ISSN 2395-8618

Algorithm Optimization Using Features In SVD & Classification In Eigenspace

Chaman Lal Sabharwal
Computer Science Department
Missouri University of Science and Technology
Missouri, Rolla, 63128 chaman@mst.edu

Abstract Singular Value Decomposition (SVD) is ubiquitous in a range of applications including computer science, economics, engineering, geology, oceanography, psychology, social networking etc. It is an unsupervised modeling technique that creates latent vectors for a subspace that reduces the dimensionality of observed data from n to k (k<<n) dimensions. Latent variables are uncorrelated variation of attribute values that are correlated in the original space. Moreover, SVD can be used to detect/remove noise/outliers, cluster similar entities and make predictions. On the other hand, classification tree is a supervised technique that accomplishes the similar tasks. It models decision trees from training data in order to make intelligent predictions. There is a close connection between SVD and decision trees, but differ in purpose, algorithm design and error analysis techniques. We present a hybrid algorithm bridges the gap between these standalone algorithms and adaptively supersedes their outcomes. For experimental analysis, we use real-world benchmark data, wines, publicly available from UCI machine learning repository. The algorithm is implemented in Matlab, supported by decision trees in Weka software, on MacOS Seirra Version 10.12.3 8GB 160MHZ.

Keywords. PCA, SVD, MDS, Dimensionality Reduction, Classification Tree.

I. INTRODUCTION

SINGULAR Value Decomposition (SVD) is ubiquitous in a range of areas including computer science, image processing (compression, enhancement), engineering, economic and social behavior models, geology, oceanography, psychology, psychophysics, social networks, visualization, and natural language processing [1], [2]. Singular Value Decomposition (SVD) is a study of the underlying structure of objects and their properties for efficient knowledge exploration.

Principal Component Analysis (PCA) and Singular Value Decomposition (SVD) are interchangeably referred in the literature. In fact, PCA is for real symmetric matrices whereas SVD is generalization of PCA applicable to any rectangular positive semi-definite matrices. However, PCA and SVD are equivalent for symmetric positive semi-definite matrices. In Section 2, we will elaborate in detail on the difference between the two, PCA and SVD. While SVD is used for unsupervised modeling, the Multi Dimensional Scaling (MDS) and Decision Tree are supervised modeling techniques for making reliable classification predictions. Both of these techniques have complimentary roles in extracting knowledge embedded in data. For example, for recommendation of a product, it may be authenticated with large consumer survey in the face of high dimensional large question answer data. We will present a hybrid algorithm that takes advantage of functionality of SVD to optimize Decision trees, resulting in substantial reduction in computational effort and reduced storage space at insignificant cost in accuracy.

The paper is organized as: Section II is background Section III is on related work, Section IV is algorithm design, prediction and accuracy measurement metrics, Section V implementation, experiments and discussion of results. Section VI is conclusions, Section VII is references and Section VIII includes the linear algebra appendix.

II. BACKGROUND

The real world business data is in the form of tables. Mathematically [see Appendix] speaking, a table is an m×n matrix whose m rows are observation vectors and n columns are properties/attributes of the observation. Principal Components are new latent vectors blending features in original vectors [3]. Because row rank and column rank of a matrix are equal, the rank of an m×n matrix A is k, where $k \leq \min(m, n)$ [4]. One of the advantages of SVD is that it is applicable to positive semi-definite rectangular matrices including symmetric square matrices [5], [6]. Linear algebra is the backbone of PCA development. Some users use it blindly without understanding the algebraic implications. We clarify it with the following examples.

A. What is Eigen-Decomposition?

If A is an n×n real matrix and has eigenvalues and eigenvectors, then (1) the eigenvectors are normalized to unit vectors and (2) the eigenpairs are arranged in the descending order of eigenvalues [7]. Let V be the matrix of eigenvectors of A. If V is invertible, algebraically $VV^{-1} = I$, and geometrically V is a latent orientation of original axes for data visualization. Let D be the matrix of eigenvalues of A arranged in descending order. The goal is to express A in terms of eigenvectors matrix V and eigenvalues matrix D as $A = VDV^{-1}$. For example, (1) the matrix $\begin{bmatrix} 0 & 1 \\ -1 & -2 \end{bmatrix}$ has two identical eigenvalues: -1, -1; and only one eigenvector $\begin{bmatrix} 1 \\ -1 \end{bmatrix}$. It does not have eigen-decomposition. (2) The matrix $\begin{bmatrix} 1 & 0 \\ -2 & -1 \end{bmatrix}$ has two distinct eigenvalues: 1, -1, and eigenvectors: $\begin{bmatrix} 1 \\ -1 \end{bmatrix}$, $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$

ISSN 2395-8618

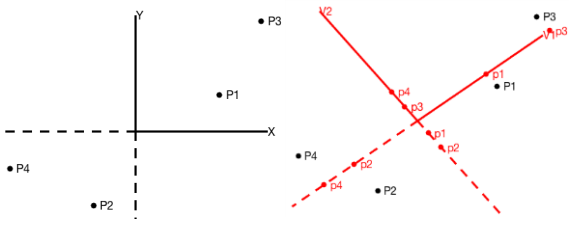
which are linearly independent, but *not orthogonal*. The eigen-decomposition is A = VDV^{-1}

$$\begin{bmatrix} 1 & 0 \\ -2 & -1 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 \\ \frac{-1}{\sqrt{2}} & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} \sqrt{2} & 0 \\ 1 & 1 \end{bmatrix}, \text{ but } V^{-1} \neq V^T.$$

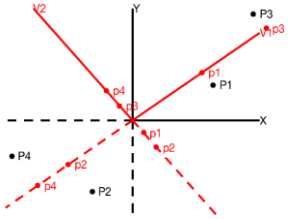
If eigenvectors are not orthogonal, this decomposition is not useful for data mining. If A is symmetric, then V is orthogonal matrix of eigenvalues and the inverse is simply the transpose of V, i.e., $V^{-1}= V^T$.

B. What is PCA?

The basic concept of PCA is to create smaller set new latent variables in place of original large set of variables/attributes used in observations [8]. Principal Component Analysis (PCA) is a generalization of eigen-decomposition to symmetric matrices leading to orthogonal eigenvectors such that A = VDV^T= VDV^{-1}. However, PCA uses the eigenvectors of covariance matrix, $A^TA = A^2$ [9]. The eigenpairs are arranged in descending order of eigenvalues. The first eigenvector gives the *direction of the largest spread*, and the first eigenvalue is *the largest spread*, see Figure2. It is equivalent to the least squares deviation meaning data points are at shortest distance from the computed eigenvector. PCA determines eigenpairs for A^2 such that the eigenvectors are (1) pairwise orthogonal, (2) normalized to unit vectors and (3) arranged in the descending order of eigenvalues, (4) that the variance along the eigenvectors are maximum in descending order. Let V be the matrix of *eigenvectors of A^2* arranged in the descending order of eigenvalues of A, then algebraically V is orthogonal matrix and is invertible. Let D be the diagonal matrix of eigenvalues of A arranged in descending order provided eigenvalues of A are non-negative. For negative eigenvalues of A, sign has to be taken into consideration. PCA reconstructs A from V and D as A = VDV^T = VDV^{-1}. Figure 1(a) shows four data points along with the standard coordinate axes. Figure 1(b) has eigenvectors and projections of data points on the eigenvectors to show the variance of data along the eigenvectors as new computed axes. It is a rotation of the vase coordinates system. Figure 1(c) displays standard and new coordinate systems, data points and their projections on the eigenvectors.



(a) (b)



(c)

Figure 1. (a) Four data points {P_1, P_2, P_3, P_4}, (b) eigenvectors, projections of data points on the eigenvectors to show data spread along new axes, (c) representative of uv, and xy frames with data points and projections.

We generalize the previous example to symmetric matrix for the PCA. The matrix $\begin{bmatrix} 1 & 0 \\ 0 & -2 \end{bmatrix}$ has eigenvalues λ =1, -2 and the eigenvectors are $\begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ which are linearly independent, and *orthogonal* eigenvectors. Then the eigen-decomposition A = VDV^{-1} = VDV^T is $\begin{bmatrix} 1 & 0 \\ 0 & -2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & -2 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$.

But PCA uses AA^T and A^TA which are $A^2 = \begin{bmatrix} 1 & 0 \\ 0 & 4 \end{bmatrix}$ (A, A^2 symmetric) for computing the eigenpairs. Except for signs, the eigenvalues of A are square roots of the eigenvalues of A^2 that are 4 and 1, the corresponding eigenvectors are eigenvectors of A^2 are $\begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}$. Matlab svd does not reconstruct A. In our algorithm, we include the proper signs. Here we used the proper sign for *square root of 4 to -2*, because -2 is eigenvalue of A.

Using eigenvectors of A^2, the eigen-decomposition A = VDV^{-1} = VDV^T is $\begin{bmatrix} 1 & 0 \\ 0 & -2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & -2 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$.

C. What is SVD?

Singular Value Decomposition (SVD) is a generalization of PCA to include (1) non-square and (2) positive semi-definite matrices [10], [11]. PCA and SVD are equivalent for symmetric positive semi-definite matrices. By definition, an m×n real matrix A is positive semi-definite, if $\mathbf{v}^T A\mathbf{u} \geq 0$ for all vectors **u** and **v**. The matrices AA^T and A^TA are symmetric and positive semi-definite. For symmetric matrices AA^T and A^TA, the eigenvalues are de facto non-negative and eigenvectors are orthogonal. SVD uses covariance matrices AA^T and A^TA to determine two orthogonal matrices of eigenvectors U, V and a diagonal matrix S for eigenvalues such that the eigenvectors in U, (and V) are (1) pairwise orthogonal, (2) normalized to unit vectors and (3) arranged in the descending order of eigenvalues. Then SVD decomposes A into three factors U, V and S such that A = USV^T. The examples where A is not both symmetric and positive semi-definite are shown in the Table 1 to understand SVD and PCA are not equivalent in general.

Example. To accommodate both PCA and SVD, we generalize the previous example matrix to symmetric, positive semi-definite A= $\begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$ whereas again $AA^T=A^TA=\begin{bmatrix} 1 & 0 \\ 0 & 4 \end{bmatrix}$.

The eigenvalues of A are 2,1; so D= $\begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}$ which is same as S. Thus for PCA/SVD of A, the eigenpairs of AA^T, A^TA are U

ISSN 2395-8618

$= \begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}$, V= $\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}$, and S = $\begin{bmatrix}2 & 0\\ 0 & 1\end{bmatrix}$.

And A = USV^T becomes

$\begin{bmatrix}1 & 0\\ 0 & 2\end{bmatrix} = \begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}\begin{bmatrix}2 & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}$ implying A = USV^T.

Summarizing this discussion, we see that several possibilities exist for an arbitrary matrix A. We show these examples in Table 1.

Consequently, we formulate the sufficiency conditions for the existence and equivalence of PCA and SVD. If a matrix A is symmetric and positive semi-definite, then A has SVD decomposition. Recall that for any matrix A, the covariance matrices AA^T, A^{T}A are symmetric and positive semi-definite. We can always get around the exceptional cases for A. In Table 1, there are some cases where A is (1) symmetric and (1.1) has PCA SVD decomposition equivalent on PSD (1.2) has PCA SVD decomposition, but not equivalent on not PSD, and (2) not symmetric and (2.1) PCA SVD decomposition equivalent on PSD (2.2) has PCA SVD decomposition not equivalent on not PSD or non-square PSD. Also for non-square matrices, there is SVD, not PCA.

TABLE1. FOUR POSSIBLE CASE OF MATRIX

Data Matrix	Positive Semi-Definite	Not Positive Semi-Definite
Symmetric	$\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$ PCA≡SVD	$\begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix}$ PCA≠SVD
Not Symmetric	$\begin{bmatrix}1 & 0\\ 0 & 1\\ 0 & 0\end{bmatrix}$ SVD but no PCA	$\begin{bmatrix}-1 & 0\\ 0 & -1\\ 0 & 0\end{bmatrix}$ no SVD, no PCA

D. What is MDS? [10], **[11]**, [12] Multidimensional Scaling (MDS) is a twist of PCA/SVD and resorts to minimization or regression optimization for solution to this problem, whereas PCA/SVD resorts to eigenpair solution to this problem. The new variables are linear combinations of original variables. Mathematically, observations samples are points in higher dimensional space. If $\mathbf{x^T} = [x_1, x_2, \ldots, x_n]$ is a point in base coordinate system, then $\mathbf{y^T} = [y_1, y_2, \ldots, y_k]$, k<<n, is the same point in new coordinate system where for p =1,…,k $y_p = \sum_i w_{ip} x_i$ the coefficients w_{ip} are also called the weights associated with x_i, namely, $\mathbf{w_p^T} = [w_{1p}, w_{2p}, \ldots, w_{np}]$ is a weight vector. This means y_p is the projection of **x** on $\mathbf{w_{\bullet p}}$. Algebraically, the matrix $[\mathbf{w_{\bullet p}}]$ is a linear transformation of **x** to **y**. Geometrically, it is a matrix of rotation from a standard vector space basis to principal component vector basis. The vector $\mathbf{w_{\bullet p}}$ is determined in such at way that variation of data points along this direction is maximum, then the data projection is $A\mathbf{w_{\bullet p}} = b_p$, $A[\mathbf{w_{\bullet p}}] = [b_p]$. Note that this $\mathbf{w_{\bullet p}}$ is the same as vector $\mathbf{v_p}$ in PCA/SVD as such V =$[\mathbf{v_p}]$ =$[\mathbf{w_{\bullet p}}]$, AV = B expresses m×n matrix A to smaller m×k matrix B. V is the matrix of eigenvectors and amount of spread along each vector is the corresponding eigenvalue.

E. What is a Decision Tree? A decision tree is a tree where the paths from the root to the leaf nodes generate rules for classification of data items [13],[14]. The conventional way to classify multidimensional data is to start with a feature space whose dimensionality is the same as that of the data. The attributes are rearranged by leveraging entropy at each step in the process of tree construction from the root node to proceed to child nodes. Entropy [15Kdnugg2017] is a computational technique for determining the best possible attribute to be used in the decision tree construction. For example, if an attribute has n values x_i, we compute the probability $p(x_i)$, to associate the entropy with **x**:

$$\text{Entropy}(x) = -\sum_{i=1,n} p(x_i) \log_2 p(x_i)$$

Decision is based on entropy of the feature values. At each stage of tree construction, we use the conditional entropy based on attributes that have been already selected.

How is decision tree used? The decision tree can be displayed graphically for easy visualization and intuitive understanding quickly. In decision tree approach prediction, search in the decision table is replaced by decision tree traversal algorithm to determine the classification. All machine learning algorithms have flaws. For a list of such flaws see [15]. It is an interesting article on what is right and what is wrong with the algorithms. So one has to understand the fundamental underpinnings of the algorithm and has to be vigilant in the selection of algorithm before using it. This paper adheres to this conviction.

III. RELATED WORK

A. What is meant by Size Reduction, Noise Reduction, and Clustering Data? The primary purpose of reduction is to optimize storage space and computation time, not necessarily to recreate the original data. The most useful contribution PCA/SVD is that one can ignore eigenvalues or variations below a certain threshold [11]. For this purpose, it is desirable to uncorrelated the correlated. Let A be an m×n real and symmetric data matrix. We may want to reduce the number of variables/properties/attributes of data (row size) or number of observations/objects for size of data (column size) or both. We create two orthonormal matrices U and V where U and V are eigenvector matrices of covariance matrices AA^T and A^{T}A. Since AA^T and A^{T}A are square, symmetric, and positive semi-definite, there exist (1) orthonormal matrices U, V, such that UU^T = 1 and VV^T = 1 and (2) a diagonal matrix S such that is AA^T = USU^T, A^{T}A = VSV^T. Consequently, the positive semi-definite rectangular matrix A can be recreated from U, V, S : A = USV^T. The covariance matrix A^{T}A is not necessarily a diagonal, but covariance matrix of rotated data AV is diagonal: $(AV)^TAV = (US)^TUS = SU^TUS = SS = S^2$ which is a diagonal matrix with non-negative values. Similarly $U^TA(U^TA)^T = SV^T(SV^T)^T = SV^TVS^T = SS = S^2$. Recall that U^{T}U = I implies UU^T = I This confirms that the orthonormal transformation matrices U and V un-correlate the correlated values because the covariance of AV and U^{T}A are diagonal. It follows from the diagonal matrix that (1) the pairwise covariance is zero, hence variables are uncorrelated in the new, rotated coordinate system. (2) eigenvalues of AA^T and A^{T}A are never negative. (3) trace(AA^T) is equal to the sum of eigenvalues of AA^T.

Now we have two orthonormal matrices U_{mxm} and V_{nxn}. The matrix A_{mxn} can be reduced by constraining U and V to (1) fewer columns AV_{nxk} by reducing the variables/attributes from n to k, k<<n, (2) fewer rows $U_{mxk}{}^TA$ reducing size of data from m observations/rows to k rows, k<<m. In case, we want to reduce both attributes and size, then we can use kxk instead of mxn A: $U_{nxk}{}^TAV_{mxk}$. The data size can be further reduced if the number of non-zero eigenvaluesis less than k<<min(m,n).

ISSN 2395-8618

These concepts and results are used in error analysis for determining the optimal number of eigenvectors sufficient for dimension reduction. However, (1) it is difficult to extract a small number of features for any learning algorithm, and (2) PCA/SVD cannot determine exactly which of the original uncorrelated variables/attributes can be dropped. Thus it cannot determine which original variables are important [1]. As we calculate eigenvectors, it becomes trivial that $A = USV^T$ implies $S = U^TAV$ that means A is diagonalizable.

When we transform the dataset, error is natural to occur due to projection for reducing dimensions. For example, for Wine data discussed in section 4.1, we find that as a result of SVD, using eigen space of dimension 1,2,3, etc. the error decreases, see Figure 2. It shows that just four dimensional transformed space data accounts for 98% of the data in the original space.

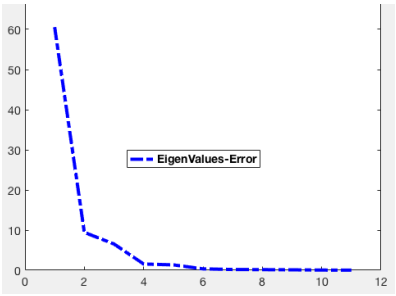


Figure 2. This shows errors due to reduced dimensionality when we use 1,2,3…,11 principal components. Reduction in data space that accounts for increased computational efficiency and increased error.

Most challenging part of PCA/SVD is interpretation of components in terms of original coordinates. Most of the time we reduce only the number of attributes (row length). It is more useful to apply SVD transformation before applying classification tree algorithm. This can be done more efficiently in transformed space. Moreover, we apply classification learning in the transformed space with substantial reduction in computational effort and storage space at insignificant cost in accuracy.

B. Algebraic Foundations and Dimension Reduction

For a positive semi-definite matrix A, the SVD decomposition $A = USV^T$ is full spectrum decomposition. If we use fewer, say k, columns of V, it reduces to k features in the new frame, $k << \min(m,n)$, in the m×n data where k indicates the number of columns used, S_k is a k×k diagonal matrix. Then $A_k = U_kS_kV_k^T$, A_k has least error from A.

C.1 Eigen-Decomposition Analysis

For a symmetric matrix A, eigenvalues exist and are real. The eigenvectors generate the vector space of matrix rows. The eigenvectors corresponding to different eigenvalues are orthogonal.

Theorem. Let A be a real symmetric matrix. Show that there exists an orthogonal matrix V and a diagonal matrix D such that $A = VDV^T$ where V is the matrix of eigenvectors of A; and D is the diagonal matrix of eigenvalues of A.

Proof. Let $A\,\mathbf{v_i} = \lambda_i\,\mathbf{v_i}$ for which $(\lambda_i, \mathbf{v_i})$ is a corresponding eigenvalue and eigenvector pair. We arrange the eigenvalues and eigenvectors on descending order of eigenvalues.

Let D be diagonal matrix of eigenvalues i.e., $d_{ii} = \lambda_i$ or $D = [\lambda_i]$, and V be the matrix whose columns are eigenvectors, i.e., $V = [\mathbf{v_i}]$. Since V is orthogonal matrix, $V^{-1} = V^T$. From eigenpair equation

$$A\,\mathbf{v_i} = \lambda_i\,\mathbf{v_i}$$

using matrix notation, it becomes

$$A\,[\mathbf{v_i}] = [\lambda_i\,\mathbf{v_i}] = [\mathbf{v_i}\,\lambda_i] = [\mathbf{v_i}][\lambda_i]$$

Since A is symmetric $V^{-1} = V^T$

$$AV = VD \rightarrow A = VD\,V^{-1} = VDV^T$$

Note that the symmetry constraint on A is sufficient, but not necessary.

Example: The matrix A *has eigen-decomposition, even when A* is *not symmetric*.

The matrix $\begin{bmatrix}0 & 0\\ 1 & 1\end{bmatrix}$ has eigenvalues: 0,1. Eigenvectors: $\begin{bmatrix}1\\ -1\end{bmatrix}$, $\begin{bmatrix}0\\ 1\end{bmatrix}$ which are linearly independent, but *non-orthogonal* eigenvectors. The eigen-decomposition is $\begin{bmatrix}0 & 0\\ 1 & 1\end{bmatrix} = \begin{bmatrix}0 & \frac{1}{\sqrt{2}}\\ 1 & -\frac{1}{\sqrt{2}}\end{bmatrix}$ $\begin{bmatrix}1 & 0\\ 0 & 0\end{bmatrix}\begin{bmatrix}1 & 1\\ \sqrt{2} & 0\end{bmatrix}$.

This example shows that eigenvectors are not orthogonal, and $V^{-1} \neq V^T$, thus $A = VDV^{-1}$, but $A \neq VDV^T$.

For PCA, the eigenvectors of A^TA and AA^T are used.

The matrix $A^TA = \begin{bmatrix}1 & 1\\ 1 & 1\end{bmatrix}$ has eigenvalues:0, 2. Eigenvectors: $\begin{bmatrix}1\\ -1\end{bmatrix}$, $\begin{bmatrix}1\\ 1\end{bmatrix}$ which are linearly independent, *orthogonal.*

The matrix $AA^T = \begin{bmatrix}0 & 0\\ 0 & 2\end{bmatrix}$ has eigenvalues:0,2. Eigenvectors: $\begin{bmatrix}1\\ 0\end{bmatrix}$, $\begin{bmatrix}0\\ 1\end{bmatrix}$ which are linearly independent, *orthogonal* forming $U = \begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}$, $V = \frac{\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}}{\sqrt{2}}$ here $U \neq V^T$, $S = \begin{bmatrix}\sqrt{2} & 0\\ 0 & 0\end{bmatrix}$,

Note 1. The PCA is $\begin{bmatrix}0 & 0\\ 1 & 1\end{bmatrix} = \begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}\begin{bmatrix}\sqrt{2} & 0\\ 0 & 0\end{bmatrix}\frac{\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}}{\sqrt{2}}$ which eigen-decomposition, SVD, but different.

Note 2. The matrix $\begin{bmatrix}0 & 0\\ 1 & 1\end{bmatrix}$ does not have PCA but SVD, $\begin{bmatrix}0 & 0\\ -1 & -1\end{bmatrix} = -\begin{bmatrix}0 & 0\\ 1 & 1\end{bmatrix}$ has no PCA , or SVD

C.2 PCA Analysis

Here we see that for a symmetric matrix, eigen-decomposition is guaranteed. For a square, symmetric matrix A, A^2 is also a square symmetric matrix, eigenvalues of A^2 are real and non-negative. Let $(\alpha, \mathbf{u})$ be eigenpair for A^2, i.e., $A^2\,\mathbf{u} = \alpha\,\mathbf{u}$. then $\alpha \geq 0$, because

$\alpha = \alpha\,(\mathbf{u}, \mathbf{u}) = (\alpha\,\mathbf{u}, \mathbf{u}) = (A^2\,\mathbf{u}, \mathbf{u})$

$= (A\,\mathbf{u}, \mathbf{A^Tu}) = (A\,\mathbf{u}, \mathbf{Au}) \geq 0$ from symmetry of A.

Theorem. If A is a symmetric square matrix and **u** is a unit eigenvector of A^2 with eigenvalue α, then

(1) A**u** is also an eigenvector of A^2 with the same

ISSN 2395-8618

eigenvalue α,

(2) $A\mathbf{u} = \sqrt{\alpha}\,\mathbf{v}$ and $A\mathbf{v} = \sqrt{\alpha}\,\mathbf{u}$ where **u, v** are unit eigenvectors of A^2.

Proof: let $A^2\mathbf{u} = \alpha\,\mathbf{u}$, then $AA^2\mathbf{u} = \alpha\,A\mathbf{u}$

then

$$AA^2\mathbf{u} = \alpha\,A\mathbf{u}$$
$$A^2(A\mathbf{u}) = \alpha\,A\mathbf{u}$$

Since $\mathbf{u} \neq 0$, and $A\mathbf{u} \neq 0$, $A\mathbf{u}$ is an eigenvector of A^2 for the same eigenvalue.

Let us denote the unit eigenvector by **v**

The $A\mathbf{u} = \beta\,\mathbf{v}$ for some beta.

We show that $\beta = \sqrt{\alpha}$ □□

α □ = α **(u, u)** = (α **u, u)** = (A^2 **u, u)** = (A **u,** A^T**u)** = (A **u,** A**u)** = (β **v,** β **v)** = β^2 **(v, v)** = β^2

therefore α □ = β^2

Since α □ = β^2

therefore $A\mathbf{u} = \pm\sqrt{\alpha}\,\mathbf{v}$

Let us use + symbol for simplicity. Also it can be verified that $A\mathbf{v} = \sqrt{\alpha}\,\mathbf{u}$. For example,

$A\mathbf{u} = \sqrt{\alpha}\,\mathbf{v}$ implies $A^2\mathbf{u} = \sqrt{\alpha}\,A\mathbf{v}$ or $\alpha\,\mathbf{u} = \sqrt{\alpha}\,A\mathbf{v}$ or $A\mathbf{v} = \sqrt{\alpha}\,\mathbf{u}$

Example This example shows that each eigenvector of A^2 is an eigenvector of A.

let $A = \begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix}$, $A^2 = \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$,

eigenvalues of A^2 are 1, 1 and eigenvectors $\mathbf{u} = \begin{bmatrix}1\\0\end{bmatrix}$, $\mathbf{v} = \begin{bmatrix}0\\1\end{bmatrix}$,

eigenvalues of A are 1, -1 and eigenvectors $\mathbf{u} = \begin{bmatrix}1\\0\end{bmatrix}$, $\mathbf{v} = \begin{bmatrix}0\\1\end{bmatrix}$,

$A^2\,\mathbf{u} = \mathbf{u}$, $A\,\mathbf{u} = \mathbf{u}$.

$A^2\,\mathbf{v} = \mathbf{v}$, $A\,\mathbf{v} = -\,\mathbf{v}$.

$A = USU^T$

Trivially, $\begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix} = \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix}\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$

Theorem. Let A be symmetric matrix. Show that there exists an orthogonal matrix U and a diagonal matrix S such that $A = USU^T$ where U is matrix of eigenvectors and S is the matrix of square roots of eigenvalues of A^2.

Proof. In PCA, we arrange the eigenvalues and eigenvectors according to descending order of eigenvalues. Let $S = [\lambda_i]$ be diagonal matrix of square roots of eigenvalues and $U = [\mathbf{u_i}]$ be matrix of eigenvectors of A^2. Since U is orthogonal matrix, $U^{-1} = U^T$ and

$$A\,\mathbf{u_i} = \lambda_i\,\mathbf{u_i}$$

which implies that

$$A\,[\mathbf{u_i}] = [\lambda_i\,\mathbf{u_i}] = [\mathbf{u_i}\,\lambda_i]$$

becomes

$AU = US \rightarrow A = US\,U^{-1}$ or $A = USU^T$

C.3 SVD Analysis

If A is not positive semi-definite, we may not have SVD decomposition. For example, let A be a rectangular matrix. Then AA^T and A^TA are symmetric and positive semi-definite. The eigenvalues of AA^T and A^TA exist, are non-negative and identical. Let U be matrix of eigenvectors of AA^T, and V be matrix of eigenvector of A^TA, also $A^TA\,\mathbf{v} = \lambda$□$\mathbf{v}$ implies $AA^T(A\,\mathbf{v}) = \lambda$□$(A\,\mathbf{v})$ or $A\mathbf{v}$ is eigenvector of AA^T. Since **u** and **v** are unit vectors, the relation between A^Tu and **v** is $A^Tu = \mu\,\mathbf{v}$ for some scalar μ; and the relation between $A\mathbf{v}$ and **u** is $A\mathbf{v} = \mu\,\mathbf{u}$ where $\mu = \sqrt{\lambda}$. For example, if $A^Tu = \mu\,\mathbf{v}$, then

$$\lambda = \lambda\,(\mathbf{u, u}) = (\lambda\,\mathbf{u, u}) = (AA^T\,\mathbf{u, u})$$
$$= (A^T\,\mathbf{u,}\ A^T\mathbf{u}) = (\mu\,\mathbf{v,}\ \mu\,\mathbf{v})$$
$$= \mu^2\,(\mathbf{v,\ v}) = \mu^2$$

Thus if we know U or V, the other is readily available.

We have now

$A\mathbf{v_i} = \mu_i\mathbf{u_i}$ for each non-zero eigenvalue μ_i,

zero value contribute nothing to the matrix of U, S, V.

$$A[\mathbf{v_i}] = [\mu_i\mathbf{u_i}]$$
$$A[\mathbf{v_i}] = [\mathbf{u_i}\,\mu_i] = [\mathbf{u_i}][\,\mu_i]$$
$$AV = US$$

This is called the *polar* decomposition. An interesting outcome of this equation is that projection of right singular vectors are scores of left singular vectors. Eigenvectors are called factors and eigenvalues are called loads. Since U and V are orthogonal, the equation AV = US leads to spectral decomposition of A:

$$A = USV^T$$

Example. The matrix A is *not positive semi-definite*, but A is symmetric. The PCA and SVD exist and are not equal to eigen-decomposition.

The matrix $A = \begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix}$ has eigenvalues 1 and -1 and eigenvectors as $\begin{bmatrix}1\\0\end{bmatrix}$ and $\begin{bmatrix}0\\1\end{bmatrix}$. It shows that the eigenvectors for $AA^T = A^TA = A^2$ are same, but the eigenvalues are 1,1. This results in

$U = \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix} = V = V^T$, $D = \begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix}$ and, $S = \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$ and not $A \neq USV^T$ though we have $A = UDU^{-1} = UDU^T$.

If A is positive definite, square or rectangular, we can have $A = USV^T$, where V is not necessarily U.

IV. THE ALGORITHM AND THE METRICS FOR ACCURACY ANALYSIS

A. The Algorithm

Routinely the algorithms for classification and dimensionality reduction are used standalone to accomplish the intended tasks. As a result, computational efficiency and storage efficiency suffer. We create a hybrid algorithm to take advantage of the computational and storage efficiency first and then apply the classification tree algorithm to data in the compressed domain. As a result we get substantial efficiency, on the average, we gain 18% computational efficiency at the cost of .08% accuracy reduction. Further optimization depends on the size of data and data properties. More the number of attributes, better the performance of the algorithm. Figure 3 describes this strategy.

Input: m×n data matrix A
Output: classification tree for A_{mxm} from reduced data B_{mxk}
The enhanced *SVD algorithm* is as follows
Ignore the missing values from consideration.
Create covariance matrices AA^T and A^TA normalized by the dimensions
Calculate eigenvalues and eigenvectors (normalized to unity) of AA^T and A^TA
Calculate eigenvalues of A to track the negative

ISSN 2395-8618

eigenvalues
Calculate the square roots of values of AA^T and A^TA
Update the signs according to the sign for eigenvalues of A
Rank the updated eigenvalues in descending order and form a diagonal matrix S
Rank the eigenvectors for AA^T using the order of eigenvalues and form matrix U
(Once U is computed V is readily available)
Rank the eigenvectors for A^TA using the order of eigenvalues and form matrix V
Determine k<<n for the number of eigenpairs acceptable for data reduction.
Transform m×n A to reduced m×k data set $B = AV_k$

The *Decision tree algorithm* in general is
Convert the classification numeric attribute to intelligent interval categorical attributes,
Apply Entropy based Greedy attribute selection
Use entropy to determine the attribute selection at each node of the tree construction using
max entropy gain
Create classification tree nodes
The leaf nodes are classes and internal nodes are tree paths to generate rules.

Hybrid approach
Collect data, clean the data items with incomplete values
Apply the enhanced PCA that may reduce classifier dimension based
Reduce the data set by ignoring contribution of unacceptable eigenvalues (near zero eigenvalues)
Apply Classification Tree algorithm in the reduced domain and original base domain
Analyze and compare the results.

Figure3. Algorithm for Hybrid approach

The enhanced PCA/SVD algorithm is implemented in Matlab and decision tree algorithm J48 is used from Weka software. Benchmark data on wines is obtained from UCI ML dataset repository. The following metrics are used to benchmark the accuracy.

B. The Metrics for Accuracy Analysis

B.1 Metrics for Decision Trees

The Data mining community uses *gold* standard, *precision, recall and F measures*, to determine the predictive accuracy of the classification [15]. For example, *Precision* is related to the accuracy of positive prediction, predicting negative as positive that means false positive, a false alarm, whereas, *Recall is* related to the accuracy of prediction on positive data, predicting positive to negative that meaning false negation, a missed case.

$$Precision = P = \frac{TP}{TP+FP}$$

$$Recall = R = \frac{TP}{TP+FN}$$

One can err on one side or the other: if one decreases the number of false negatives to ensure that it is less likely to miss an actual value, or one can reduce the number of false positives at the cost of misses. One can "fine tune" the detection algorithms. One way to tune is the *F-measure*, which is the *weighted Harmonic mean* of Precision and Recall. For example, the F-measure [15], is defined by Harmonic mean of P and R,

$$F = \frac{1}{\frac{\alpha}{P} + \frac{1-\alpha}{R}} \quad or \quad F = \frac{PR}{(1-\alpha)\,P + \alpha\,R}$$

where $\alpha \in [0, 1]$. The weight α can be fine-tuned by decreasing the missed at the cost of increasing the false alarms. The *default* balanced F-measure is with equally weighted precision and recall, which means making $\alpha = 1/2$ making

$$F = \frac{2PR}{P+R}$$

B.2 Metrics for Dimensionality Reduction SVD

Based on k<<n, k highest eigenvalues out of n eigenvalues are selected to analyze dimensionality reduction. Recall A is the original matrix; V is the matrix whose columns are eigenvectors of A^TA. There are three equivalent ways to measure errors; experimental simulations verify this and the theorems confirm it. The value of k is determined as follows: First by dropping near zero eigenvalues, the error is the based on the dropped eigenvalues λ_p, p=k+1…n. Second since corresponding eigenvector are dropped, the projection in eigenspace becomes AV and projection space constrained to k dimensions becomes newAV. Thirdly, the projection translated in the base space becomes newA.

Error is measured in three ways:

$\frac{\Sigma p=k+1,n\ \lambda p}{\Sigma p=1,n\ \lambda p}$ relative error in the eigenvalues

$\frac{|AV-newAV|}{|AV|}$ relative error in projection space.

$\frac{|A-newA|}{|A|}$ relative error in the original space

For logical data, the data is coerced to numerical integer data, thus quantization is performed before error checking.

V. IMPLEMENTATION AND EXPERIMENTATION

The SVD algorithm, implemented in Matlab, is enhanced by prioritizing the directions of eigenvectors produced by the Matlab svd function. In addition to entropy heuristic used in the Weka tree construction algorithm J48, we updated it to include svd as part of the tree construction. The metrics in section 3 are used to benchmark the overall accuracy obtained in the confusion matrix.

Benchmark data on Wine dataset is obtained from UCI [16] where it has been extensively studied. Besides the classic site, there are numerous sites where the same data sets are available. For our purposes data is ready to use for analysis. Many times it may still not be practical to use directly due to the data size and computational bottlenecks. Data customization may need to be performed, e.g. data reduction in size and removal of attributes irrelevant to classification. If data gets transformed, it may be mapped in the original space for analysis or the analysis may be performed in the transformed domain.

We experimented it both ways and found that either

ISSN 2395-8618

way the algorithm performs better that brute force decision tree algorithm. We show results of experiments in Table 2 and Table 3. Our purpose is to show the improvement in reduced computations of the hybrid algorithm on classification without loss of accuracy.

V. WINES DATASET

A predictive model on Wine dataset is useful to provide guidance to vineyards regarding quality and price expected on their produce without heavy reliance on volatility of wine tasters [16]]. Two datasets are available, of which one dataset is on red wine and has 1599 different varieties. All wines are produced in a particular area of Portugal. Dataset has 12 attributes: (fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, alcohol, quality). Quality is based on sensory data and the rest are function of chemical properties of the wines. All 11 chemical properties of wines are real variables, whereas Quality is an ordinal variable with possible ranking from 1 (worst) to 10 (best). Since Weka J48 uses non-numeric categorical attribute for classification, we replaced the numbers 1 to 10 by number strings One, Two, Three, Four, Five, Six, Seven, Eight, Nine, Ten.

Before we use our algorithm, we analyze by using standalone Weka algorithm J48, classification model is constructed and confusion matrix for accuracy calculated for baseline comparison. We used SVD to determine the eigenvalues and found that the ratio of two smallest eigenvalues to largest eigenvalue is .0004 (all 11 eigenvalues are 60.573958, 9.433888, 6.540730, 1.550106, 1.338038, 0.346191, 0.188387, 0.148977, 0.101789, 0.040841, 0.024550). We decided to drop the smallest two eigenvalues out of 11 eigenvalues and apply the classification to the reduced dataset. The result show the there is 18% reduction in data with no loss of accuracy, in fact accuracy improved slightly. This may account for noise reduction. Computation is also speeded up by applying the algorithm in the compressed space with retaining slightly better accuracy because conversion to base space is eliminated. Thus we reduced the data by 18% and applied the decision tree algorithm in two ways: in original space and in compressed space. The results are shown as confusion matrix in Table 2 and Table 3

The latent variables do not tell much about the original variables. Prediction of Quality ranking from the chemical properties of the wines can be inferred from the latent variables directly.

TABLE 2. CONFUSION MATRIX FOR DECISION TREE GENERATED BY J48 ALGORITHM

Precision	Recall	F-Measure	Class
0.2	0.111	0.143	Eight
0.492	0.472	0.482	Seven
0.609	0.635	0.622	Six
0.71	0.72	0.715	Five
0.054	0.038	0.044	Four
0.167	0.1	0.125	Three
0.612	**0.622**	**0.616**	**weighted average**

TABLE 3. DATA REDUCTION WITH SVD AND J48 ALGORITHM IN RAW, REDUCED, AND TRANSFORMED DOMAIN.

	PRECISION	RECALL	F-MEASURE
Decision Tree J48 Raw data	0.612	0.622	0.617
Decision Tree in Original domain after 18% data reduction	0.615	0.625	0.620
Decision Tree in reduced domain after 18% data reduction	0.617	0.626	0.621

VI. CONCLUSION

We have given a hybrid algorithm that takes advantage of reducing data by dropping near zero eigenvalues and applying classification algorithm on latent variables. We applied the hybrid algorithm to a well-known benchmark dataset of a collection of Wines. For comparison the classification algorithm is applied in both the full base space and reduced eigenspace. We have shown that hybrid algorithm outperforms the usual standalone algorithm applied to data reduction and classification on their own for the intended tasks. Though accuracy result shown by confusion matrices are comparable, but the reduction in computational effort and storage space is significant. The application developers working in this area will find it useful in real time computations.

VII. REFERENCES

[1] Kirk Baker, Singular Value Decomposition Tutorial https://www.ling.ohiostate. edu/~kbaker/pubs/Singular_Value_Decomposition_Tutorial.pdf , January 2013.

[2] Jonthan Shlens A Tutorial on Principal Component Analysis, arXiv:1404.1100 [cs.LG], pp. 1-15,2014]

[3] Karen Bandeen-Roche Nov 28, 2007, An Introduction to Latent variable Models, http://www.biostat.jhsph.edu/~kbroche/Aging/Intro to Latent VariableModels.pdf

[4] Jim Hefferon, Linear Algebra, Free Book, http://joshua.smcvt.edu/linearalgebra, 2014.

[5] Abdi, Hervé, Beaton, Derek, Principal Component and Correspondence Analyses Using R, Springer, ISBN 978-3-319-09256-0, Digitally watermarked, DRM-free, 2017

[6] Suresh kumar Gorakala, Principal Component Analysis in R, Percept Psychology (2016)78:3-20

[7] Caroline J Anderson, Psychology Lecture Notes: Principal Component Analysis 2017

[8]Sebastian Raschka Principal Component Analysis in 3 Simple Steps LSA-Least Squares Approximation http://sebastianraschka.com/Articles/2015_pca_in_3_steps.html, 2015.

[9] Jure Leskovec, Anand Rajaraman, Jeffrey D Ullman, Datamining of Massive Datasets, 2014

ISSN 2395-8618

[10] Patrick J.F. Groenen, Michel van de Velden, Multidimensional Scaling, Econometric Institute EI 2004-I5, Erasmus University Rotterdam, Netherlands, 2015.

[11] An SVD-Entropy and Bilinearity based product ranking algorithm using heterogeneous data, Journal of Visual Languages & Computing, https://doi.org/10.1016/j.jvlc.2017.06.001, 2017.

[11] Christopher K. I. Williams 1998 - On a Connection between Kernel PCA and Metric, Multidimensional Scaling, Machine Learning, Volume 46, Issue 1–3, pp 11–19, January 2002.

[12] Rita Osadchy , Unsupervised learning 2011 DIMENSIONALITY REDUCTION: PCA, MDS http://www.cs.haifa.ac.il/~rita/uml_course/lectures/PCA_MDS.pdf

[13] Matthew Mayo, All machine learning models have flaws, http://www.kdnuggets.com/2015/03/all-machine-learning-models-have-flaws.html

[14] Matthew Mayo, Simplifying Decision Tree Interpretability with Python & Scikit-learn, 2017 http://www.kdnuggets.com/tag/decision-trees.

[15] Wikipedia, the free encyclopedia https://en.wikipedia.org/wiki/Precision_and_Recall

[16] Center for Machine Learning and Intelligent Systems, UCI Machine Learning Repository, http://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.data, https://onlinecourses.science.psu.edu/stat857/node/223, 2016.

VIII. APPENDIX A

The linear algebra definitions used here can be found in any textbook [4]. Briefly we describe in frequent terms definitions. We adopt the convention that the vectors are *column vectors* by default. The linear algebra concepts of *vector*, *transpose* of a vector, *scalar* product of a vector, Euclidean *norm or length* of a vector, *unit* vector, *sum* of two vectors, *dot product* of two vectors, *orthogonal* vectors, *Gram-Schmidt* orthogonalization, *matrix*, *square* matrix, *identity* matrix, *diagonal* matrix, *transpose* of matrix, *symmetric* matrix, *sum* of matrices, *product* of matrices, *inverse* of a matrix, *orthogonal* matrix, *norm* of a matrix, *orthogonal* matrix, *rotation* matrix, *rank* of a matrix, *determinant* of a matrix, *vector space*, and *basis* of a vector space, are standard terms in linear algebra. Additional terms that we use are an *eigenvector*, and an *eigenvalue*. By conventions of linear algebra, all vectors are column vectors unless categorically and specifically stated. For details on linear algebra, reader may consult references

For convenience in reading this paper, the notation necessary for readability is: matrices are described in *uppercase*, vectors are in *lowercase bold*, and the elements in matrices and elements in vectors are *lowercase italic*. For example,

Definition. If **u** and **v** are both *row* vectors or both *column* vectors, and **v** is a *unit* vector, then the scalar *projection* of **u** on **v** is given by **u•v** which is expressed as a matrix product
if **u** and **v** are column vectors, then **u•v** = $\mathbf{u}^T\mathbf{v}$ = $[u_1 \quad \dots \quad u_n]\begin{bmatrix} v_1 \\ \dots \\ v_n \end{bmatrix}$

Definition. The vectors u and v are *orthogonal* if **u•v = $\mathbf{u}^T\mathbf{v}$ = 0.**

Definition. A set of vectors is *orthonormal* if each vector is a unit vector, and any two different vectors are mutually orthogonal.

A matrix is a 2D array consisting of rows and columns. For a matrix A, we use the shorthand notation for matrix A = $[a_{ij}]$, where a_{ij} is the ij-th element of A. If A is a matrix, $\mathbf{a_{i\bullet}}$ is a row vector representing the i-th row of matrix A, and $\mathbf{a_{\bullet j}}$ is a column vector representing the j-th column of A**.** Thus ith row is $\mathbf{a_{i\bullet}}$ = [$a_{i1}, a_{i2}, \dots, a_{in}$]. Similarly jth column is $\mathbf{a_{\bullet j}} = \begin{bmatrix} a_{1j} \\ a_{2j} \\ \dots \\ a_{mj} \end{bmatrix}$.

Definition. The matrix A= $[a_{ij}]$ is said to be *symmetric* if $a_{ij} = a_{ji}$ for all i,j. That is, the elements across the main diagonal are identical, that is, A is equal to its transpose, A = A^T.

Definition. The norm of a matrix A is defined as the square root of the sum of squares of all elements in A. If A = $[a_{ij}]$ then $|A| = \sqrt{(\sum_{i,j} a_{ij}^2)}$

Definition. The trace of a matrix A is defined as the sum of entries on its main diagonal. If A = $[a_{ij}]$ then trace(A) = $\sum_i a_{ii}$

Definition. The matrix A is *orthogonal*, if the rows (and columns) are pairwise orthogonal, and $AA^T = A^TA = I$, the identity matrix.
There are three statistical terms associated with a matrix A: trace , rank, determinant,

Proposition. For a square matrix A,

$\text{trace}(AA^T) = \text{trace}(A^TA) = \text{trace}(A^2) = |A|^2$

$\text{trace}(AA^T) = \sum_i \mathbf{a_{i\bullet}}\, a_{i\bullet}^T = \sum_i \mathbf{a_{i\bullet}} \bullet a_{i\bullet}$

$\text{trace}(A^TA) = \sum_i \mathbf{a_{\bullet i}}^T a_{\bullet i} = \sum_i \mathbf{a_{\bullet i}} \bullet a_{\bullet i}$

$= \sum_i \sum_k a_{ik}\, a_{ik,} \quad = \sum_i \sum_k a_{ki}\, a_{ki,}$

In either case $= \sum_i \sum_k a_{ik}^2 = |A|^2$

It is the sum of squares of all the entries in A, $|A| = \sqrt{\text{trace}(AA^T)} = \sqrt{\text{trace}(A^TA)} = \sqrt{\text{trace}(A^2)}$

Definition. The *rank* of a matrix A is the number of linearly independent rows (and columns) in a matrix. It is equivalent to number of non-zero eigenvectors.

Definition. The *determinant* of a matrix A is denoted by det(A) and is defined as product of the eigenvalues of A.

Definition. For a matrix A, if there exists a non-zero vector **u** and a real number λ such that A**u** = λ **u**, then λ is called an *eigenvalue* and **u** is called the corresponding *eigenvector*. The term singular value and singular vecror are also used.
If λ is an eigenvalue, it is computed by solving the *characteristic* equation det(A- λ I)=0.
Note. Any non-zero multiple of a eigenvector is again an

ISSN 2395-8618

eigenvector. To make them unique, they are normalized to unit vectors, see Figure A1 (a). But if **u** is unit eigenvector, then –**u** is also a unit vector. In the literature, they use the convention of making the first non-zero component positive in the eigenvector, see Figure A1(b). Since eigenvectors are ordered, we propose to make the k-th element of k-th vector to be positive, see FigureA1(c) that makes the vectors look like a right handed system.

Theorem. If the matrix is *symmetric*, then for different eigenvalues, the eigenvectors are orthogonal.

Definition. The matrix A is diagonalizable if there is an invertible matrix V such that $A = V D V^{-1}$ where D is diagonal matrix of eigenvalues. If there are n distinct eigenvalues for n×n matrix, then it is diagonalizable.

Theorem. If the matrix is *symmetric or diagonalizable*, then there are as many eigenvectors as eigenvalues.

Proposition The eigenvalues of a real symmetric matrix are real.

Let $A\mathbf{u} = \lambda \mathbf{u}$, **u** is a unit vector.

$\lambda = \lambda(\mathbf{u},\mathbf{u}) = (\mathbf{u},\lambda\mathbf{u}) = (\mathbf{u},A\mathbf{u})$

$= \overline{(A\mathbf{u},\mathbf{u})} = \overline{(\lambda\mathbf{u},\mathbf{u})} = \overline{\lambda(\mathbf{u},\mathbf{u})}$

$= \bar{\lambda}$

Therefore $\lambda = \bar{\lambda}$, that is λ is equal to its complex conjugate, thus eigenvalues are real.

Definition. A matrix A is positive semi definite if $\mathbf{v}^T A\mathbf{u} \geq 0$ for all vectors **u,** and **v**.

Proposition. For a positive semi definite matrix A, eigenvalues must be non-negative.

Proof. If **u** is an eigenvector of A, then $A\mathbf{u} = \lambda \mathbf{u}$, that is, $\lambda = \mathbf{u}^T A\mathbf{u} \geq 0$, thus eigenvalues are non-negative .

Proposition. If A is a positive semi-definite matrix, then

• *Eigenvalues of AA^T and A^TA are non-negative*

Let λ be eigenvalue of A^TA and AA^T.

$\lambda = \lambda(\mathbf{v},\mathbf{v}) = (\lambda\mathbf{v},\mathbf{v}) = (A^TA\,\mathbf{v},\mathbf{v}) = (A\mathbf{v}, A\mathbf{v}) \geq 0$ and is the variance of A**v**

• *Eigenvalues of AA^T and A^TA are identical.*

Let λ be eigenvalue of A^TA.

If $A^TA\,\mathbf{v} = \lambda \mathbf{v}$, then $AA^TA\,\mathbf{v} = \lambda A\mathbf{v}$

Since $\mathbf{v} \neq 0$, $A\mathbf{v} \neq 0$, and $AA^T(A\mathbf{v}) = \lambda(A\mathbf{v})$

Therefore A**v** is an eigenvector of AA^T, hence λ is an eigenvalue of AA^T.

Similarly, if λ be eigenvalue of AA^T, then λ be eigenvalue of A^TA

• *Eigenvectors for different eigenvalues of AA^T and A^TA are orthogonal.*

Since AA^T and A^TA are symmetric, the eigenvectors are orthogonal.

Also, let **u**, **v** be eigenvectors and α, β be corresponding eigenvectors of AA^T and A^TA respectively. Then

$\alpha(\mathbf{u},\mathbf{v}) = (\alpha\,\mathbf{u},\mathbf{v}) = (AA^T\mathbf{u},\mathbf{v})$

$= (\mathbf{u}, AA^T\mathbf{v}) = (\mathbf{u},\beta\mathbf{v}) = \beta(\mathbf{u},\mathbf{v})$

$\alpha(\mathbf{u},\mathbf{v}) = \beta(\mathbf{u},\mathbf{v})$

if $\alpha \neq \beta$, $(\mathbf{u},\mathbf{v})$ must be zero.

• If $AA^T\mathbf{u} = \lambda\mathbf{u}$ and $A^T\mathbf{u} = \mu\mathbf{v}$, **v** is unit vector then $\mu = \sqrt{\lambda}$

$\lambda = \lambda(\mathbf{u},\mathbf{u})$

$= (\lambda\mathbf{u},\mathbf{u}) = (AA^T\mathbf{u},\mathbf{u})$

$= (A^T\mathbf{u}, A^T\mathbf{u}) = (\mu\mathbf{v},\mu\mathbf{v})$

$= \mu^2(\mathbf{v},\mathbf{v}) = \mu^2$

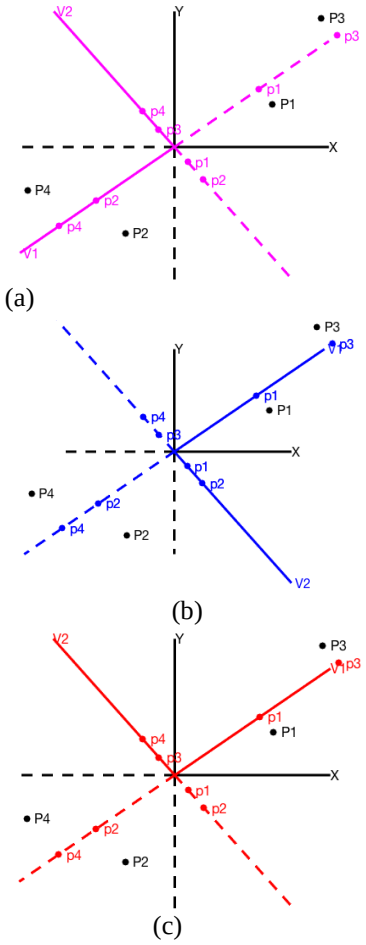


Figure A1. Matrix A representing four points P_1, P_2, P_3, P_4, axes are in black. V1 and V2 vectors resulted by using Matlab functions (a) Magenta: eig() function is used, we get arbitrary signs of vectors (b) Blue: svd() function is used and first component coerced to be positive for all eigenvectors, (c) Red: we organized it to look more natural right handed system, adaptively making the sequential component positive.

Proposition. For a symmetric matrix A, trace(A) = trace(UDU^T) = trace(D) = sum of eigenvalues of A, that is, $\text{trace}(A) = \sum_i \lambda_i$

Proof.

trace(A) = trace(UDU^T) =

$= \sum_i \boldsymbol{u_{i\bullet}} D \boldsymbol{u_{i\bullet}}^T$

$= \sum_i [\, u_{ik}\, \lambda_k] \boldsymbol{u_{i\bullet}}^T$ *where* $[\, u_{ik}\, \lambda_k]$ is a row vector with index k

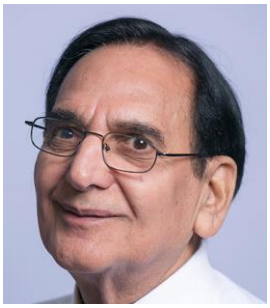
$= \sum_i \sum_k u_{ik} \lambda_k u_{ik}$

$= \sum_k \lambda_k \sum_i u_{ik} u_{ik}$

ISSN 2395-8618

$= \sum_k \lambda_k \boldsymbol{u}_{\bullet k} \bullet \boldsymbol{u}_{\bullet k}$ $\mathbf{u}_{\bullet k}$ is a unit column vector.
$= \sum_k \lambda_k$

Thus it shows that trace(A) is the sum of eigenvalues of A.



Chaman L. Sabharwal was born at Ludhiana, Punjab, India, in 1937. He received his B.A.(Hons) in 1959, M.A.(Math) in 1961 from Panjab University, Chandigarh, India. He received his M.S.(Math) in 1966 and Ph.D.(Math) from the University of Illinois, Urbana, Champaign, Illinois, USA, in 1967.

He is professor of Computer Science at Missouri University of Science and Technology (1986-). He was assistant professor (1967-971), associate professor (1971-9175), full professor (1975-1982) at Saint Louis University. He was Senior Programmer Analyst(1982), Specialist(1983), Senior Specialist(1984) Lead Engineer(1985) at Boeing Corporation. He published several technical reports and journal articles on CAD/CAM. He was consultant at Boeing (1986-1990). He was National Science Foundation fellow (1979) at Boeing and NSF Image Databases Panelist (1996).

Dr. Sabharwal has been member of American Mathematical Society, Mathematical Society of America, IEEE Computer Society, ACM, and ISCA. He has been on editorial board of International Journal of Zhejiang University Science (JZUS), Editorial Board CAD(Computer Aided Design), Progress In Computer Graphics Series, Modeling and Simulation, Instrument Society of America. He has been a reviewer for numerous books, journals and conferences. He was awarded service awards by NSF Young Scholars George Engelmann Institute, and ACM Symposium on Applied Computing for Multimedia and Visualization track. He is on program committees of several conferences including IEEEicSoftComm, MICAI, MIKE, MDS etc. including general Chair of MIKE2017.