Image Compression using SVD & Dimensionality Reduction using PCA

Sainadh Chilukamari (sc249)
Project-2 Report - Advanced Algorithms

INTRODUCTION

The first goal of this project is to understand and use the concept of Singular Value Decomposition to compress the image, which reduces the size of the original image. The second goal of this project is to understand the concept of PCA (Principal Component Analysis) and use it on high dimensional space to reduce the dimensionality of the data.

SINGULAR VALUE DECOMPOSITION (SVD)

Singular value decomposition is a factorization of a real or complex, square or non-square matrix. The main idea of SVD is to take a high dimensional data, and reducing it to a lower-dimensional space (Eg: In the case of images, it reduces the size of the image which is called as an image compression). If we consider a matrix A (m rows * n columns) and with rank k, then matrix A can be decomposed into three matrices: $\mathbf{A = USV^T}$.

$$A = \begin{bmatrix} u_1 & \cdots & u_r & \cdots & u_m \end{bmatrix} \begin{bmatrix} \sigma_1 & & & & \\ & \ddots & & & \\ & & \sigma_r & & \\ & & & \ddots & \\ & & & & 0 \end{bmatrix} \begin{bmatrix} v_1^T \\ \vdots \\ v_r^T \\ \vdots \\ v_n^T \end{bmatrix}$$

- **U** is an m * m orthogonal matrix.
- $\mathbf{V^T}$ is the conjugate transpose of the n * n orthogonal matrix.
- **S** is an m * n diagonal matrix that has the singular values of A. (decreasing order)

After decomposing the original matrix A then we reduce the image if size. To reduce it we will remove the singular values which are not important. Since S matrix is a diagonal matrix with singular values arranged in decreasing order, we can discard singular values with the chosen rank (k) of the matrix. The value of K is selected in such a way that the image quality is maintained with a good compression rate. The reduced matrix with K value will contain K singular values from sigma, m* K elements from U, and n* K elements from V.

When performing image compression using SVD, the sum is not performed to the very last Singular Values (SV's). The values which are outside the value of K are equated to zero. The total storage for A_k will be: $A_k = 2 * k(m + n + 1) + 6$. The image corresponding to A_k will still have a very close resemblance to the original image.

PRINCIPAL COMPONENT ANALYSIS (PCA)

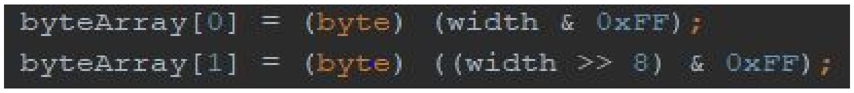
Principal component analysis (PCA) is a dimensionality reduction technique for a data consisting of many variables correlated with each other while retaining the variation present in the data (extracting important information from a high-dimensional space and projecting it into a lower-dimensional space). When the data is projected into a lower dimension from a higher space, the dimensions on which the original data points projected are called the Principal Components (PCs).

Principal components have both direction and magnitude. Direction represents across which principal axes the data is mostly spread out. Magnitude signifies the amount of variance that Principal Component captures. Principal components are orthogonal and ordered such that the retention of variation present in the original variables decreases as we move down in the order. So, the 1st PC retains the maximum variation that was present in the original components. The principal components are the eigenvectors of a covariance matrix, and hence they are orthogonal.

ALGORITHMS DESCRIPTION

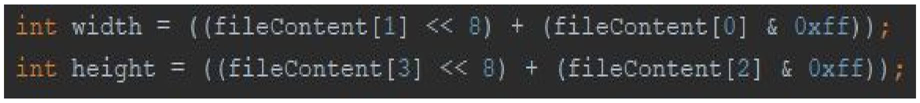
Programming Language: All the algorithms are written in Java (Except PCA - which was written in Python).

Ascii file to binary file: We have used an array of bytes data structure to store all the values in the byte array and then write it directly to the output file (binary file). The array of bytes is an appropriate data structure to write and read easily from a binary file. Firstly, 2 bytes are used to save the width of the image and 2 bytes to save the height of the image and one byte for the greyscale levels and then one byte was used for each pixel. The size of the byte array equals w*h+2+2+1. The below figure shows how the width value was stored in 2 bytes. Finally, all the other values are also stored in 1 byte.



```
byteArray[0] = (byte) (width & 0xFF);
byteArray[1] = (byte) ((width >> 8) & 0xFF);
```

Binary file to Ascii file: The binary file which was converted using the above algorithm was read as an array of bytes. The first four bytes are converted into integers and wrote it as the width and the height of the image. Similarly, all the other values which are stored in one byte are converted into integers and wrote it to the output file. The below figure shows how the byte values of width are height is converted back to integers. Finally, the resultant file was a copy of the original file.



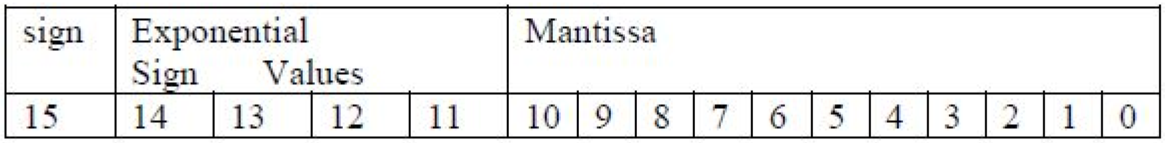
```
int width = ((fileContent[1] << 8) + (fileContent[0] & 0xff));
int height = ((fileContent[3] << 8) + (fileContent[2] & 0xff));
```

SVD Compression: Below are the steps that are used to decompose a matrix and then compress the image file.

- Decompose a matrix A into **USV^T.**
- Set the Singular values in decreasing order and accordingly adjust the other matrices.
- Select the desired rank - remove redundant information.
- Truncate the matrices based on the k value

Compress SVD double to SVD binary with K value:

- Read each value and convert it to a 16-bit format.
- We have used the IEEE standard 754-2008 half-precision floating-point (binary16) as our storing strategy.
- Then take each value store the information into a file by using the byte array as the 1st algorithm (Ascii to binary).



sign	Exponential Sign	Values			Mantissa										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Recover Image: Read the SVD file which was saved by using the above algorithm and the decode two-byte values as did in 2nd algorithm (Binary to Ascii).

Error Calculations: For this part, we have used two measures namely the Rate of compression (C_r) and Mean Square Error (MSE). For calculating the C_r and MSE we have used the original image, approximated image, and recovered the image.

<u>The rate of compression:</u>

((Size of the original image - Size of approximated image) / size of the original image)

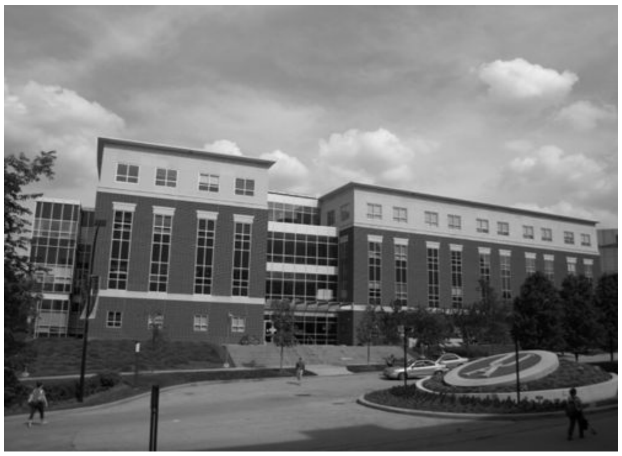
<u>Mean Square Error (MSE):</u> is the measure of degradation of compressed image quality as compared to the original image.

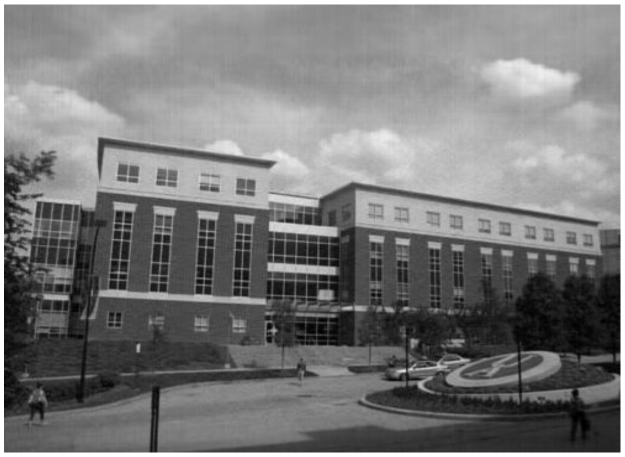
$$MSE = \Sigma\ \Sigma\ (\ (A(i,j) - A'(i,y))^2\ )\ /\ (m*n);$$

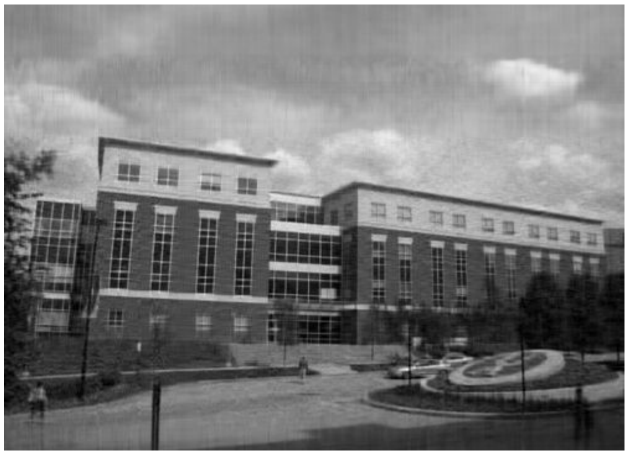
Principal Component Analysis: PCA was done in python by using the python library called sklearn.decomposition. Below steps explain how PCA was done and how principal components are extracted.

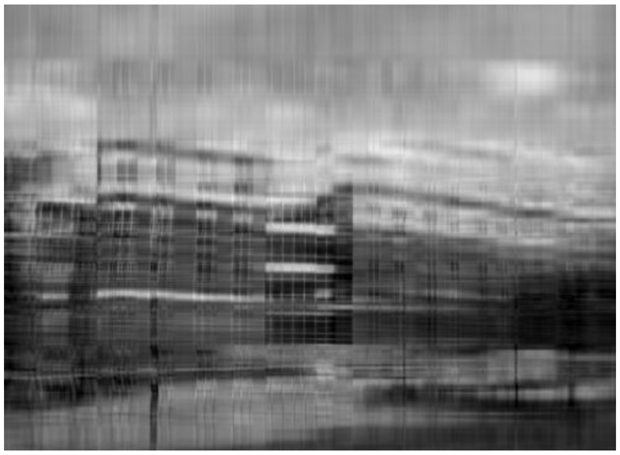
- Let A be m * n matrix, the first step is to find the Mean values for each variable and then matrix B was calculated (subtracted using mean value).
- The second step is to find Co-Variance Matrix S =B' * B/ (n-1) and then find eigenvalues and eigenvectors of S = VDV'
  - D is the diagonal matrix that contains eigenvalues of S.
  - V is the matrix that contains eigenvectors of S.
  - V' is the transpose of V and it is also the rotational matrix. Principle Components is the product of attributes and V.

Compressed Image using SVD Results: Converting a PGM file to a binary file and SVD part has been tested on grayscale CAS.pgm (500x376) and Baboon.pgm (512x512) with different ranks and we have calculated the rate of compression and Mean Square Error. Below are images of CAS.pgm of different k values. Observations are discussed in the next phase.


Original Image: CAS.PGM


Rank 100, MSE: 11.91%, CR:74.49%

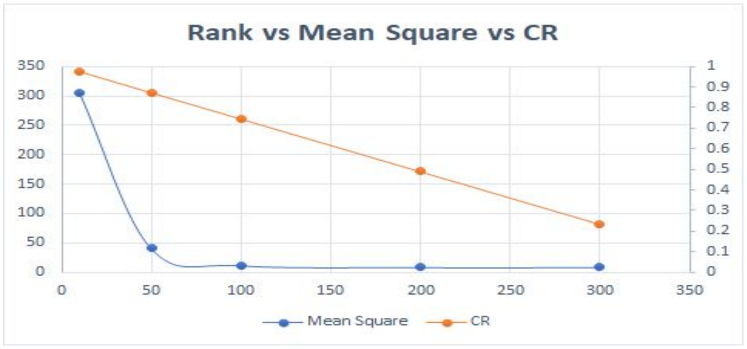




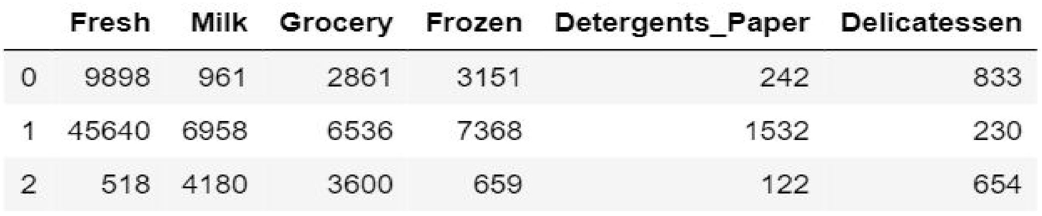
Rank 50, MSE: 40.84%, CR: 87.24% **Rank 10, MSE: 304.94%, CR: 97.44%**

Observation:

- The below graph shows the relation between k (rank) vs MSE and C_R. K represents the number of singular values (x-axis) used in the reconstruction of the compressed image.
- We came to know that smaller the value of k, more is the compression ratio but image quality decreases.
- As the value of k increases, image quality improves (MSE is small) and more storage space is required for an image $(2 * k(m + n + 1) + 6)$.



Wholesale Customers: This the dataset we have used for PCA. It consists of various customer's annual spending amounts (reported in monetary units) of diverse product categories for internal structure. Data has 440 samples composed of six important product categories: 'Fresh', 'Milk', 'Grocery', 'Frozen', 'Detergents_Paper', and 'Delicatessen'. The below figure shows the sample data points.

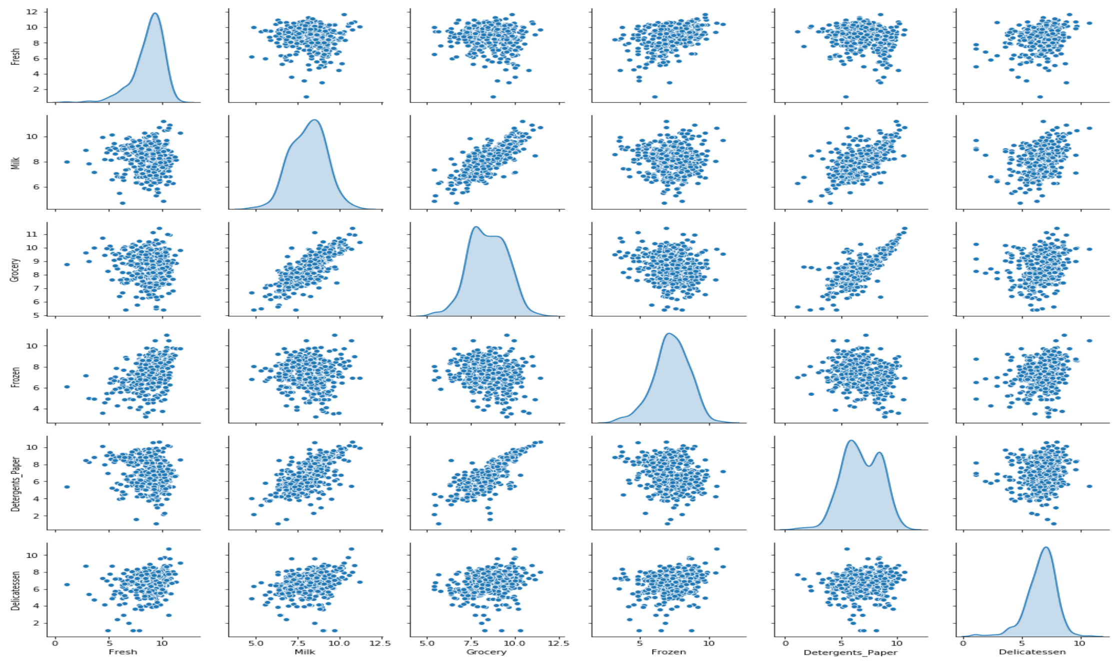


	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicatessen
0	9898	961	2861	3151	242	833
1	45640	6958	6536	7368	1532	230
2	518	4180	3600	659	122	654

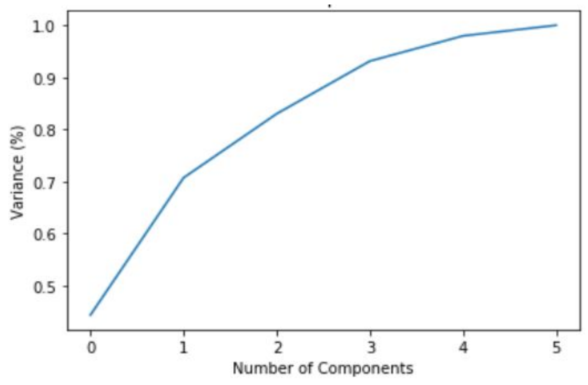
To check the correlation between the variables, we have tried to predict the 'Fresh' (also tried other variables) by using the remaining variables and resulted in very low accuracy. So, we have preprocessed the entire dataset to create a better representation of customers by performing a scaling on the data and detecting (and optionally removing) outliers.

- Feature scaling was done by using natural logarithm - this was done because if we take detergents_paper and delicatessen all the data points are along the axis of detergents_paper.

- Outliers do not add value to the algorithm (for eg: k-means) and make the algorithm perform worse in a few cases. An outlier step is calculated as 1.5 times the interquartile range (IQR). A data point with a feature that is beyond an outlier step outside of the IQR for that feature is considered abnormal.

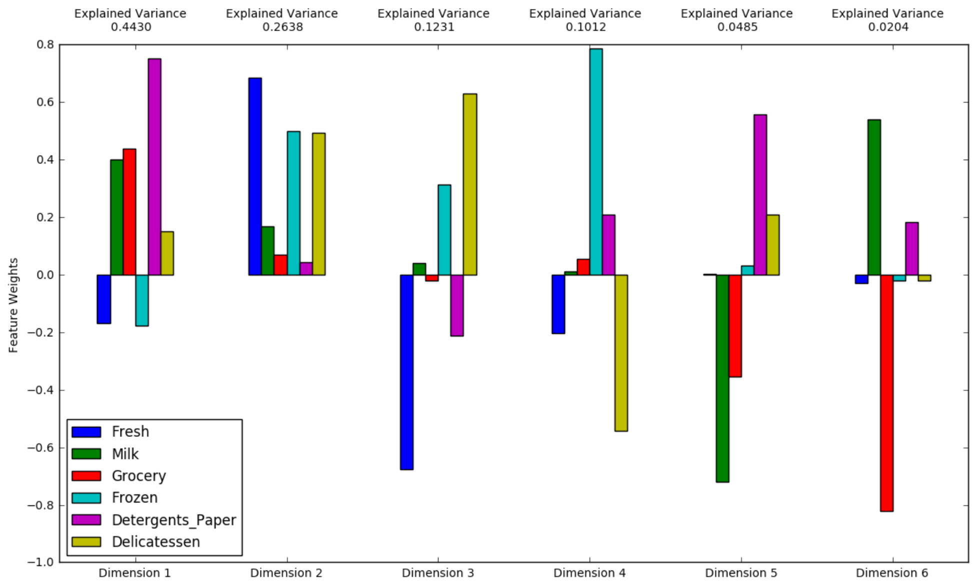


After scaling and removing the outliers, we are left with 435 good samples and now the data looks so better now when compared to earlier. The above figure tells that features now follow a normal distribution. Then PCA was applied to this data.

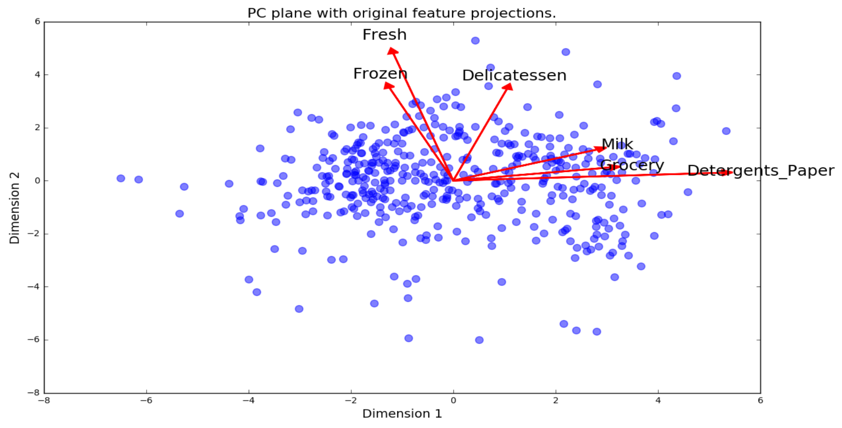


The above figure shows that the first three principal components explain 80% of the variance in the data. This means we can reduce the data from six dimensions to 3 dimensions with 20% of information lost. By adding the fourth PC we can achieve a 90% variance of the data.

- Dimension 1 has a high positive weight for Milk, Grocery, and Detergents_Paper features.
- Dimension 2 has a high positive weight for Fresh, Frozen, and Delicatessen.



- Dimension 3 has a high positive weight for Deli and Frozen features, and low positive weight for Milk, but has negative weights for everything else.
- Dimension 4 has positive weights for Frozen, Detergents_Paper, and Groceries, while being negative for Fresh and Deli.



From the above image, a point the lower right corner of the figure will likely correspond to a customer that spends a lot on 'Milk', 'Grocery' and 'Detergents_Paper', but not so much on the other product categories.

Contribution

- Converting Ascii to binary and reverse it and PCA analysis - Sainadh Chilukamari.
- Image compressing using SVD and testing: done by Saransh Bhalla.