Implementing Dimensionality Reduction Techniques using Vector Models

Group Members:

1. Dhruvil Deepakbhai Shah / dshah47@asu.edu
2. Deep Vijaykumar Patel / dpatel94@asu.edu
3. Nihar Samirbhai Patel / npatel69@asu.edu
4. Manikanta Mudara / mmudara@asu.edu
5. Pramukh Belam / pbelam@asu.edu
6. Vijay Aravindh Viswanathan / vviswa19@asu.edu

Abstract

During this phase, images are labeled image-X-Y-Z.png, where – X (cc, con, emboss, jitter, neg, noise1, noise2, original, poster, rot, smooth, stipple) specifies the image type, Y denotes the subject ID in range [1-40], and Z denotes the image sample ID [1-10]. The three feature models and similarity/distance functions generated in the previous phase are used in this phase's tasks. Various dimensionality reduction techniques like Principal Component Analysis, Latent Dirichlet Allocation, Singular Valued Decomposition, K-means clustering are applied to the extracted features. Using this computed latent semantics data matrix and input query image we return most similar images based on the specified parameters.

Keywords

Type-type similarity, Subject-subject similarity, SVD, PCA, LDA, Color Moments, ELBP, HOG

Introduction

The dataset contains images of 12 classes. Each class contains 40 subjects, and each subject is having 10 various images. The dimensions of the images are 64 x 64. Dimensionality reduction techniques cannot be applied if the length of feature vectors is not uniform. To tackle this issue, we dropped some images from some classes and got 360 images for each class type. For each image, Color Moments returns a feature vector of length 192 (8x8x3), Extended Local Binary Pattern return a feature vector of length 256 (0-255), Histogram of Oriented Gradients returns a feature vector of length 1764 (49x36). In the dimensionality reduction functions, we give the data matrix of NxM (where N = number of objects, M = number of features), and it returns the Nxk (where N = number of objects, k = number of reduced dimensions) dimension matrix.

Terminology

This section highlights the basic terminologies used throughout the implementation of the tasks of this project phase.

Singular Value Decomposition:

The factorization of a matrix A into the product of three matrices $A = UDV^T$, where the columns of U and V are orthonormal and the matrix D is diagonal with positive real entries, is known as singular value decomposition. The SVD is useful for a variety of activities. First, the data matrix A is often near to a low rank matrix, and it is useful to locate a low rank matrix that is a good approximation to the data matrix.

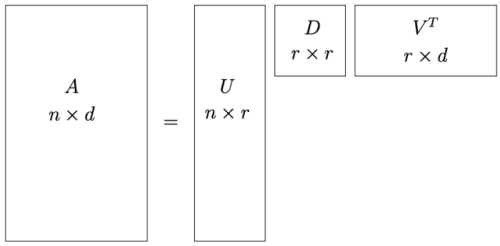


Figure 1.2: The SVD decomposition of an $n \times d$ matrix.

Custom Implementation:
Given the data matrix of (N x M), this function returns the data matrix of reduced dimensions (N x k) and the top-k eigenvalues for the data matrix.

```
# This Function reduces dimensions using SVD

def do_svd(data_mat, k):

    u, s, vt = np.linalg.svd(data_mat, full_matrices=True)
    u = u[:, :k]
    s = s[:k]
    vt = vt[:k, :]

    return [u, s]
```

Where, u is our reduced data matrix and s is the matrix comprising top k eigenvalues.

Principal Component Analysis:

Large datasets are becoming more frequent, and they might be challenging to decipher. PCA is a method for lowering the dimensionality of such datasets, boosting interpretability while minimizing information loss. It accomplishes this by generating new uncorrelated variables that optimize variance in a sequential manner. PCA is an adaptive data analysis technique because it simplifies finding new variables, the principal components, to solving an eigenvalue/eigenvector problem, and the new variables are specified by the dataset at hand, not a priori.

Custom implementation:

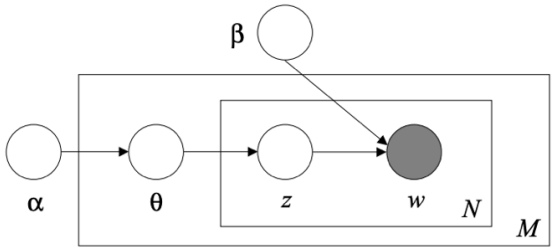
Given the data matrix of (N x M), this function returns the data matrix of reduced dimensions (N x k) and the top-k eigenvalues for the data matrix.

```
# This Function reduces dimensionality of the data matrix (n x k) using PCA

def do_pca(data_mat,k):
    #Calculating Eigen values and Eigen vectors of the data matrix
    X_meaned = data_mat - np.mean(data_mat , axis = 0)
    cov_mat = np.cov(X_meaned , rowvar = False)
    eigen_values , eigen_vectors = np.linalg.eig(cov_mat)

    #Sort the eigenvalues in descending order
    sorted_index = np.argsort(eigen_values)[::-1]
    sorted_eigenvalue = eigen_values[sorted_index]

    #Sort the eigenvectors
    sorted_eigenvectors = eigen_vectors[:,sorted_index]
    eigenvector_subset = sorted_eigenvectors[:,0:k]

    #Transform data into lower dimensions
    X_reduced = np.dot(eigenvector_subset.transpose(),X_meaned.transpose()).transpose()

    #Eigen values in diagonal in decreasing order
    LS = np.identity(k)
    for i in range(k):
        LS[i][i] = sorted_eigenvalue[i]

    return X_reduced, LS
```

Here, X_reduced is the newly generated reduced data matrix of shape (Nxk) and LS is matrix of eigenvalues of shape (kxk).

Latent Dirichlet Allocation:

LDA is a three-level hierarchical Bayesian model in which each collection item is represented as a finite mixture over an underlying set of topics. Each topic is thus represented as an infinite mixture over a collection of topic probabilities. The topic probabilities give an explicit representation of a document in the context of text modeling. For empirical Bayes parameter estimation, we provide efficient approximate inference strategies based on variational methods and an EM algorithm.



Custom implementation:

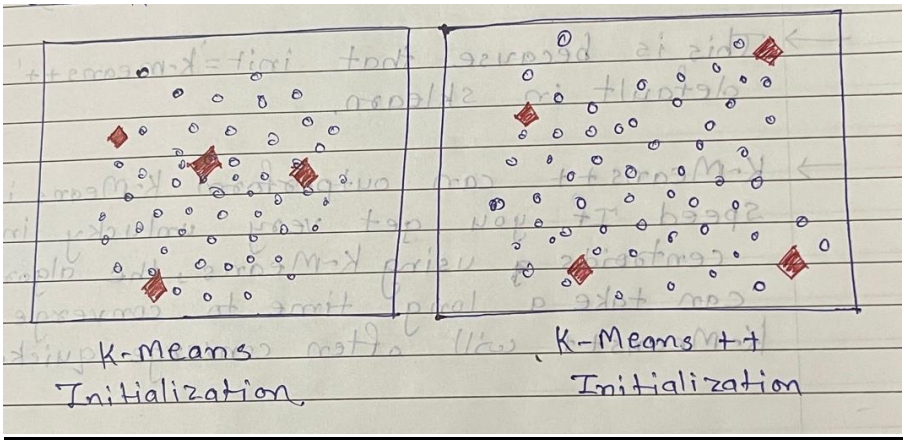
Given the data matrix of N x M, this function returns the data matrix of reduced dimensions (Nxk).

```
# This Function reduces dimensions using LDA

def do_lda(data_mat, k):

    lda = LatentDirichletAllocation(n_components=k, batch_size = 10)
    lda_f = lda.fit(data_mat)
    lda_weights = lda_f.transform(data_mat)

    return [lda_weights, lda_f.components_]
```

Here, lda_weights is the data in reduced dimension of shape (Nxk) and lda_f.components_ is the concept-term matrix of shape (kxM) dimensions given input image of shape (NxM).

KMeans++ clustering as dimensionality reduction:

K-means algorithm can be used to reduce the dimensions of a data matrix. When we cluster N-dimensional data into C-dimensional clusters, we have effectively lowered the dimension from N to C. If we employ a hard K-means clustering, the output will almost certainly be binary and useless for dimension reduction. Thus, we start with K-means and then utilize the cluster centroids as a starting point for Radial basis functions. For this phase, we have used KMeans++ clustering algorithm as it solves the problem of Poor Clustering by picking the next cluster according to a probability proportional to the distance of each point to its nearest cluster centroid. This makes it likely for the next cluster centroid to be far from the initially assigned centroids.



Custom implementation:

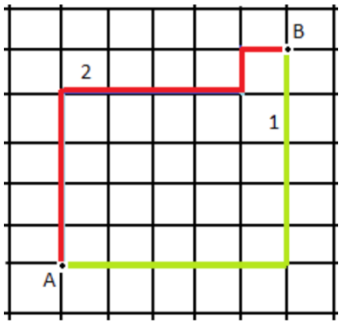
Given the data matrix of N x M, this function returns the data matrix of reduced dimensions N x K.

```
# This Function reduces dimensions using K-Means++

def do_kmeans(data_mat, k):
    kmeans = KMeans(n_clusters=k).fit(np.array(data_mat).T)
    centroids = kmeans.cluster_centers_

    return np.array(centroids).T
```

Here, centroids is the newly generated reduced data matrix of shape (kxN). We return the transpose of this matrix of shape (Nxk).

Manhattan Distance:

Manhattan distance is utilized when we need to determine the distance between two data points in a grid-like pattern, also known as city block distance or taxicab geometry.



$$d = \sum_{i=1}^{n} |x_i - y_i|$$

According to Aggarwal et al. (2001), it may be advantageous to utilize smaller values of p for a particular problem with a fixed (high) dimensionality d. This suggests that for high-dimensional applications, the L1 distance metric (Manhattan Distance metric) is the best choice [1].

Personalized Page Rank:

The Pagerank algorithm is used to determine a webpage's quality or importance. The Pagerank of a webpage is determined by counting the number of web pages that connect to it and the number of outgoing links from each of those web pages. In general, Pagerank can be deduced by examining a person who visits a specific webpage. The user jumps to any random webpage with probability 1 - Beta, and with probability Beta, he jumps to one of the web pages pointed by the current webpage.

$$PR(t+1) = \beta * \big(T\ X\ PR(t)\big) + (1-\beta) * s$$

Where T denotes the Transition Matrix, Beta denotes the likelihood that a user will jump to one of the nodes in the current node's outbound set, PR(t) denotes the set of Page Rank of all nodes at iteration t, and s denotes the seed matrix.

Problem Specification

Task 1: Implement a program which (a) given one of the three feature models, (b) a user specified value of X, a user specified value of k, (d) one of the four dimensionality reduction techniques (PCA, SVD, LDA, k-means) chosen by the user, reports the top-k latent semantics extracted using images of this type. Each latent semantic should be presented in the form of a list of subject-weight pairs, ordered in decreasing order of weights. Store the latent semantics in a properly labeled output file.

Task 2: Implement a program which (a) given one of the three feature models, (b) a user specified value of Y, a user
specified value of k, (d) one of the three dimensionality reduction techniques (PCA, SVD, LDA,k-means) chosen by the user, reports the top-k latent semantics extracted using images of this subject. Each latent semantic

should be presented in the form of a list of type-weight pairs, ordered in decreasing order of weights. Store the latent semantics in a properly
labeled output file.

Task 3: Implement a program which, (a) given one of the three feature models and (b) a value k,
– creates (and saves) a type-type similarity matrix,
– performs a user selected dimensionality reduction technique (PCA, SVD, LDA,k-means) on this type-type similarity
matrix, and
– reports the top-k latent semantics.
Each latent semantic should be presented in the form of a list of type-weight pairs, ordered in decreasing order of weights.
Store the latent semantics in a properly labeled output file.

Task 4: Implement a program which, (a) given one of the three feature models and (b) a value k,
– creates (and saves) a subject-subject similarity matrix,
– performs a user selected dimensionality reduction technique (PCA, SVD, LDA,k-means) on this subject-subject similarity matrix, and
– reports the top-k latent semantics.
Each latent semantic should be presented in the form of a list of subject-weight pairs, ordered in decreasing order of
weights. Store the latent semantics in a properly labeled output file.

Task 5: Implement a program which, given the filename of a query image which may not be in the database and a latent
semantics file, identifies and visualizes the most similar n images under the selected latent semantics.

Task 6: Implement a program which, given the filename of a query image which may not be in the database and a latent
semantics file, associates a type label (X) to the image under the selected latent semantics.

Task 7: Implement a program which, given the filename of a query image which may not be in the database and a latent
semantics file, associates a subject ID (Y) to the image under the selected latent semantics.

Task 8: Implement a program which (a) given a subject-subject similarity matrix, (b) a value n, and (c) a value m,
– creates a similarity graph, G(V, E), where V corresponds to the subjects in the database and E contains node pairs vi, vj such that, for each subject vi, vj is one of the n most similar subjects in the database
– identifies the most significant m subjects in the collection using ASCOS++ measure.

Task 9: Implement a program which (a) given a subject-subject similarity matrix, (b) a value n, (c) a value m, and three subject IDs
– creates a similarity graph, G(V, E), where V corresponds to the subjects in the database and E contains node pairs vi, vj such that, for each subject vi, vj is one of the n most similar subjects in the database
– identifies the most significant m subjects (relative to the input subjects) using personalized PageRank measure.

Assumptions

The following assumptions are made during the implementation of this phase of the project.

1. All images in the folder are of the same size and shape 64 x 64.
2. The images in the folder are grayscale and have normalized values [0-1].
3. The libraries used to compute the distance give accurate results.
4. If two photos have the same distance/similarity score, the image with the lowest numerical image ID is picked as the more similar.
5. For this implementation, we are given 12 types, 40 subjects and 10 images per subject. After Exploratory analysis, it was observed that some of the subjects were having 9 images. To address this issue, we have dropped the images with image_id = 10. Hence, we now have the final dataset comprising 4320 images (12 x 40 x 9).

Description of the proposed solution/implementation

Task 1:

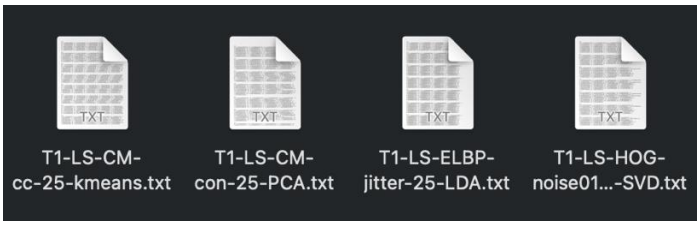
Input: Feature model (Color Moments, Extended Local Binary Pattern, Histogram of Oriented Gradients), X (One of the 12 types), k (Number of desired latent semantics), Dimensionality reduction technique (Principal Component Analysis, Latent Dirichlet Allocation, Singular Valued Decomposition, K-means clustering).

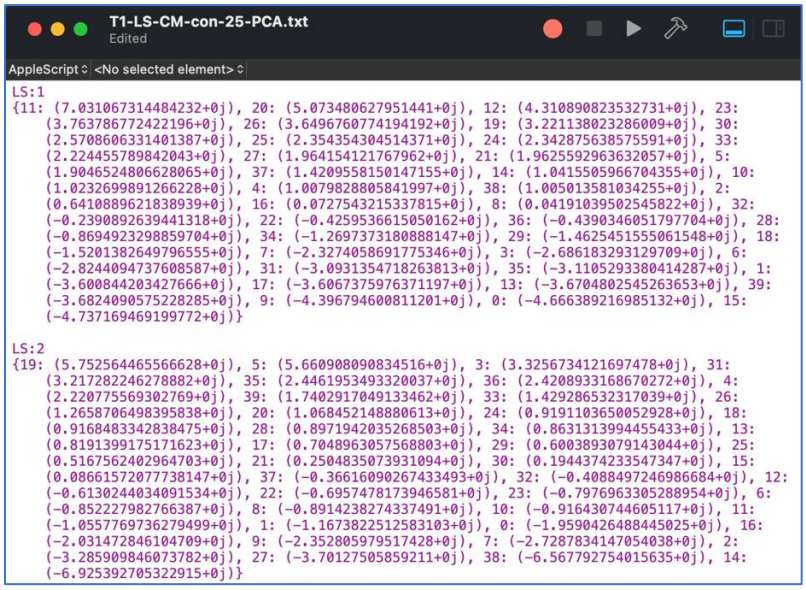
Output: Each latent semantic is presented in the form of a list of subject-weight pairs, ordered in decreasing order of weights and stored in a properly labeled output file. For example, when the main function of this task is given the parameters – CM, con, 4, kmeans, the file "T1-LS-CM-con-4-kmeans.txt" will be created.

In this task, we first create a dictionary having Subjects as keys given the class X. The values will be lists containing the images of that subject. Now, we calculate the features for all 9 images per subject using the specified feature model and create a single feature vector by taking an average of them. The feature vectors will be of length [192 (Color moments) / 256 (ELBP) / 1764(HoG)]. Thus, we will convert our dictionary to data matrix of length 40, where each row has a length of (192/256/1764). We pass this array to reduce the dimensionality and get the (40 x K) latent semantics matrix. For this, a Dimensionality Reduction method is passed as string to a function and that returns latent semantics array.

```
Enter a feature model - CM, ELBP, HOG :CM
Enter a value of X:con
Enter a value of k:4
Enter a dimentionality reduction technique - PCA, LDA, SVD, k-means:kmeans

Latent Semantic file created :  T1-LS-CM-con-4-kmeans.txt
```

Output snapshots:





Task 2:

Input: Feature model (Color Moments, Extended Local Binary Pattern, Histogram of Oriented Gradients), Y (One of the 40 subjects), k (Number of desired latent semantics), Dimensionality reduction technique (Principal Component Analysis, Latent Dirichlet Allocation, Singular Valued Decomposition, K-means clustering).

Output: A file containing top-k latent semantics of the images in the database. For example, when the main function of this task is given the parameters – HOG, noise02, 34, kmeans, the file “T2-LS-CM-3-23-PCA.txt” will be created.

In this task, we first create a dictionary having Classes as keys given the subject Y. The values will be lists containing the images of each Class of that subject. Now, we calculate the features for all 9 images using any of three models and create a single feature vector by taking an average. The feature vectors will be of length (192 (Color moments) / 256 (ELBP) / 1764(HoG)). Thus, we will convert our dictionary to data matrix of length 12, where each row has a length of (192/256/1764). We pass this array to reduce the dimensionality and get the (12 x K) latent semantics matrix. For this, a Dimensionality Reduction method is passed as string to a function and that returns latent semantics array.

```
Enter a feature model - CM, ELBP, HOG :CM
Enter a value of Y:3
Enter a value of k:23
Enter a dimentionality reduction technique - PCA, LDA, SVD, k-means:PCA
Latent Semantic file created :  T2-LS-CM-3-23-PCA.txt
```

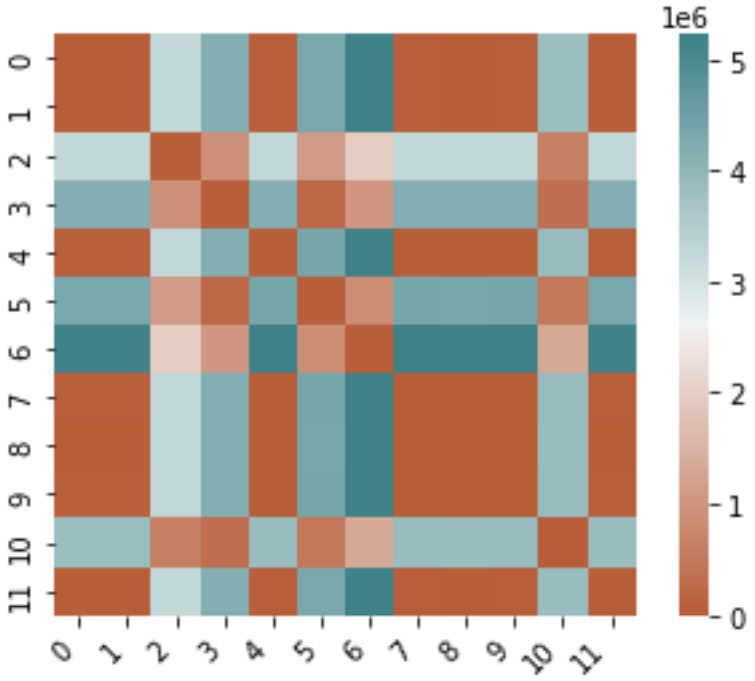
Output snapshots:

Task 3:

Input: Feature model (Color Moments, Extended Local Binary Pattern, Histogram of Oriented Gradients), k (Number of desired latent semantics), Dimensionality reduction technique (Principal Component Analysis, Latent Dirichlet Allocation, Singular Valued Decomposition, K-means clustering).

Output: A file containing top-k latent semantics of the images in the database. For example, when the main function of this task is given the parameters – CM, 6, SVD, the file “T3-LS-CM-6-SVD.txt” will be created.

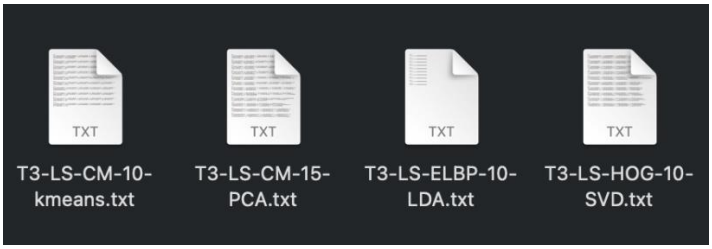
First, we create a dictionary that has Classes (type) as keys and 192 (Color moments) shape vector as value. We get this 192-length vector by taking an average of all the feature vectors received by calculating Color moments of 40 (subjects) * 9 (images per subject) images. Now, to create a type-type similarity matrix of shape 12 x 12, we implement Manhattan distance to compute the similarity between each type from the above dictionary. Thus, in the new data matrix we have null values in diagonal. The rest of the values portray the similarity/ distance between two Classes (types).



We plot the 12 x 12 type- type similarity matrix as above. Where [i,j] depicts the distance between i^{th} class and j^{th} class. It is palpable from the above figure that the plot is symmetric to its diagonal.

Finally, we give this matrix for dimensionality reduction (using Principal Component Analysis, Singular Valued Decomposition, KMeans) operation. Next, we write the final 12 x k shape matrix into a file and save it.

Output snapshots:

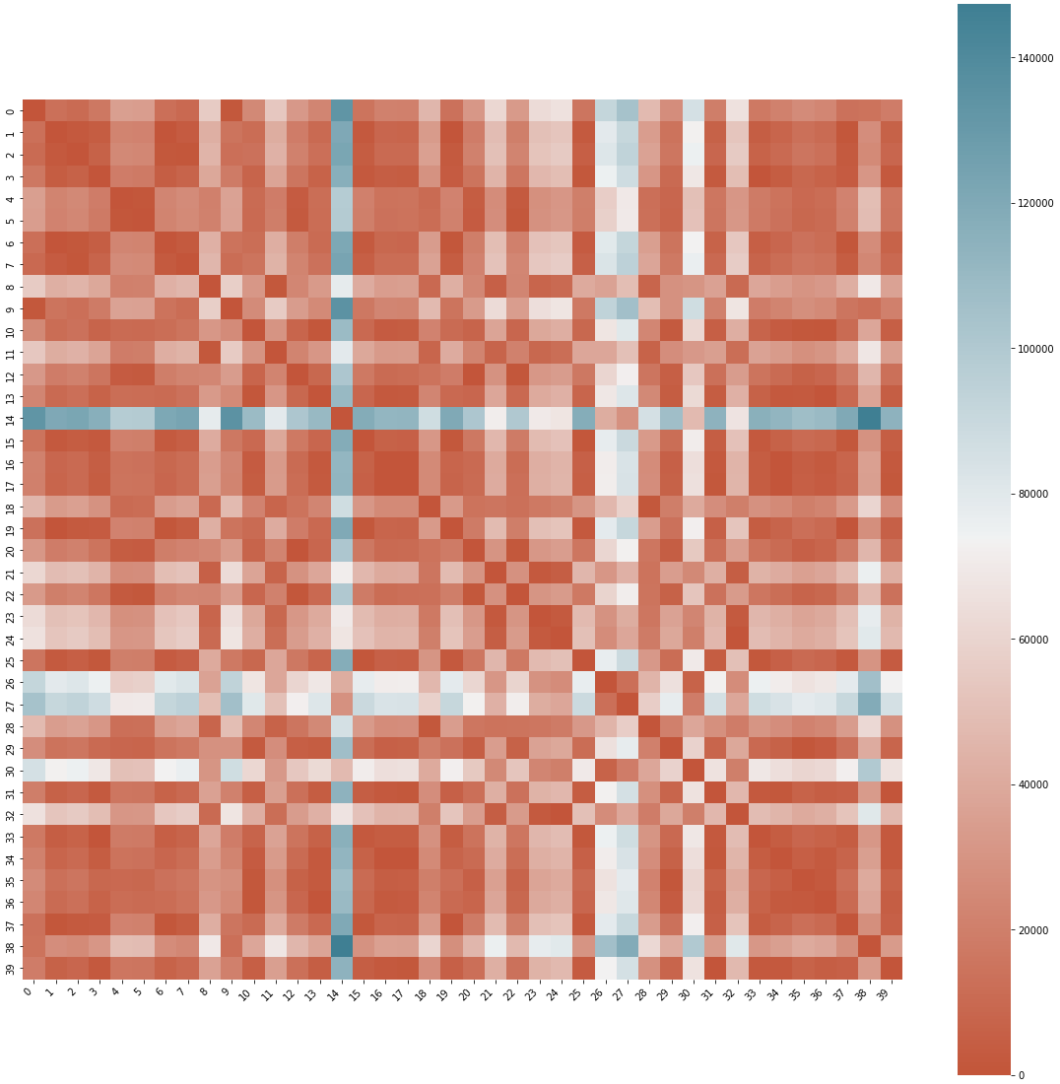


Task 4:

Input: Feature model (Color Moments, Extended Local Binary Pattern, Histogram of Oriented Gradients), k (Number of desired latent semantics), Dimensionality reduction technique (Principal Component Analysis, Latent Dirichlet Allocation, Singular Valued Decomposition, K-means clustering).

Output: A file containing top-k latent semantics of the images in the database. For example, when the main function of this task is given the parameters – CM, 6, SVD, the file "T4-LS-HOG-5-SVD.txt" will be created.

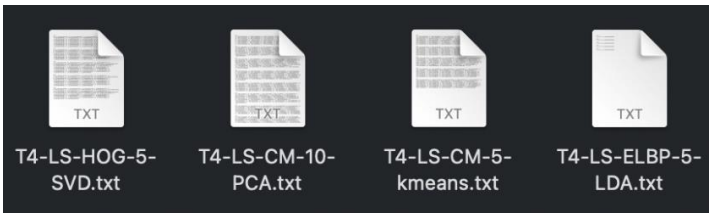
First, we create a dictionary that has Subjects as keys and 192 (Color moments) shape vector as value. We get this 192 length vector by taking an average of all the feature vectors received by computing color moments of 12 (Types) * 9 (each Class's images) images. Here, we consider Color Moments to compute the features of the images. Now, to create a subject-subject similarity matrix of shape 40 x 40, we implement Manhattan distance to compute the similarity between each subject from the above dictionary. Thus, in the new data matrix we have null values in diagonal. The rest of the values portray the similarity/ distance between two Subjects.

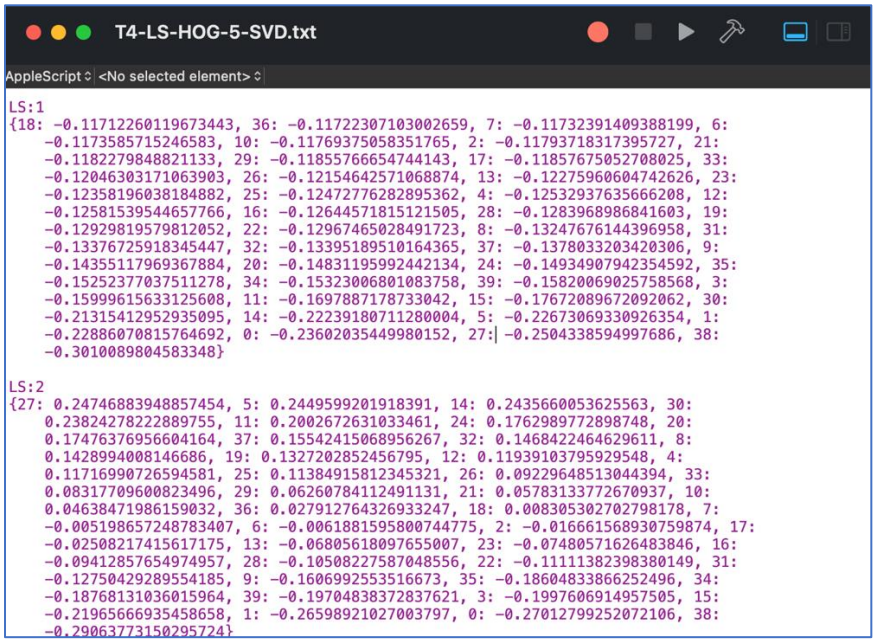


We plot the 40 x 40 subject- subject similarity matrix as above. Where [i,j] depicts the distance between i^{th} subject and j^{th} subject. It is palpable from the above figure that the plot is symmetric to its diagonal

Finally, we give this matrix for dimensionality reduction (using Principal Component Analysis, Singular Valued Decomposition, KMeans) operation. Next, we write the final 40 x k shape matrix into a file and save it.

Output snapshots:





Task 5:

Input: Query image name (string), Feature model, Dimensionality reduction technique, Value of k, Value of n

Output: Most similar n-images

In this function, we first create a data matrix of shape 4320 x (192/256/1764) where, 4320 represents all the images in the dataset and 192 is the length of the feature vector (for color moments) for each image. Now we append our query image in this data matrix and compute the same feature vector for it. Now, we calculate the euclidean distance between all images and the query image. Lastly, we return the most similar n images from the distances computed.

```
query_image = input("Enter name of input image : ")
FM = input("Enter a feature model - CM, ELBP, HOG :")
k = int(input("Enter a value of k:"))
DR = input("Enter a dimentionality reduction technique - PCA, LDA, SVD, kmeans:")
n = int(input("Enter value of n:"))

#task5(query_image='image-cc-5-3.png', FM='CM', DR='PCA', k=50, n=10)
task5(query_image, FM, DR, k, n)
```

```
['image-original-1-1.png',
 'image-poster-1-1.png',
 'image-original-1-3.png',
 'image-original-16-3.png',
 'image-poster-1-3.png',
 'image-original-16-2.png',
 'image-original-1-7.png',
 'image-original-24-2.png',
 'image-poster-16-3.png',
 'image-poster-16-2.png']
```

Task 6:

Input: Query image name (string), Feature model, Dimensionality reduction technique, Value of k, Subject id Y

Output:
Most similar Types based on the distance

Here, first we compute the features in a way like Task 2. We get two matrices when we reduce the dimensions of the data matrix given the Subject id. Thus, we have 2 arrays of shapes (12 x k) and (k x (192/256/1764)). Now, we compute the features for the query image that will be of shape (1, 192/256/1764). Next, we project this vector into our (k x 192/256/1764) shape matrix by performing a dot product to get the vector of shape (k x 1). Finally, we find the distance between each type's feature vector from the matrix of shape (12 x k) that we created in the initial step. We return the Type ids along with the distance score.

```
img_name = input("Enter name of input image : ")
FM = input("Enter a feature model - CM, ELBP, HOG :")
k = int(input("Enter a value of k:"))
DR = input("Enter a dimentionality reduction technique - PCA, LDA, SVD, kmeans:")
Y = input("Enter value of Y:")

a = task6(img_name, FM, k, DR, Y)
#a = task6('image-cc-1-1.png', 'CM', 15, 'PCA', '5')
print(a)
```

```
Enter name of input image : image-cc-1-1.png
Enter a feature model - CM, ELBP, HOG :CM
Enter a value of k:5
Enter a dimentionality reduction technique - PCA, LDA, SVD, kmeans:PCA
Enter value of Y:5
{2: 497.78932703700076, 0: 551.7037140614211, 1: 552.8894284161947, 11: 552.915079294854,
```

Task 7:

Input:
Input: Query image name (string), Feature model, Dimensionality reduction technique, Value of k, Type id X

Output:
Most similar Subjects based on the distance

Here, first we compute the features in a way like Task 1. We get two matrices when we reduce the dimensions of the data matrix given the Subject id. Thus, we have 2 arrays of shapes (40 x k) and (k x (192/256/1764)). Now, we compute the features for the query image that will be of shape (1, 192/256/1764). Next, we project this vector into our (k x 192/256/1764) shape matrix by performing a dot product to get the vector of shape (k x 1). Finally, we find the distance between each Subject's feature vector from the matrix of shape (40 x k) that we created in the initial step. We return the Subject ids along with the distance score.

```
img_name = input("Enter name of input image : ")
FM = input("Enter a feature model - CM, ELBP, HOG :")
k = int(input("Enter a value of k:"))
DR = input("Enter a dimentionality reduction technique - PCA, LDA, SVD, kmeans:")
X = input("Enter value of X:")

a = task7(img_name, FM, k, DR, X)
#a = task7('image-cc-10-1.png', 'CM', 15, 'PCA', 'cc')
print(a)
```

```
Enter name of input image : image-cc-1-1.png
Enter a feature model - CM, ELBP, HOG :CM
Enter a value of k:5
Enter a dimentionality reduction technique - PCA, LDA, SVD, kmeans:PCA
Enter value of X:cc
{0: 7.243229396261195, 29: 7.953443267497938, 6: 8.009671435494896, 16: 8.127719057418789,
```

Task 8:

Input:
Transition Graph, n value, m value

Output:
Most significant m objects using the PageRank

For this task, we have created a transition matrix that has 40 x 40 (subject-subject) shape. Here, in each row represents the distances with that subject to all other subjects. In each row, the most similar n values (distances with most similar n images) are non-zero while the other values are set to zero. If we consider subjects to be nodes, we now have the edges between different subjects. Each subject will have edge towards its most similar n images. The distance represents the edge weight between two subjects. We pass this matrix into our function along with the m value (most significant nodes) and 3 subject ids. Here, we give equal probability to each node for random jump unlike in Task 9.

```python
def task8(FM, n, m):

    transition_mat = compute_sub_sub(FM)

    #Setting jump factor to 0.85
    jumpFactor = 0.85

    #Initialization of seeds and distance matrices
    seeds = np.zeros((40, 1))
    dist = np.zeros(40)

    #Result matrix
    ans = []

    #Setting the probabilty to 1/40 for the given subject Ids
    for i in range(40):
        seeds[i] = 1/40

    #Computing the rankings of the nodes using PPR algorithm
    rankings = np.dot((np.identity(40) - (1-jumpFactor) * transition_mat), jumpFactor * seeds)
    rankings = rankings.reshape(40)

    for i, _ in enumerate(rankings):
        if i+1 not in seeds:
            dist[i] = rankings[i] / (1-jumpFactor)
        else:
            dist[i] = (rankings[i] - (jumpFactor / 40)) / (1-jumpFactor)


    for _, r in enumerate(dist):
        ans.append((_+1, r))

    #Returning the most significant m subjects
    return [i[0] for i in sorted(ans, reverse=True, key=lambda x: x[1])[:m]]
```

Output:

The three most significant subjects as the m value is passed as 3. The sub_sub variable represents the graph where each value represents the edge weight between two subject nodes.

```python
FM = input("Enter a feature model - CM, ELBP, HOG :")
n = int(input("Enter the value of n:"))
m = int(input("Enter the value of m:"))

top_m = task8(FM, n, m)
```

```
Enter a feature model - CM, ELBP, HOG :CM
Enter the value of n:5
Enter the value of m:5
```

```python
print(top_m)
```

```
[20, 38, 2, 18, 35]
```

Task 9:

Input:
Transition Graph, n value, m value, 3 subject IDs

Output:
Most significant m objects using the PPR

For this task, we have created a transition matrix that has 40 x 40 (subject-subject) shape. Here, in each row represents the distances with that subject to all other subjects. In each row, the most similar n values (distances with most similar n images) are non-zero while the other values are set to zero. If we consider subjects to be nodes, we now have the edges between different subjects. Each subject will have edge towards its most similar n images. The distance represents the edge weight between two subjects. We pass this matrix into our function along with the m value (most significant nodes) and 3 subject ids. Here, we give the probability to the given subject Ids only.

```
def task9(FM, n, m, s):

    transition_mat = compute_sub_sub(FM)

    #Setting jump factor to 0.85
    jumpFactor = 0.85

    #Initialization of seeds and distance matrices
    seeds = np.zeros((40, 1))
    dist = np.zeros(40)

    #Result matrix
    ans = []

    #Setting the probabilty to 1/40 for the given subject Ids
    for seed in s:
        seeds[seed-1] = 1/40

    #Computing the rankings of the nodes using PPR algorithm
    rankings = np.dot((np.identity(40) - (1-jumpFactor) * transition_mat), jumpFactor * seeds)
    rankings = rankings.reshape(40)

    for i, _ in enumerate(rankings):
        if i+1 not in s:
            dist[i] = rankings[i] / (1-jumpFactor)
        else:
            dist[i] = (rankings[i] - (jumpFactor / 40)) / (1-jumpFactor)


    for _, r in enumerate(dist):
        ans.append((_+1, r))

    #Returning the most significant m subjects
    return [i[0] for i in sorted(ans, reverse=True, key=lambda x: x[1])[:m]]
```

Output:
The three most significant subjects as the m value is passed as 3. The sub_sub variable represents the graph where each value represents the edge weight between two subject nodes.

```
FM = input("Enter a feature model - CM, ELBP, HOG :")
n = int(input("Enter the value of n:"))
m = int(input("Enter the value of m:"))

top_m = task9(FM, n, m, s=[1, 2, 30])
```

```
Enter a feature model - CM, ELBP, HOG :CM
Enter the value of n:10
Enter the value of m:9
```

```
top_m
```

```
[2, 30, 9, 12, 15, 17, 18, 19, 22]
```

System requirements /installation and execution instructions:

The solution will work errorless with Python 3.6+.

System: Google Colab or Local Machine (Tested only on this platform)

Requirements:
numpy==1.19.5
scipy==1.4.1
scikit-image==0.16.2
scikit-learn==0.22.2.post1
matplotlib==3.2.2
seaborn==0.11.2

Interface Specifications

Run the “CSE515 Phase2.ipynb” file into Google colab because there will be no dependency and environment issues and code will run smoothly.

Execution steps: -

1) Open the CSE515 project phase2.ipynb file.
2) Hit “**Run all**” from “Runtime” tab.
3) After successful run, the cells above TASK 1 cell will import the libraries along with initializing helper functions.
4) After successful run, TASK 1 cell will generate the text file (T1-LS-CM-con-4-kmeans.txt) in the local directory given the specific parameters.
5) After successful run, TASK 2 cell will generate the text file (T2-LS-CM-3-23-PCA.txt) in the local directory given the specific parameters.
6) After successful run, TASK 3 cell will generate the text file (T3-LS-CM-6-SVD.txt) in the local directory given the specific parameters.
7) After successful run, TASK 4 cell will generate the text file (T4-LS-HOG-5-SVD.txt) in the local directory given the specific parameters.
8) After successful run, TASK 5 cell will return the most similar n images.
9) After successful run, TASK 6 cell will give the most similar Types along with the distances score.

10) After successful run, TASK 7 cell will give the most similar Subjects along with the distances score.
11) After successful run, TASK 8 cell will give the most significant m nodes using ASCOS++.
12) After successful run, TASK 9 cell will give the most significant m nodes using PPR.

Conclusions

It can be observed from the results of this study's research that different feature models contain variable quantities of information, and that information loss is critical for similarity comparisons. Varied models provide different outcomes from time to time, and the discrepancies are attributable to the intrinsic loss induced by the model selected.

Bibliography

1. Aggarwal, C.C., Hinneburg, A. and Keim, D.A., 2001, January. On the surprising behavior of distance metrics in high dimensional space. In *International conference on database theory* (pp. 420-434). Springer, Berlin, Heidelberg.
2. Page, L., Brin, S., Motwani, R. and Winograd, T., 1999. The PageRank citation ranking: Bringing order to the web. Stanford InfoLab.
3. Lofgren, P., Banerjee, S. and Goel, A., 2016, February. Personalized pagerank estimation and search: A bidirectional approach. In Proceedings of the Ninth ACM International Conference on Web Search and Data Mining (pp. 163-172).

Appendix
Specific roles of the group members

Each team member was responsible for implementation of the assigned tasks of this phase of the project on their own. The collaboration was confined to the overall solution design and idea sharing. The team members' ideas were also taken into account during design decisions and overall flow of the solution.