COL 380	Mar 6, 2019
Lab 2	
Instructor: Subodh Sharma	**Due: Mar 16, 23:55 hrs**

1 Principal Component Analysis (PCA)

Principal component analysis, or PCA for short, is a mathematical procedure that transforms a number of (possibly) correlated variables into a smaller number of uncorrelated variables called principal components. The objective of principal component analysis is to reduce the dimensionality (number of features) of the dataset, but retain as much of the original variability in the data as possible. The first principal component accounts for the majority of the variability in the data, the second principal component accounts for the majority of the remaining variability, and so on.

PCA can be thought of as a projection method where data with n-features is projected onto a subspace with n (or fewer)-featuress, while retaining most of the information.

The article `https://towardsdatascience.com/a-one-stop-shop-for-principal-component-analysis-5582fb7e0a9c` has a nice intuitive explanation of PCA. The article `http://setosa.io/ev/principal-component-analysis/` contains some interesting examples.

Computing PCA

1. Input a dataset D with s samples of f features, *ie* dimension of D is s x f.
2. Obtain the Eigenvectors and Eigenvalues by performig *Singular Vector Decomposition* of D^T (explained below).
3. Sort eigenvalues in descending order and choose the k eigenvectors that correspond to the k largest eigenvalues. Here, k is the number of dimensions of the new feature subspace ($k \leq f$)
4. Construct the projection matrix W of principal components from the selected k eigenvectors.
5. Transform the original dataset D via W to obtain a k-dimensional feature subspace $\widehat{D}$.

2 Singular Value Decomposition (SVD)

The Singular Value Decomposition, or SVD for short, is a matrix decomposition method for reducing a matrix to its constituent parts in order to make certain subsequent matrix calculations simpler.

SVD of a matrix M is the factorization of M into the product of three matrices $M = U\Sigma V^T$ where the columns of U and V are orthonormal and the matrix Σ is diagonal with positive real entries. Given that M is a real m x n matrix that we wish to decompose, U is a m x m matrix, Σ is a m x n diagonal matrix, and V^T is the transpose of a n x n matrix.

SVD is the most well known and widely used matrix decomposition method. All matrices have an SVD, which makes it more stable than other methods, such as the eigendecomposition. As such, it is often used in a wide array of applications including compressing, denoising, and data reduction. Read the article `https://blog.statsbot.co/singular-value-decomposition-tutorial-52c695315254` for more information on SVD.

Computing SVD

1. Compute the Hermitian (conjugate transpose), M^T, of a matrix M.
2. Compute multiplication $M^T.M$.
3. Compute the eigenvalues of $M^T.M$ and sort them in descending order (in the absolute sense).
4. Square root the eigenvalues to obtain the singular values of M.
5. Construct diagonal matrix Σ by placing singular values in descending order along its diagonal. Also compute Σ^{-1}
6. Use the ordered eigenvalues and compute the eigenvectors of $M^T.M$.
7. Place these eigenvectors along the columns of a matrix to obtain the matrix V. Compute its transpose, V^T.
8. Compute the matrix U as $U = MV\Sigma^{-1}$.

SVD computation example

$$\text{let } M = \begin{bmatrix} 4 & 0 \\ 3 & -5 \end{bmatrix}$$

$$\text{since, } M^T = \begin{bmatrix} 4 & 3 \\ 0 & -5 \end{bmatrix} \text{ thus, } M^T.M = \begin{bmatrix} 24 & -15 \\ -15 & 25 \end{bmatrix}$$

$$\text{eigenvalues: } |M^T.M - \lambda I| = \begin{bmatrix} 25-\lambda & -15 \\ -15 & 25-\lambda \end{bmatrix}$$

$$= \lambda^2 - 50\lambda + 400, \text{ thus, } \lambda_1 = 40, \lambda_2 = 10$$

$$\text{Singular values: } \sigma_1 = \sqrt{40} = 6.3245, \ \sigma_2 = \sqrt{10} = 3.1622$$

$$\text{Thus, } \Sigma = \begin{bmatrix} 6.3245 & 0 \\ 0 & 3.1622 \end{bmatrix}, \text{ and, } \Sigma^{-1} = \begin{bmatrix} \frac{1}{6.3245} & 0 \\ 0 & \frac{1}{3.1622} \end{bmatrix} = \begin{bmatrix} 0.1581 & 0 \\ 0 & 0.3162 \end{bmatrix}$$

$$\text{For } \lambda = \lambda_1 = 40, \ M^T.M - \lambda I = \begin{bmatrix} 25-40 & -15 \\ -15 & 25-40 \end{bmatrix} = \begin{bmatrix} -15 & -15 \\ -15 & -15 \end{bmatrix}$$

$$\text{Compute } X_1 \text{ as: } (M^T.M - \lambda_1 I).X_1 = 0, \text{ i.e.}$$

$$\begin{bmatrix} -15 & -15 \\ -15 & -15 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \text{ thus, } X_1 = \begin{bmatrix} 0.7071 \\ -0.7071 \end{bmatrix}$$

$$\text{Similarly, for } \lambda = \lambda_2 = 10, \text{ compute } X_2 = \begin{bmatrix} 0.7071 \\ 0.7071 \end{bmatrix}$$

$$V = \begin{bmatrix} X_1 & X_2 \end{bmatrix} = \begin{bmatrix} 0.7071 & 0.7071 \\ -0.7071 & 0.7071 \end{bmatrix} \text{ and } V^T = \begin{bmatrix} 0.7071 & -0.7071 \\ 0.7071 & 0.7071 \end{bmatrix}$$

$$\text{Thus, } U = MV\Sigma^{-1} = \begin{bmatrix} 0.4472 & 0.8944 \\ 0.8944 & -0.4472 \end{bmatrix}$$

Correctness Argument
Multiplying the matrices U, Σ and V^T gives back the original matrix M, i.e. $M = U\Sigma V^T$

For the example above,

$$U\Sigma V^T = \begin{bmatrix} 0.4472 & 0.8944 \\ 0.8944 & -0.4472 \end{bmatrix} \begin{bmatrix} 6.3245 & 0 \\ 0 & 3.1622 \end{bmatrix} \begin{bmatrix} 0.7071 & -0.7071 \\ 0.7071 & 0.7071 \end{bmatrix} = \begin{bmatrix} 3.9998 & 0 \\ 2.9999 & -4.9997 \end{bmatrix} \approx M$$

PCA computation example Consider the "Iris" dataset deposited on the UCI machine learning repository (`https://archive.ics.uci.edu/ml/datasets/Iris`). The dataset contains 150 instances of Iris plants with 4 features.

Step 1 The dimension on input dataset D = 150 x 4 ($s = 150$, $f = 4$)

Step 2 Computing SVD of D^T,

$$\Sigma = \begin{bmatrix} 2.9108 & 0 & 0 & 0 & ... \\ 0 & 0.9212 & 0 & 0 & ... \\ 0 & 0 & 0.1473 & 0 & ... \\ 0 & 0 & 0 & 0.0206 & ... \end{bmatrix} \text{ and, } U = \begin{bmatrix} -0.5223 & -0.3723 & 0.7210 & 0.2619 \\ 0.2633 & -0.9255 & -0.2420 & -0.1241 \\ -0.5812 & -0.0210 & -0.1408 & -0.8011 \\ -0.5656 & -0.0654 & -0.6338 & 0.5235 \end{bmatrix}$$

Step 3
Eigenvalues in descending order: $2.9108 > 0.9212 > 0.1473 > 0.0206$.
Sum of eigenvalues = 4
Explained variance of eigenvalues:
2.9108/4 = 0.7277, 0.9212/4 = 0.2303, 0.1473/4 = 0.0368 and 0.0206/4 = 0.0051.

It is clearly seen that most of the variance (72.77% of the variance to be precise) can be explained by the first principal component alone. The second principal component still bears some information (23.03%) while the third and fourth principal components can safely be dropped without losing too much information. Together, the first two principal components contain 95.8% of the information.
Hence, the reduced dimension, $k = 2$.

Step 4 Although, the name "projection matrix" has a nice ring to it, it is basically just a matrix of the concatenated top k eigenvectors (columns of U). The top k eigenvectors represent the principal components.

$$W = \begin{bmatrix} -0.5223 & -0.3723 \\ 0.2633 & -0.9255 \\ -0.5812 & -0.0210 \\ -0.5656 & -0.0654 \end{bmatrix}$$

Step 5 Use the 4 x 2-dimensional projection matrix W to transform the samples onto the new subspace via the equation, $\widehat{D} = D$ x W. The resulting $\widehat{D}$ is a 150 x 2 matrix of transformed samples.

3 QR Algorithm for eigenvalues and eigenvectors

QR algorithm is one of the most important algorithm for eigenvalue computations, computed as:

1. $D_0 = D$

2. $E_0 = I$ /*Eigenvector*/
3. Until convergence:
   (a) Q_i, R_i = QRfactors(D_i) /*QR factorization*/
   (b) $D_{i+1} = R_i.Q_i$
   (c) $E_{i+1} = E_i.Q_i$
4. eigenvalues = diagonal of D_∞
5. eigenvectors = columns of E_∞

QR factorization (Step 3(a)) can be computed using *Gram-Schmidt process, Householder transformation* or *Givens rotation*. You can refer the document at `http://www.cs.nthu.edu.tw/~cherung/teaching/2008cs3331/chap4%20example.pdf`. The wikipedia page on QR decomposition is also a good stating point (`https://en.wikipedia.org/wiki/QR$_$decomposition`).

4 Lab submission details

4.1 Problem Statement

- Your task is to implement a parallel PCA algorithm over rectangular datasets using OpenMP.
- You will be provided a dataset, D (such that #samples $\geq$ #features, *ie* $s \geq f$), and minimum percentage data retention, R. You will be required to perform parallel PCA through parallel SVD, as follows:
  1. Perform SVD of D^T. Return U, Σ and V^T.
  2. Recognize the minimum value of dimension k such that the sum of top k eigenvalues is $\geq R$. Return k.
  3. Transform D to $\widehat{D}$. Return $\widehat{D}$.

4.2 Correctness check

- Multiplication of $U.\Sigma.V^T$ must give back D^T.
- Your returned k must match our correct k.
- Provided that you have computed k correctly, your matrix $\widehat{D}$ must match our corresponding matrix.

Note: We will keep it in consideration that real values may not be exact matches. We will accept values within a $\leq$0.001 error tolerance.

4.3 Input format

The input file format and the main file to calculate the time taken by your implementation can be cloned from `https://github.com/dvynjli/col380_lab2_suite`. Read the `README` file for details.

4.4 Plagiarism policy

Make sure the code is not plagiarized. We have a repository of online codes. If you are found copying code from online sources or your peers, you will be penalized with a grade-drop + zero in the lab.

4.5 Late submission policy

10% penalty for each day over the deadline. Maximum of 2 days allowed after which summary zero will be awarded (unless there is a medical emergency – for which you will have to provide a letter from the doctor).

4.6 Word of advice on time management

We believe that this will be an involving task. You are strongly advised to start working on the assignment at the earliest possible.

4.7 Evaluation scheme

1. Correct computation of U, Σ, V^T	**45 Marks**
2. Correct computation of k	**eligibility for component 3**
3. Correct computation of $\widehat{D}$	**25 Marks & eligibility for component 4**
4. Performance	**30 Marks**

Note: The grading for component 4 (*Performance*) will be relative.
The grading of components 1, 2 and 3 will be absolute.