Singular Value Decomposition Applied to Principal Component Analysis

David Clarkson, Stephanie Allen, Shayne O'Brien

State University of New York, College at Geneseo

dac17@geneseo.edu, saa7@geneseo.edu, sto2@geneseo.edu

FALL 2016

Background

- Singular Value Decomposition (SVD):
  $A = USV^t$
  A mxn, and U mxm, V nxn, S mxn
- Has myriad numerical applications, particularly in areas of data compression
- Principal Component Analysis (PCA):
  Data transformation of linearly correlated variables into linearly uncorrelated variables along axes with highest variance

Introduction: Singular Value Decomposition

- $A = U\Sigma V^t$
  A mxn, and U mxm, V nxn, S mxn
- U is composed of m *left singular vectors* $\boldsymbol{u}_i$
  V is composed of n *right singular vectors* $\boldsymbol{v}_i$
  Σ is rectangular diagonal with $Min(m, n)$ singular values σ_i
- Related to each other by:
  $A\boldsymbol{v} = \sigma\boldsymbol{u}$
  $A^t\boldsymbol{u} = \sigma\boldsymbol{v}$
- Also related to Eigenvalues and Eigenvectors:
  $\boldsymbol{v}_i$ are eiegenvectors of A^tA
  $\boldsymbol{u}_i$ are eiegenvectors of AA^t
  σ_i are the square roots of eigenvalues of A^tA and AA^t

Quick statistics review

- $\widetilde{X}$ is an mxn data matrix, with n variables and m observations.
- The new axes are the "principal components," which are the eigenvectors of the correlation matrix

Introduction: Principal Component Analysis

- PCA transforms a data set into a system with zero covariance-i.e. one that is uncorrelated.
- The new axes are the "principal components," which are the eigenvectors of the correlation matrix

Introduction: Applications

Applications of SVD with PCA include:

- Machine learning
- Stock Market data analysis
- Characterization of political positions
- Examining entanglement in quantum computation

We focused on:

- Dimensionality reduction of United Nations Development Programme data
- Image compression

Procedure: SVD Generalized Algorithm

- Assume $m \geq n$
- S Matrix (mxn): Find eigenvalues of A^tA, sort in descending order, and then take the square roots to get the singular values. Then create an mxn matrix of zeros and put singular values on the diagonal
- V Matrix (nxn): Find the *right eigenvectors* of A^tA, sort them as the eigenvalues were sorted, and these vectors will become the columns of V. Transpose V to get V^t

Procedure: SVD

- U Matrix (mxm):
  - Use the formula $u_i = \frac{1}{s_i} A v_i$ for $i = 1, 2, ..., k$
  - Concatenate these k columns with the mxm identity matrix, I_{mxn}
  - Perform the Gram-Schmidt procedure on the concatenated matrix to produce a set of m orthogonal column vectors
  - During the process of orthogonalizing each column v to the previous columns, use the partially orthogonalized v vector. This increases the accuracy of the algorithm and is known as the Modified Gram-Schmidt procedure
  - Remove zero columns from the concatenated matrix
  - Normalize the m columns

Procedure: Image Compression

SVD is designed to be applied to matrices. This is a problem because images in MATLAB are 3D arrays consisting of the red, green, and blue values. We can modify our code to accommodate for this by separating and then applying SVD on each layer of the array in turn.

Procedure: Image Compression

In doing this, there are a few considerations:

- The number of singular values that we will maintain, k, must be the same for all three layers of RGB
- Error is computed three times: one for each layer of the array. We average these values to get an estimate of aggregate error
- MATLAB requires RGB values to be in [0,1]

Procedure: PCA with SVD

- Center the set of data A by subtracting the mean of each column, $\bar{a}_i$, from each variable, $\boldsymbol{a}_i$, and standardize it by dividing each column by its standard deviation, s_i, to create the new matrix $X = [\frac{a_i - \bar{a}_i}{s_i}]$
- Select the desired number of dimensions q and in turn take q column vectors from V. Project X onto the first two principal components to create feature vectors $F = XV$, which are each observation's scores on the first two principal components. This reduced our data set from four to two dimensions, at the cost of losing information.

Procedure: PCA with Covariance Matrix

A similar procedure was followed for the covariance matrix method.

- The sole difference between these two tests was that instead of calculating the right singular vectors of X using SVD, the eigenvectors of the covariance matrix $\frac{XX^t}{m-1}$ were found instead.
- The data was then treated identically in that it was projected onto the first two principal components, separated, plotted, and then compared.

Considerations: Covariance Versus SVD for PCA

- Principal Component Analysis relies on each variable to be measured using the same units. If this is not the case, variables with greater variance will be given more weight during and relationships between smaller variables will not be accurately reflected or captured. This problem can be avoided by standardizing each variable before applying PCA.

- Furthermore, for $m \times n$ data matrices in which m is significantly greater than n, it is less computationally expensive to use the PCA algorithm in which the eigenvectors of the covariance matrix are found. However, the SVD method is more numerically stable. While neither of these points affected the quality of our results, the difference could be sizable for tasks requiring very high precision or working with very large data sets.

Results: Image Compression

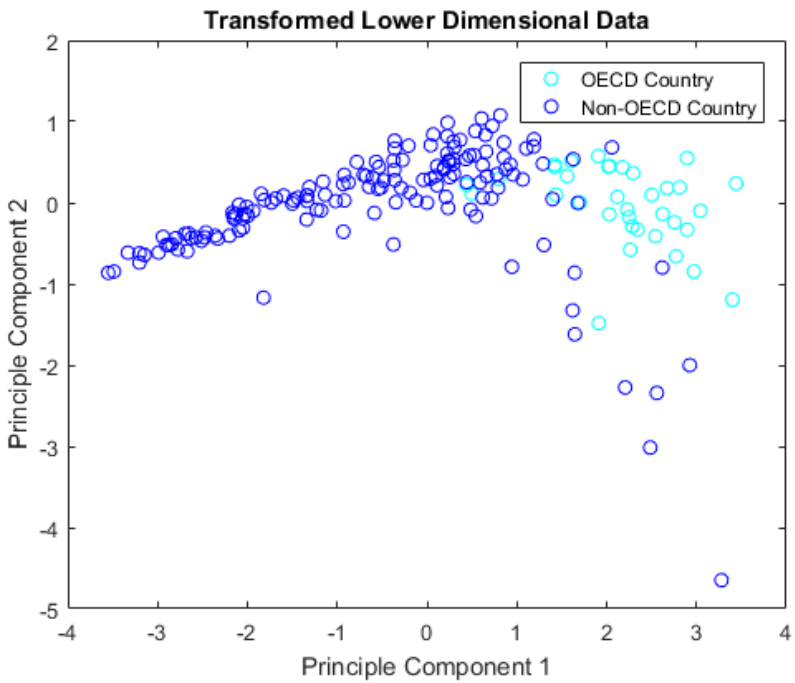
Table: Error for SVD Image Compression

Matrix	l_2**-norm error**
Red	$1.962103 * 10^{-8}$
Green	$3.202683 * 10^{-8}$
Blue	$1.616040 * 10^{-8}$
Average	$2.260275 * 10^{-8}$

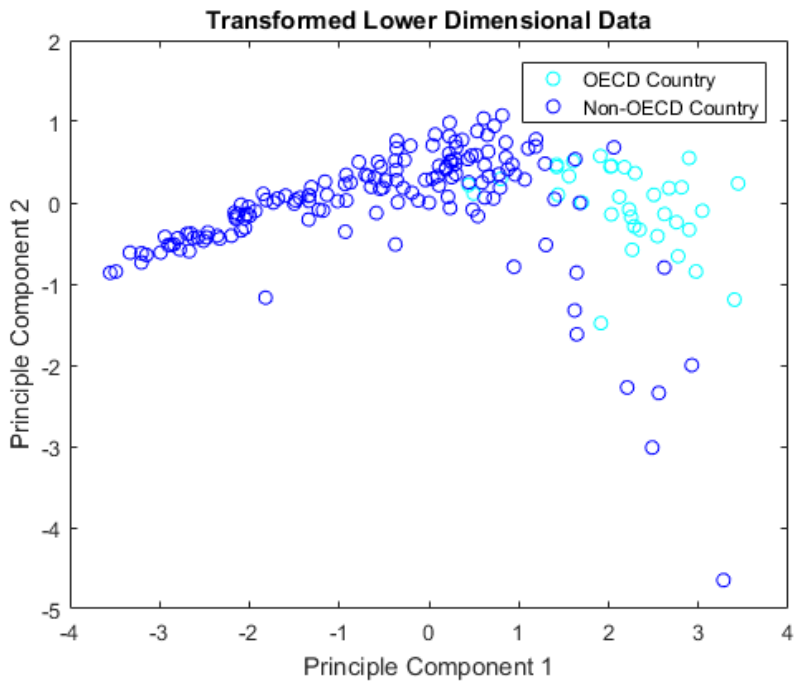
Results: Image Compression

Figure: Image Compression for a .JPG image of the *Canis lupus familiaris*, Tucker.

Results: PCA



(a) Covariance Matrix PCA Graph



(b) SVD PCA Graph

Figure: PCA score graphs produced by two separate methods. Notice that there is no discernible difference between the two.

Principal Components

Table: Principal Components of SVD Applied to PCA

Variable	PC 1	PC 2
GNI Per Capita	0.447326	-0.888765
Life Expectancy	0.505390	0.180040
Mean Years of Schooling	0.513468	0.340196
Expected Years of Schooling	0.529935	0.248894

Table: Singular Values of A

Singular Values
1.73891276816121
0.70456665802325
0.551531756251022
0.419023783304168

Error Calculations

Table: Error Calculations for Covariance versus SVD PCA Methods

Method	**Absolute Error**	**Relative Error**
SVD Applied to PCA	7.542083	0.317171
PCA with Covariance Matrix	7.542083	0.317171

Conclusions

We have shown two major applications of SVD and PCA:

- Reducing storage requirement for images
- Reducing the number of variables in data

Given more time, we would look for a better data set to represent the difference between SVD applied to PCA and using the covariance matrix of A to do PCA. We would also implement numerical techniques to further reduce dimensionality.

Any Questions?