3460:635 Advanced Algorithms

Project 2: Image compression using SVD and dimensionality reduction using PCA

Submitted by: Sejuti Banik (UANet ID:4724011)

11/21/2019

EigenValues and EigenVectors:

For mxm square matrix S, if $Sv=\lambda v$ then v: eigenvectors ($v\neq 0$ and $v\in R^m$) and λ: eigenvalue ($\lambda\in R$). For example, we need to solve $(S-\lambda I)\,v=0$ to determine the eigenvector v and $Det(S-\lambda I) = 0$.

The number of eigenvalues depend on the dimension of matrix. So if $S=\begin{pmatrix} 8 & 3 & 0 \\ 3 & 7 & 2 \\ 0 & 2 & 4 \end{pmatrix}$ then

$Det\begin{vmatrix} 8-\lambda & 3 & 0 \\ 3 & 7-\lambda & 2 \\ 0 & 2 & 4-\lambda \end{vmatrix}=0$ => $\lambda_1\approx 2.562$, $\lambda_2\approx 5.636$, $\lambda_3\approx 10.801$, and after calculations $v_1=\begin{pmatrix} 0.397 \\ -0.719 \\ 1 \end{pmatrix}$ $v_2=\begin{pmatrix} -1.039 \\ 0.818 \\ 1 \end{pmatrix}$ and $v_3=\begin{pmatrix} 3.642 \\ 3.401 \\ 1 \end{pmatrix}$

Rank: Rearranging row and columns, the dimension of the largest sub-matrix whose determinant $D\neq 0$ is the rank of the matrix. For example the rank of $\begin{pmatrix} 1 & 0 & -1 \\ 0 & 1 & 2 \\ 0 & 0 & 0 \end{pmatrix}$ is 2 because the submatrix $\begin{pmatrix} 0 & -1 \\ 1 & 2 \end{pmatrix}$ has $D\neq 0$

Singular Value Decomposition (SVD):

For a mxn linear matrix A having rank r, SVD $(A_{mxn}) = \mathbf{U}_{mxm}\,\mathbf{\Sigma}_{mxn}\,\mathbf{V^T}_{nxn}$. Here **U**, **Σ** and $\mathbf{V^T}$ are three very special matrices where

Columns of **U**: orthogonal eigenvectors of $A.A^T$

$\mathbf{\Sigma}=diag(\sigma_1\sigma_2\ldots\sigma_n)$ where $\sigma_i = \sqrt{\lambda_i}$, $\sigma_i \geq \sigma_{i-1}$ the singular values are in decreasing order

and eigenvalues $\lambda_1\ \lambda_2\ldots\lambda_r$ of $A.A^T$ and $A^T.A$ are same

Columns of $\mathbf{V^T}$: orthogonal eigenvectors of $A^T.A$

The method of compression through SVD is based on the assumption that all the Singular values in the **Σ** matrix are not equally significant. Some of them are important while the others are small and non-important. Keeping the significant values of σ and discarding the other values, and keeping the corresponding columns in **U** and rows in $\mathbf{V^T}$ is the low rank approximation. As we include more and more singular values and increase the rank, the quality and representation of the image compared to the original image improves.

Through SVD compression the goal is to obtain $A_k= \min\,(X_{rank(X)=k}): \lvert\lvert A-X \rvert\rvert_2$ where

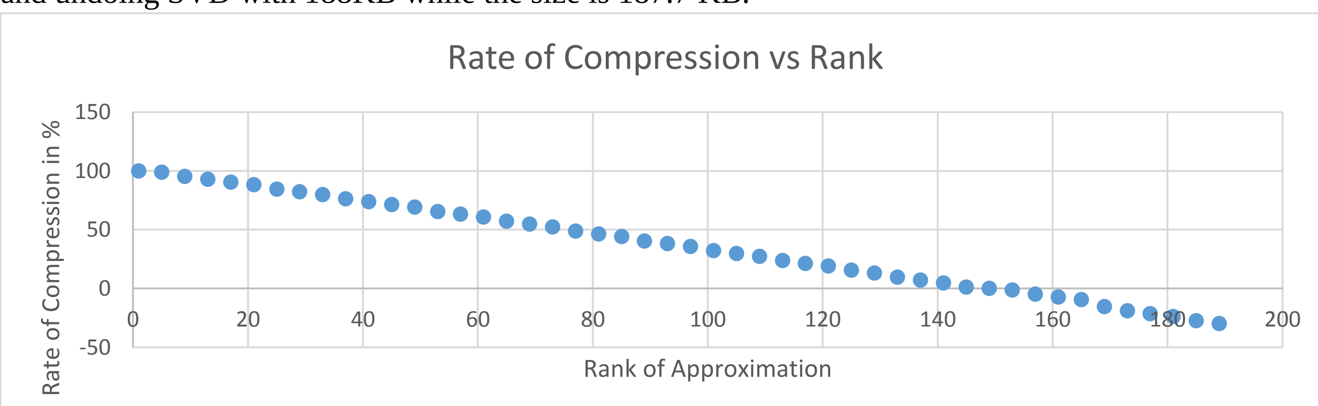
$\lvert\lvert A \rvert\rvert_2=\sqrt{\,(\sum_{i=1}^{m}\sum_{j=1}^{n}\lvert a_{ij}\rvert^2)}$

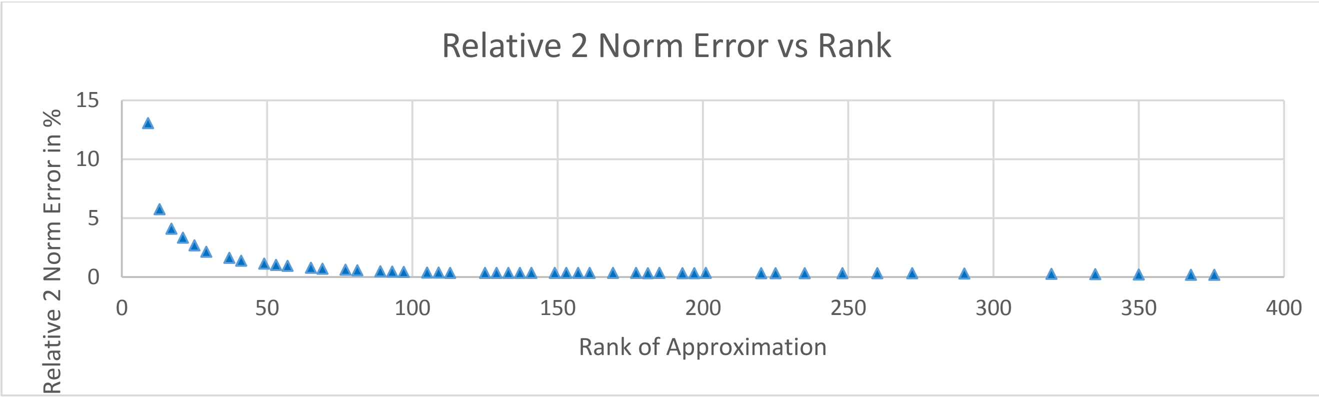
SVD Compression in Implemented Code:

In my implemented java code, the supplied SVD decomposition of the original matrix was obtained from the Original Matrix=SVD(U,,V^T) in MATLAB. Then I stored only the k Singular values, and the corresponding columns of U and rows of V^T in the file image_b.pgm.SVD and the necessary magic number, width, height, grayscale, k to retrieve a low value approximation of the original matrix. The background methodology is that it is assumed that only the significant Singular values is considered. The others are small and can be considered 0 as they are unimportant. In the image conversion, the order of multiplication is as follows ($\mathbf{U}(\mathbf{\Sigma V^T})$). Additionally, I used bitwise operations (10 bits mantissa, 5 bits exponent and 1 bit for sign) to accommodate the float value into 2 bytes and save it short datatype. Then while undoing the SVD, the low rank image should be with as less error as possible along with a good compression rate.

Theoretical analyses & experimental runs in SVD:

In testing SVD compression, I used the provided "CAS.pgm". From the graph below, we observe that as we increase the rank, the compression decreases. When k=1 there is almost 100% compression rate, but the relative 2norm error is 13%. But after k=150, there is negative compression that means, there is expansion. The file becomes larger than the original file. On the other hand, the relative 2 norm form is almost around 34% for k=109 till little less than 235. That means in those k values there is not much pattern change that could be retrieved by increasing k in that range. And lastly until the k_{max} =376=min(width, height) the best relative 2norm error rate is 0.18%. From the results it can inferred that after a definite k rank for an image, image cannot be compressed anymore, and the 2 norm relative error cannot be improved anymore. Moreover, the compression ratio C= (2*k*(w+h+1))/w*h. So when C=1, the k-value of SVD compression will strictly worsen than binary compression. k=107 provides binary file after SVD and undoing SVD with 188KB while the size is 187.7 KB.





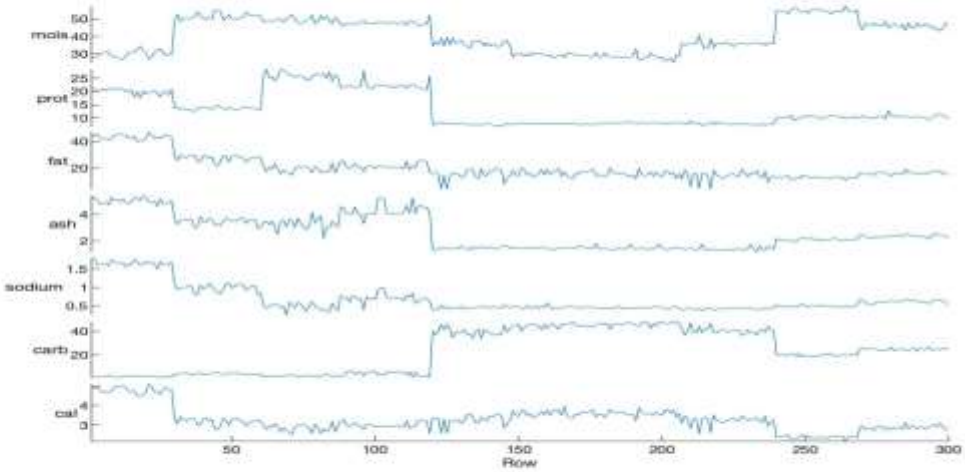
Principal Component Analysis (PCA):

PCA uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. The first Principal component represents the maximum variance direction from the origin, the next Principal Components are orthogonal to the first and provides residual variance. So the goal of PCA is to provide new features of uncorrelated data. PCA uses the SVD to avoid the less important data. But it considers the highest variances to keep. The scale featuring occurs before the SVD in sample covariance matrix = $B^TB/n-1$. Next from the Covariance matrix we obtain SVD. From the SVD, we observe the singular values, and based on the singular values the highest values are chosen that after those the singular values are very small. After that data is visualized and decisions are inferred on the basis of Principal components.

Code Implementation in PCA: for PCA, the MATLAB implementation for centralization is used to centralize the data with respect to the center of the data. The command includes the following:
coeff: coefficients of new data
newdata: orthogonal data in the new dimensional space (**U**)
latent: variance of each PC (Principal Component)
tsd: how far data is from center of data set (Singular values)
variance in % explained by each Principal component

Analysis of data through PCA:

Our dataset consisted of 7 components of 10 kinds of pizza 30 samples each. Among the seven, three components are chosen- moisture, protein and carbohydrate on assumption from the following graph to identify which components make the Pizza tasty. (total variance 99.96%)



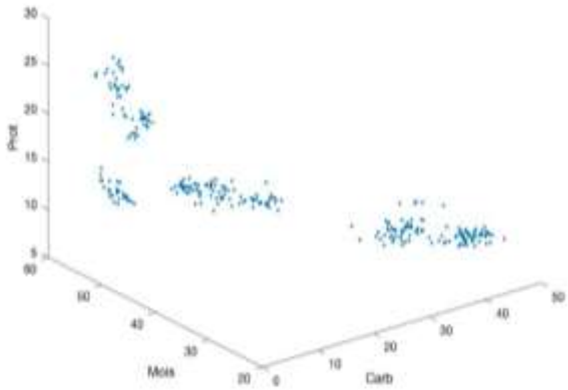
Then calculating the mean of columns and subtracting from respective columns we get B matrix, and transposing that we get B'. Then we calculate the covariance= $B^T B/n-1$. These 3 variables make up 99.96% of the variance within the data. Then the SVD provides us with SVD=VSV^T. The S matrix is in decreasing order and we see that after the first 3 singular values the rest are very small. So we consider the first three Principal components from the V matrix where column 1, 2, 3 represent principal components y1, y2, y3 which previously represented carb, moisture and protein respectively.

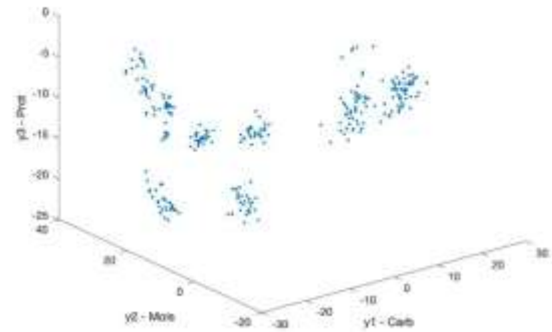
	1	2	3	4	5	6	7
1	421.5885	0	0	0	0	0	0
2	0	101.8202	0	0	0	0	0
3	0	0	16.8253	0	0	0	0
4	0	0	0	0.1844	0	0	0
5	0	0	0	0	0.0084	0	0
6	0	0	0	0	0	0.0010	0
7	0	0	0	0	0	0	2.4383e-...

Fig: S Matrix

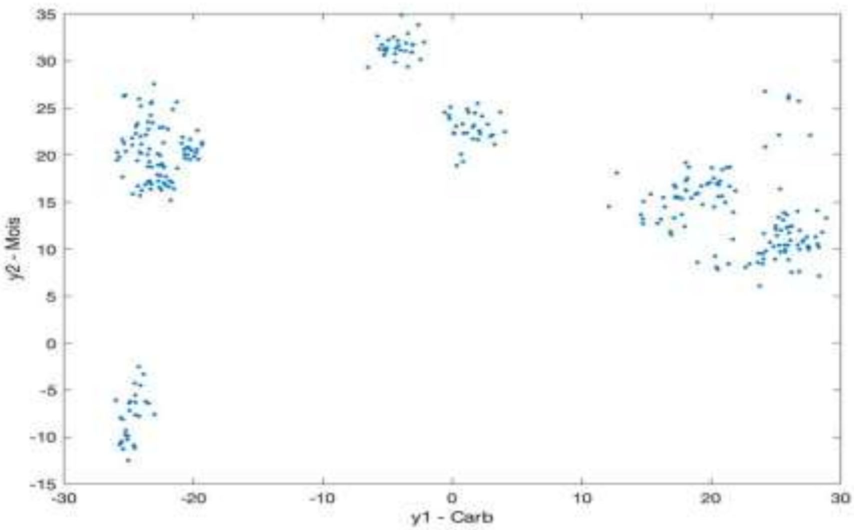
1	2	3
-0.2770	0.7471	-0.3520
-0.2669	-0.0557	0.8097
-0.2789	-0.6578	-0.4680
-0.0554	-0.0406	0.0222
-0.0111	-0.0238	-0.0262
0.8781	0.0068	-0.0125
-6.0329e-04	-0.0613	-0.0101

Fig: V Matrix





The above two graphs represent the plots before and after PCA. And if we separate the first two components we get the following graph.



Conclusion From PCA:
Finally we can assume that, the combination of protein, moisture and carbohydrates make a pizza a tasty and not tasty (total variance 99.96%). If there was an identification of breads, decision like thin crust or thick crust is tastier could be inferred. But there is also a fact that Carbohydrate starts the breakdown into particles in the mouth. So carbohydrate in fact might have a strong role in making pizza tasty.