in v_c^p \wedge \arg\max_k \mu_j^k = t} \mu_j^t}{\sum_{q_j \in v_c^p} \max_k \mu_j^t}$$

...meaning the confidence is encoded as the sum of ranks u_j^t over all points $q_j \in v_c^p$ having attribute t as the top-rank, normalized by the sum of all top-ranks over all points in v_c^p. This definition will ensure the confidence acts as a smoothing filter where homogeneous regions with many same top-ranks are assigned high confidence, while more mixed regions are assigned a lower confidence. The visualization of the confidence allows users to more easily distinguish between these regions.

Visualization

How does this theory work in practice? Fig. 2.1 includes a snapshot of this visualization method in action. Using the distance contribution metric, we examine a multidimensional dataset describing social, economic, and environmental data from 3138 USA counties.[22] For each point, we examine the local neighborhood and calculate the top-ranked attribute which are found to locally contribute the most to the distance in that neighborhood. The size C of our colormap is equal to 9. So we compute the top 8 most frequently top-ranked dimensions over all points. The first color, purple-like, in the colormap is assigned to the most frequent top-ranked dimension, in this example the *under18* attribute. The second color in the colormap, yellow-like, is assigned to the second most-frequent top-ranked dimensions, in this case being the *percentage of high-school graduates* attribute, and so on. The ninth color, blue, is assigned to all attributes which have a frequency higher than zero but are not in the top 8 most frequently top-ranked dimensions. Once the colormap has been assigned, we can generate the visualization. The top ranks and their confidences are displayed over the projection using the dense map technique based on Voronoi interpolation.[23] In practice, this will mean that we will generate a texture for the projection. For each pixel in the projection, we look at which points are within the neighborhood of that pixel for within a certain radius. For each of the points, we take the top-ranked dimension; and from the set of top-ranked dimensions within the neighborhood we compute the influence of each dimension. Very basically, the dimension which

influences the pixel the most is used to color the pixel, while we interpolate for confidence. The texture is visualized. Meanwhile, we also generate a ranking histogram as displayed to the left Fig. 2.1. Each of the top most frequent top-ranked dimensions are given a histogram, colored by their assigned color in the colormap, where the length of the histogram is based on the number of points for which the dimension was the top-ranked dimension during the ranking phase. This number is also shown to the right of the histogram. The histogram allows users an easy insight into which dimensions where the most frequent over the entire projection.

The ranking method devised by da Silva et al. [3] can be demonstrated to be an effective and intuitive way to label points by a dimension that is most similar in their proximate neighborhood. A shortcoming in the metrics used by da Silva et al. [3] is that only the top-ranked dimensions values for each point are used for labeling, thus the presence of any dimensions k that have a rank u_i^k weighted very closely to the rank u_i^j of top ranked dimension j is potentially obscured. Furthermore, information about relationships **between** dimensions within v_i is also something that is not provided by these metrics.

The theory drafted by da Silva et al. [3] was implemented in a $C++$ program called "Projection Explainer". The program utilizes OpenGL and the NVidia CUDA library for visualization and several open source libraries such as ANN [24] and $Eigen$ [16] to improve computational performance. For this project, nearly all software development served as an extension to the original "Projection Explainer" code maintained by R. da Silva and was dependent on the theory discussed within this section and its implementation in the project.

We see that there are global/local metrics; none of them is by itself sufficient to fully explain a complex dataset; moreover, we see that some metrics only come in the global version. Our work next is to show how we can create local metrics based on these global ones (correlation, PCA), and embed them in the local visual explanation proposed by da Silva et al.[4]

Chapter 3

Implementation

The aim of this chapter is to aim to enrich the local explanations provided by da Silva et al [4] by creating additional ones based on the global PCA and correlation metrics. For this, we globally proceed very much like da Silva:

- We define a way to compute a neighborhood of our dataset and/or projection, which defines the unit of computation of our local metrics (Sec. 3.1)
- We next adapt the global metrics PCA and correlation (Sec 2.3, 2.4) to work locally, and also to deliver simple-to-understand characterizations of the data (Sec. 3.2, 3.3). In the same time, we also extend/refine the visual presentation used for these metrics.

3.1 Neighborhood calculation

Dimensionality reduction techniques seek to identify the principal attributes in high-dimensional data which allow for expressing the data on a more limited basis. When we project high-dimensional points to a lower 2D plane, the transformation is a form of compression of the points in the nD space into the 2D space. As the nD space is much larger than the 2D space, the ability to maintain the similarity between points in nD to the converted points in 2D presents a challenge that some projection techniques are better capable of handling than others. Essentially, we want points which are close to each other in nD to be as close to each other in $2D$, but some projection techniques such as principal component analysis tend to be limited in their ability to maintain the similarity structure of high-dimensional data which is not fundamentally low-dimensional. This issue presents a problem for the method of neighborhood calculation based on radii as introduced in section 2.5, which exactly does assume that distance similarities are preserved during the projection. If we use this method to explain projections generated using PCA, we run the risk of generating wrong or confusing explanations. To deal with this limitation, we propose an alternative way to calculate the point neighborhood that combines the radius-based method with a variation of the k-nearest neighbors algorithm.

As before, we define a dataset $D = \{p_1, .., p_N\} \in R^n$ of N nD elements $p_i = \{p_i^1, .., p_i^n\}$. We project D onto a lower m-dimensional space using dimensionality reduction techniques resulting in projected elements $D^p = \{q_i = P(p_i \in D)\}$. For each point q_i, we define a 2D neighborhood

$v_i^p = \{ \quad q \in D^p \quad | \quad || \quad ||q - q_i|| \leq \rho \quad \}$ given a radius ρ. The nD neighborhood v_i is then defined as $v_i = \{ \quad p \in D \quad | \quad P(p) \in v_i^P \quad \}$. Next, the size of the neighborhood is calculated as $|v_i|$. The size of $|v_i|$ is set as the size of k. We then compute the set of Euclidean distances between p_i and all other points in the dataset D. The k-nearest points, based on Euclidean distances, are then defined as the point-neighborhood of p_i. The squared Euclidean distance $e_{p,r}$ between a point p and a point r is defined as:

$$e_{p,r} = \sum_{j=1}^{n} (p^j - r^j)^2$$

The set of pairs of points and squared distances $v_{p_i}^e$ for a point within v_i can be defined as:

$$v_{p_i}^e = \{ \quad (r, e_{p,r}) \quad | \quad r \in D \quad \}$$

We then sort $v_{p_i}^e$ in ascending order, based on the value of $e_{p,r}$, such that

$$v_{p_i}^e = \{ \quad (r_i, e_{p,r_i}) \quad | \quad e_{p,r_i} <= e_{p,r_j} \quad | \quad 1 \leq i \leq N \quad | \quad i < j \leq N \quad \}$$

the point-neighborhood v_i^k can be selected as the first k or $|v_i|$ points r in $v_{p_i}^e$.

$$v_i^k = \{ \quad r_i \quad | \quad 1 \leq i \leq k \quad | \quad (r_i, e_{p,r_i}) \in v_{p_i}^e \quad \}$$

Unlike for the calculation of v_i, the calculation of v_i^k will **always** involve the calculation of the nearest neighbors of p_i in nD space, thus effectively serving as a precautionary measure to deal with the issue of point-space compression. The calculation of k serves as a way to tell how large the area we want to encode should be. By making it dependent on the size of the 2D neighborhood we are directly relating it to what is displayed in the projection.

This new method, in contrast with the conventional one, is not **necessarily** better. The advantage gained depends on the quality of the projection, and the degree to which compression has taken place. In Chapter 2, section 4, we detailed several different projection methods that can be used to achieve dimensionality reduction of data. Some methods, like PCA, do not deal too well with the issue of point-compression, especially for datasets that are not inherently low-dimensional. The new method minimizes the inaccuracy produced by compression as much as possible, by directly taking the neighborhood from the points close in the unprojected plane. As such, when we analyze datasets that were projected using PCA, the new method will have our preference over the conventional approach. For projection methods such as LAMP however, which reduce the problems of other projection methods as much as possible, using the conventional method will be the standard, mainly because it will be more efficient. Furthermore, we will cross-reference our newly proposed explanatory visualizations with results and analysis performed by da Silva [4]; which relied on using the conventional radii based method. Varying the parameters could potentially lead to faulty comparisons and conclusions.

3.2 Correlation metric

3.2.1 Motivation

Multidimensional data is frequently the subject of a process of regression analysis, researchers will for example try to examine chemical properties of wine [5] to find a relationship between these chem-

ical properties and the quality of the wine, or try to find an association between the evaluation of software using software metrics and the quality of that software. In both examples, researches try to find relationships between independent variables, those that can predict, and dependent variables, those that are predicted. Any dataset might contain hidden relationships that allow us to differently interpret the data. So intuitively, as the main subject of this thesis concerns analyzing multidimensional projections, and therefore multidimensional data, it makes sense to attempt to implement a metric that allows for some kind visualization of these relationships in the projection. Specifically, what we want to do is pinpoint the strongest relationships in each point-neighborhood of the projection. If the projections shows the existence of different kinds of relationships within different areas of the projection, that might tell us that the way those attributes of the data relate to each other is fundamentally different per region. It might give us an indication which attributes are dependent and which are independent. Moreover, if we know that a certain region of points mostly belongs to a certain category, and we can visualize a strong correlation within that region, that might tell us something about the behavior of sets of measurements that belong to that category. Think of for example of regions of points that can be divided as belonging to "good" software and "bad" software, or points associated with "high quality" wine and "low quality" wine. If we find different kinds of correlations in the "high quality" and "bad quality" regions, then we might get an indication which attributes can together predict low or high quality wine/software.

The established metrics [3][4] of analyzing contributions to distance and variance are of course, less capable or not entirely capable of all of detecting those relationships. As such the benefit is clear. Still, the benefit of the metric is not just that it is able to show different information than the established ones, the visualizations it provides can also serve as a complement to the explanatory capabilities of those established metrics. If regions where a particular attribute contributes the most also uniformly feature a particular correlation, then that could allow us to infer a pattern between the attributes in the relationship and the attribute that dominates the region. The inverse is of course, also true, if a relationship is present in regions specifically where an attribute does not dominate, then it might tell us something about the ability of that attribute to affect the relationship.

3.2.2 Specification

Local correlations

As discussed in section 2.4, correlation coefficients can be used to measure the association between two variables. Given an MP for a $N \times n$ dataset D, consisting of N samples and n attributes, the purpose of the correlation metric will be to explain a point-neighborhood by correlation of a pair of dimensions, instead of just a single dimension, by examining which pair of dimensions is calculated to have the highest correlation coefficient within the point-neighborhood. Phrased differently, we seek to color-code points by which pair of dimensions is most heavily correlated within their point-neighborhood.

For a dataset consisting of n attributes, the total count of unique pair combinations will equal $\frac{n(n-1)}{2}$. We can define the set S of pairs of attributes as:

$$S = \{ \quad (k,l) \quad | \quad 1 \leq k < n \quad | \quad k < l \leq n \}$$

For each nD point $p_i = \{p_i^1, ..., p_i^n\}$, we calculate a ranking vector $\{s, \mu_i^s\}_{\forall s \in S}$ for each pair of attributes in S. the value of each rank is calculated using a correlation coefficient algorithm, with this thesis focusing primarily on the application of the Pearson correlation measure and secondarily considering the Spearman rank correlation measure, both detailed in section 2.4.

Given a neighborhood v of size g, we can define the set of observations X^i for attribute i as:

$$X^i = \{ \quad r_t^i \quad | \quad 1 \leq t \leq g \quad \}$$

Given a neighborhood, attribute i and j, we can define the Pearson correlation coefficient $pearson(v, i, j)$ as:

$$pearson(v, i, j) = \frac{cov(X^i, X^j)}{\sigma_{X^i}\sigma_{X^j}} \quad or \quad pearson(v, i, j) = \frac{\sum_{t=1}^{g}(r_t^i - \bar{r^i})(r_t^j - \bar{r^j})}{\sqrt{\sum_{t=1}^{g}(r_t^i - \bar{r^i})^2}\sqrt{\sum_{t=1}^{g}(r_t^j - \bar{r^j})^2}}$$

Given a neighborhood v_i for a point p_i, the rank μ_i^s is calculated as

$$\mu_i^s = \frac{abs(pearson(v_i, s_k, s_l))}{\sum_{\forall s \in S} abs(pearson(v_i, s_k, s_l)}$$

If we use the Spearman method to calculate the correlation coefficient, we have:

$$\mu_i^s = \frac{abs(spearman(v_i, s_k, s_l))}{\sum_{\forall s \in S} abs(spearman(v_i, s_k, s_l)}$$

with

$$spearman(v, i, j) = \frac{cov(X^i_{ranked}, X^j_{ranked})}{\sigma_{X^i_{ranked}}\sigma_{X^j_{ranked}}} \quad or \quad spearman(v, i, j) = 1 - \frac{6\sum d_i^2}{g(g^2 - 1)}$$

The *abs* function returns the absolute value of its input. With correlation coefficients, the value can range from -1 to 1. By taking the absolute value, both negative and positive correlations can be compared and quantified with each other. This allow us to differentiate between strong correlations, negative correlations or no correlations for pairs within different regions of the data, as opposed to simply analyzing for strong correlation or no correlation at all. The benefit of our method is that we are able to assign more colors to different pairs of attributes, thereby enhancing the information gain for the visualization and the user.

Visualization and inverse relationships

For the basic visual encoding of the projection, we use the same basis used for the variance and distance contribution metric as discussed in section 2.6. We use the correlation metrics to compute, for each point p_i, a ranking vector $\{j, \mu_i^j\}_{1 \leq j \leq \frac{n(n-1)}{2}}$ for all $\frac{n(n-1)}{2}$ pairs. Each ranking vector is sorted in ascending order. For each rank, the frequency of that rank being the top rank for a point is calculated for the set of ranking vectors. Using a colormap containing C colors, the C most frequent top-ranked ranks are selected and assigned to one of $C-1$ colors. Pairs that are relatively

infrequently top ranked are collectively assigned the same color as they are not deemed relevant enough to enhance the visualization.

For the correlation metric, we will add another part to the visualization that uses saturation to differentiate between regions where there are many inverse correlations versus regions that are mixed in terms of normal and inverse correlations. In Section 2.6, recall we defined the terms for encoding the confidence of a rank by adjusting the strength of the luminance of that point's color. We will encode the *inverse confidence* ∇ in a similar way, but instead of visualizing this confidence by a point's luminance, we will adjust the the *saturation*. Regions which are homogeneous, with many strong inverse correlations, will show stronger saturation compared to mixed regions where there are both many strong inverse and normal correlations. We again recall we use a radius ρ to calculate the neighborhood v^p for a point p_i. To calculate the inverse confidence, we use a smaller radius ρ_i such that $\rho_{inv} < \rho$ to define a 2D neighborhood v^p_∇ centered at q_i. For each rank u^j_i, we add an attribute *inv* which is set to 1 if that rank encodes an inverse relationship, and 0 if it encodes a normal linear relation. Again, we identify t as the rank *ID* or *number* of the top-ranked attribute for the point p_i such that $t = \arg\max_{1 \leq j \leq n} \mu^j_i$. The inverse confidence ∇_i for point p_i is then defined as

$$\nabla_i = \frac{\sum_{q_j \in v^p_\nabla \wedge \arg\max_k \, inv(\mu^k_j)=1} \mu^t_j}{\sum_{q_j \in v^p_\nabla \wedge \max_k} \mu^t_j}$$

...meaning the inverse confidence is encoded as the sum of top-ranks u^t_j over all points $q_j \in v^p_c$ which encode the strength of an inverse relationship, normalized by the sum of all top-ranks over all points in v^p_∇. Recall we calculate one rank for each pair of attributes. The most frequent top-ranked pairs are assigned unique colors. The correlation value of a rank might be either strong as either a linear or linear inverse relationship, and we would like to indicate this difference in the visualization. If we did not use saturation, we would have to change our visualization scheme so that we calculate twice as many ranks, where for each pair we would create a rank for the strength of the linear inverse relationship, and the strength of the linear relationship. As always, we would then assign colors to the top most frequent top-ranked pairs. But under this scheme, because the colors would be unique, it would be harder to distinguish between ranks that have the same pairs but differ in what kind of relationship they encode. Furthermore, the colormap is always of a limited size, so ranks between the same pairs might take up more spots in the colormap, leaving less visualization options for other pairs. The use of saturation solves this issue, as it allows us to indicate different kinds of relationships between the same pair of attributes more easily, while also maintaining the property of the colormap to assign unique colors to qualitatively different top-ranked ranks.

An additional tool implemented for the correlation metric is a GUI matrix "pair color picker". As with the contribution and variance metrics, the C most frequent top ranked correlated pairs are encoded by assigning them to a colormap of size C. Ranked pairs that have a frequency higher than zero but lower than the C top ranked pairs are assigned to a common color B. The $n \times n$ matrix picker is organized as a set of squares, each associated with a pair of dimensions. The C most frequent top ranked pairs will have their square in the matrix colored according to their assigned color in the color map. The pairs assigned to color B will all have their squares colored using B. The utility of the matrix comes from the ability to select a square with the color B, such that that the pair associated with this square will replace the C-th most frequent top ranked pair, i.e. the least frequent top ranked pair assigned a unique color, in the projection. Once the user selects

a square with color B, the two pairs will switch colors in the colormap and the matrix and the projection will be re visualized based on the re-assignment.

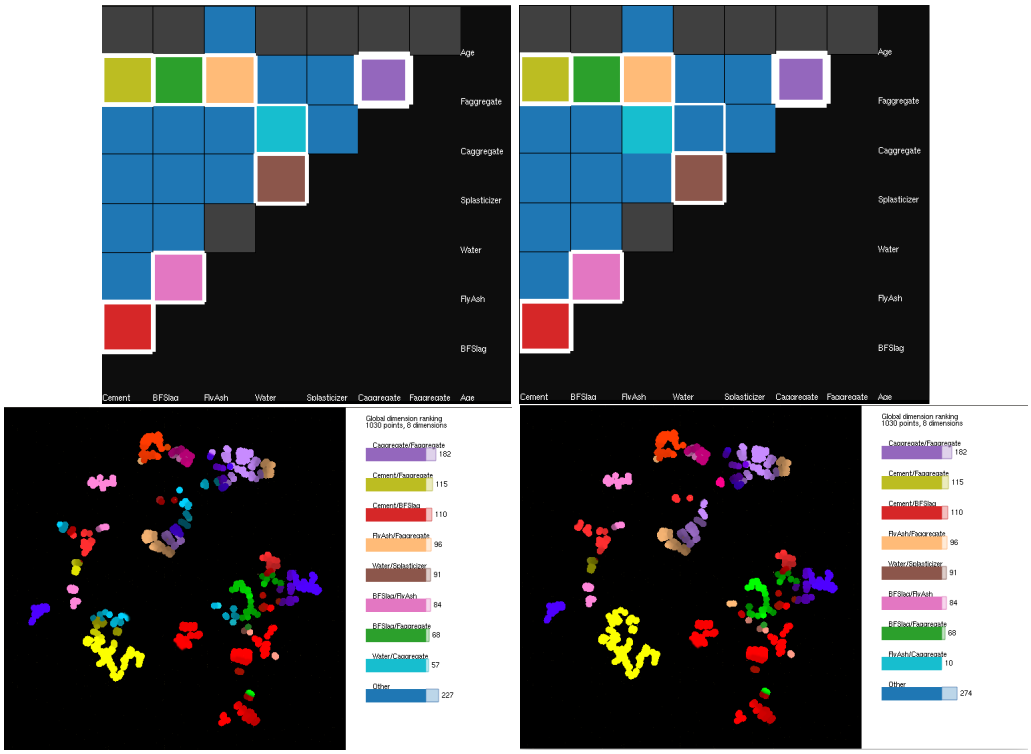


Figure 3.1: A example of a matrix color picker for the "concrete" ranked using the Pearson correlation metric [25] dataset

The main benefit of the matrix picker is to let one analyze where, on the data, ANY dimension-pairs are well correlated (even though these pairs are not the most frequent ones globally). In Fig. 3.1 we can see an illustrative example of the matrix picker. On the left we see the state of the original visualization and matrix, while on the right we see the state of the visualization with the pair $FlyAsh/Caggregrate$ replaced by the pair $Water/Caggregrate$. Note that the original C most frequent top ranked pairs will have their squares outlined in white regardless whether or not they've been switched; which is to make it easy for the user to return to the original visualization.

3.3 Dimensionality metric

3.3.1 Motivation

What purpose does the dimensionality metric serve? What information can it help identify through visualization and what kind of benefits are achieved through implementing it as a locally explanatory metric? The dimensionality metric should be able generate new information, to allow us to visualize the projection in a different way so that we can interpret the data differently compare to what we can achieve with established methods. Specifically, we want to be able to find, for each point, an integer k that estimates the inherent dimensionality for the point-neighborhood. By "inherent dimensionality" we mean to express the inherent "structure" or "shape" of the neighborhood, the count of dimensions that are considered relevant enough to adequately explain the neighborhood according to a fixed parameter. In each neighborhood, all attributes will, to some degree, contribute to the variance in that neighborhood. Some attributes will inevitably contribute more to that variance than other attributes, as is demonstrated by the variance contribution [3][4] metric.

As is detailed in Sec. 2.3, Principal Component Analysis (PCA) is a technique that allows us to derive a set of eigenvectors, or principal components, from a multidimensional dataset, that allow us to re-express the data with a limited basis composed of those eigenvectors, revealing the inherent structure of the data. By applying PCA to a point-neighborhood, we can derive eigenvectors for that neighborhood, which describe the structure of that neighborhood, and a set of associated eigenvalues which signify the contribution of each eigenvector to describing the variance in the neighborhood. If we know the eigenvalues of the point-neighborhood, then in their relation to the fixed parameter, we should able to estimate the size of k. Once we know the value of k for all points, we can generate a **heatmap** of dimensionality; where points with a higher value of k are assigned a more intense color compared to points associated with a lower value of k. As a result, we can differentiate between regions of the projection with different dimensionality.

But why is this useful? The dimensionality analysis tells us how many INDEPENDENT variables you really need to describe a phenomenon in some region of our data. Say we have a 100D dataset and locally, we are able to derive an inherent 5D dimensionality. Then, if we were trying to find a law which explains, say, how all the variables are linked by an equation, then we know that we could write 95 equations each which has only five 'free' variables, and each which explains one of the 'dependent' variables.

As a practical example of the usefulness of the metric, we consider the example of applying the dimensionality metric to a projection of a nD dataset which is fundamentally "random" i.e. **noise**. At each point, within each neighborhood, the inherent structure would be nD. If there is no pattern to the data, then it should not be possible to find pockets of the data where one dimension varies significantly more than the other. So already, we've found a use of our metric; allowing the user to find areas of noise within the projection. In other words, the metric allows the user to distinguish between areas with different structures, which vary and behave fundamentally differently compared to other regions. If our nD dataset, instead of being purely noise, would consist of both noisy areas and areas being fundamentally limited in their dimensionality, the user would be able to distinguish between areas of interest and areas that are too random or complex to consider. If we would apply the variance metric [3][4] to such a dataset, the amount of information we would be able to derive would be much more limited. Specifically, while it might easily be able to distinguish between

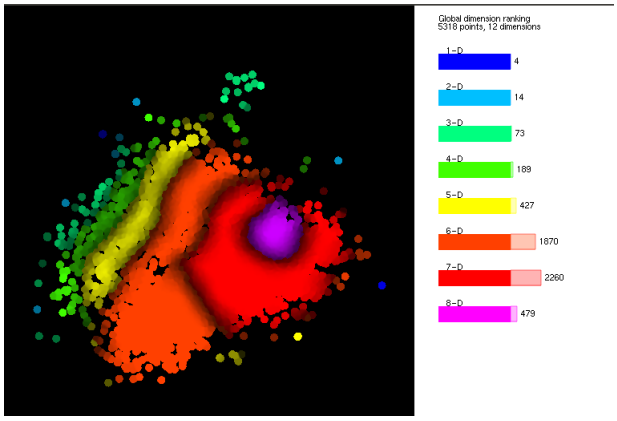


Figure 3.2: An example of a dimensionality "heatmap" of a MP of a dataset [5] used to analyze quality of wine

regions dominated by a particular attribute, it does not specify what the shape of those regions is inherently, and therefore omits information about how those regions are structured.

The dimensionality metric allow us to differentiate between regions of the projection with a different shape, a different classification of dimensionality of their variance, thereby giving us the ability to locate regions of interest in the projection that can be expressed and behave in a fundamentally more limited way compared to other regions. This can be used to complement established metrics [3][4], by giving us the ability to place those visualizations in perspective.

3.3.2 Specification

To calculate the eigenvalues of a point-neighborhood, we follow the directions given for PCA in Sec. 2.3. Here, the dataset X for which we calculate the PCA statistic would consist of the set of nD points in the neighborhood. In the matrix X, each point would be included as row vector of length n, or the number of attributes in the dataset. We derive the covariance matrix from X, and then derive the eigenvalues using eigenvalue decomposition. Once the eigenvalues have been calculated, it is possible to define a metric for characterizing the dimensionality k of the neighborhood according to a fixed parameter.

Cumulative Percentage of Total Variation

An intuitive way to define this parameter would be as a threshold value; which we can apply and define in two main variations. Both methods will assume that the eigenvalues are sorted in descending order, such that we have for the set of eigenvalues E:

$$E = \{ \quad \lambda_i >= \lambda_j \quad | \quad 1 \leq i < n \quad | \quad i < j \leq n \quad \}$$

The first thresholding method can be characterized as form of "cumulative thresholding". Namely, for a set of (sorted) eigenvalues λ_1 to λ_n, we can define the dimensionality k informally, as by the count of the largest eigenvalues that cumulatively contribute to a certain percentage, defined by θ, of the variance in the region.

$$\frac{\lambda_1 + ... + \lambda_k}{\sum_i^n \lambda_i} \geq \theta$$

where θ is used as a thresholding value between 0 and 1. We can find examples of descriptions of this method in I.T. Jolliffe's [10](Chapter 6.1.1) literature on principal component analysis. For our application, we also need a confidence value to encode luminance. We can encode the confidence by the fraction at which much the sum deviates from the mean:

$$conf_\theta = 1 - \frac{\sum_{i=1}^k \lambda_i - \bar{\lambda}}{\sum_{j=1}^n \lambda_j}$$

If we set θ to 0.8, then the k largest eigenvalues that contribute to at least 80% of the variance are considered to define the structure of the variance in the neighborhood region. As an example, assume we have a set of eigenvalues

$$E_\alpha = \{1.7, 1.2, 1, 0.1, 15.65e^{-4}, 13.25e^{-4}\}$$

In this example, it would be intuitive to declare that the three largest values in E_α significantly differ from the rest of the values. For any dataset however, the eigenvalues for the large number of neighborhoods that need to be calculated will differ significantly in their composition. As such, the use of θ allows for a fixed way to characterize the different regions which is also both simple and intuitive.

Size of Variances of Principal Components

While the composition of the eigenvalues of E_α allows for a very easy designation of dimensionality, other possible compositions of the sets of eigenvalues are more arbitrary. We can consider for example, an E_β that consists of one large eigenvalue, but also many, smaller eigenvalues which approximately have the same value:

$$E_\beta = \{0.2, \approx 0.05, \approx 0.05, \approx 0.05, \approx 0.05, \approx 0.05, \approx 0.05\}$$

In this example, it is obvious that the first eigenvalue contributes significantly to the variance, but it is not very clear where we should "cut-off" the rest of the eigenvalues to define k, provided we use θ to set a threshold. Given that the eigenvalues are about as equally large as each other, the value of θ could lead to arbitrarily and unfairly characterizing the neighborhood as a lower dimensionality than would seem appropriate. To deal with this limitation, we could define an alternative thresholding strategy which involves a minimum thresholding value η:

$$k = |\{ \quad \frac{\lambda_i}{\sum_{j=1}^{n} \lambda_j} \geq \eta \quad | \quad 1 \leq i \leq n \quad \}|$$

Here we simply count the number of eigenvalues that contribute **at least** a certain percentage to the variance. Examples of using similar kinds of thresholding method is can be found within explanatory factory analysis following Kaiser's criterion [26] and were also described in I.T. Jolliffe's [10](Chapter 6.1.2) literature on principal component analysis. To define the confidence, we calculate the normalized sum of all the eigenvalues that contribute to at least a fraction η of the variance:

$$conf_\eta = \frac{\sum_{i=1}^{k} \lambda_i}{\sum_{j=1}^{n} \lambda_j}$$

It should be stressed that neither of these thresholding methods would be absolutely better than one or the other. As we will show in Section 4.2, they can both be used to achieve the same result, but also to generate visualizations unique to each other.

In the software, both methods have been implemented, and both will be tested and compared to each other in Ch. 4. To make an appropriate choice between the two methods, it would be useful if we could more closely inspect the eigenvalues in the point-neighborhoods for ourselves. As such, the ability has been given to print the eigenvalues of a point-neighborhood. If the user clicks on a point in the MP, then the eigenvalues of that point's point-neighborhood will be printed.

Cumulative Differences in Similarity measured in distance ratios

A third alternative way to define k is through a calculation of ratios of differences. This technique is used in areas of research such as diffusion tensor analysis [18]. This method was considered for this project as it stands as an empirical, scientifically applied method for deriving the dimensionality of a set of values. More details about this process can be found at the end of Section 2.3. In diffusion tensor image analysis, the eigenvalues are limited to a set of three and are therefore more easy to differentiate i.e. simply picking the largest ratio among the three would provide for a solid basis to characterize the dimensionality. If the set of eigenvalues is potentially large, as it often will be

for neighborhoods of multidimensional data, then the choice becomes more arbitrary. To solve this issue, we could define a thresholding method to define the dimensionality $k = h + 1$:

$$\frac{\frac{\lambda_1 - \lambda_2}{\lambda_1} + ... + \frac{\lambda_h - \lambda_{h+1}}{\lambda_1}}{\sum_{i=1}^{n} \frac{\lambda_i - \lambda_{i+1}}{\lambda_1}} \geq \nu$$

Note that we define λ_{n+1} is as zero. Furthermore, if the first eigenvalue is already larger than ν, then the dimensionality of the set classified as 1. If the first eigenvalue is smaller than ν, the value of k is set to zero. Otherwise, the value of k equals $h + 1$. Instead of thresholding by a raw cumulative sum of eigenvalues, this method would represent a method thresholding cumulative ratios of difference divided by the largest eigenvalue. In diffusion tensor image analysis, deciding which ratio best characterizes the diffusion is done by examining which ratio is the largest i.e. looking for a certain point where the difference between the eigenvalues become the largest. The point is to find the place where two sorted eigenvalues most significantly differ and thus signify a "break" between contributions to the variance, thus giving an indication for a structure of dimensionality. The benefit of using a cumulative sum is the ability to capture several large "breaks" or difference ratios by thresholding them to a single value. What this method would emphasize is using the **similarity** of eigenvalues to determine the number of relevant principal components. In his review of methods that can be used to distinguish between components that are relevant and which are noise [10](Chapter 6.1.7) I.T. Jolliffe concludes that using *similarity* of components as a sensible standard in this respect, citing work from Besse and de Falguerolles (1993) [27] and North et al (1982)[28] which argue a similar position.

To imagine how this works in action, imagine if we applied this ratio thresholding metric to the set of eigenvalues E_α. The differences between the first three eigenvalues would be very small, and thus would contribute little to sum. However, the large difference between the third and the fourth eigenvalue would contribute much more, which makes sense at the fourth eigenvalue is relatively speaking less significant than the first three and could constitute a solid cut off point. The main benefit of this method lies in its ability to methodically account for the **differences** between the largest eigenvalues. To encode the confidence for the ratio metric, we can define:

$$conf_\nu = 1 - \frac{\sum_{i=1}^{h} \frac{\lambda_h - \lambda_{h+1}}{\lambda_0} - \frac{1}{n} \sum_{l=1}^{n} \frac{\lambda_l - \lambda_{l+1}}{\lambda_0}}{\sum_{j=1}^{n} \frac{\lambda_j - \lambda_{j+1}}{\lambda_0}}$$

Additional features and visualization

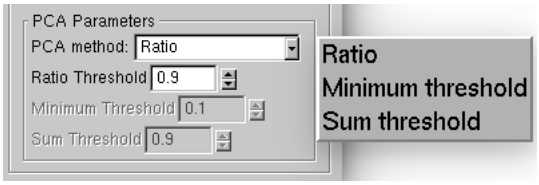


Figure 3.3: PCA GUI snippet for selecting parameters

Choosing appropriate values for θ, η, ν can be somewhat arbitrary and depend on the impression the user has with the dataset and what kind of information he wants to see in the visualization. For cumulative thresholding values θ and ν, a relatively large value makes more sense while picking small values is more sensible for η. As illustrated in Fig. 3.2, the option has been provided for the user to choose his thresholding method in a GUI and also set the thresholding value for each method. With the dimensionality metric, we no longer compute a ranking vector. Or rather, for each point i, we calculate ranking vectors of size 1, containing a single ranking value μ_i^k. The value of k indicates the dimensionality of the rank, while the weight of μ_i^k is encoded as either the value of $conf_\theta$ when applying the sum threshold method, the value of $conf_\eta$ when applying the minimum thresholding method, or the value of $conf_\nu$ when applying the ratio threshold method. The value of μ_i^k is also automatically the top-ranked value.

As mentioned earlier in this section, we want to visualize the projection as a heatmap of dimensionality. Formally, this "heatmap" is a *ordinal colormap* that gives us a way order the colors and ranks of dimensionality used during the visualization. With the dimensionality metric, we do not rank points by attributes any longer, but instead assign a number k as the point-neighborhood's rank. Therefor, it would not make sense to make use of a categorical colormap of a limited size. With a categorical colormap, there will be a clear distinction between regions because of the distinctive colors. When the ranks are *qualitatively* different i.e. describe different attributes, a colormap which assigns unique colors to the top most frequently top-ranked attributes makes sense. With the dimensionality metric however, the ranks are only *quantifiable* different from each other. Because there is a quantitative relation between the different ranks, where ranks can be sorted against each other in order, it makes sense to implement the colormap on a similar basis. In this sense, the configuration of the colormap should allow us to see the transitions between areas of lower dimensionality to areas of higher dimensionality and vice versa. As such, it is more appropriate to generate the colormap for ourselves as a heatmap, which assigns more intense colors, for example red, to ranks with a higher dimensionality but assigns softer colors, for example blue, to ranks with a lower dimensionality. Specifically, we calculate the count C of ranks that have a frequency larger than zero. These ranks are then sorted in ascending order, to generate a ranking vector F of size C. The rank with the lowest dimensionality is assigned the color blue, with $hsv(240, 100, 100)$. The color of the f-th rank with a frequency larger than zero is then calculated as $hsv(240 - f * (240/C), 100, 100)$.

Chapter 4

Testing and Results

This chapter features analysis, validation and exploration of the correlation and dimensionality metrics through testing on datasets and examination of those datasets. We validate a metric by testing the data on known, simple synthetic datasets where we are familiar with the shape and composition of the data. By visualizing synthetic data using the metric, we can test the correctness and behavior of the visualization in a controlled environment. We explore a metric by applying it on real-world data, allowing for evaluation of the robustness of the metric on more complex, unknown data and how it can be used for exploration of this data. Section 4.1 provides an overview of all data used, including generation methods and sources. Section 4.2 evaluates the correlation metric, while Section 4.3 evaluates the dimensionality metric. We perform testing of the software under hardware parameters including a Intel Core 2 Duo CPU P8600 @ 2.40GHz CPU, with a GeForce 320M/integrated/SSE2 GPU running on 64-bit Ubuntu 14.04

4.1 Test data organization

This section is dedicated to itemizing the terminology and shortcuts used when discussing and specifying figures in this section and will also specify the details behind each synthetic and real-world, measured data.

4.1.1 Figure tags and parameters

These tags are used to specify information about the figure listed.

- data : The data that is being analyzed.
- metric: the metric used to label and classify the projection.
  - *variance* = Variance contribution metric.
  - *contribution* = Distance contribution metric.
  - *pearson* = Pearson correlation coefficient metric
  - *spearman* = Spearman rank correlation coefficient metric

  - pca_{ratio} = Thresholding of eigenvalue difference ratios derived using principal component analysis

    The threshold value is specified by ν

  - pca_{sum} = Thresholding of cumulative sums of eigenvalues derived using principal component analysis

    The threshold value is specified by θ

  - pca_{min} = Thresholding of eigenvalues derived using principal component analysis using a min based threshold.

    The threshold value is specified by η

- projection : The projection method used to reduce the multidimensional data to a two-dimensional dataset, which then mapped as a multidimensional projection.
  - pca = Principal Component Analysis [10]
  - $lamp$ = Local Affine Linear Projection [14]
  - $tsne$ = t-distributed stochastic neighbor embedding [13]
- neighborhood : The method used to calculate the neighborhood v_i of a point
  - v^r : For all 2D projected points q_i 2D neighborhood $v_i^p = \{q \in D^p \quad || \quad | \quad ||q - q_i|| \leq \rho\}$ given a radius ρ. The nD neighborhood v_i^r is then defined as $v_i^r = \{p \in D | P(p) \in v_i^P\}$.
  - v^n Given a radius ρ, calculate v_i^r for all points q_i. Take $k = |v_i^r|$ and set v_i^n as the k-nearest n-D neighbors to p_i
- ρ: The size of the radius used to calculate the neighborhood. Continuous variable. Has a range $0 \leq \rho \leq 1$. The value of the radius indicates the percentage of the diameter of the projection.

4.1.2 Data specification

We want to test our metrics on real-world data to get an impression of how well they can help explore complex datasets in a scientific context. Examples of real-world data include a dataset measuring chemical content, such as alcohol, sugar or chloride, of different wines; a dataset containing of software quality, quantified by metrics such as lack of cohesion, coupling and structural complexity, of software projects on sourceforge.net. Each of these datasets contain thousands of measurements with about a dozen attributes. Their complexity will lend itself well to explore of how the metrics introduced in this thesis will function when applied to real-world research data, and validate their usefulness to discover patterns in the data. With synthetic data, we can decide for ourselves what shape the data takes. Unlike with real-world data, we'd know beforehand what the shape of the data is and therefore can also rationalize about what results we should expect if use metrics to visualize the data. This will allow us to establish so-called "ground-truths" about the correctness of our metrics; in the sense that we can validate that they are working correctly by applying them to these synthetic datasets.

(A) Synthetic data

i *linearnoise*
3D, 2000 samples. The data features 2D linear correlation, embedded in 3D, between the first and second dimension, with Gaussian distributed noise with $\mu = 0$ and $\sigma = 4$ between all dimensions. Generated using *Distribution Painter*

ii *linear_inverse*
3D, 2000 samples. The data features 2D linear correlation, embedded in 3D, between the first and second dimension, and also an inverse 2D correlation between the first and second dimension. Generated using *Distribution Painter*

iii 3*planes*
10D, 3000 samples. The data is created by first creating a 3D dataset and then embedding in 10D. This was created as 3 sets of 1000 samples with each set sampling a 2D plane. The first plane lies on the normal vector $[0, 1, 1]$, the second plane lies on normal vector $[1, 0, 1]$, and the third plane lies on normal vector $[1, 1, 0]$. Randomly distributed noise is added to the set with a range of -1 to 1.

iv *gauss_noise*
10D, 2000 samples. Gaussian distributed noise dataset with a very small standard deviation $\sigma = 0.0001$ with $\mu = 0$. Generated using a Python script.

v *cube_3d*
3D, 3000 samples. Three faces of a 3D cube. The points consist of 3000 random samples of each faces, 1000 for each face. Generated using a Python script.

vi $halfsphere$
3D, circa 2000 samples. The shape of the data represents a halfsphere defined by cosine and sinus functions over the x, y and z axes. Generated using a Python script.

vii *emb_cube_3d*
10D, 8000 samples. A uniformly sampled cube embedded in 10D space. Generated using a Python script.

viii *nd_clusters*
7D, 5000 samples. Each 1000 samples is sampled from a Gaussian distributed cluster. There are five clusters in total. The mean of the ith cluster is equal to $\mu_i = 5 * i$. The standard deviation for all clusters is equal to 1. The i-th cluster has a normally distributed random sampling from $i + 2$ dimensions. For each i-th cluster, we add small levels of noise with $\sigma = 1e^{-6}$ for $i + 7$ dimensions. Generated using a Python script.

ix *circles*
2D, 400 samples. Two circles generated using Scikit's *make_circles*() function [29] with a scale factor of 0.3 and noise factor of 0.05

x *circles2k*
2D, 2000 samples. Two circles generated using Scikit's *make_circles*() function [29] with a scale factor of 0.3 and noise factor of 0.05

(B) Measured data

1. *indian_liver* = A data set containing 416 liver patient records and 167 non liver patient records. The data set was collected from north east of Andhra Pradesh, India. Selector is a class label used to divide into groups(liver patient or not). This data set contains 441 male patient records and 142 female patient records. Dataset retrieved and description cited from the UCI Machine Learning Repository [30][31][32]

2. *winequality* = A dataset of 6497 samples of Portuguese vinho verde wine with 4898 samples of red wine, and 1599 samples of white wine. Each sample contains measurements of 12 physico-chemical such as acidity, residual sugar, and alcohol rate. The dataset is often viewed as a part of solving classification or regression tasks.[30][5]

3. $software\_ln$ = dataset which describes 6773 software projects from sourceforge.net written in C. Each project has 12 dimensions (11 software quality metrics and the projects total download count). These metrics were extracted using static analysis tools. [6]

4. *uscounties* = A 12D dataset which describes social, economic, and environmental data from 3138 USA counties.[22]

5. *concrete* = A 8D dataset of 1030 samples measuring the way 8 ingredients influence concrete strength.[30][25]

4.2 Correlation metric

4.2.1 Synthetic datasets and ground-truths

Figures 4.1, 4.2 and 4.3 are meant to demonstrate the ability of the metric to detect basic linear and inverse correlations while dealing with noise levels and curvatures. Figures 4.4, 4.5, 4.7 and 4.8 are meant to demonstrate the ability of the metric to detect and distinguish between different kinds of correlations.

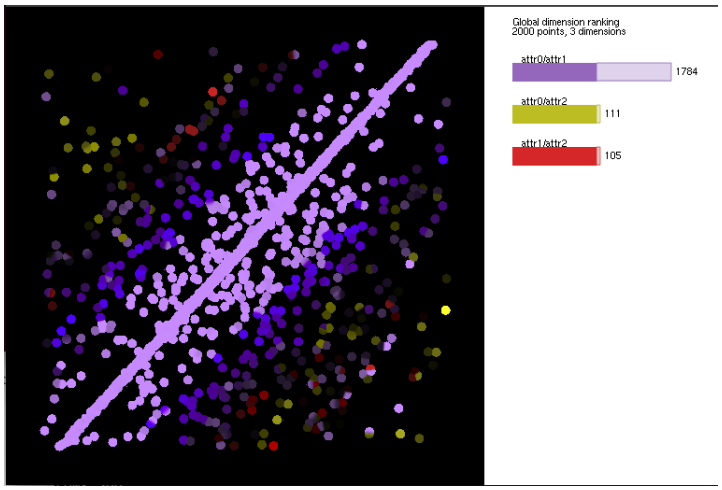


Figure 4.1: data=*linearnoise*, metric=*pearson*, projection=*pca*, neighborhood=v^n, $\rho = 0.1$

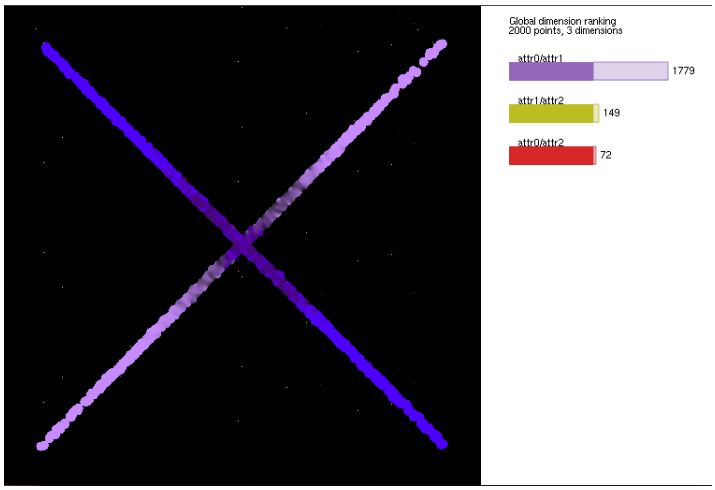


Figure 4.2: data=*linearinv*, metric=*pearson*, projection=*pca*, neighborhood=v^n, $\rho = 0.1$

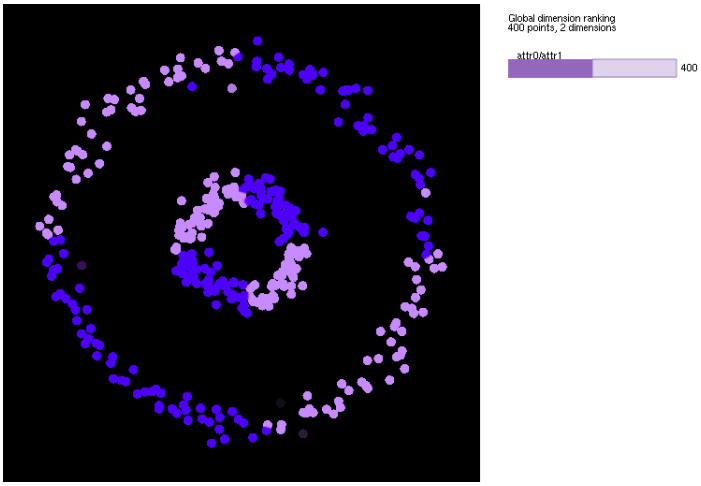


Figure 4.3: data=*circles*, metric=*pearson*, projection=*pca*, neighborhood=v^n, $\rho = 0.1$

Fig. 4.2 shows a dataset consisting 3 dimensions, 2000 samples. The data features 2D linear correlation, embedded in 3D, between the first and second dimension, with Gaussian distributed noise with $\mu = 0$ and $\sigma = 4$ between all dimensions.

Fig. 4.2 shows a dataset consisting 3 dimensions, 2000 samples. The data features 2D linear correlation, embedded in 3D, between the first and second dimension, and also an inverse 2D correlation between the first and second dimension.

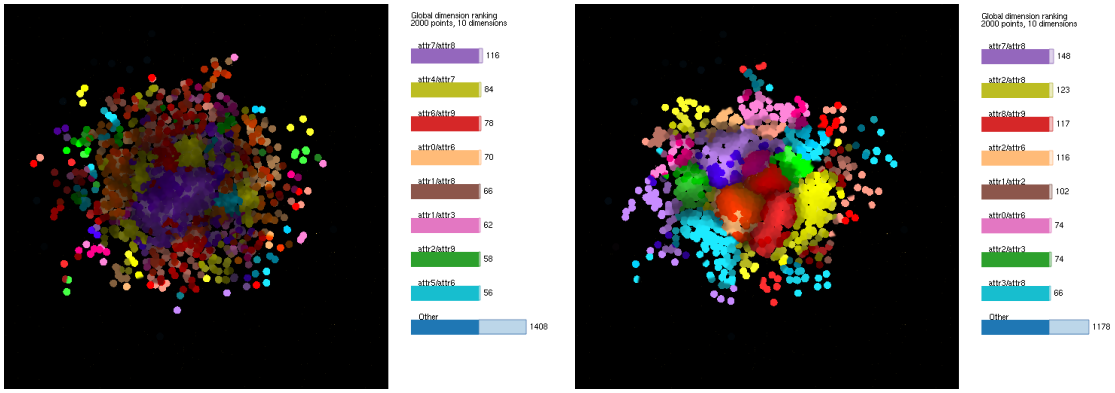
Fig. 4.3 features a dataset that is 2D, 400 samples. Two circles generated using Scikit's *make_circles*() function [29] with a scale factor of 0.3 and noise factor of 0.1

The main point of Fig. 4.1 is to demonstrate the ability of the metric to detect very linearly correlated 2D data, embedded in a low-dimensional space, with additional noise added to test fault detection. We can note some faulty correlation detections due to the noise. The first and third dimensions and the second and the third dimension are shown to be correlated for about 100 points each. This error is caused by the fact the metric must assign a ranking to each point and these can thus be simply be indicated as random errors.

Fig. 4.2 is meant show the ability of the tool to detect both linear and inverse correlations. In this dataset the 1st and 2nd dimension are very strongly correlated, both positively and negatively although both with two distinguishable distributions. Fig. 4.3 to show the same ability, but this time with circles created from signal functions that oscillate between negative and positive correlations. While Fig. 4.2 just shows the ability of the metric to detect different kinds of the same correlations, Fig. 4.3 also demonstrates its ability to changes in the kind of correlation **within** a single band of data.

Fig. 4.4 sees the application of the Pearson metric on a 10D dataset of Gaussian distributed noise with a mean of 0 and a standard deviation of 1. The result of the classification is a very noisy

image, with most a large amount of correlations detected as indicated by the large count of the "Other" category. For a continuous dataset, where local variation in attribute values is very small, we expect to see a more or less continuous picture were we to apply a categorical explanatory visualization. The idea being that the variation in the visualization is closely similar to the variation in the data. In this sense, the results in Fig. 4.4 are to be expected, as we see a large amount of variation in colors and variation.



(a) neighborhood=v^n (b) neighborhood=v^r

Figure 4.4: data=$gauss_noise$, metric=$pearson$, projection=$tsne$, $\rho = 0.1$

In Fig. 4.5, we see a LAMP projection of three 2D planes embedded in 10D space. One plane lies on normal vector $[0, 1, 1]$, another on normal vector $[1, 0, 1]$ and a third one lies on a normal vector $[1, 1, 0]$. This is a good testing example because if a plane lies orthogonal or parallel with two axes (i.e. another plane) then we should expect those axes to be correlated with each other and not with much else. The resulting projection results in three "bands" were we should expect correlations to be detected. Individually, because we have one plane on normal vector $[0, 1, 1]$ and one plane on normal vector $[1, 0, 1]$, we have one plane that lies parallel with the xz-plane and one that lies parallel with the yz-plane, and therefore expect to see at least one band that is primarily correlated with the first and third dimension and another that shows a correlation between the second and third dimension. If we examine Fig. 4.5, we can indeed see this to be the case. Because these planes are also orthogonal to each other, they intersect (both in nD and in 2D) and we see a mixed correlation in the intersection area.

For the final plane band, lying on normal vector $[1, 1, 0]$, we see a classification with oscillating correlations between the xz-plane and the yz-plane. This is the result of the fact that this plane is orthogonal to both the yz-plane and the xz-plane.

Figures 4.6, 4.7 and 4.8 show classifications of the *cube3d* dataset. In Fig. 4.6, we include the classification of the dataset using the variance contribution metric from da Silva et al. [3] for the sake of comparison. In Fig. 4.7, we see two classifications of the dataset using Pearson classification using the v^n neighborhood method (left) and the v^r neighborhood method (right). In Fig. 4.8, we see a Pearson classification of the dataset using the v^n method but using a radius twice as large.

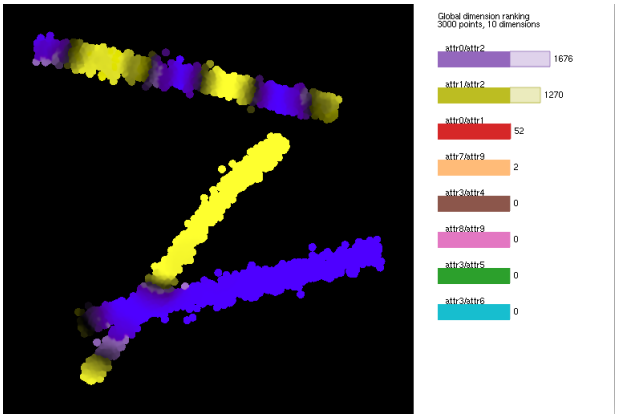


Figure 4.5: data=3*planes*, metric=*pearson*, projection=*lamp*, neighborhood=v^r, $\rho = 0.1$

Fig. 4.7 serves to illustrate two points: The first about how well the metric is at catching correlations between dimensions that are highly related to each other; and also to illustrate the effect and benefit of the v^n neighborhood calculation method. First, if we examine Fig. 4.6, we can see that a face is always dominated by one particular attribute. In Fig. 4.7, we can see that while both visualizations differ the classification that they return for the "edges" on the cube, the point at which the faces intersect, are the same (note that the colors are mapped differently in each visualization, which can be confusing). The only difference being that the illustration using the v^n method has a larger area classified with the correlations. If we now again compare Figures 4.6 and 4.7, we can see that the "edge", in Figure 4.7, that is classified as a correlation between x and z, is the intersection of the face labeled by x and the face labeled by z in Fig. 4.6. We detect a correlation at the edges as these are transitional regions between faces; which means point dimensions will start increasing/decreasing in value at approximately the same rate; causing the detection of a correlation. The edge between the x designated face and the z-designated face in Fig. 4.6 will have values that vary and increase/decrease most strongly along these dimensions. A similar relationship exists between the other faces and edges of the cube. This illustrates the ability of the metric to find intersections between different regions of MP by classifying them as (inverse) correlations thus creating an indication of structure in the MP.

About the second point, namely the difference between v^n neighborhoods and v^r neighborhoods, we can see that in the v^r visualization the faces of the cube are labeled with only one correlation, while in the v^n visualization the correlations in the faces seem mostly random. Now because the faces of the cube are randomly sampled, we should actually not expect to see a correlation in the area of the faces of the cube, and for that reason should judge the v^n visualization as the more accurate version. The reason for the difference is related to the fact that the v^n method takes the neighbors from the nD space, where differences are preserved, while the v^r is most likely the result of the compression of the PCA projection onto the 2D plane. The visualization in Fig. 4.8 also uses the v^n neighborhood method, but with a much larger radius. We now see a much more uniform classification, which is the result of the larger ρ value leading to more occurrences of sampled from two faces being included in the neighborhood of a point.

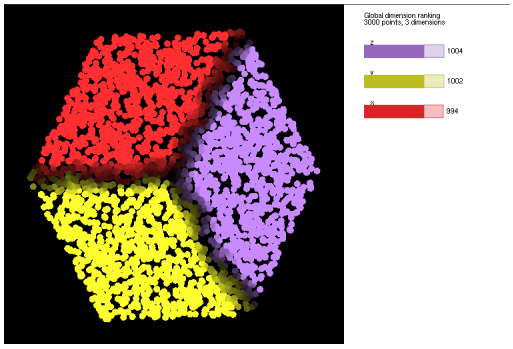
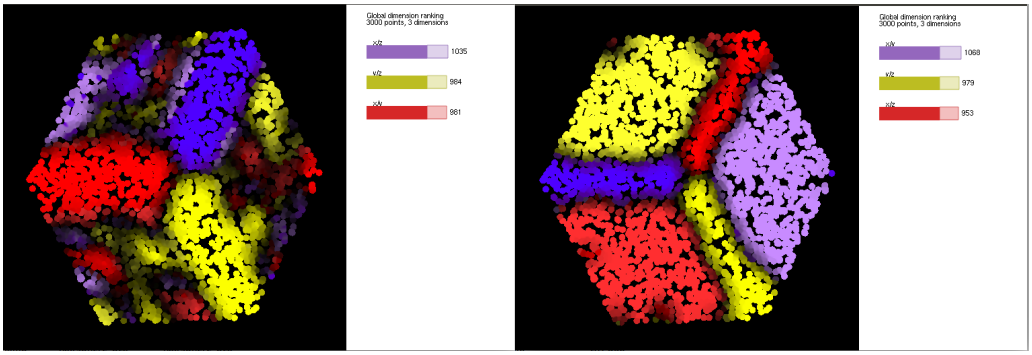


Figure 4.6: data=*cube3d*, metric=*variance*, projection=*pca*, neighborhood=v^r, $\rho = 0.1$



(a) neighborhood=v^n (b) neighborhood=v^r

Figure 4.7: data=*cube3d*, metric=*pearson*, projection=*pca*, $\rho = 0.1$

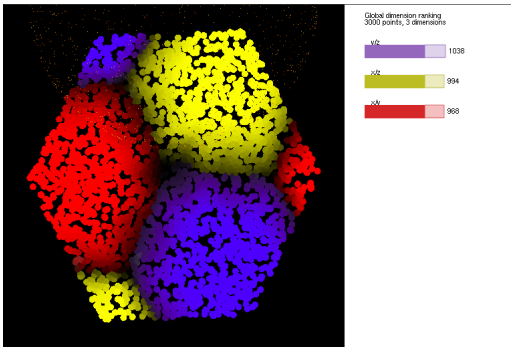


Figure 4.8: data=*cube3d*, metric=*pearson*, projection=*pca*, neighborhood=v^n, $\rho = 0.2$

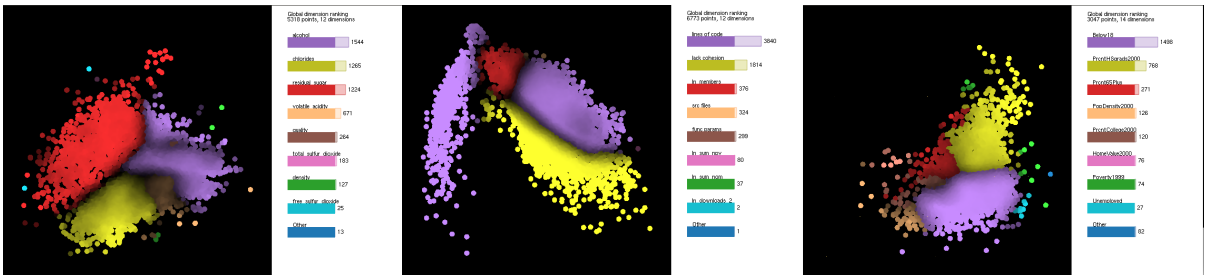
4.2.2 Real-world datasets

Wine quality, Software quality and United States Counties

To demonstrate the use of the correlation metrics, we first examine three sets of data:

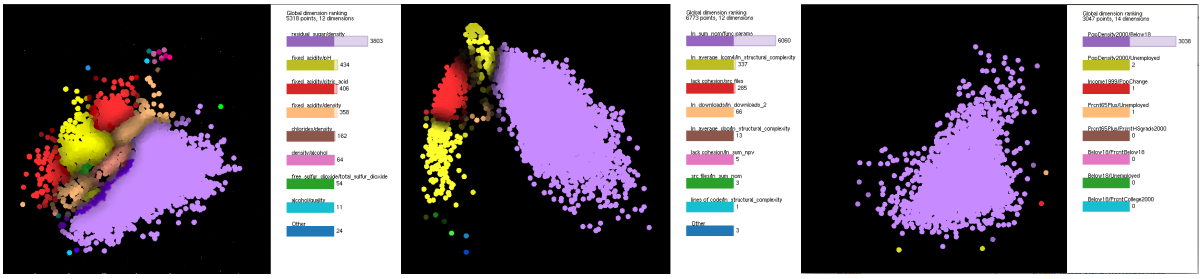
- *winequality* = A dataset of 6497 samples of Portuguese vinho verde wine with 4898 samples of red wine, and 1599 samples of white wine. Each sample contains measurements of 12 physico-chemical such as acidity, residual sugar, and alcohol rate. The dataset is often viewed from the perspective of solving classification or regression tasks.[30][5]
- *software_ln* = dataset which describes 6773 software projects from sourceforge.net written in C. Each project has 12 dimensions (11 soft- ware quality metrics and the projects total download count). These metrics were extracted using static analysis tools.[6]
- *uscounties* = A 12D dataset which describes social, economic, and environmental data from 3138 USA counties.[22]

Fig. 4.9 gives the visualization of each of the three sets using the *contribution* metric. Fig. 4.10 visualizes each of the three sets using the Pearson correlation metric. Each of the three sets has previously been used as examples to illustrate the use of the similarity metrics as applications on real-world by da Silva et al.[3][4].



(a) data=*winequality* (b) data=*software_ln* (c) data=*uscounties*

Figure 4.9: metric=*contribution*, projection=*lamp*, neighborhood=v^r, $\rho = 0.1$



(a) data=*winequality* (b) data=*software_ln* (c) data=*uscounties*

Figure 4.10: metric=*pearson*, projection=*lamp*, neighborhood=v^r, $\rho = 0.1$

The first visualization that has our interest is the Pearson correlation for the *uscounties* dataset in Fig. 4.10, which gives a correlation for the *population density* and *under*18 attributes over the entire projection. In Fig. 4.9, we can see that the MP is partitioned into a number of regions, one of which is the *under18* attribute. Another smaller region on the left is assigned to the *population density* attribute. In previous research performed by da Silva et al. [3][4] the *population density* attribute was also found to have a significant contribution to the similarity of points in the area of the *under18* region. The Pearson correlation analysis implies that the rate at which their contribution expands or decline throughout the projection is consistent with each other at a linear rate, which is emphasized by the fact that their strongest contributions originate in a close general area. Because these values are so heavily correlated, at least in comparison to the other pairs, there is also an implication that if other attributes differ in comparison to the *under18* attribute, then they differ in the same way to the *population density* attribute.

Next we can examine the classifications of the $software\_ln$ dataset. In Fig. 4.9(b), we can see that the *lines of code* attribute defines two basic regions, which is separated by the *number of members* attributed region. There is also a secondary yellow region defined by the *lack of cohesion* attribute. In Fig. 4.10(b), we can see that the right *lines of code* region, the *number of members* region and the *lack of cohesion* feature correlations between the *function parameters* and *number of modules summed* attributes. Meanwhile, in the left *lines of code* there are two correlations: one between *structural complexity/average lack of cohesion on methods* and attributes and another between the *source files/lack of cohesion* attributes. In the original study [6](Table ii), the authors analyzed the different variables with different correlation metrics, and the *structural complexity/average lack of cohesion on methods* pair, were found to correlate at a value of 0.786, which is notably the strongest correlation for these attributes in the table. From the table, we can also see a strong correlation between the *structural complexity* and *lines of code* region at 0.666, the third strongest for *structural complexity*, and also a correlation between *average lack of cohesion on methods/lines of code* with coefficient 0.472, the second strongest for the *lines of code* attribute; which rhymes with the observation that we find the *structural complexity/average lack of cohesion on methods* correlation within a region strongly [4] dominated by the *lines of code* region. That we find a correlation featuring the *lack of cohesion* attribute within this region also makes sense in the same manner. Information about the *function parameters* attribute in the study [6] is lacking, but the *number of members* attribute is also shown to correlate heavily with the *lines of code* attribute. We can conclude that the correlation metric, in conjunction with the *contribution/variance* metrics, can reveal layers of relationships between attributes.

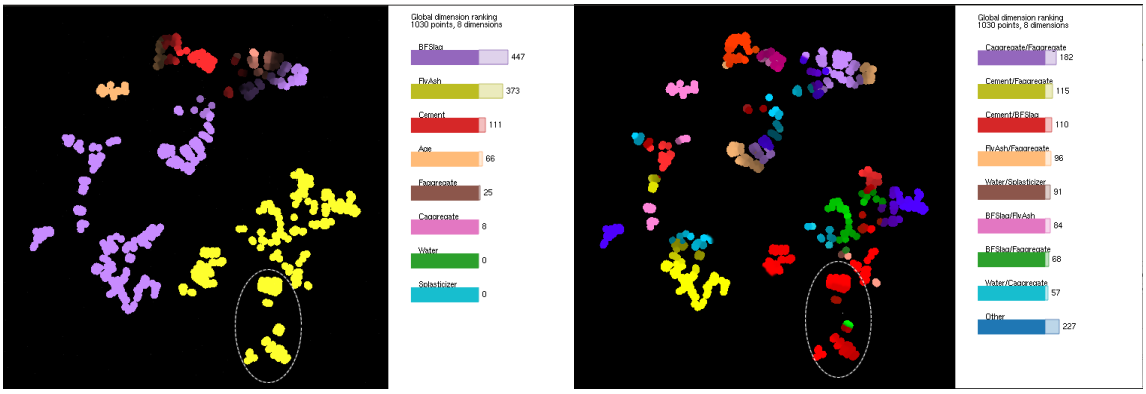
The results for the *winequality* dataset are also notable. The *residual sugar* attribute is shown to correlate over a broad region with the *density* in a region which in Fig. 4.9(a) where most of the similarity is explained by the *chloride* and *alcohol* attributes. Within the region where *residual sugar* explains most of the similarity between points, the *fixed acidity* attribute is shown to correlate with three different attributes. In the original study [5](Fig. 4), the authors postulate that *alcohol* is one of the most important factors in explaining a higher wine quality. If *residual sugar* and *density* are correlated over the *alcohol* region, then that can imply that significant variations in alcohol, and therefore wine quality, can also feature an added relationship of *residual sugar* and *density*. Indeed, increasing alcohol content tends to lead towards increases in wine quality [5]. *Residual Sugar* is also shown to significantly affect the quality of white wine in particular [5](Fig. 4), and as such the presence of many correlations featuring *fixed_acidity* imply a similar relationship i.e. increases in

wine quality and *residual sugar* might coincide with correlations featuring *fixed_acidity*.

Beh E J and Holdsworth C I [34] investigated this wine quality dataset from the perspectives of correspondence analysis, multiple regression analysis, classification and visual evaluations. Using the classification technique of P. Cortez et al. [5], they examine the mean quantity of each attribute for the classification as scored by assessors [34](Table 3). This process leads the authors establish a relationship between lower values of *residual sugar*, *density*, *fixed acidity* and *volatile acidity* and higher quality white wine. On the other hand, stronger values of *alcohol*, *pH* and *sulphates* are implied to lead to higher quality wine. For red wine, high levels *alcohol*, *sulphates* are also implied to be a strong indicator of quality, while low levels of *chloride* might lead to higher quality red wine and *citric acid*, *residual sugar* and *density* are considered to be statically irrelevant in predicting red wine quality [34](Table 6). If we relate Fig. 4.10 with the findings of Beh E J and Holdsworth C I [34], we can confirm an additional layer behind correlation between *residual sugar* and *density*, specifically in regions where most of the similarity is explained by *chloride* and *alcohol*, as all of these attributes add to predicting the quality of wine; and that we can further conclude that *residual sugar* and *density* might express a property of multicollinearity in their inverse relationship with higher quality wine.

Concrete

In this section we analyze the *concrete* dataset, which is a 8D dataset of 1030 samples measuring the way 8 ingredients influence concrete strength.[30][25]. The ingredients are *cement*, *blast furnace slag*, *fly ash water*, *superplasticizer*, *coarse aggregate*, *fine aggregate*, each measured in kg in a m3 mixture. There is also a additional *age* attribute measured in Days (1 365) . In Fig. 4.11 we compare the visualizations produced from the *contribution* and *pearson* metrics.

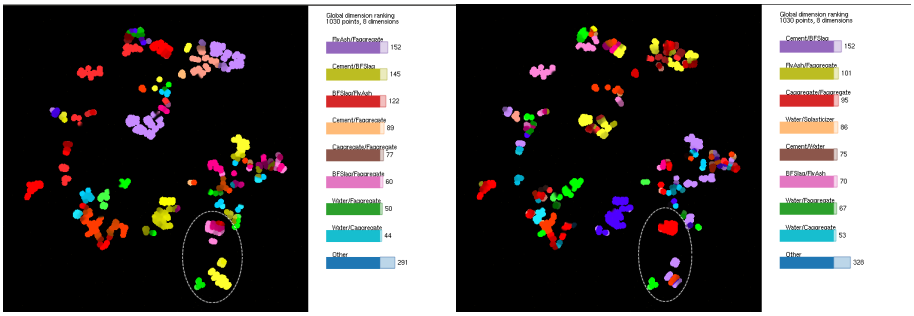


(a) metric=*contribution* (b) metric=*pearson*

Figure 4.11: data=*concrete*, projection=*tsne*, neighborhood=v^r, $\rho = 0.1$

This dataset was analyzed as part of validation of the similarity metrics by da Silva R [4](Fig. 5.10). The author concluded that the group of clusters encircled in Fig. 4.11 were associated with high concrete strength. The author also concluded[4] that the *BFSlag* (*blast furnace slag*) attribute had interesting correlations with this group of points, with the *blast furnace slag* also tending

towards high values in this region. In Fig. 4.11, we see a correlation between the *cement* and *blast furnace slag* attributes in this region. Now, if *cement* and *blast furnace slag* are correlated with each other, and *blast furnace slag* is correlated with high concrete strength values, the it is also likely that *cement* is in some form related to concrete strength as well. In Fig. 4.12, we decrease the radius ρ to see if any additional subregions of correlations can be identified with the encircled group of clusters. We find additional strong correlations finely separating the clusters in the encircled region. In Fig. 4.12(a), we can note a correlation, partially inverse and partially non-inverse, between *blast furnace slag* and the $Faggregate$ (*fine aggregate*) attributes in the pink upper cluster of the encircled region, and in turn also a correlation between *water* and *fine aggregate* in the *green* lower cluster of the encircled region. The *cement*/*blast furnace slag* (yellow) correlation remains strong in the middle cluster. In Fig. 4.12, we see the *cement*/*blast furnace slag* and *water*/*fine aggregate* correlations remain in the lower and middle clusters, and the upper cluster locating an additional correlation between $Caggregate$ (*coarge aggregate*) in the red upper cluster. Now because the *blast furnace slag* was found to (inversely) correlate with the *fine aggregate* attribute in this region, the *fine aggregate* might be inversely related to high concrete strength values (especially in combination with increasing *blast furnace slag* values). And because the *fine aggregate* might be inversely related, and we located an inverse *water*/*fine aggregate* correlation, and also a *coarge aggregate*-*fine-aggregate* correlation, both the *water* and *coarge aggregate* attributes might help explain high concrete strength values, in the sense that they inversely contribute to high concretes strength values. Taken together, the *blast furnace slag*, *cement*, *fine aggregate*, *water* and *coarse aggregate* might all together help to shape a regressive model that models concrete strength in the dataset. Wu S S et al. [33] conducted a study of this dataset from the perspective of predictive modeling, for which the authors among other metrics index a table of Pearson Correlation coefficients between the attributes in the data[33](Table II). The study found a relatively strong positive correlation between *cement*/*blast furnace slag* of 0.29, negative correlations between *blast furnace slag*-*coarse aggregate*($= -0.31$) and *blast furnace slag*/*fine-aggregate*($= -0.31$) and a inverse correlation between *fine-aggregate*/*water*=(-0.44). The results seem to rhyme with our findings in the visualizations. However, the authors concluded there was little chance for multicollinearity[33], a phenomenon where two or more attributes are highly correlated in a regressive model. Hench, the visualized correlations in the encircled region might not be very significant. This gives us an insight in the limitations of the correlation metric. If however, we did not do an extensive analysis of the data using complicated models, the visualizations would be useful in giving us an initial indication of the behavior of the data.



(a) $\rho = 0.005$ (b) $\rho = 0.003$

Figure 4.12: data=*concrete*, metric=*pearson*, projection=*tsne*, neighborhood=v^r

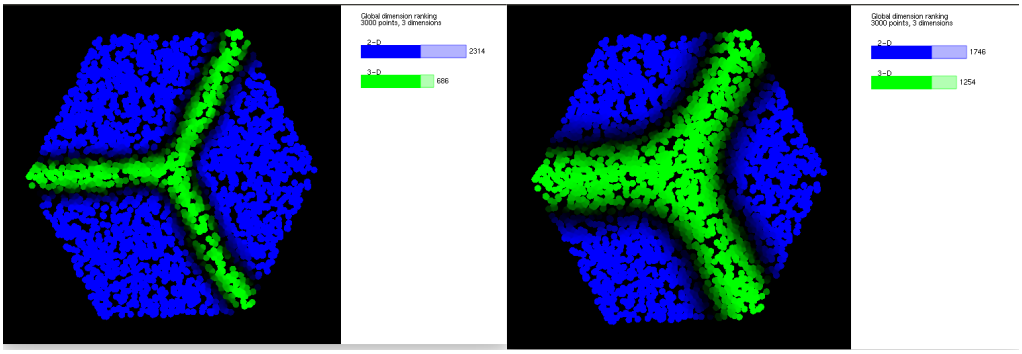
4.3 Dimensionality metric

4.3.1 Synthetic datasets and ground-truths

Figures 4.13, 4.14, 4.15 analyze the *cube3d* dataset using all three variations of thresholding metrics. For PCA_{min} and PCA_{sum} metrics, the visualizations for a larger radius are also given in Fig. 4.13(b) and 4.14(b). The visualization show that all three thresholding methods are capable of indicating the same overall structure in the cube. Specifically, we can see that the faces of the cube are classified with a dimensionality of 2 while at the edges of the cube, the intersections of the faces, are given a dimensionality of 3. This should intuitively make sense. The faces are, if we recall the description of the dataset, defined primarily by two dimensions each. When the faces intersect, at the edges of the cube, all three dimensions vary much closer to each other. Increasing the radius in 4.13(b) and 4.14(b), we see that the area designated as 3D becomes somewhat larger. This is because more points have other points in their neighborhood taken from all three faces. Now if we pay attention to the values of η and θ, we see their values are quite high and the thresholding is therefore quite strict.

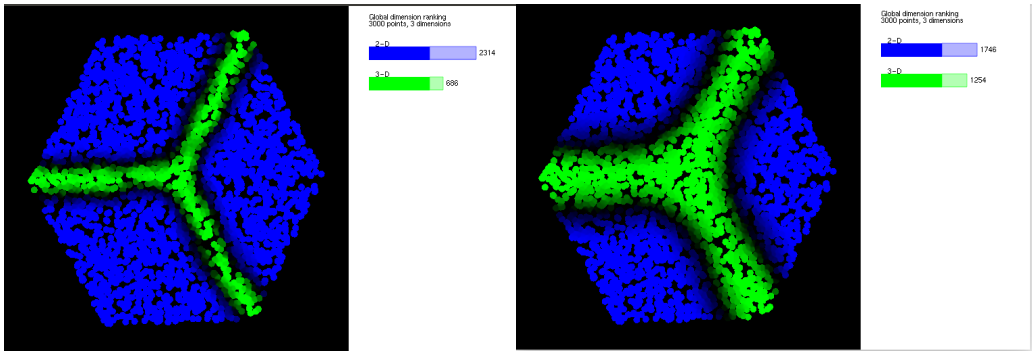
In Fig. 4.16 , the visualizations using the PCA_{sum} metric using less strict θ values is given and can be compared to Fig. 4.14. When using a smaller $\theta = 0.9$ value, the PCA_{sum} metric is only capable of detecting the most intense areas of intersection as 3D. In Fig. 4.17, we can see the same holds true for the PCA_{min} metric when using a less strict value for $\eta = 0.1$. When the thresholding value has to be set so high, it betrays a certain weakness for these metrics, as small deviations in thresholding values can apparently have them make inaccurate visualizations. In contrast if we examine Fig. 4.15, the PCA_{ratio} method shows no such issues with being able to detect the edges using looser thresholding values; albeit with arguably less accuracy when using the PCA_{sum} and PCA_{min} methods with a very strict thresholding value. Overall, this shows the PCA_{ratio} is the most flexible of the three methods and the least sensitive to small differences in parameters of the three.

Fig. 4.18 shows the visualization, using the PCA_{min} metric, for a collection of clusters where each cluster is distributed along a different count of dimensions. The clusters all have slightly shifted means, but have the same standard deviation. The purpose of the visualization is to show the ability of the dimensionality metric to distinguish between different groups of dimensionality. In Fig. 4.19, 10D Gaussian noise, centered at 0 with a very small $\sigma = 0.0001$, labeled using the PCA_{ratio} metric is visualized using both v^r and v^n neighborhoods. Areas that are very noisy do not have any clear relationship between the underlying attributes, and as such should always be classified as very high-dimensional. As such, what we expect to see is a projection that is colored with very "intense" colors within the heatmap. For Fig. 4.19, we can indeed see this is the case for both neighborhood calculations. The v^n neighborhood calculation method seems to have an edge over the radius-based method, as we expect the noisy area to be classified as entirely 10D. Less accuracy in the v^r visualization is most likely again the result of PCA compression where points that are part of a noisy area in nD get placed far away in 2D.
Fig. 4.20 shows the visualization, using the PCA_{ratio} metric, on a 3D cube uniformly sampled between an interval of 1 and -1, which is embedded in 10D space with 10D Gaussian distributed noise with $\mu = 0$ and $\sigma = 1$ added. The point of the visualization is to test the ability of the dimensionality metric to detect "hidden" areas, which are fundamentally m-D, but which embedded in a nD space with $n > m$.



(a) $\rho = 0.1$ (b) $\rho = 0.2$

Figure 4.13: data=*cube*3*d*, metric = PCA_{min}, $\eta = 0.05$, projection=*pca*, $neighborhood = v^n$



(a) $\rho = 0.1$ (b) $\rho = 0.2$

Figure 4.14: data=*cube*3*d*, metric = PCA_{sum}, $\theta = 0.95$, projection=*pca*, $neighborhood = v^n$

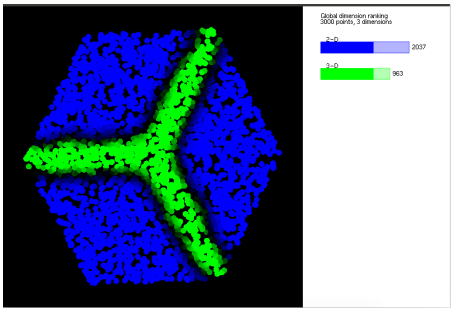
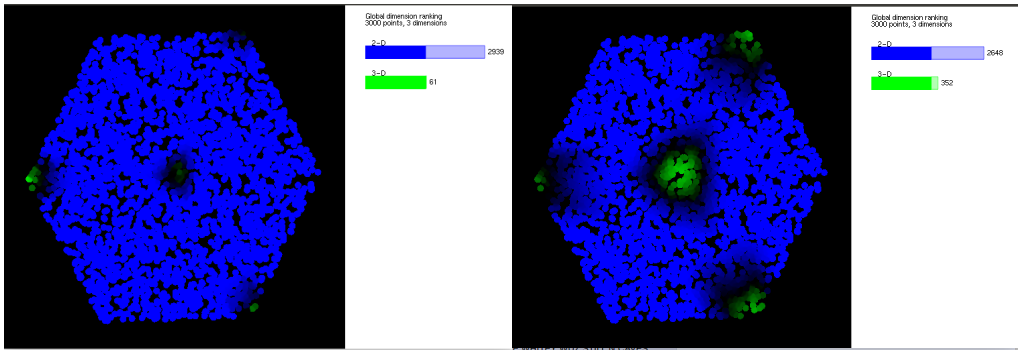
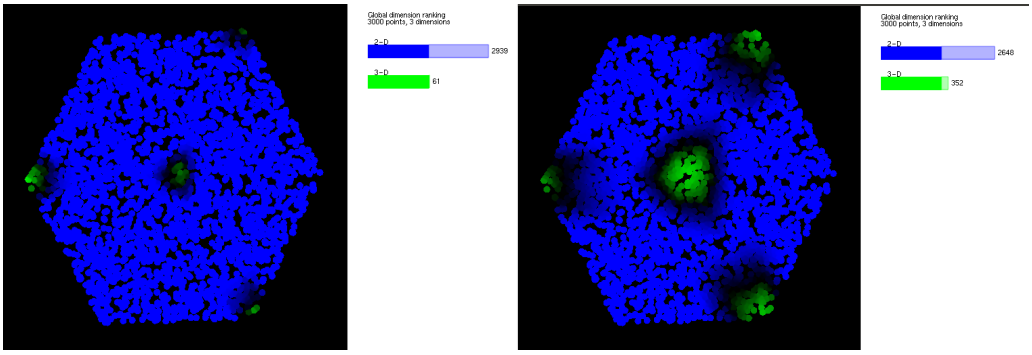


Figure 4.15: data=*cube*3*d*, metric = PCA_{ratio}, $\nu = 0.90$, projection=*pca*, $neighborhood = v^n$



(a) $\rho = 0.1$ (b) $\rho = 0.2$

Figure 4.16: data=*cube*3*d*, metric = PCA_{sum}, $\theta = 0.9$, projection=*pca*, neighborhood=v^n



(a) $\rho = 0.1$ (b) $\rho = 0.2$

Figure 4.17: data=*cube*3*d*, metric = PCA_{min}, $\eta = 0.1$, projection=*pca*, neighborhood=v^n

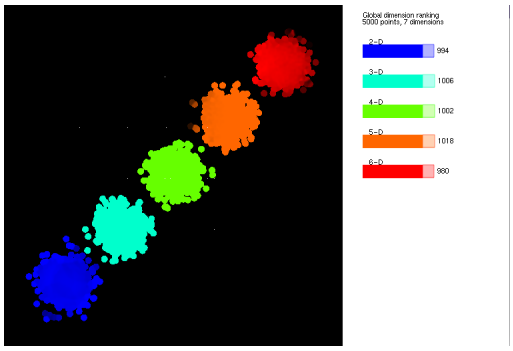
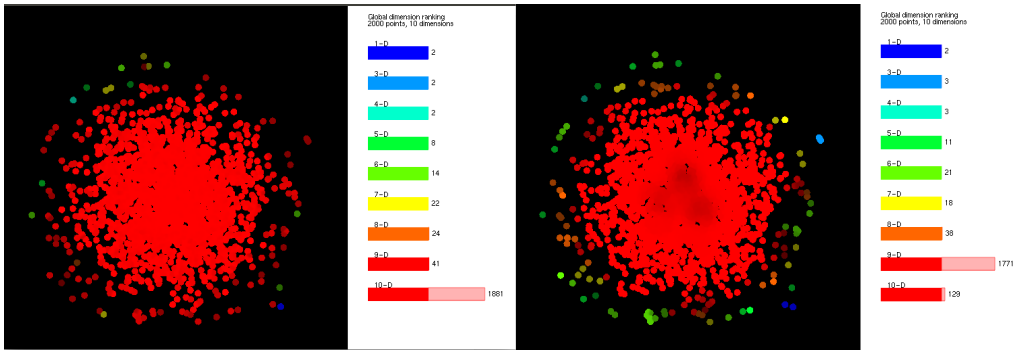


Figure 4.18: data=*nd_clusters*, metric = PCA_{min}, $\eta = 0.1$, projection=*pca*, *neighborhood* = v^n, with $\rho = 0.1$



(a) neighborhood=v^n (b) neighborhood=v^r

Figure 4.19: data=*gauss_noise*, metric = PCA_{ratio}, $\nu = 0.9$, projection=PCA, , $\rho = 0.1$

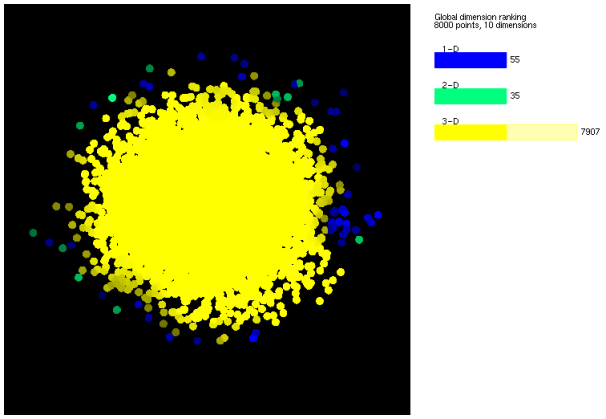


Figure 4.20: data=emb_cube_3d, metric = PCA_{ratio}, $\nu = 0.9$, projection=PCA, neighborhood=v^n, $radius = 0.1$

Fig. 4.21 applies the PCA_{sum} metric to a 2D dataset generated using Scikit's *make_circles*() function [29] with a scale factor of 0.3 and noise factor of 0.05. As the circle is closed and therefore not a disk, the variance on the edge is dominated primarily by one dimension. In Fig. 4.22 and Fig. 4.23, the sum and ratio thresholding metrics are applied to a 3D halfsphere. Again, both v^n and v^r neighborhoods are used to emphasize the benefit of the v^n method. Specifically, the area of the surface of a halfsphere is essentially 2D, and the projection of the halfsphere is visualized as a plane, so we expect the entire projection to be labeled as 2D. Again, using PCA leads to compression of the points near the border of the sphere. As a result, points which are close to the border in nD are denser in the projection in 2D. If in the 2D projection we select 2D points which are in a round neighborhoods of a border point, then in 3D, along the tangent-to-border direction, the size is about the same in 2D and nD; along the radius direction, the size is much larger in nD than in 2D. As a result, the circular region we take in 2D would not correspond to a circle-like region in nD.

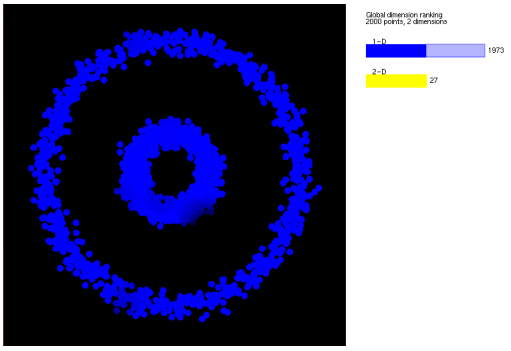
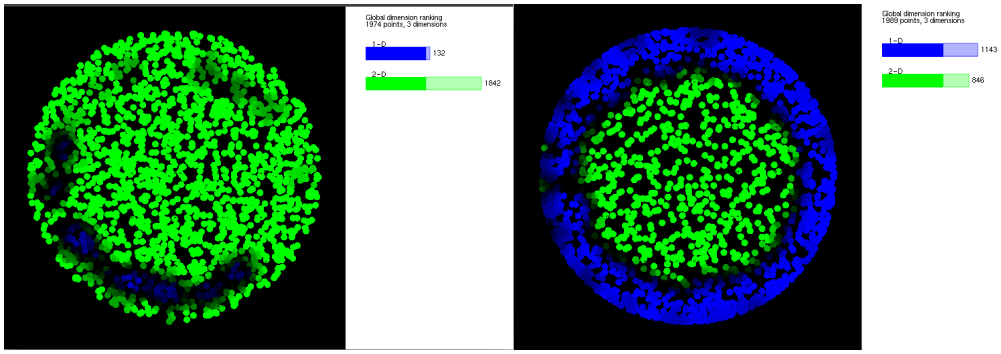
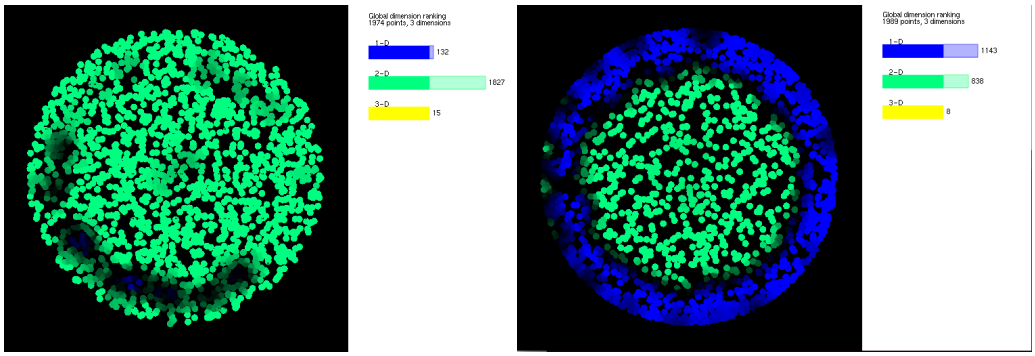


Figure 4.21: data=$circles2k$, metric=PCA_{sum}, $\theta = 0.8$, projection=pca, neighborhood=v^n, $\rho = 0.1$



(a) neighborhood=v^n (b) neighborhood=v^r

Figure 4.22: data=$halfphere$, metric=PCA_{sum}, $\theta = 0.9$, projection=pca, $\rho = 0.1$



(a) neighborhood=v^n (b) neighborhood=v^r

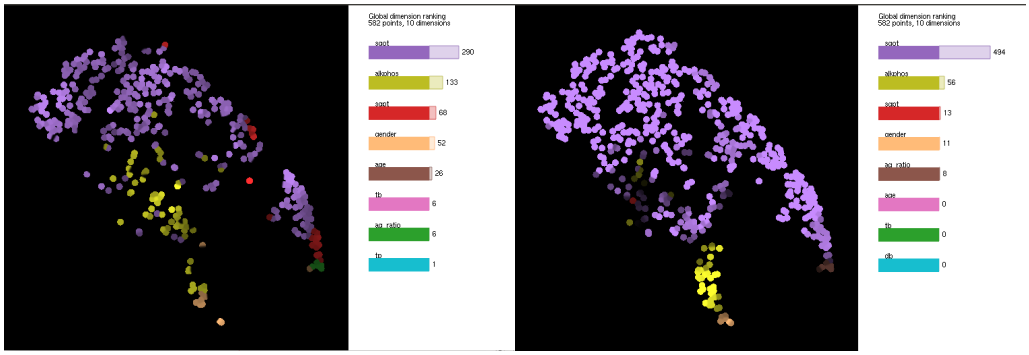
Figure 4.23: data=$halfphere$, metric=PCA_{ratio}, $\nu = 0.9$, projection=pca, $\rho = 0.1$

4.3.2 Real-world datasets

Indian Liver

One of the key limitations[3] of the original variance and contribution metric devised by da Silva et. al was that while they could reliably attribute regions to dimension within MPs, they lacked the ability to identify to which degree those dimensions explained the position of the points within the region. Applying the dimensionality metrics to the real world *indian_liver* dataset allows us to demonstrate how the metrics complement the original metrics by resolving this limitation. The dataset contains patient records from 416 liver patient records and 167 non liver patient records, collected from north east of Andhra Pradesh, India. Such datasets can be useful for identifying attributes that correlate with liver illnesses. Using a several different classification Algorithms variance[32](Table 14), identified the *sgpt* (Sgpt Alamine Aminotransferase), *alkphos* and *sgot* (Sgot Aspartate Aminotransferase) as a significant factors in diagnosis of liver disease. In Figure 4.24, we have applied the *contribution* and *variance* metrics to the dataset. In both Fig. 4.24(a) and Fig. 4.24(b), we see that the *sgpt*, *alkphos* and *sgot* attributes are listed as the top 3 most frequent top-ranked dimension ranks in the projection. The results of the visualization are quite interesting, as they seem to rhyme with the results of the more complicated classification algorithm[32] Noticeably, the *sgot* seems to dominate more in the projection than the other two attributes. But what other information can we get from the visualizations?

In Fig. 4.25, we have applied the PCA_{sum} metric with increasing values of θ. Of particular interest is Fig. 4.25(a), where we can see that the region roughly attributed to *sgpt* in Fig. 4.24 also has 50% of its variance explained by a single dimension. This indicates that the inherent structure of the region dominated by *sgpt* in the visualization in Fig. 4.24 is overall quite low.

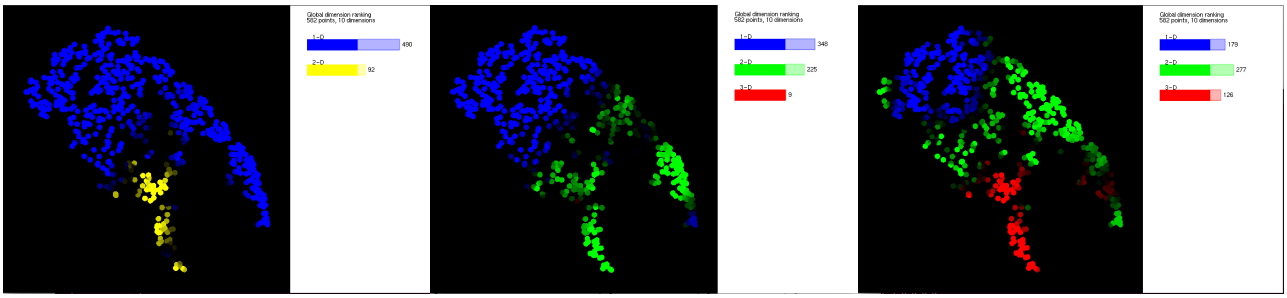


(a) metric=*contribution* (b) metric=*variance*

Figure 4.24: data=*indian_liver*, , projection=*tsne*, neighborhood=v^n, $\rho = 0.1$

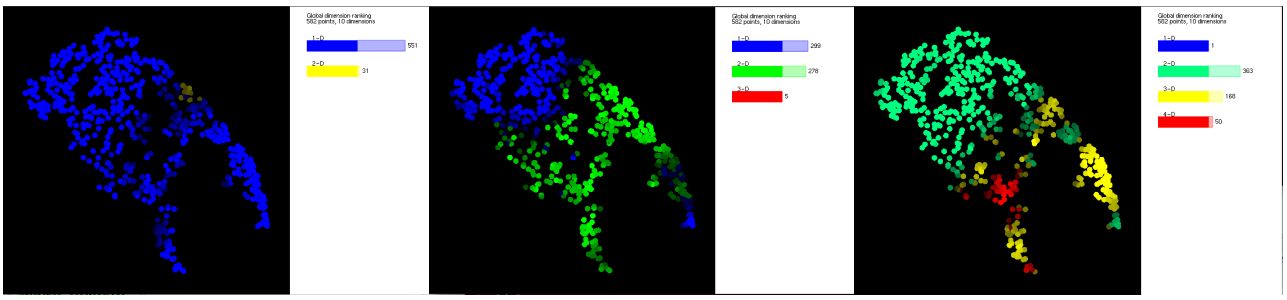
In Figs. 4.25(b) and 4.25(c) however, we see that the 1D region recedes with an increased thresholding value. In Figures 4.26, we analyze the data using the PCA_{min} metric with a decreasing η value. Here we see that the recession of the 1D region in Fig. 4.25 is caused by by mostly a single dimension with variance contribution of 20%, which gradually "erodes" into the northwest from the southeast. In Fig. 4.27 the results of applying the PCA_{ratio} metric are also given, which rhyme

with the results in Fig. 4.25 and 4.26. However, we can also note the visualizations in Fig. 4.27 also contain a more detailed transition between regions. Taken together, these results tell us that there is a very deep "pocket" of the region in the north-west that is only explained by one significant dimension and where the *sgpt* attribute might be more important than usual. In Fig. 4.24, we see that in the purple region encoded the *sgpt* attribute, there are small spots of points which are labeled by the *sgot* or *alkphos* attributes, which gradually seem to decrease in frequency as we approach the firm 1D region identified in Fig. 4.25. In Fig. 4.28, the dataset is analyzed using two very high threshold values with both the PCA_{ratio} and PCA_{sum} metrics, the visualizations show the low-dimensional pocket of the projection to be close to 3D in its dimensionality even when using a very high threshold values. The fact that the region remains relatively low-dimensional would indicate a high confidence for the inherently low-structure of the dimensionality of the region. If we take all of the results together, given the fact that we know from other research[32] that the *sgot*, *sgpt* and *alkphos* are related to each in predicting liver disease, and that we can guess that they tend to contribute more to the variance for points closer to the low-dimensional pocket, we can say that there is a possibility that this pocket is very strongly associated with points related to patients with liver disease. While this conclusion is far from definite, the visualizations showcase the ability of the metric to visualize the data in such a way that allows us to make conclusions or assumptions that are more wider-reaching compared to what we could assume using the regular metrics.



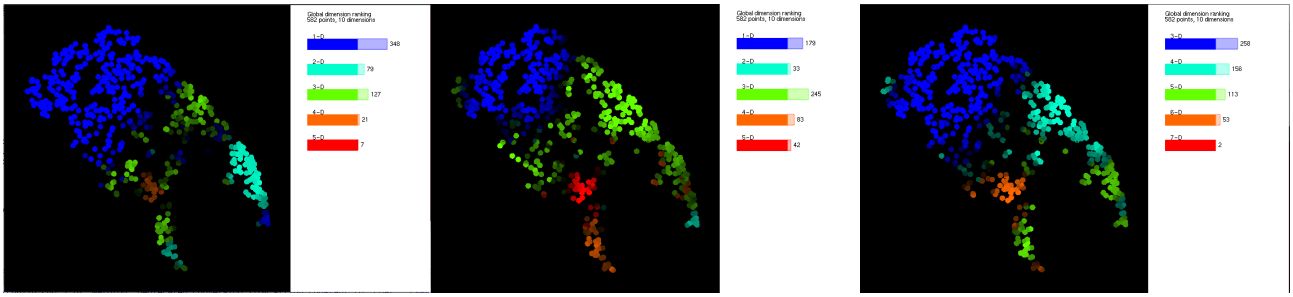
(a) $\theta = 0.5$ (b) $\theta = 0.6$ (c) $\theta = 0.75$

Figure 4.25: data=*indian_liver*, metric=PCA_{sum}, projection=*tsne*, neighborhood=v^n, $\rho = 0.1$



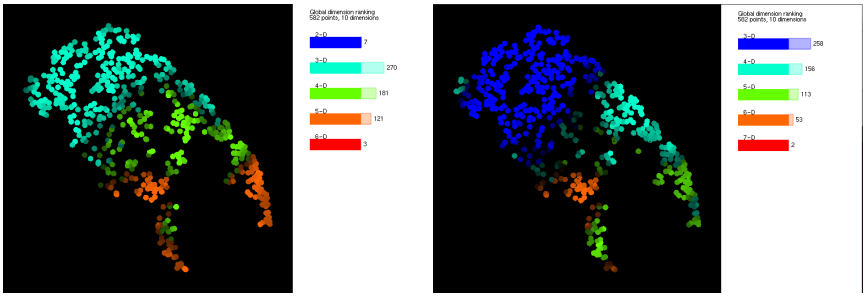
(a) $\eta = 0.3$ (b) $\eta = 0.2$ (c) $\eta = 0.1$

Figure 4.26: data=*indian_liver*, metric=PCA_{min}, projection=*tsne*, neighborhood=v^n, $\rho = 0.1$



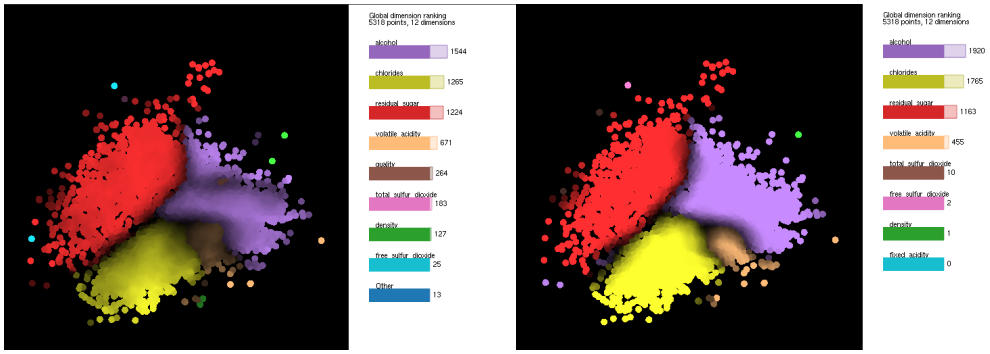
(a) $\nu = 0.6$ (b) $\nu = 0.75$ (c) $\nu = 0.9$

Figure 4.27: data=*indian_liver*, metric=PCA_{ratio}, projection=*tsne*, neighborhood=v^n, $\rho = 0.1$



(a) metric=PCA_{sum}, $\theta = 0.95$ (b) metric=PCA_{ratio}, $\nu = 0.9$

Figure 4.28: data=*indian_liver*, metric=PCA_{sum}, projection=*tsne*, neighborhood=v^n, $\rho = 0.1$



(a) metric=*contribution* (b) metric=*variance*

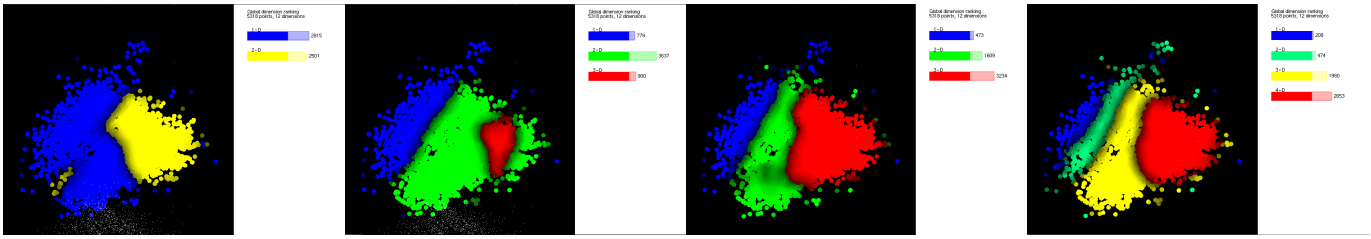
Figure 4.29: data=*winequality*, projection=*lamp*, neighborhood=v^r, $\rho = 0.1$

Wine quality

Next, we examine the *winequality* dataset. This is a dataset consisting of 6497 samples of Portuguese vinho verde wine with 4898 samples of red wine, and 1599 samples of white wine. Each sample contains measurements of 12 physico-chemical such as acidity, residual sugar, and alcohol rate. The dataset is often viewed from the perspective of solving classification or regression tasks.[5]. This dataset was also analyzed in da Silva's et al. paper on analyzing the variance and contribution metrics[3] and was also part of analysis in da Silva's PhD thesis [4]. Fig. 4.29 demonstrates applying the *variance* and *contribution* metrics to this dataset, where for both visualizations we can see that the MP is clearly split into three significant regions dominated by *alcohol*, *chlorides* and *residual sugar*. In Figures 4.30 4.31, 4.32 results are shown for applying all three dimensionality metrics on the data using various thresholding values.

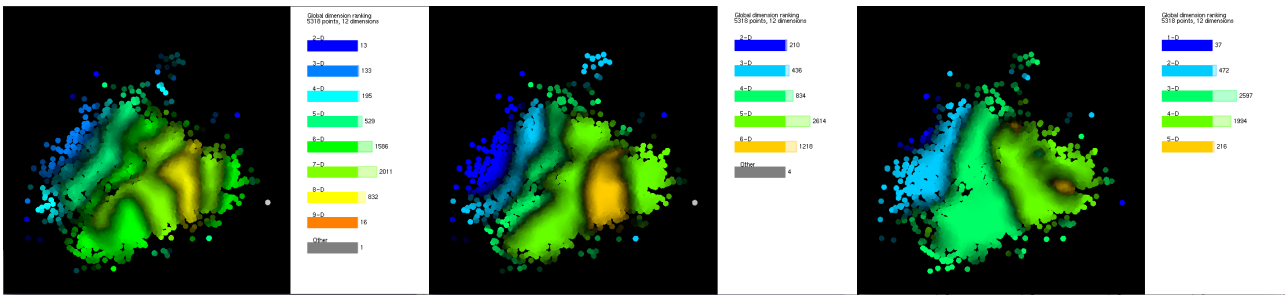
If we start by examining Fig. 4.30, particular 4.30(c) and 4.30(d), we can see that not only do a few dimensions only account for a majority of the variance, but there is also a very clear and smooth transition from lower dimensionality to higher dimensionality regions. In Fig. 4.30(d) we can see that the area roughly attributed to the *alcohol* attribute in Fig. 4.29 is designated as 4-D using a $\theta = 0.75$, but from Fig. 4.31(a), we derive that only two dimensions contribute more than 20% to the variance while the other two contribute only 10%, the rest of the dimensions contribute even less than this. Taken together, these reveal that while the *alchol* is the most powerful predictor for similarity in terms of distance and variance contribution in this region, the dimensionality of the region is quite high; indicating a complex structure for the data in this region. Meanwhile, the region dominated by *residual sugar* is fundamentally of a lower dimensionality. This indicates a fundamentally different structure between points which have most of their similarity explained by *alchol* and those that have their similarity mostly explained by *residual sugar*. Moreover, relative to *alcohol*, *residual sugar* might form a more powerful explanatory variable because of the lower dimensionality in its region. In previous work [3] [4], this region was further analyzed by partitioning it into several additional regions, revealing additional significance of the *winequality* and *volatileacidity* within this region. Because of the significance of the *winequality* subregion, the points in region attributed to *alcohol* in Fig. 4.29 might tend to be from samples of high quality wine. In turn, samples not in this region are not sampled (or to a lesser degree sampled) from high

quality wine. Because this region is very high-dimensional, it could be argued that qualifying or predicting what "high quality" wine constitutes based on the values of the attributes is very difficult. If the lower-dimensional area attributed to *residual sugar* in Fig. 4.29 is in turn associated with less quality wine, then it might however be possible to predict more easily what qualifies as "low quality" wine. In Fig. 4.33 we can investigate the effect of increasing θ even more to see if the smooth transition of regions with decreasing dimensionality is maintained. Indeed this seems to be the case. What this reveals is that for the majority of dimensions, their contribution to the variance in the projection is tucked away in an eastern corner, but gradually decreases as we move to the west of the projection.



(a) $\theta = 0.3$ (b) $\theta = 0.4$ (c) $\theta = 0.5$ (d) $\theta = 0.75$

Figure 4.30: data=*winequality*, metric=PCA_{sum}, projection=*lamp*, neighborhood=v^r, $\rho = 0.1$

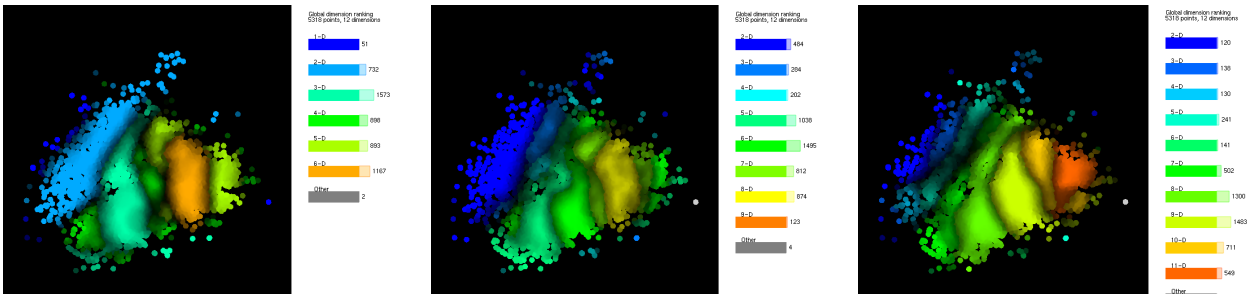


(a) $\eta = 0.05$ (b) $\eta = 0.075$ (c) $\eta = 0.1$

Figure 4.31: data=*winequality*, metric=PCA_{min}, projection=*lamp*, neighborhood=v^r, $\rho = 0.1$

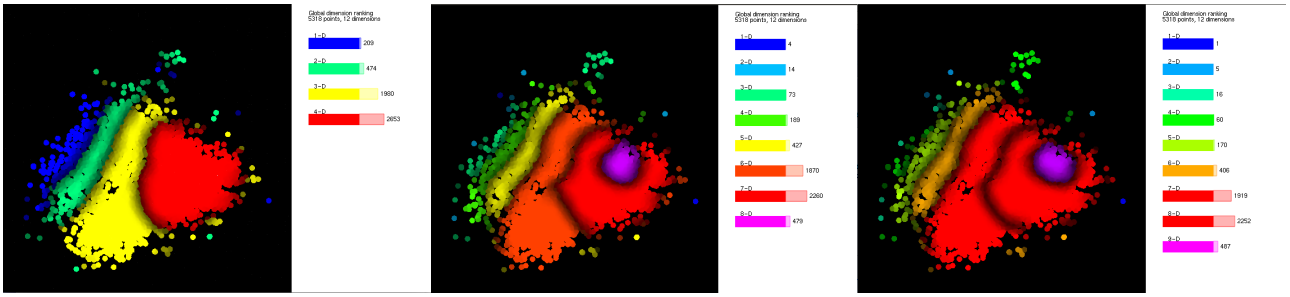
Fig. 4.31 shows the results of visualizing the projection with relatively low values of η using the PCA_{min} metric. The distribution of dimensionality in the three sets Fig. 4.31 roughly follows the same pattern as those in Fig. 4.30. This is to be expected. If 10 dimensions contribute to 90% of the variance, then the chance is high most of those values will be relatively small. With the results of Fig. 4.31, we get a very good indication that there are only a few dimensions which contribute to most of the variance in the lower-dimensional regions. We can conclude that both the PCA_{min} metric and PCA_{sum} metric are capable of detecting the same patterns in terms of dimensionality in the data. However, the division between regions in Fig. 4.31 is much more chaotic compared to the divisions between regions in Fig. 4.30 and Fig. 4.33. This is because the PCA_{min} metric's method

of thresholding is more sensitive differences in the value of the eigenvalues between different regions. Some dimensions might contribute slightly more or less in different regions close to each other, but according to the PCA_{sum} metric, they all could contribute together to a certain percentage of the variance regardless of small differences of their value. Those differences are more easily picked up by the PCA_{min} metric however, as it tests individual values instead of agglomerating them. This inherent sensitivity is not necessarily a bad thing, but it also does not necessarily mean the PCA_{min} metric inherently produces more "accurate" assessments of dimensionality either. Whether or not it performs adequately is entirely up to what the user considers as a good metric to define dimensionality. However, overall the visualizations of the PCA_{sum} metric, at least for this dataset, show visualizations that provide for the best oversight and structure of the distribution of dimensionality in the projection.



(a) $\nu = 0.7$ (b) $\nu = 0.8$ (c) $\nu = 0.9$

Figure 4.32: data=*winequality*, metric=PCA_{ratio}, projection=*lamp*, neighborhood=v^r, $\rho = 0.1$



(a) $\theta = 0.75$ (b) $\theta = 0.85$ (c) $\theta = 0.9$

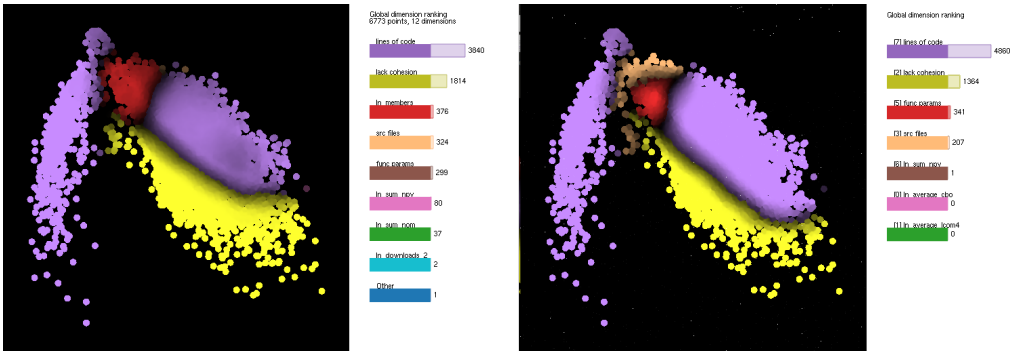
Figure 4.33: data=*winequality*, metric=PCA_{sum}, projection=*lamp*, neighborhood=v^r, $\rho = 0.1$

The results for Fig. 4.32, using the PCA_{ratio} metric, are also interesting. Just like with the PCA_{sum} metric, we see a general transition from low dimensionality regions to higher dimensionality regions. The difference is that with the PCA_{ratio} metric, the subdivision of regions is less generalized, with a more chaotic partitioning of dimensionality, just like with the PCA_{min} metric. And just like with the PCA_{min} metric, the PCA_{ratio} metric is more dependent on small differences between eigenvalues in different regions. Comparing the visualizations in Fig. 4.31 and Fig.

4.32, the PCA_{ratio} indeed appears to be the most sensitive, and perhaps also the most accurate in terms of finding small differences in eigenvalues and dimensionality contribution. So while using the PCA_{sum} metric can give us an overall impression of the data, the PCA_{ratio} metric might prove to be very useful when we start identifying subregions in the projections and would like to investigate their "sub" dimensionality more accurately.

Software Quality

Next, we examine the software quality [6] data set. This is a dataset which describes 6773 software projects from sourceforge.net written in C. Each project has 12 dimensions (11 software quality metrics and the projects total download count). These metrics were extracted using static analysis tools.[6]. Fig. 4.34 gives the visualizations when using the *variance* and *contribution* metrics. Overall, the results are similar to each, with notably a distinction between a western region attributed to *lines of code* and a eastern region attributed to *lines of code*. In Fig. 4.35, we see visualizations for the projection of the data using all three different metrics. The results are interesting, because all three visualizations show a small pocket, with very high dimensionality, that separates the western region indicated by the *lines of code* attribute and the eastern region indicated by the *lines of code* in Fig. 4.34 . In this region, attributes such as the *average count of function parameters*, the *number of members* and the *number of source files* also become more important, as indicated in Fig. 4.34. The consistency of the pocket is confirmed by using different thresholding values with sum thresholding in Fig. 4.36.

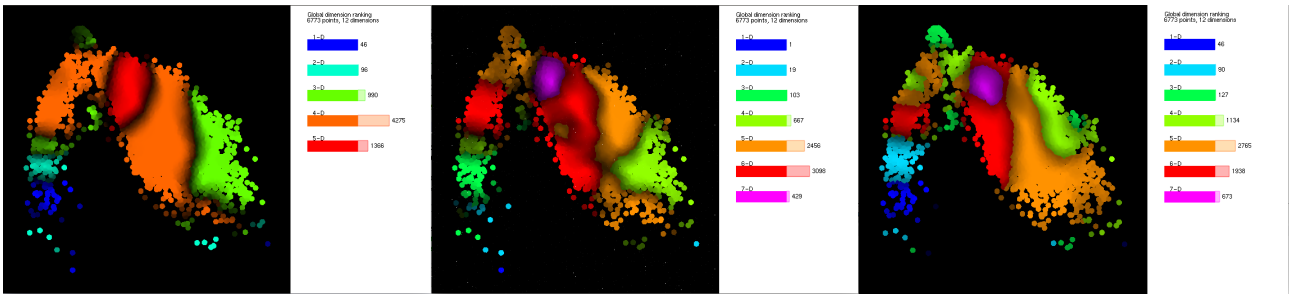


(a) metric=*contribution* (b) metric=*variance*

Figure 4.34: data=$software\_ln$, projection=*lamp*, neighborhood=v^r, $\rho = 0.1$

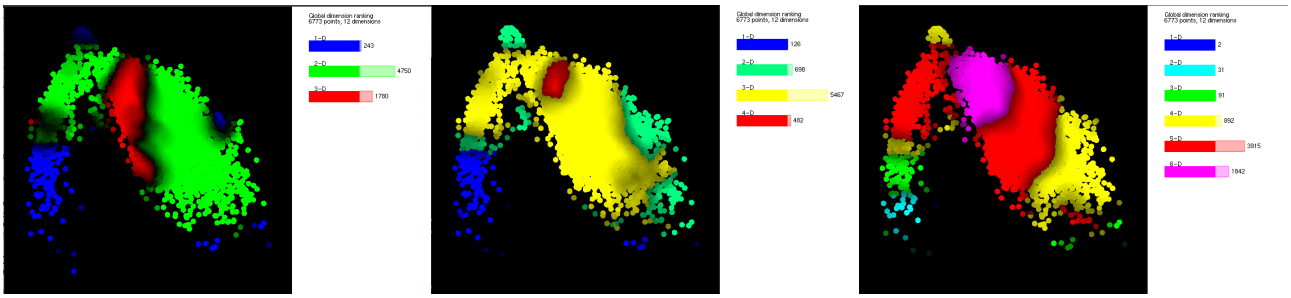
Now we know that the area of high dimensionality in Fig. 4.35, 4.36 roughly overlaps with the region in Fig. 4.34 where attributes such as *average count of function parameters*, the *number of members* and the *number of source files* become more important, and we can assume that the *lines of code* attribute remains relatively important in this region, there is a causal relationship between the increased contribution of the *average count of function parameters*, the *number of members* and the *number of source files* attributes and the increased dimensionality. This in turns allows us to make assumptions about the the extent of the influence of these attributes. The pocket of high-dimensionality has a limit, and is relatively small in comparison to other regions. In turn, because of the assumption of a causal relationship, there is an implication that the influence of

the *average count of function parameters*, the *number of members* and the *number of source files* attributes is also limited. Specifically we can say that the extent of their influence roughly overlaps with the pocket of higher-dimensionality.



(a) metric=PCA_{sum}, $\theta = 0.750.75$ (b) metric=PCA_{min}, $\eta = 0.05$ (c) metric=PCA_{ratio}, $\nu = 0.75$

Figure 4.35: data=$software\_ln$, projection=$lamp$, neighborhood=v^r, $\rho = 0.1$



(a) metric=PCA_{sum}, $\theta = 0.5$ (b) metric=PCA_{sum}, $\theta = 0.6$ (c) metric=PCA_{sum}, $\theta = 0.85$

Figure 4.36: data=$software\_ln$, projection=$lamp$, neighborhood=v^r, $\rho = 0.1$

Being able to easily analyze the extent of influence of attributes is an important task when working with multidimensional projections. Being able to distinguish areas where attributes influence or not can impact our assessment of their characterization in the data. For example, if we assume that leftmost region of the software quality projection would be mostly made up points projected from measurements of smaller projects, while the larger rightmost region is made up points projected from measurements of larger, monolithic projects, then the visualizations in Fig. 4.35 and Fig. 4.36 can also inform us about the extent of the influence of *average count of function parameters*, the *number of members* and the *number of source files* attributes in both regions. Furthermore, a distinction between those regions also means that the analysis of the dimensionality in the projection serves an additional method to note the differences between the inherent structure of those regions.

Chapter 5

Discussions and Conclusions

This final chapter discusses the limitations of the progress made over the course of this project, what future work should focus on to remedy those limitations, but also reflects on what was achieved and to what degree the aim of the project was attained.

5.1 Discussion

- **Advantages & Way of Use**
  Both additional visualization methods allow for a detailed degree of control and customization by the user using the large amount of parameter settings and computational options. The freedom to vary the thresholding value in the dimensionality metric lends itself well to detailed explorations of the data while the freedom to switch thresholding methods is suitable for experimentation with visualizations of the data. The utilization of the heatmap for the dimensionality metric allows for a more intuitive way to represent different levels of dimensionality, while the use of added saturation correlation metric presents a partial solution to finding a suitable visualization method for inverse correlations. As detailed in sections 4.2.2 and 4.3.2, the visualizations are sufficient enough in their explanatory capabilities that they can be used to investigate and infer detailed information from real-world, complex data. Many advantages of the metrics established by da Silva et al. [4] remain, such as being able to use the projection technique as a black box, with the added option for a different neighborhood calculation technique allowing users a workaround for compression issues with some projection techniques. Like with the established metrics from da Silva et al. [4], the partitioning of the projection into regions of dimensionality or regions defined by correlated pairs is automatic and emphasizes ease-of-use.

- **Computational Scalability**
  We can briefly contrast the computational scalability of all different metrics (dimensionality, correlation, distance contribution, variance contribution) for large datasets. Scalability gives us an overview of how efficiently the metrics are able to deal with large input. To test this, we use the large software quality dataset [6] containing 6773 observations 12 attributes. Testing was performed using a Intel Core 2 Duo CPU P8600 @ 2.40GHz processor, with a GeForce 320M/integrated/SSE2 videocard running on 64-bit Ubuntu 14.04. After completing pre-

processing steps (reading and storing data, computing distances between points, generating GUI elements), we test with a radius of 0.1 of the diameter. We measure the time in seconds necessary starting from selecting the metric from the GUI to completing the visualization. The *variance* metric performs best at 28.802s, second comes the *distance* contribution metric at 36.352s. Despite its larger amount of ranks, the correlation metric comes close at 38.203s. The dimensionality metric using sum thresholding comes last, at approximately 1m8s. The correlation metric scales closely compared to the variance and distance metrics, while the longer processing time necessary for the dimensionality metric is the result of the computations necessary to complete the eigenvalue decomposition for many neighborhoods.

- **Limitations and Future Work**
  The correlation metric suffers the same limitations as noted for established techniques[4]. As input datasets increase in datasets increase in dimensionality, the amount of unique colors that cannot be allocated to frequent top-ranked pairs will increase also. However, with the implementation of the matrix picker, the information loss associated with this issue is severely decreased. Likewise, with the dimensionality metric we face a similar issue. As the amount of dimensions increase, the size of the rainbow heatmap will increase also. As a result, the colors in the heatmap will become more and more indistinguishable from each other. However, because of additions to the software tool the user is able to customize the frequency necessary to be allocated a color, thus partially resolving this issue as well. For a long-term solution, we believe increasing the options for users to customize their visualization will prove to be best solution for resolving the limitations of color-mapped labeling of projections. For the correlation metric, this could include additional visualization of a complete pair-wise correlation table indexing the values of all correlations coefficients, or adding the ability to only rank all pair-wise correlations for a single attribute with all other attributes in the visualization. The correlation metric would also benefit by allowing it to be integrated with its own variation on the dimension-set options used with the established metrics of da Silva et. al[4].

  The visualization of the metrics can be enhanced by improving the way in which different properties are highlighted. With the correlation metric, we use added saturation for indicating inverse relationships. This is a serviceable but flawed method, because some colors in the colormap might be less sensitive to additional saturation, leading to little distinction between inverse and linear correlated regions. A core future solution to this problem should focus on resolving the limitations of using colors to illustrate explanatory visualizations, by for example illustrating inverse regions with different shapes or sizes of points. The matrix picker can also be enhanced, by adding visualizations that allow the user to easily observe the significance of the correlations in the matrix. The calculation of the correlation metric can be enhanced by integrating different statistics of correlation, such as Spearman Rank Correlation method and the Kendall rank correlation coefficient.

  As with the similarity metrics established by da Silva et. al[4] the dimensionality and correlation metrics have a limited scope with respect to their ability to service complex classifier or predictor system designs. Additionally, as we saw in sections 4.2.2 and 4.3.2 the information that we are able to infer from the visualizations is currently sometimes restricted to attempting to compare the results found with those found using more complicated models.

  Another key limitation is the sensitivity of the metrics. Each of the metrics is enhanced

by giving it several options for setting the parameters of the metric(neighborhood calculation, thresholding). As we observed in section 4, varying these parameters can produce very different results for the same dataset. The sensitivity means slight changes in the parameter settings can lead to producing inaccurate visualization from which users may derive false conclusions. It is left up to the user to select the right parameters, but as the aim of this thesis was to automate the process of generate explanatory visuals as much as possible, we would like to define models and methods in the future that enable us to reduce the burden put on the user.

With respect to future work for the correlation metric, we could investigate an alternative approach to visualization which uses the example of correlation heatmaps as a basis. An example of an application of such a heatmap is provided in the research of Rao T, Srivastava S[35](Figure 2) in investigating the relationship between microblog discussions and market movements, with the former measured in search volume indices, and the value of various securities index used to quantify the latter. The idea is to represent the range of the Pearson correlation coefficient can take as a color scale from -1 to 1, where "-1" would equal a very "red" color while "1" would equal a very "blue" color. Values in between -1 to 1 would interpolate between these "red" and "blue" colors. Similar to idea of the "matrix picker" introduced in section 3.2, we can then generate an interactive grid matrix where each of the squares is colored by interpolating between red and blue using the value of the Pearson correlation coefficient between the pairs of attributes. Like with the matrix picker, the user can click on a square in the grid. For the pair associated with that square, the projection would be visualized by generating a heatmap of the Pearson correlation coefficient just for that pair over the entire plot, by locally examining the value of the coefficient much like we locally examine the value of dimensionality as with the dimensionality metric. While this approach would lack providing a "global" perspective, it would provide users with a large amount of freedom to investigate dependency relationships between individual pairs of attributes over the entire region of the data.

5.2 Conclusion

This thesis presents new techniques that provide a novel method to automatically generate explanatory visualizations which can enhance 2D scatter-plots of multidimensional projections. The techniques build on previous research, and differentiate themselves by offering different kinds of explanations, while also being able to complement the visual explanations using existing techniques. The first technique focuses on explaining the data by visually indicating where in the data correlations between attributes are present. Additional visualizations, by varying the color compositions, indicate the strength of the correlation, and whether the correlation is a linear or inverse one. With the addition of the matrix picker, user interaction with the software tool is enhanced by allowing users to select which correlations they want to compare and visualize. The testings results indicate that the correlation metric is capable of accurately representing correlations between attributes in the projection in different regions, which allows for a more intuitive representation in comparison to established methods. The second technique focused on explaining data based on their dimensionality. By visualizing the projection with a heat-map, the data is partitioned into areas that are distinguished by estimations of the shape of their variance. This allows for a new way to interpret the data, allowing us to how different regions in the projection are shaped. The technique enhanced

by giving the user different ways to estimate the dimensionality, with additional options for parameter selection. By setting additional parameters for how to generate the heat-map, we offer a workaround of the danger of having an unworkable size for the heat-map. Additional contributions include:

- The metrics we propose are *local* explanations of the data (with a simple way for the user to define locality). Thus allowing us to generate and indicate different explanatory visualizations for different regions in the data.
- The ordinal (heat) colormap for the dimensionality metric and encoding of hue/brightness encoding with an additional color legend simplify the process of *intuitively* developing a *visual understanding* of the data
- *Scalability* for both metrics is achieved both visually and computationally, as they are able to cope with projections having tens or hundreds of thousands of data-points and tens..hundreds of dimensions.
- Our proposed metrics are *generic*, what kind of dataset is used as input is not relevant as long as it is numeric/quantitative.
- The *interaction* with the GUI for changing parameters in the visualizations is very simple and maintains a easy learning curve in terms of interacting with the controls or application.
- The metrics can generate explanatory visualizations *independently* of which projection-technique is used.
- The metrics can be re-used and added to any tool that uses 2D projections, atop of other exploratory mechanisms, showcasing the *adaptability* of the metrics.
- A new technique for neighborhood calculation was implemented, which can compensate for limitations of some projection techniques, thus allowing for more accurate evaluations and visualizations of projections.

We can conclude the research question and both sub-questions have been addressed. By applying our metrics on synthetic data, we were able to indicate patterns and shapes that could not otherwise be visualized using the metrics based on local variance and distance contribution. By applying our metrics to real-world data, comparing the resulting visualizations to those generated local variance and distance contribution metrics, and cross-referencing the results with other research examining the same data, we showed that we can simplify the task of understanding and detecting trends in highly complex scientific datasets in ways the local variance and distance contribution metrics were not able to. Both the dimensionality metric and the correlation metric can be used to generate explanatory visuals which indicate fundamentally different patterns, behavior and trends in the data. Not only are the proposed metrics different from established metrics, they are also fundamentally different from each other. More work remains to be done on researching different ways to generate explanatory visuals; for example developing methods to indicate local dimensionality per pair of correlations over the entire projection, or in developing new metrics which can be used to generate fundamentally different explanatory visualizations from those established in this thesis.

Bibliography

[1] Peral J, Mate A, Marco M. *Application of Data Mining techniques to identify relevant Key Performance Indicators*. COMPUTER STANDARDS & INTERFACES, Volume: 54, Pages: 76-85, Part: 2, Special Issue: SI, 2017

[2] Zhang L, Zhang D, Sun M M, Chen F M. *Facial beauty analysis based on geometric feature: Toward attractiveness assessment application*. EXPERT SYSTEMS WITH APPLICATIONS, Volume:82, Pages: 252-265, 2017

[3] da Silva R O, Rauber P E, Martins R m, Minghim R, Telea A C. *Attribute-based Visual Explanation of Multidimensional Projections*. Proc. EuroVA, 2015.

[4] da Silva R O. *Visualizing multidimensional data similarities: Improvements and applications*, [Groningen]: University of Groningen (2016)

[5] Cortez P, Cerdeira A, Almeida F, Matos T and Reis J. *Modeling wine preferences by data mining from physicochemical properties*. In Decision Support Systems, Elsevier, 47(4): Pages 547-553, 2009.

[6] Meirelles P, Santos C, Miranda J, Kon F, Terceiro A, and Chavez C. *A study of the relationships between source code metrics and attractiveness in free software projects*. Proceedings of the 2010 Brazilian Symposium on Software Engineering (SBES), Pages 1120, Sept 2010. doi: 10.1109/SBES.2010.27.

[7] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, and Duchesnay E. *Scikit-learn: Machine Learning in Python*. The Journal of Machine Learning Research, Volume 12, 2/1/2011 Pages 2825-2830 URL http://scikit-learn.org/stable/documentation.html

[8] Albuquerque, G., Löwe, T., & Magnor, M. *Synthetic generation of high-dimensional datasets*. IEEE Transactions on Visualization and Computer Graphics,Volume: 17, Issue: 12, Pages 2317 - 2324, Dec. 2011

[9] Shlens J. *A Tutorial on Principal Component Analysis*. Systems Neurobiology Laboratory, Salk Insitute for Biological Studies La Jolla, CA 92037 and Institute for Nonlinear Science, University of California, San Diego La Jolla, CA 92093-0402

[10] Jolliffe I T. *Principal Component Analysis, 2nd Edition*. Springer-Verlag New York 2002, ISBN 978-0-387-95442-4, 978-1-4419-2999-0

[11] Collins M, Dasgupta S, Schapire R E. *A Generalization of Principal Components Analysis to the Exponential Family*. Conference: Advances in Neural Information Processing Systems 14 [Neural Information Processing Systems: Natural and Synthetic, NIPS 2001, December 3-8, 2001, Vancouver, British Columbia, Canada]

[12] Üzömcü M, Frangi A F, Sonka M, Reiber J H C, Lelieveldt B P F. *ICA vs. PCA Active Appearance Models: Application to Cardiac MR Segmentation*. Medical Image Computing and Computer-Assisted Intervention – MICCAI 2003, 6th International Conference, Montréal, Canada, Nov 2003, Proceedings Part I.

[13] van der Maaten L, Hinton G. *Visualizing Data using t-SNE*. Journal of Machine Learning Research 9, Pages 2579-2605 (2008)

[14] Joia P, Paulovich FV, Coimbra D, Cuminato JA, Nonato LG. *Local Affine Multidimensional Projection.*. IEEE Transactions on Visualization and Computer Graphics, Volume 17 Issue 12, Pages 2563-2571, December 2011

[15] Dick S, Kendal A. *Computational Intelligence in Software Quality Assurance*. Series in Machine Perception and Artificial Intelligence: Volume 63, ISBN: 978-981-256-172-5

[16] http://eigen.tuxfamily.org/index.php?title=Main_Page

[17] Politi T, Popolizio M. (2006) *Schur Decomposition Methods for the Computation of Rational Matrix Functions*. In: Alexandrov V.N., van Albada G.D., Sloot P.M.A., Dongarra J. (eds) Computational Science ICCS 2006. ICCS 2006. Lecture Notes in Computer Science, vol 3994. Springer, Berlin, Heidelberg

[18] O'Donnell L J, Westin C F. *An introduction to diffusion tensor image analysis*. Neurosurgery Clinics of North America 2011;22:185.

[19] Altman D G. *Practical Statistics for Medical Research* (2006). Chapman & Hall/CRC, ISBN:1584880392

[20] Swinscow T D V. *In: Statistics at square one*, 9th Edition. Campbell M J, editor. University of Southampton; Copyright BMJ Publishing Group 1997.

[21] Zhongheng Z, *Introduction to machine learning: k-nearest neighbors*, Annals of Translational Medicine 4.11 (2016): 218. PMC. Web. 9 July 2017.

[22] U. of Maryland. US counties dataset, 2014. URL http://archive.ics. uci.edu/ml

[23] Matins R, Coimbra D, Minghim R, Telea A C. *Visual analysis of dimensionality reduction quality for paramaterized projections*. Computers & Graphics 41 (2014), Pages 2642.

[24] https://www.cs.umd.edu/ mount/ANN/

[25] I-Cheng Yeh. *Modeling of strength of high performance concrete using artificial neural networks*. Cement and Concrete Research, Vol. 28, No. 12, Pages: 1797-1808 (1998).

[26] Cliff N. *The Eigenvalues-Greater-Than-One Rule and the Reliability of Components*. Psychological Bulletin, Vol. 103(2), Mar 1988, Pages: 276-279.

[27] Besse P, Falguerolles A,, *Application of resampling methods to the choice of dimension in principal component analysis*, Computer Intensive Methods in Statistics, eds. W.Härdle and L. Simar, Pages: 167176. Heidelberg: Physica-Verlag

[28] North G R, Bell T L, Cahalan R F, Moeng F J, (1982). *Sampling errors in the estimation of empirical orthogonal functions*, Mon. Weather Rev., 110, Pages: 699706.

[29] http://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_circles.html

[30] Lichman M (2013). *UCI Machine Learning Repository [http://archive.ics.uci.edu/ml].* Irvine, CA: University of California, School of Information and Computer Science.

[31] Ramana B V, Babu P, Venkateswarlu N B. *Critical Comparative Study of Liver Patients from USA and INDIA: An Exploratory Analysis.* International Journal of Computer Science Issues, ISSN :1694-0784, May 2012.

[32] Ramana B V, Babu P, Venkateswarlu N B. *Critical Study of Selected Classification Algorithms for Liver Disease Diagnosis.* International Journal of Database Management Systems (IJDMS), Vol.3, No.2, ISSN : 0975-5705, Pages: 101-114, May 2011.

[33] Wu S S, Li B Z, Yang J G, Shukla S K. *Predictive Modeling of High-Performance Concrete with Regression Analysis* . 2010 IEEE International Conference on Industrial Engineering and Engineering Management, IEEE, 7-10, Dec. 2010

[34] Beh E J, Holdsworth C I. *A visual evaluation of a classification method for investigating the psysicochemical properties of Portugese Wine.* CURRENT ANALYTICAL CHEMISTRY, Volume: 8, Issue: 2, Pages: 205-217, Apr 2012

[35] Rao T, Srivastava S. *Modeling Movements in Oil, Gold, Forex and Market Indices using Search Volume Index and Twitter Sentiments.* 2010 IEEE International Conference on Industrial Engineering and Engineering Management, IEEE, 7-10 Dec. 2010

Appendix A

Guide to the software tool

To build the software, several dependencies need to be installed, and proper guidelines need to be followed to correctly utilize the program. This appendix serves as a comprehensive guide to the user for these purpose.

A.1 CUDA

To use the software, you need CUDA, a parallel computing platform and programming model invented by NVIDIA.

- First install NVIDIA video driver via "Softwares and Updates" tool, tab "Additional Drivers".
- Install additonal packages: `sudo apt-get install freeglut3-dev build-essential libx11-dev libxmu-dev libxi-dev libgl1-mesa-glx libglu1-mesa libglu1-mesa-dev`
- Download CUDA installation run file, and extract its parts: `./cuda_6.5.14_linux_64.run -extract=dir_to_extract`
- Install CUDA toolkit: `sudo ./cuda-linux64-rel-6.5.14-18749181.run` and create a symbolic link pointing to `/usr/local/cuda`
- Create the file `/etc/profile.d/cuda.sh` to environment variables:
  - `export PATH=/usr/local/cuda/bin:$PATH`
  - `export LD_LIBRARY_PATH=/usr/local/cuda/lib64:$LD_LIBRARY_PATH`
- Execute ldconfig: `sudo ldconfig` * Install Cuda Samples (included in the downloaded .run file)
  - `sudo ./cuda-samples-linux-6.5.14-18745345.run`
  - execute `cuda-install-samples-6.5.sh /home/user/`
- **Note:** There is a bug on Ubuntu 14.04 running optimus systems that does not allow execution of CUDA programs.

- **Workaround:** The user has to execute a sample as sudo first:
  `sudo NVIDIA_CUDA-6.5_Samples/1_Utilities/deviceQuery`
- Add this line in the end of `.bashrc`: * `export LD_PRELOAD=/usr/lib/x86_64-linux-gnu/mesa/libGL.so.1`

A.2 Dependencies

CUBu (`https://bitbucket.org/rmmartins/cubu`)
* All platforms: download the sources and follow the instructions on `README.md` to build and install.
* Open `CMakeLists.txt` and point the CUBU variable to the base folder.

Triangle (`http://www.cs.cmu.edu/ quake/triangle.html`) – Last tested version: 1.6 (10/08/14)
All platforms: Download the sources on the site and extract to directory "lib" under the project root but do not build it. We'll build it together with our own code.
Open `CMakeLists.txt` and point the TRIANGLE variable to the base folder.
Ubuntu: Use your package system (ex.: `libtriangle-dev`)

ANN (`http://www.cs.umd.edu/ mount/ANN/`) – Last tested version: 1.1.2 (10/08/14)
* Win32: There are pre-compiled binaries on the site.
* Linux: You can probably find it in your package system (ex.: `libann-dev`). If not, download the sources and use `make linux-g++` to build.

FreeGLUT (`http://freeglut.sourceforge.net/`) – Last tested version: 2.8.1 (10/08/2014)
* Win32: There are pre-compiled binaries here(`http://www.transmissionzero.co.uk/software/freeglut-devel/`).
* Linux: just get it from your packaging system (ex.: `freeglut3-dev`).

GLUI (`http://glui.sourceforge.net/`) – Last tested version: 2.35 (10/08/2014)
* Win32: Download the sources and build the (ancient) solution found in `src/msvc`.
* It depends on (Free)GLUT so add `$(INCLUDE)` to the project's include dirs.
* Change the Target Name (on Properties/General) to `glui32` and the Output File (under `Properties/Librarian`) to "`Debug/glui32.lib`. * This will not generate a DLL. Install `glui32.lib` and `GL/glui.h` as defined previously.
* Linux: just get it from your packaging system (ex.: `libglui-dev`). On Ubuntu it has been removed from recent repos, but you can find it [here](`http://packages.ubuntu.com/raring/libglui-dev`), for example.
* After building the static lib, move the project directory to the "lib" folder under the project root.

FreeImage (`http://freeimage.sourceforge.net/`) – Last tested version: 3.16.0 (10/08/2014)
* Win32: There are pre-compiled binaries on the site.
* Linux: just get it from your packaging system (ex.: `libfreeimage-dev`).

GLEW (`http://glew.sourceforge.net/`) – Last tested version: 1.10.0 (10/08/14)
* Win32: There are pre-compiled binaries on the site. Install the "glew32" (not "s") lib (together with the DLL).
* Linux: just get it from your packaging system (ex.: `libglew-dev`)

Boost (`http://www.boost.org/`) – Last tested version 1.59.0 (22/08/2015)
* Unix: just get it from the packaging system (`libboost-all-dev`)

FTGL (`http://sourceforge.net/projects/ftgl/`) – Last tested version 2.1.3-rc5 (22/08/2015)
* Linux: just get it from the packaging system (`libftgl-dev`)

FreeType 2 (`http://packages.ubuntu.com/precise/libfreetype6-dev`) – Last tested version `libfreetype6-dev 2.5.2-1ubuntu2.5` (22/08/2015)
* Linux: just get it from the packaging system (libfreetype6-dev)

Eigen 3 (`http://packages.ubuntu.com/precise/libeigen3-dev`) – Last tested version `libeigen3-dev 3.2.0-8` (07/05/2016) * Linux: just get it from the packaging system (`libeigen3-dev`)

A.3 Building

1. The preferred way is to build with CMAKE. For example:
   * Win32: run `CMAKE-GUI`, point to the source folder, create a new "`build_win32` folder for the binaries, configure and generate. This will create a VS solution (`ProjectionExplain.sln`) that you can (hopefully) build without problem. Building the INSTALL project will put the executable in the right place (root of the solution).
   * Unix: `mkdir -p build && cd build && cmake ..  && make install`

2. If you do not want to or cannot use CMAKE, try `makefile.old` at your own risk. A simple `make` should work. If it does not, check `Makefile.old` to see if everything is correct. Some notes:
   * ANN code is expected to be in "`../lib/ann_1.1.2` (or change the Makefile).
   * Triangle code is expected to be in "`../lib/triangle` (or change the Makefile)

3. If you want to use Netbeans, start a new Netbeans C++ project from existing code and point it to the projection-explainer root directory (where the `CMakeLists.txt` is). Then Netbeans will create some new files like nbproject and CMakeFiles.
   * Whether you need to update anything, like including new files or moving them, update the `CMakeLists.txt` and remove `CMakeCache.txt`. Then rebuild the project.

A.4 Running the program and data format

You need to run the program as a command line command. Example:

```
./projwiz -e -f data/static/segmentation lamp -d
```

Here:

- `projwiz` is the name of the executable
- `-e` tells the program to use the functionality implemented for this project. This includes all functionality of the new metric and visualization methods. Use `-n` to use the functionality implemented by da Silva such as point triangulation, dimension set explanations and label placements.
- `-f` tells the program to expect an input file argument. `data/static/segmentation` is the directory location of the file.
- `lamp` is the name of the projection technique used to reduce the data.

In this example, the name of the input data is `segmentation`. You will need one file that contains the nD data, which in this case would be called `segmentation.data`. You will also need one file containing the projected data. In this case, because we use LAMP, it would be called `segmentation.lamp.2d`. Generally, the format for the files is:

1. dataname.data
2. dataname.projectiontechnique.2d

Examples of projection techniques are “lamp”, “pca” “tsne” or “lsp. You can name the “projectiontechnique” anything you like. The number of measurements in both files needs to be the same.