Understanding Principal Component Analysis Using a Visual Analytics Tool

Dong Hyun Jeong, Caroline Ziemkiewicz, William Ribarsky and Remco Chang
Charlotte Visualization Center, UNC Charlotte
{dhjeong, caziemki, ribarsky, rchang}@uncc.edu

Abstract

Principle Component Analysis (PCA) is a mathematical procedure widely used in exploratory data analysis, signal processing, etc. However, it is often considered a black box operation whose results and procedures are difficult to understand. The goal of this paper is to provide a detailed explanation of PCA based on a designed visual analytics tool that visualizes the results of principal component analysis and supports a rich set of interactions to assist the user in better understanding and utilizing PCA. The paper begins by describing the relationship between PCA and single vector decomposition (SVD), the method used in our visual analytics tool. Then a detailed explanation of the interactive visual analytics tool, including advantages and limitations, is provided.

1. Introduction

PCA is a widely used mathematical tool for high dimension data analysis. Just within the fields of computer graphics and visualization alone, PCA has been used for face recognition [19], motion analysis and synthesis [16], clustering [11], dimension reduction [9], etc. PCA provides a guideline for how to reduce a complex dataset to one of lower dimensionality to reveal any hidden, simplified structures that may underlie it.

Although PCA is a powerful tool capable of reducing dimensions and revealing relationships among data items, it has been traditionally viewed as a "black box" approach that is difficult to grasp for many of its users [10]. The coordinate transformation from original data space into eigenspace makes it difficult for the user to interpret the underlying relation. This paper focuses on providing a detailed explanation of PCA based on our designed visual analytics tool, iPCA, which helps to give the user an intuitive feel for the details of this black box operation. Specifically, we describe how the visual analytics tool has been developed and how the math behind PCA is covered in the tool. In addition, we will explain how PCA is intimately related to the mathematical technique of singular value decomposition (SVD). This understanding will lead us to a prescription for how to apply PCA in visual analytics. We will discuss the assumptions behind the visual analytics tool as well as possible extensions.

First we explain basic principals of PCA and SVD computations. Detailed explanations of how iPCA is designed and what functions it provides are given in Sections 4 and 5, respectively. Then, we conclude this paper and discuss its implications in Section 6.

2. Principal Component Analysis

PCA is a method that projects a dataset to a new coordinate system by determining the eigenvectors and eigenvalues of a matrix. It involves a calculation of a covariance matrix of a dataset to minimize the redundancy and maximize the variance. Mathematically, PCA is defined as a orthogonal linear transformation and assumes all basis vectors are an orthonomal matrix [10]. PCA is concerned with finding the variances and coefficients of a dataset by finding the eigenvalues and eigenvectors.

The PCA is computed by determining the eigenvectors and eigenvalues of the covariance matrix. The covariance matrix is used to measure how much the dimensions vary from the mean with respect to each other. The covariance of two random variables (dimensions) is their tendency to vary together as:

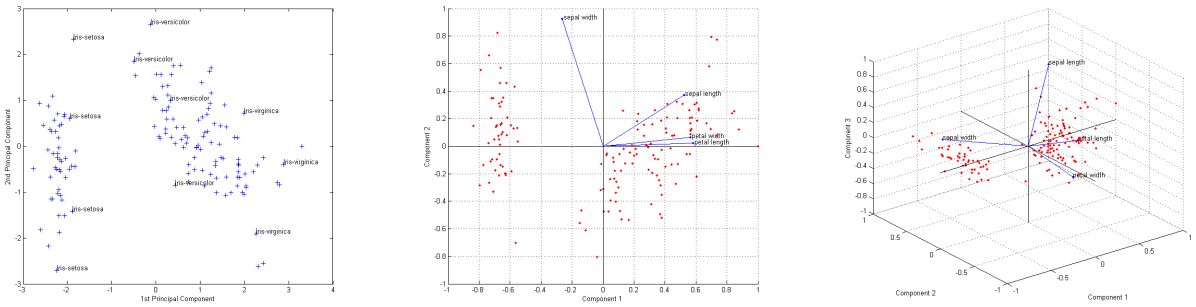
$$cov(X,Y) = E[E[X] - X] \cdot E[E[Y] - Y] \tag{1}$$

where $E[X]$ and $E[Y]$ denote the expected value of X and Y respectively. For a sampled dataset, this can be explicitly written out.

$$cov(X,Y) = \sum_{i=1}^{N} \frac{(x_i - \bar{x})(y_i - \bar{y})}{N} \tag{2}$$

with $\bar{x} = mean(X)$ and $\bar{y} = mean(Y)$, where N is the dimension of the dataset. The covariance matrix is a matrix A with elements $A_{i,j} = cov(i,j)$. It centers the data by subtracting the mean of each sample vector.

In the covariance matrix, the exact value is not as



(a) (b) (c)

Figure 1: Examples of PCA with public Iris dataset [2]. A generic representation of data with two principal components (a), and 2D (b) and 3D (c) representations with principal components and the coefficients for each variable. The red dots in (b) and (c) indicate individual data items and blue lines represent the coefficients.

important as its sign (i.e. positive or negative). If the value is positive, it indicates that both dimensions increase, meaning that as the value of dimension X increased, so did the dimension Y. If the value is negative, then as one dimension increases, the other decreases. In this case, the dimensions end up with opposite values. In the final case, where the covariance is zero, the two dimensions are independent of each other. Because of the commutative attribute, the covariance of X and Y $(cov(X, Y))$is equal to the covariance of Y and X $(cov(Y, X))$.

With the covariance matrix, the eigenvectors and eigenvalues are calculated. The eigenvectors are unit eigenvectors (lengths are 1). Once the eigenvectors and the eigenvalues are calculated, the eigenvalues are sorted in descending order. This gives us the components in order of significance. The eigenvector with the highest eigenvalue is the most dominant principle component of the dataset (PC_1). It expresses the most significant relationship between the data dimensions. Therefore, principal components are calculated by multiplying each row of the eigenvectors with the sorted eigenvalues.

As mentioned above, PCA is used as a dimension reduction method by finding the principal components of input data. But to map a high-dimensional dataset to lower dimensional space, the best low-dimensional space has to be determined by the eigenvectors of the covariance matrix. The best low-dimensional space is defined as having the minimal error between the input dataset and the PCA by using the following criterion:

$$\frac{\sum_{i=1}^{K} \lambda_i}{\sum_{i=1}^{N} \lambda_i} > \theta \qquad \text{(e.g., } \theta \text{ is 0.9 or 0.95)} \qquad (3)$$

where K is the selected dimension from the original matrix dimension N, θ is a threshold, and λ is an eigenvalue. Based on this criterion, the $N \times N$ matrix is linearly transformed to an $N \times K$ matrix. Even though the number of dimensions is decreased through the PCA calculation, the difference between the input and the output matrix is minor. A K value of 2 or 3 is often used to map the dataset into a 2D or 3D coordinate system. Figure 1(a) shows a projection with principal components (PC_1 and PC_2). And Figures 1(b) and 1(c) show both the coefficients for each variable and the principal components in different dimensional spaces. Each variable is represented as a vector, and the direction and the length of the vector indicates how each variable contributes to the principal components. For instance, Figure 1(b) shows that the first principal component has positive coefficients with three variables (sepal length, petal width, and petal length) and a negative coefficient with sepal width. However, the second principal component has positive coefficients with all variables. Figure 1(c) shows that the third principal component has positive coefficient only with the sepal length. If the Iris data are plotted based on the first two principal components (Figure 1(b)), there is about 60% confidence ($\theta = 0.61$). Using the first three principal components (Figure 1(c)), there is about 94% confidence ($\theta = 0.94$), indicating that the error between the original dataset and the projected dataset is less than 6%.

A common method for finding eigenvectors and eigenvalues in non-square matrices is singular value decomposition [14] (SVD). In our implementation of PCA, we use an approximation method based on SVD called Online SVD to maintain real-time interaction when interacting with large scale datasets [4, 5]. In Section 3, we describe SVD and how PCA is intimately related to SVD. A detailed explanation of Online SVD and how it is applied to our visual analytics tool is included in Section 4.3.

3. Singular Value Decomposition

SVD is a PCA-like approach which is widely used in face recognition [15], microarray analysis[20], etc. SVD decomposes a matrix into a set of rotation and scale matrices, which is used in computing the pseudoinverse, matrix ap-

proximation, and determining the rank[1], range and null space of a matrix. There are four types of SVD: full-SVD, thin-SVD, compact-SVD, and truncated SVD. Full-SVD is a full unitary decomposition of the null-space of the matrix. The other three types of SVD are revised approaches that minimize computation time and storage space. Although most applications adopt one of these SVDs (thin-SVD, compact-SVD, or truncated SVD), we use the full-SVD throughout the paper when explaining mathematical equations since it is easy to understand and uses straightforward matrix calculations. The general form of computing SVD is:

$$A = USV^T \tag{4}$$

where $A \in R^{m \times n}$ (with $m \geq n$), $U \in R^{m \times m}$, $V \in R^{n \times n}$, and S is a diagonal matrix of size $R^{m \times n}$. U and V are called the *left singular vector* and the *right singular vector* respectively, and are both orthogonal. The elements of S are only nonzero on the diagonal and are called the *singular values*. The ordering of the singular vectors is determined by high-to-low sorting of singular values, with the highest singular value in the upper left index of the matrix S. Because of its characteristics, if the matrix M is unitary ($M \in R^{m \times m}$), it can be represented by a diagonal matrix Λ and a unitary matrix Γ ($\Gamma \in R^{m \times m}$) by the formula:

$$M = \Gamma \Lambda \Gamma^T \tag{5}$$

where

$$\begin{array}{l} \Lambda = diag(\lambda_1, \lambda_2, \ldots) \\ \Gamma^T \Gamma = \Gamma \Gamma^T = I \end{array} \tag{6}$$

SVD constructs orthonormal bases[2] of a matrix. The columns of U, whose same-numbered elements λ_j are nonzero, are an orthonormal set of basis vectors that span the range; the columns of V, whose same-numbered elements λ_j are zeros, are orthonomal bases for the nullspace. The left and right singular vectors are constructed as orthonomal bases.

$$A = \begin{pmatrix} \cdots & u_{1j} & \cdots \\ \cdots & \cdot & \cdots \\ \cdots & u_{mj} & \cdots \end{pmatrix} \begin{pmatrix} \cdots & \cdot & \cdots \\ \cdots & \lambda_j & \cdots \\ \cdots & \cdot & \cdots \end{pmatrix} \begin{pmatrix} \cdot & \cdots & \cdot \\ v_{j1} & \cdots & v_{jn} \\ \cdot & \cdots & \cdot \end{pmatrix} \tag{7}$$

where $U = (u_{11}, u_{12}, ..., u_{mm})$, $S = (\lambda_1, \lambda_2, ..., \lambda_n)$, and $V = (v_{11}, v_{12}, ..., v_{nn})$.

In SVD calculation, the left and right singular vectors are determined where the following two relations hold:

$$\begin{array}{l} A^T A = VS^T U^T USV^T = V(S^T S)V^T \\ AA^T = USV^T VS^T U = U(SS^T)U^T \end{array} \tag{8}$$

where A^T is the conjugate transpose of A. The right hand sides of these relations describe the eigenvalue decompositions of the left hand sides. Consequently, the squares of the non-zero singular values of A are equal to the non-zero eigenvalues of either $A^T A$ or AA^T. When calculating the left and right singular vectors, QR-decomposition is often used when the matrix has more rows than columns ($m > n$) [1]. The QR-decomposition factors a complex $m \times n$ matrix (with $m \geq n$) as the product of a $m \times m$ unitary matrix Q and an $m \times n$ upper triangular matrix R as

$$A = QR = QQ^T A \tag{9}$$

where Q is an orthogonal matrix (meaning that $Q^T Q = I$) and R is an upper triangular matrix. The factor R is then reduced to a bidiagonal matrix.

SVD is closely related to PCA in terms of computing eigenvalues and eigenvectors. From the SVD calculation, the PCA output is calculated by the outer product of the columns of U with the singular values $diag(S)$.

In order to better understand PCA calculation, we designed a visual analytics tool (iPCA). Section 4 provides a detailed explanation of iPCA.

4. A Visual Analytics Tool (iPCA)

The design of iPCA is based on multiple coordinated views. Each of the four views in the tool represents a specific aspect of the input data either in data space or eigenspace, and they are coordinated in such a way that any interaction with one view is immediately reflected in all the other views (*brushing & linking*). This coordination allows the user to infer the relationships between the two coordinate spaces.

4.1 Interface Design

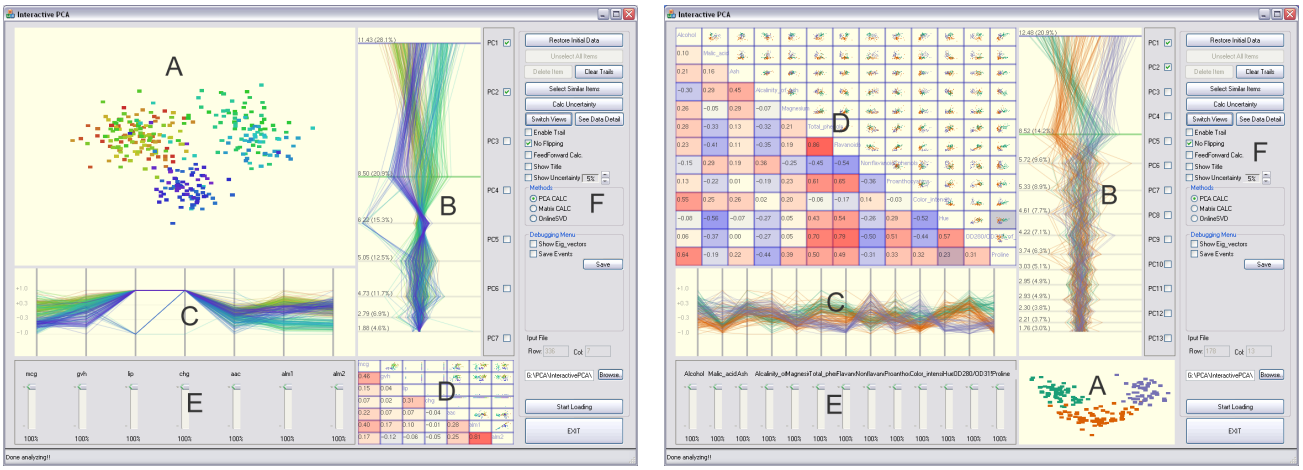
iPCA is designed with four distinct views and two control panels. Figure 2 shows the system overview with sample datasets.

Projection View (Figure 2A): In the Projection View, selected principal components are projected in 2D coordinate system. By default, the first two dominant principal components are used to project data items onto a two-dimensional coordinate system. In the coordinate system, the x-axis represents the first principal component and the y-axis represents the second principal component. However, the mapping between the principal components and the coordinate system can be changed by the user in the control panel.

Eigenvector View (Figure 2B): The calculated eigenvectors and their eigenvalues are displayed in a vertically projected parallel coordinates visualization in the Eigenvector View, with eigenvectors ranked from top to bottom by dominance. The distances between eigenvectors in the parallel coordinate view vary based on their eigenvalues, separating the eigenvectors based on their mathematical weights.

[1]The ranks of a matrix is the number of linearly independent rows or columns.

[2]An orthogonal set of vectors $V = v_1, v_2, ..., v_k$ is called orthonormal if all vectors are unit vectors (magnitude 1) and the inner product $v_i \cdot v_j = 0$ when $i \neq j$.



(a) (b)

Figure 2: Two examples of the system with the E.Coli (a) and Wine (b) dataset. (A) Projection view. Data items are projected onto the two user-selected eigenvectors (in this case, the primary and secondary principle components). (B) Eigenvector view. Each eigenvector is treated as a dimension in this parallel coordinates view, and every data item is drawn as a line. (C) Data view. Another parallel coordinates view, but this time each dimension represents the dimensions in the original data, and each line represents each data item. (D) Correlation view. Pearson-correlation coefficients and relationships (scatter plot) between each pair of variables are represented. (E) Dimension sliders. Each slider controls the amount of contribution of a dimension in the PCA calculation. (F) Control options. The projection view and the correlation view are switchable as shown.

Data View (Figure 2C): The Data View shows all data points in a parallel coordinates visualization. In this view, an auto-scaling function is applied to increase the readibility of data.

Correlation View (Figure 2D): The Correlation View contains three regions: the diagonal, the bottom triangle, and the top triangle. The diagonal displays the name of the dimension as a text string. The bottom triangle shows Pearson-correlation coefficients between two dimensions with a color indicating positive (red), neutral (white), and negative (blue) correlations. The top triangle contains cells of scatter plots in which all data items are projected onto the two intersecting dimensions. A selected scatter plot is enlarged to show the detail. The colors of the data items are the same as the colors used in the other three views so that clusters are easily identified.

It is important to note that the selection operation in all views and the zooming-in mechanism in the Projection and Correlation views help users to focus their interest on a data item or items. Also, the Projection View and the Correlation View can be switched (see Figure 2), which allows the user to utilize the visual real estate for focusing either on a single projection of data or to examine in detail all (or one) scatter plot(s) in the Correlation View. Initially, the color of each data item is allocated depending on the class to which the item belongs. If the data items are the same class, the same color is used.

Control Panels (Figure 2E and 2F): The two control panels include a set of dimension sliders (Figure 2E) that can be used to decrease or increase the contributions of each of the original data dimensions, whose purpose will be discussed further in the following section (Section 5). Several additional modes can also be specified in the other control panel to enhance understanding of the visual changes during data analysis (Figure 2F). For instance, the user can enable *trails* so that the path of each data item's recent motion is painted to the screen, making the movement of each point during interaction operations more apparent.

4.2 Resolving the Sign Ambiguity

There is an inherent ambiguity in the signs of values resulting from the SVD calculation (Figure 3) that is not mathematically solvable [8, 15]. While Bro et al. [15] proposed a solution that determines the signs based on the sign of the sum of the signed inner products, there is still a limitation. For example, as the magnitude of the inner products approaches to zero, the signs remain ambiguous and undetermined. Because of this, the method that Bro et al. proposed does not produce consistent results and can destroy frame-to-frame coherence. Instead we designed a simple method that maintains the frame-to-frame coherence at all times. It predicts the sign based on the previously calculated singular vector as:

$$\vec{U}_k^c = \vec{U}_k \times sign,$$
$$sign = \begin{cases} -1 & \text{if } sum(\vec{U}_{k-1} \times \vec{U}_k) < 0, \\ 1 & \text{otherwise} \end{cases} \quad (10)$$

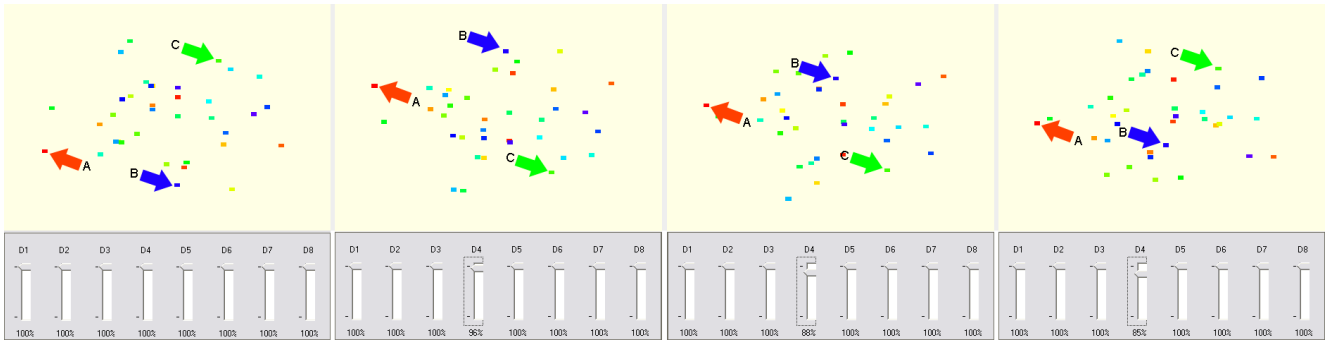


Figure 3: Sign-flipping caused in the random dataset. All the projected points are flipped over along the horizontal axis.

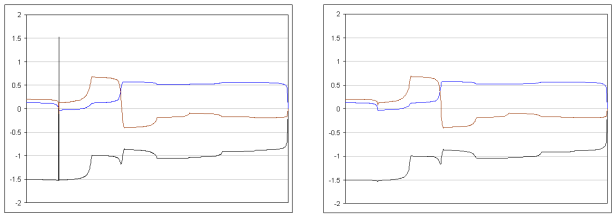
where $\vec{U}_k$ is the left singular vector (before the sign is corrected), $\vec{U}_{k-1}$ is the vector in previous time frame, and $\vec{U}_k^c$ is the vector with the sign corrected. From the sign corrected left singular vector, the right singular vector V_k^c can easily be calculated as:

$$\begin{aligned} A &= U_k^c S V_k^{cT} \\ U_k^{cT} A &= U_k^{cT} U_k^c S V_k^{cT} \\ U_k^{cT} A &= S V_k^{cT} \qquad (U_k^c \times U_k^{cT} = I) \\ S^{-1} U_k^{cT} A &= S^{-1} S V_k^{cT} \\ S^{-1} U_k^{cT} A &= V_k^{cT} \qquad (S^{-1} \times S = I) \end{aligned} \tag{11}$$

where I is the identity matrix and S^{-1} represents the pseudo-inverse of the matrix S. From the sign-corrected left singular vector $\vec{U}_k^c$, the sign-corrected PCA output is calculated as:

$$PCA = \vec{U}_k^c \times diag(S) \tag{12}$$

Figure 4 shows PCA results before and after the sign is corrected. The lines indicate the first three principal components from a randomly generated dataset (36×8) where all points on all dimensions range from -1 to +1. A peak in Figure 4(a) indicates where the sign-flip occurred. Figure 4(b) shows that the sign-flip problem is completely solved.



(a) (b)

Figure 4: First three principal components of a randomly generated dataset before (a) and after (b) the sign-flip problem is fixed. The spike in (a) indicates where the sign-flip occurred. These two graphs are created decreasing the first variable in steps of 0.1% along the horizontal axis.

4.3 Applying the Online SVD

Rank-1 modifications (update, downdate, revise, and recenter) were proposed by Brand [4, 5], which use an update rule that increases the rank of the SVD by 1 until a user-specified output is reached. This update rule can build the SVD in $O(mnr)$ time, where m and n are the dimensions of the matrix, and r ($r \leq \min(m,n), r \doteq O(\sqrt{\min(m,n)})$ is the ranking. In iPCA, we uses the rank-1 modification (update) to approximate the changed SVD. It initially performs calculations of U, S, and V^T. These matrices are then used to identify updates: $U'S'V'^T = USV^T + AB^T$ where A represents the column vector with the updated values and B^T is a binary vector indicating which columns have been updated.

The projection (m) from the column A to the eigenvalue (U) is $m \doteq U^T A$ and the amount of orthogonality to the eigenvalue is $p \doteq A - Um$. The projection (n) from the column of B to the eigenvalue (V) is $n \doteq V^T B$ and the amount of orthogonality to the eigenvalue is $q \doteq B - Un$. Simply by increasing the rank to $r+1$, the updated point is calculated as:

$$U' = [U; m]R_u \qquad V' = [V; n]R_v \tag{13}$$

The rotations (R_u and R_v) and the updated eigenvalue (S') are computed by rediagonalizing the $(r+1) \times (r+1)$ matrix.

$$\begin{bmatrix} diag(S) & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} U^T A \\ \|p\| \end{bmatrix} \begin{bmatrix} V^T B \\ \|q\| \end{bmatrix}^T \rightarrow \begin{bmatrix} R_u, S', R_v \end{bmatrix} \tag{14}$$

where $\|p\|$ and $\|q\|$ are the measures of how different the updated point is from the approximation. They are calculated as:

$$\|p\| \doteq \sqrt{p^T p} = \sqrt{A^T p} \qquad \|q\| \doteq \sqrt{q^T q} = \sqrt{B^T q} \tag{15}$$

Brand [5] demonstrates that Online SVD has an RMS error[3] of less than 0.3×10^{-10} when computing the SVD of a 1000×1000 matrix with a ranking of 30.

[3] RMS error is a measure of the dispersion of points around a center.

5. Understanding PCA Through Interaction

This section describes how data, eigenvectors, eigenvalues, and uncertainty can be understood through using iPCA. To explain these procedures in detail, we use four different types of datasets [2] as examples: a randomly generated dataset (36×8), the Iris dataset (150×4 matrix), the E.coli dataset (336×7 matrix), and the Wine dataset (178×13). The Iris dataset is well-known and broadly used in pattern recognition. It contains 3 classes of 50 instances where each class refers to a species of iris flower and each instance has 4 attributes (sepal length, sepal width, petal length, and petal width). The original E.coli dataset has 336 instances and 8 attributes. Each attribute was measured using biological analysis methods. For the purposes of our PCA analysis, one non-numerical attribute in E.coli dataset was removed. Finally, the Wine dataset has 3 classes of 178 instances and 13 attributes.

5.1 Understanding Data

In PCA, it is known that understanding the relation between data and PCA is difficult. Without a clear understanding of the math behind PCA, unveiling the underlying meaning from the data is almost impossible. In order to help the user easily understand relationships and find underlying knowledge, iPCA provides a technique of changing the values of individual data items and the dimension contribution in a specific dimension. Based on the modification of data items and dimensions, iPCA performs the PCA calculation and provides the changed PCA results visually to the user. These visual changes help the user perceptually recognize the relationship between data and PCA.

As mentioned above, iPCA performs the PCA calculation whenever modifications are made. However, the PCA calculation takes high computation time especially when the dimensionality of input data is high. To address this, iPCA provides two alternative solutions: Online SVD and Fast-estimation. Both are approximation approaches that perform the PCA calculation by referencing the previously measured PCA results. As we explained in Section 4.3, the Online SVD computes singular vectors based on changes in the original data, which guarantees calculation of the PCA output in $O(mnr)$ time. Fast-estimation, on the other hand, takes a minimum computation of $O(mn)$ because it is performed based on an assumption that the right singular vector is not going to change. Fast-estimation is performed as:

$$AV = US \rightarrow A'V = U'S' \tag{16}$$

where A and A' indicate the original and the modified input data, respectively. From the equation 4, it can be rewritten as $AV = US$ since V^TV is I (identity matrix). The Fast-estimation computes the PCA results based on multiplying the modified input data and the right singular vector. Although the Fast-estimation can minimize the computation time, it has the limitation of estimating the rotational movement as shown in Figure 5(b).

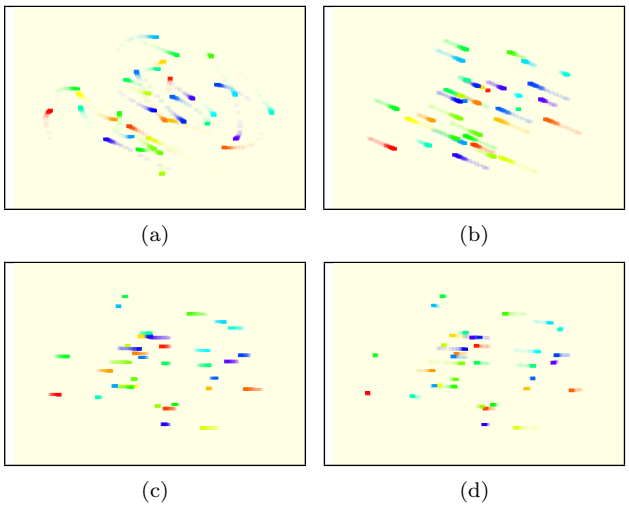


Figure 5: Examples when the Online SVD (a, c) and the Fast-estimation method (b, d) are applied to the random dataset. The transitional movements are created when the amount of the data values is changed in different dimensions.

Figure 5 shows examples when the Online SVD and the Fast-estimation method are applied. In principal, SVD has a characteristic of preserving the orthogonality of dimensions by rotating the principal components to search for a significant components whenever the values of data are changed. Since the Online SVD has the same characteristic as the SVD has, there is no visual difference between them. However, the Fast-estimation method does not have such characteristic of preserving the orthogonality of dimensions. Because of the reason, it does not show the rotational movement (Figure 5(b)). Instead, it only shows a directional movement (Figure 5(d)).

5.2 Understanding PCA

To support the user's understanding of the relationship between SVD and data, it is important to note that there is a common property of orthogonality ($U \times U^T = I$ and $V \times V^T = I$). Based on this property, SVD is often used to predict unknown values [3, 12]. As mentioned above, if the values of data items are changed, its relevant eigenvectors should be changed to maintain the orthogonality of dimensions. Similarly, eigenvector changes cause the modification of input data. In iPCA, the PCA calculation is performed whenever the modification of an eigenvector is made. However, the input data and the right singular vector are not deterministic and are difficult to determine from eigenvector changes. To resolve this limitation, we use an approximation method based on the assumption that the eigenvalue persists even if the eigenvector is changed.

The changed data and the modified right singular vector can be mathematically calculated with the following steps.

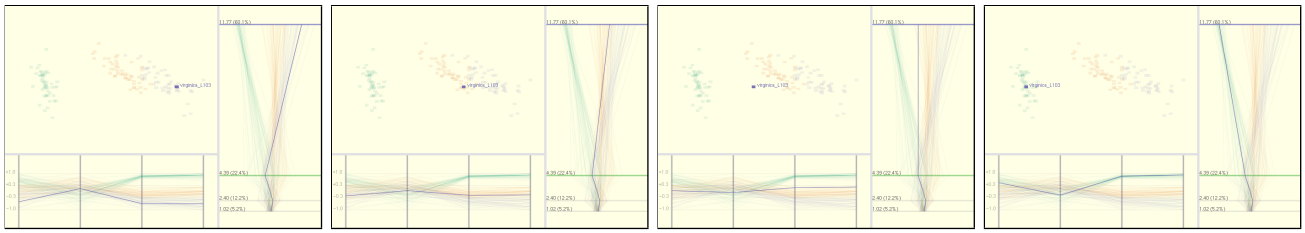


Figure 6: Interactively changing an eigenvector in the Iris dataset. Whenever the eigenvector is changed, the input data and the PCA output are changed correspondingly.

First, an eigenvalue decomposition method [1] is used ($A = X\Lambda X^T$ or $AX = X\Lambda$). We use a common relation that the SVD has:

$$\begin{aligned} A'A'^T &= (U'S'V'^T)(U'S'V'^T)^T \\ &= (U'S'V'^T)(V'S'^TU'^T) \\ &= U'S'S'^TU'^T \\ &= U'S'^2U'^T \end{aligned} \tag{17}$$

where A', U', S', and V' indicate the modified vectors. The singular value decomposition in general can be applied to any $m \times n$ matrix, whereas the eigenvalue decomposition[4] can only be applied to square matrices. Since the S' is the diagonal matrix, the $S'S'^T$ is equal to the matrix product of S'^2. The squares of the non-zero singular values of A' are equal to the non-zero eigenvalues of either $A'A'^T$. U' is the eigenvector of $A'A'^T$.

Second, the matrix $A'A'^T$ needs to be decomposed to find the individual matrixes of A' and A'^T. Cholesky decomposition [1] is applied. Since it is a decomposition method of a symmetric and positive-definite matrix into the product of a lower triangular matrix and its conjugate transpose, LU decomposition is applied to the symmetric matrix ($A'A'^T$). LU decomposition writes a matrix as the product of a lower triangular matrix and an upper triangular matrix ($A' = LU$). From the matrix $A'A'^T$, the lower triangular matrix L is decomposed to create the matrix L^T. Although the exact A' and A'^T cannot be found, approximate L and L^T are deterministic because a matrix (M) can be rewritten as $M = LL^T$. Hence, the generalized square matrix $A'A'^T$ is rewritten as the product of a lower triangular matrix L and a transpose of the matrix L^T. After finding the matrices L and L^T, A'^TA' can be computed by simply multiplying L^TL as:

$$A'A'^T = LL^T \longrightarrow A'^TA' = L^TL \tag{18}$$

From the equation $A'^TA' = L^TL$, the updated right singular vector (V') can be computed based on rediagonalizing $A'^TA' = V'S'^TS'V'^T$. The modified input data matrix A' can be found from the equation of $A' = U'S'V'^T$.

As we mentioned above, it is difficult to understand the relation between the eigenvector and other related collineation vectors in the PCA analysis. The eigenvector change shows how the change is related to the input data as well as the eigenvalue. However, it is impossible to compute the modifications of the input data and the eigenvalue correctly from the eigenvector change. In addition, performing the matrix decomposition is computationally expensive. Therefore, iPCA performs a simple method to find the modifications based on the previously computed right singular vector and the eigenvalue. Figure 6 shows an example of how the modification of the selected eigenvector may affect others. In this case, the selected data item is shifted from left to right in PCA space. At the same time, its values in the parallel coordinate view (data view) are changed. This movement helps the user understand the characteristics of the eigenvector.

5.3 Understanding Eigenvalue

Similar to the eigenvector, the eigenvalue changes show the modified results in the data space as well as in the PCA space. It helps the user to understand the relationship between eigenvalue and other matrixes (input data and eigenvector). Mathematically the eigenvalue is computed based on the matrix decomposition (left and right singular vector). However, it is still difficult to find the relevant input data and the left or right singular vector from the eigenvalue change. Hence, the modified input data can be computed based on the previously computed left and right singular vectors as $A' = US'V^T$, where A' indicate the modified input data matrix and S' represents the changed eigenvalue. A limitation of this approach is that it cannot find the exact corresponding matrixes correctly, but this assumption does support finding an approximate result (input data). Once the input data are found, the modified left and right singular vectors are computed by performing singular value decomposition on the computed input data.

Figure 7 shows an example of how the eigenvalue change affects the PCA output. Here we directly put the eigenvalue from 11.77 to 4.47. Although it uses an approximation method, it clearly shows that whenever the eigenvalue

[4] In eigenvalue decomposition ($AA^T = US^2U$), U is not unitary and S is not positive semi-definite, while U and V are unitary matrices and S is a diagonal positive semi-definite in the singular value decomposition ($A = USV^T$).

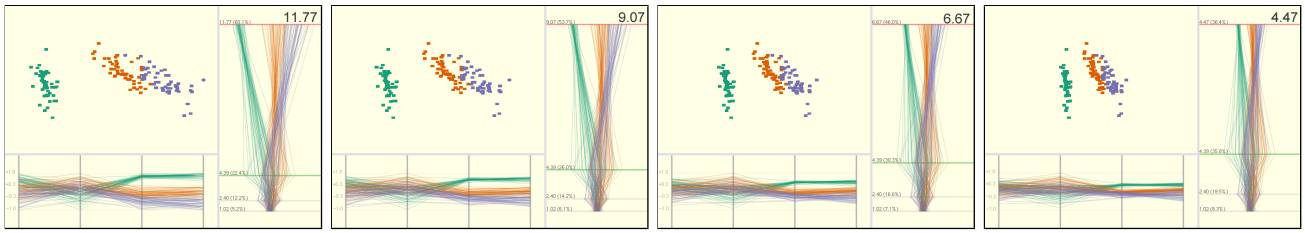
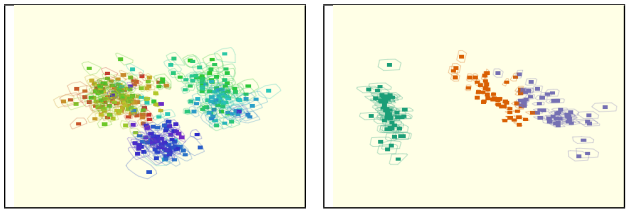


Figure 7: Interacting with an eigenvalue in the Iris dataset. The numeric values in each figure indicate the changed eigenvalue of the first principal component (from 11.77 to 4.47). These figures show that the eigenvalue changes destroy the overall structure of the Iris dataset.

in the first principal component is modified, the data structure projected onto the PCA space is changed correspondingly. This feature is useful for the user to understand how each principal component has a major role in creating the distinctive data structure.

5.4 Understanding Uncertainty

There are many situations in which a user may wish to understand the uncertainty of a given dataset, such as when data has been recorded with some potential for error. iPCA can display the uncertainty level of data to increase overall understanding of the dataset. If a dataset has a known level of potential error or uncertainty, it is visualized. In iPCA, an uncertainty ratio of 10% is conceived as a 10% variation in a point's data values across all of the original data dimensions. If there are n dimensions in the dataset, we represent the uncertainty projection for each point as an n-dimensional hypercube (Figure 8).

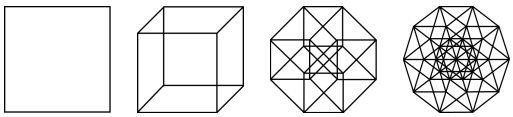


(a) (b)

Figure 8: Assuming 10% uncertainty in the data (in all dimensions) of the E.Coli dataset (a) and Iris dataset (b), the possible locations for each data item are drawn in outlines.

Whenever a modification has been made in the projected vectors, the represented uncertainty volume is changed in the projection space (Figure 9). It indicates that the range of uncertainty becomes more or less restricted depending on where the selected data item is projected. This interaction can visualize the shape of the high dimensional projection in a way that is very difficult using static views of PCA results. iPCA can display the uncertainty as either a wireframe hypercube or an outline. The n-dimensional hypercube is created from the n-dimensional dataset with a defined uncertainty range (in percent of potential error), and the outline simply indicates the boundary of the hypercube.

Displaying the uncertainty as a hypercube In geometry, a hypercube is a n-dimensional analogue of a square (n = 2) and a cube (n = 3) [6]. To project the n-dimensional object, we use polytopes [6], which are geometrical figures bounded by portions of lines, planes or hyperplanes (for polytopes of 2, 3 and 4 or more dimensions respectively). A polygon and a polyhedron are regarded as the polytopes in two dimensions and three dimensions, respectively. A cube of $(n-1)$ dimensions creates a cube of n dimensions by taking the $(n-1)$-dimensional cube and drawing it along a new dimension with a distance equal to the length of one of its sides. Then the vertices of each of the old cubes and the new cube are joined. Figure 10 shows examples of hypercubes.



(a) (b) (c) (d)

Figure 10: n-dimensional hypercubes as (a) square (n=2), (b) cube (n=3), (c) tesseract (n=4), (d) penteract (n=5).

Displaying the uncertainty as an outline Alternatively, the uncertainty can be viewed as an outline by applying a convex hull algorithm. In iPCA, we use Graham's algorithm [7], which runs in $O(n \log n)$ time. It initially finds a pivot point with largest y-coordinate. And all data points are sorted in order of increasing angle about the pivot. Then, the convex hull is built by connecting edges. Figure 8 shows examples of uncertainties created as outlines on each data item. Although there is no difference in representing the uncertainty as a hypercube or an outline, the outline representation helps to avoid visual clutter.

6. Discussion

We believe that interactive principal component analysis (iPCA) helps the user to understand the math behind PCA. However, there are some limitations to correctly

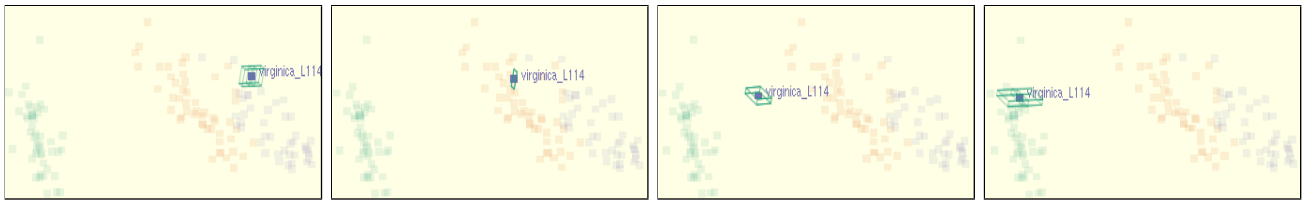


Figure 9: This series of images show the change of the uncertainty volume (tesseract) as a data item is moved across the screen in the Iris dataset. It clearly shows that as the selected data item moves (from a cluster group to another), the size of the uncertainty volume decreases. This implies that the area in which the data item could appear in the new location becomes more restricted.

grasping the underlying relations among data, eigenvectors, and eigenvalues. In this section, we provide several considerations that are relevant to understanding the importance and the usefulness of iPCA.

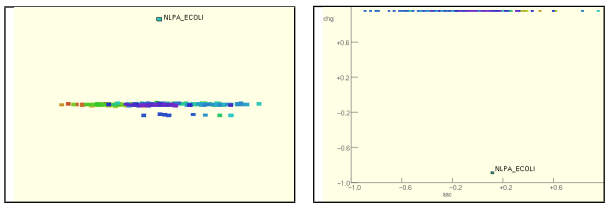
6.1 Effects of Combination of Dimensions

In the Iris dataset, it is already known that one class is linearly separable from the other two classes, and that the latter two classes are not linearly separable from each other. In iPCA, similar results can be seen in the Projection view. By interacting with the dataset, we find that it is the petal length dimension that makes one of these class linearly inseparable from the others. Also, we can see that removing the effect of both petal length and petal width obliterates the overall structure. On the other hand, we find that sepal length and width have less of an impact on the PCA results. Even when we change the contributions of these dimensions a great deal, the classes remain visually separated. However, both the E.coli and the Wine dataset are not linearly separable by PCA calculation. In the E.coli dataset, we find that changes in three attributes do not significantly affect the appearance of the default view of PC1 and PC2. Instead, the changes in the Wine dataset continuously produce different and inseparable results.

6.2 Exploring Outliers

In data analysis, it is often helpful to remove outliers from calculations because they are numerically distant from the rest of the data and often represent errors in the data. On the other hand, outliers can be important results and, in this case, their relation to the rest of the data should be studied in detail. With an outlier skewing calculations, the data cannot be fully analyzed or can give a misleading understanding. Detecting outliers can be difficult, and there has been a great deal of research on automated outlier detection. PCA calculation is one of the methods used to detect outliers in data [11, 13].

iPCA provides the capability to investigate these outliers and discover what attributes set them apart from the rest of the data. In the E.coli dataset, we can discover an obvious outlier by selecting PC1 and PC7 (Figure 11(a)). When we move this outlier's value on dimension 5 in the Data view to be closer to the mean, we can see that the point moves towards the rest of the data in the Projection view. This tells us that this point's separation from the cluster is due to its value on that single dimension. This can be verified through the scatter plot in the correlation view (Figure 11(b)).



(a) (b)

Figure 11: Finding an outlier in the E.coli dataset based on the first and the seventh principal components in the projection view (a) and the coefficient correlation between dimensions of *chg* and *acc* in the correlation view.

6.3 Understanding the Math behind PCA

Commercial software such as MATLAB [18] and SAS/INSIGHT [17] performs PCA and visualizes its results. Although these are powerful tools that utilize PCA, understanding the math behind PCA using the tools is limited. The strength of iPCA is that it helps the user to understand the math as well as the relation between original space and projected PCA space. However, there are some issues with this understanding that arise through changing the dimension contributions and approximating the results.

Changing the data can result in misleading views, for obvious reasons. If the user maintains an awareness of those changes and what they mean, the disadvantages of changing the dimension contribution can be outweighed by the advantages. But if the meaning of changes is unclear, users can easily become disoriented. However, there is a clear mathematical precedent to our use of dimension contributions. In weighted principal component analysis (WPCA), different variables can have different weights $s_1, s_2, ..., s_n$ [10]. This approach assumes that data are not always linearly increasing or decreasing, and there may be reason to allow different observations to have different weights. Based on this assumption, some researchers

adopt WPCA when analyzing complex data to set different weights to each variable, to find missing data by giving zero weight to missing elements, to create new approaches such as nonlinear multivariate data analysis, etc [10]. In iPCA, changing the dimension contribution by moving the slider bar in each variable provides the ability to analyze the data non-linearly. However, a correct justification between dimension contribution and weight change should be further studied.

Projecting visual changes might increase the user's understanding of PCA. In iPCA, approximation methods are used to solve mathematically unsolvable computations in case more than two unknown variables exist. It approximately calculates the newly updated values by referencing the previously calculated elements. Although approximation methods have the potential problem of not finding exact solutions, they can support the finding of nearly exact solutions within a limited computational time. Brand demonstrates the error rate when using Online SVD [5] (Section 4.3). Similarly, the degree of errors encountered when applying the approximation methods in iPCA should be computed to guarantee the reliability of these methods.

7. Conclusion and Future Work

PCA is a complex mathematical tool whose underlying operations cannot easily be understood by people. Although the mathematical understanding of PCA and SVD helps people to gain a concrete idea of how it works, we found that it is still difficult to create a clear figure that imposes the mathematical operations. Because of the difficulty, we designed a visual analytics tool (iPCA) to help the user easily understand PCA. iPCA is designed with four views: the projection view, the eigenvector view, the data view, and the correlation view. All views are closely connected, so that an action in one view can affect the other views. If the user interactively changes the elements in one view, its corresponding results are updated in other views. This interactivity helps the user understand principal component analysis by creating a visual model in her mind.

For future work, we are going to extend our visual analytics tool (iPCA) having different uncertainties for different dimensions, which would be more like a realistic case.

8. References

[1] E. Anderson and et al. *Principal Component Analysis*. the Society for Industrial and Applied Mathematics, thrid edition, 1999.

[2] A. Asuncion and D. Newman. UCI machine learning repository, 2007.

[3] M. Brand. Incremental singular value decomposition of uncertain data with missing values. In *In ECCV 2002*, pages 707–720, 2002.

[4] M. Brand. Fast online svd revisions for lightweight recommender systems. In *SDM 2003*, pages 83–91, 2003.

[5] M. Brand. Fast low-rank modifications of the thin singular value decomposition. *Linear Algebra and its Applications*, 415(1):20–30, 2006.

[6] H. S. M. Coxeter. *Regular complex polytopes*. Cambridge University Press, second edition, 1991.

[7] R. L. Graham. An efficient algorithm for determining the convex hull of a finite point set. *Information Processing Letters*, 1(4):132–133, 1972.

[8] Q. Hu. On the uniqueness of the singular value decomposition in meteorological applications. *Journal of Climate*, 10:1762–1766, 1997.

[9] S. Huang, M. O. Ward, and E. A. Rundensteiner. Exploration of dimensionality reduction for text visualization. In *CMV '05: Proceedings of the Coordinated and Multiple Views in Exploratory Visualization*, pages 63–74, Washington, DC, USA, 2005. IEEE Computer Society.

[10] I. T. Jolliffe. *Principal Component Analysis*. Springer, second edition, 2002.

[11] Y. Koren and L. Carmel. Visualization of labeled data using linear transformations. *InfoVis*, 00:16, 2003.

[12] M. Kurucz, A. A. Benczúr, and K. Csalogány. Methods for large scale svd with missing values. In *In KDDCup 2007*, 2007.

[13] M. Mohtashemi, K. Kleinman, and W. K. Yih. Multisyndrome analysis of time series using pca: A new concept for outbreak investigation. *Statistics in Medicine*, 26(29):5203–5224, 2007.

[14] G. E. Plassman. A survey of singular value decomposition methods and performance comparison of some available serial codes. NASA Technical Report CR-2005-213500, October 2005.

[15] E. Rasmus Bro and T. Kolda. Resolving the sign ambiguity in the singular value decomposition. *Journal of Chemometrics*, 22(2):135–140, 2008.

[16] A. Safonova, J. K. Hodgins, and N. S. Pollard. Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. *ACM Trans. Graph.*, 23(3):514–521, 2004.

[17] SAS Institute, Inc. SAS/INSIGHT. http://sas.com/technologies/analytics/statistics/insight.

[18] The MathWorks, Inc. Matlab. http://www.mathworks.com/products/matlab.

[19] M. Turk and A. Pentland. Face recognition using eigenfaces. In *Computer Vision and Pattern Recognition*, pages 586–591. IEEE Computer Society, 1991.

[20] M. E. Wall, A. Rechtsteiner, and L. M. Rocha. Singular value decomposition and principal component analysis. in A Practical Approach to Microarray Data Analysis D.P. Berrar, W. Dubitzky, M. Granzow, eds. pages 91-109, Kluwer: Norwell, MA, 2003.