Phase #2

Group Members

Daniel George Mathew	Nawendu Singh	Sai Adithya Reddy Ch.
1216989033	1215146543	1217392228
dgmathe1@asu.edu	nsingh49@asu.edu	schinth3@asu.edu

Varoon Parthasarathy	Vaibhav Bharuka	Zain Mustakali Momin
1216905976	1217205665	1217167757
vparth10@asu.edu	vbharuka@asu.edu	zmomin1@asu.edu

Abstract:

In this Project, we are experimenting with multivariate time series data sets and vector models and trying to create a gesture recognition system.In phase 1 we explored a smaller time series to create, visualize and find similarities between different gestures.

In phase 2, We will build upon our previous work and create 4 new programs to identify the Latent semantics/topics in our data and then find and rank 10 most similar gestures in the Database based on user options.These user options will include dot product, latent sensor semantics, edit distance and DTW. The latent topics will be presented in the form of a <**word,score**> pairs sorted in descending order.

We will also be creating a gesture-gesture similarity matrix and report the top principal components in the form of <**gesture,score**> pairs sorted in descending order.These will be based on user inputs which has the options of dot product, latent sensor semantics, edit distance and DTW.

Finally , We would be applying clustering algorithms to our gesture-gesture similarity matrix and cluster the gestures into groups.We would specifically be using k-means clustering and Laplacian-based clustering techniques for this task.

Keywords:

Feature descriptors, Euclidean distance, Term factor(Tf), Term factor-Inverse document factor(Tf-IDF), Latent Semantics, Principal component analysis, Latent Dirichlet analysis, Singular value decomposition, Non-negative matrix factorization, Graph Laplacian matrix, Dot product, Edit distance, Dynamic time warping, k-means Clustering, Laplacian-based clustering,Dimensionality Reduction.

Introduction:

The goal of a gesture recognition system is to interpret human gestures via mathematical algorithms.These gesture can originate from any part of the body and captured through sensors attached to them.These help a computer understand human body language and can be used to develop systems that can help them interact way better than text or even GUIs.

Terminology:

- **Feature Descriptors:** Feature descriptors encode compelling information into a series of numbers and act as a sort of numerical "fingerprint" that can be used to distinguish one feature from another.
- **Euclidean Distance:** The distance between two points in a plane is given by Euclidean distance. The formula is given by:

  Euclidean distance (X,Y) = $\sqrt{\Sigma}(xi-yi)^2$
- **Term factor (TF):** Term factor is the value which accounts for how good the feature is in describing the contents of the object.
- **Term factor-Inverse document factor (TF-IDF):** Term factor-Inverse document factor is the value which accounts for both how good the feature is in describing the contents of the object and how differentiating the feature is in the whole database.
- **Principal Component Analysis (PCA):** Principal component analysis is a statistical procedure that 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.
- **Latent Dirichlet Allocation (LDA):** Latent Dirichlet allocation is a generative statistical model that allows sets of observations to be explained by unobserved groups that explain why some parts of the data are similar.
- **Singular Value Decomposition (SVD):** Singular value decomposition is a factorization of a real or complex matrix.
- **Non-Negative Matrix Factorization (NMF):** Non-negative matrix factorization is a group of algorithms in multivariate analysis and linear algebra where a matrix V is factored into two matrices W and H, with the property that all three matrices have no negative elements.

- **Graph Laplacian Matrix :** The graph laplacian matrix is the difference between the degree matrix and the weight matrix. Degree matrix is defined as the diagonal matrix with the degrees of each of n points in the graph as its diagonal. Weight matrix of a graph is just a NxN matrix, whose elements are the weights of the edges between any 2 points of the graph.
- **Dot Product:** Dot product is the sum of product of corresponding dimensions in a vector space. It can also be calculated by multiplying the magnitude of both vectors with the cosine angle between them.
- **Edit Distance:** Edit distance is a way of quantifying how dissimilar two sequences are, by calculating how many operations are required to transform a sequence into another.
- **Dynamic Time Warping:** Dynamic time warping is an algorithm used to measure similarity between two temporal sequences, which may vary in speed.

Goal:

The goal of this phase is to apply dimensionality reduction techniques to the feature descriptors obtained from the feature extraction techniques implemented in the previous phase. In this phase we focus on the following:

1. The feature descriptors have to be decomposed based on the chosen dimensionality reduction technique.
2. The extraction of k-latent semantics for the dimensionality reduction technique.
3. The retrieval of 10 similar gestures after dimensionality reduction.
4. The extraction of top-p principle components and latent semantics (in terms of gesture membership) underlying the gesture-gesture similarity matrix
5. The partition of the entire gesture folder into groups based on their degree of contribution to p latent-semantics.
6. Clustering of the entire gesture folder based on gesture-gesture similarity matrix based on K-means as well as laplacian based spectral clustering.

Assumptions:

- All input data should be correct. No validation checks are done.
- Each gesture file is in csv format and is stored in a single directory.
- Each gesture file contains the data from 20 sensors.
- Every gesture file has the data from the similar sensor placed in similar body positions.
- The input file given will be part of the directory and the correct path and file name will be provided as input.
- The rest of the user inputs will be of correct data type.
- Both tf_vectors_fi.txt and tf_idf_vectors_fi.txt for all files are combined into one dictionary written to one ‘vectors.txt’ file
- Results of task2 and task 3 are dependent on the option selected in task 1.
- Input of task 4 is dependent on the output of task 3.

Dataset:

We have 93 gesture files in 4 subfolders X,Y,Z and W which represent the 3 spatial coordinates and angular coordinates respectively. Each gesture file has 20 sensors and each sensor has a certain number of values.
The rows map to body parts in the following way:

1 HipCenter
2 Spine
3 ShoulderCenter
4 Head
5 ShoulderLeft
6 ElbowLeft
7 WristLeft
8 HandLeft
9 ShoulderRight
10 ElbowRight
11 WristRight
12 HandRight
13 HipLeft
14 KneeLeft
15 AnkleLeft
16 FootLeft
17 HipRight
18 KneeRight
19 AnkleRight
20 FootRight

Implementation:

Task 0a

1. We compute the average of the values in every component's file's sensor and output them into a file named means. Similarly, we save the standard deviations into a file named stdev.

   The means are stored in the format: `('X', '1', '1'): 0.19093430799999994`
   Where 'X' is the component name, '1' is the file name, '1' is the sensor number and `0.19093430799999994 is the average of values across a sensor.`

2. Then we normalize the values between -1 and 1 using the min max scaler. If all values are the same, then they are normalized to 0.

$$normalized\ value = \frac{value}{max - min}$$

3. Then based on the following formula we calculate the bin lengths and the values are quantized into 2r levels, where r is the resolution:

$$length_i = 2 \times \frac{\int_{(i-r-1)/r}^{(i-r)/r} Gaussian_{(\mu=0.0,\sigma=0.25)}(x)\ \ \delta x}{\int_{-1}^{1} Gaussian_{(\mu=0.0,\sigma=0.25)}(x)\ \ \delta x}$$

4. By moving a window of size "w" as given by user input, we compute the symbolic quantized window descriptor. These values are calculated for each component, every file, every sensor. In each sensor, we move a w length window and extract the word and save it to a file. For every file(gesture) there is a corresponding ".wrd" file in the format:

"('X', '1', 0, 0)",333

In this example, 'X' is the component, '1' is the file name, 0 is the sensor id and 0 is the timestamp. 333 is the computer word.

5. Similarly, the average values are calculated and stored into a file with extension ".awrd" file. Each line in the file is of the form

"('X', '58', 0, 4)",-0.5870797507850672

Where 'X' is the component name, '58' is the file name, 0 is the sensor id, 4 is the time stamp and -0.5870797507850672 is the average of normalized value.

Task 0b

1. Tf values for each sensor of every gesture are written into a file named vectors.txt in the output folder. Each word's tf value is written in the form of

X_0_3333: 0.00208333333333333

Here X_0_3333 represents the word and 0.0020833333333 represents the tf value.
These values are calculated for each file(gesture)

2. Tf value is calculated using the formula:

$$tf_{value} = \frac{no\ of\ times\ a\ word\ occurs\ in\ the\ gesture}{no\ of\ times\ the\ word\ occurs\ across\ all\ gestures}$$

3. Tf_idf values for each sensor of every gesture are written into a file named vectors.txt in the output folder. Each word's tf_idf value is written in the form of

X_0_3333 : 0.1486623675428065

Here X_0_3333 represents the word and 0.0020833333333 represents the tf_idf value

4. Tf_idf value is calculated using the formula:

$$tf\_idf_{value} = \left(0.5 + 0.5 * \frac{\frac{n}{k}}{\max freq}\right) \log\left(\frac{n}{m}\right)$$

where n/k is frequency of term in the query
And n/m represents the no of files the word exists in by the total no of files

Task 1

The data from word files are taken into a dataframe where rows are file names and columns are word files. The cells have tf values/ idf values based on the user selection.
Then we scale the values such that the results are positive and between 0, 1.
Then pca, svd, nmf, lda are calculated using relevant sklearn libraries for these scaled values.Where
Pca: principal component value
Svd: singular value decomposition
Nmf: Non negative matrix factorization
Lda: latent dirichlet allocation
The output is in the form of

{LT-1 :{ X_4_333: 17.2384, W_16_333: 12.3521….}
LT-2 :{ X_4_333: 20.9815, W_16_333: 18.9046….}....}

Where LT-1, LT-2 are the latent semantic names and are dependent on the no of components the user has selected. The words inside latent semantics are contributions given by each word to the latent semantic and they are ordered in non-increasing order.
Then we also output transformed file, which is in the following format:

{1:{LT-1:0.00016, LT-2:0.00023….},
2:{LT-1:0.00025, LT-2:0.00014….}

Where 1, 2,.. Represent the file name and LT-1, LT-2 are the latent semantics. The values are the transformed tf values/idf values.

Task 2

In this task, we will be finding the 10 most similar gesture files in the directory with respect to the query file name passed by the user. In order to achieve this we have implemented the following algorithms -

1. Dot Product - The dot product similarity of a vector not only gives us the absolute distance between two vectors but also considers the composition of the vectors. We have used the TF values for calculating the dot product.
2. Edit Distance - We use edit distance to find the dissimilarity between two sequences. We use the quantized window descriptor as a sequence. Since edit distance gives us the cost of transforming the source file to the target file, the greater the cost less the file is similar

to the query file. In order to achieve this we use dynamic programming. The cost of a single operation is taken as 1. We take one sensor from a source and query file, compute the cost for that sensor among all X, Y, Z and W components and then add the cost for all the 20 sensors. As we know this takes quadratic time which is one of the drawbacks of using edit distance.

3. Dynamic Time Warping - Dynamic time warping is an algorithm used to find the similarity of two sequences, which may vary in time. We use average quantized amplitude as the measure to utilize inplace of the word sequence. Dynamic time warping returns the minimum cost required to match each window of the sequence with one in the second sequence. So lesser the cost, more similar the sequences are. The cost of matching is the sum of the absolute difference between the set of matched pairs. It is achieved using dynamic programming. We take one sensor from a source and query file, compute the cost for that sensor among each of X, Y, Z and W components and then add the cost for all the 20 sensors. It is solved in quadratic times which makes it slow.
4. Top k latent sensor semantics - In Task 1, we have ranked the top k most latent semantics in order to achieve k dimension where k is less than the number of original dimensions. In this step we use those values to form a new vector space with k dimensions. We use this newly formed vector space to compute how far the vector files are to the query vector. To calculate the distance we have used the Euclidean distance. The smaller the distance between the two vector files, higher is the similarity between them.

After finding the costs defined as per user input, the gestures are then ranked according to their similarity score and the top 10 similar gestures to the given query gesture are displayed to the user.

Task 3

In this task(3a and 3b), we first generate a gesture-gesture similarity matrix based on user input. We have used the transformation 1 / (1 + distance measure) to convert distance to similarity measure. The user can choose from the following options to generate the similarity matrix:

1. Dot product
2. PCA
3. SVD
4. NMF
5. LDA
6. Edit distance
7. DTW

Task 3a: Upon generating the similarity matrix, we perform Single value decomposition over this similarity matrix to obtain the dimension reduced matrix with k dimensions. The top p(user input) principle components underlying this gesture-gesture similarity matrix is reported in the form of <*gesture, score*> values, in non-increasing order of score.

Task 3b: Upon generating the similarity matrix, we perform Non-negative matrix factorization over this similarity matrix to obtain the dimension reduced matrix with k dimensions.The top p(user input) latent semantics underlying this gesture-gesture similarity matrix is reported in the form of <*gesture, score*> pairs, in non-increasing order of score.

Task 4ab

In these two tasks, we receive the top p most relevant latent semantics with the corresponding gesture contribution scores. We have taken this data in the form of a dictionary as input to both these sections from task 3a and 3b for SVD and NMF respectively. The form of the dictionary is :

{'23': {'LT-1': 0.10563676179091053, 'LT-2': -0.06547131363172337, 'LT-3': -0.021871693679146376, 'LT-4': 0.0069224396259833605, 'LT-5': -0.04394485518931037, 'LT-6': -0.0173031285397O249, ….}}

For each gesture id i we find the LatentSemantic LT-j that contributes the most to it and assign that gesture i to Latent semantic LT-j.

Task 4cd:

In these two tasks, we are clustering the given gesture-gesture similarity matrix into p clusters using k-means and laplacian based spectral clustering algorithm.
The input is not exactly a similarity matrix. It is a dictionary of similarities and is converted to a NxN numpy array of similarities where there are N gestures. A list of gesture names are maintained for indexing later.

K-means based clustering:

K-Means clustering clusters the given input into k clusters, where k is the user input. The algorithm starts with initializing the centers followed by clustering each dataset into one of the centers and then recalculating the centers based on the data in it. The algorithms run iteratively till the old center and the new center are the same which means we can't optimize the cluster any further.

There are three assumptions/implementations strategies used for implementing k-means.

1. The initial k centers are chosen randomly from the given dataset. There is another strategy to choose the centers one by one such that each center is far from the other chosen centers. However, to keep things simple, we have implemented a function which randomly selects the k centers.
2. To find the cluster of the data point, we use the euclidean distance as the metric. The distance is calculated between the datapoint and all of the centers and the data belongs to the cluster whose center is the closest to the data point.
3. The center update strategy simply averages the values of each dataset belonging to the cluster in each of the dimensions.

Laplacian based spectral clustering:

The degree matrix is calculated by summing up the degrees of all N gestures and creating a diagonal matrix. The weight matrix(similarity matrix in this case) is subtracted from the degree matrix to get the unnormalized laplacian graph matrix. The following pseudo code is then implemented to get the p clusters using this laplacian matrix.[1]

Unnormalized spectral clustering

Input: Similarity matrix $S \in \mathbb{R}^{n \times n}$, number k of clusters to construct

- Construct a similarity graph by one of the ways described in Section 2. Let W be its weighted adjacency matrix.
- Compute the unnormalized Laplacian L.
- **Compute the first k eigenvectors $v_1, \ldots, v_k$ of L.**
- Let $V \in \mathbb{R}^{n \times k}$ be the matrix containing the vectors $v_1, \ldots, v_k$ as columns.
- For $i = 1, \ldots, n$, let $y_i \in \mathbb{R}^k$ be the vector corresponding to the i-th row of V.
- Cluster the points $(y_i)_{i=1,\ldots,n}$ in $\mathbb{R}^k$ with the k-means algorithm into clusters $C_1, \ldots, C_k$.

Output: Clusters $A_1, \ldots, A_k$ with $A_i = \{j | y_j \in C_i\}$.

Interface Specification and Results:

Task 0a

```
(env) ➜  Code git:(phase2) ✗ python task0a.py
Enter the path of directory: ../3_class_gesture_data/
Enter resolution: 3
Enter window size: 3
Enter split size: 3
Calculating Guassian Band Length
Calculating Guassian Band Range
Processing folder: X
Processing folder: Y
Processing folder: Z
Processing folder: W
Saving words to file
(env) ➜  Code git:(phase2) ✗
```

The following are the output files in the task 0a:

1. .wrd: contains the symbolic quantized window descriptor
2. .awrd contains the average quantized amplitude
3. Means.json contains the average amplitude in sensor level
4. Std_dev.json contains he standard deviations in sensor level

Task 0b

```
(env) ➜  Code git:(phase2) ✗ python task0b.py
Enter the path of directory: ../3_class_gesture_data/
Reading files..
Calculating tf values..
Calculating tf idf values..
Saving vector space in the output directory. The file name is vector.txt
(env) ➜  Code git:(phase2) ✗
```

Task 1

```
(env) ➜  Code git:(phase2) ✗ python task1.py
Enter the path of directory: ../3_class_gesture_data/
Choose one of the vector representation from the following:
1. TF
2. TF-IDF
Choose 1 - 2: 2
Choose one of the model from the following:
1. PCA
2. SVD
3. NMF
4. LDA
Choose 1 - 4: 1
Enter the number of components(k): 4
Latent Semantic and new feature set created.
(env) ➜  Code git:(phase2) ✗
```

Task 2

```
(env) ➜  Code git:(phase2) ✗ python task2.py
Enter file name to compare without extention: 1
Choose one of the option to compare :
1. Dot product
2. PCA
3. SVD
4. NMF
5. LDA
6. Edit Distance
7. DTW
Enter your choice: 1
Choose one of the vector model from the following:
1. TF
2. TF-IDF
Choose 1 - 2: 2

The top 10 gestures similar to  1  are: ['1', '564', '269', '2', '259', '9', '567', '251', '7', '11']
(env) ➜  Code git:(phase2) ✗ python task2.py
Enter file name to compare without extention: 2
Choose one of the option to compare :
1. Dot product
2. PCA
3. SVD
4. NMF
5. LDA
6. Edit Distance
7. DTW
Enter your choice: 2
Choose one of the vector model from the following:
1. TF
2. TF-IDF
Choose 1 - 2: 2

The top 10 gestures similar to  2  are: ['2', '6', '577', '4', '576', '255', '3', '5', '18', '8']
(env) ➜  Code git:(phase2) ✗
```

Task 3

```
(env) ➜  Code git:(phase2) ✗ python task3.py
Choose one of the vector model from the following:
1. SVD
2. NMF
Choose 1 - 2: 1
Enter the number of components(p): 4
Choose one of the option to generate similarity matrix :
1. Dot Product
2. PCA
3. SVD
4. NMF
5. LDA
6. Edit Distance
7. DTW
Enter your choice: 1
Choose one of the vector model from the following:
1. TF
2. TF-IDF
Choose 1 - 2: 2
(env) ➜  Code git:(phase2) ✗
```

Results are stored in a file named nmf_transformed_similarity or svd_transformed_similarity depending on the user input in the output folder.

Similarity matrix creation for first five user options is considerably fast, while the edit distance and dtw takes noticeable time. It depends on the number of gesture files available as the input.

Task 4

```
(env) → Code git:(phase2) ✗ python task4ab.py
Choose one of the vector model from the following:
1. SVD
2. NMF
Choose 1 - 2: 1
LT-1 :  ['23', '256', '31', '278', '585', '569', '17', '29', '272', '265', '254', '255', '269', '22', '257', '21', '19', '25', '252', '253', '30', '18', '26', '2
51', '279', '250', '6', '586', '579', '578', '587', '7', '5', '584', '4', '580', '581', '1', '3', '583', '568', '582', '2', '564', '570', '571', '565', '559', '5
73', '567', '566', '572', '9', '589', '576', '562', '563', '577', '588', '8', '561', '575', '574', '560', '16', '249', '261', '275', '274', '260', '276', '262',
'277', '10', '273', '267', '266', '11', '13', '264', '270', '258', '259', '271', '268', '15', '263', '28', '14', '12']
LT-2 :  ['24', '20', '27']
(env) → Code git:(phase2) ✗ python task4ab.py
Choose one of the vector model from the following:
1. SVD
2. NMF
Choose 1 - 2: 2
LT-1 :  ['569', '21', '18', '251', '6', '579', '578', '5', '583', '568', '571', '573', '566', '572', '9', '562', '563', '561', '560', '249', '10', '13']
LT-2 :  ['23', '31', '24', '17', '29', '272', '22', '20', '19', '25', '30', '26', '27', '570', '8', '270', '28', '14']
LT-3 :  ['256', '278', '585', '265', '254', '269', '257', '252', '253', '279', '586', '587', '7', '584', '4', '580', '581', '1', '582', '2', '589', '588', '16',
'261', '275', '274', '276', '262', '277', '273', '267', '11', '264', '258', '271', '268', '15', '263', '12']
LT-4 :  ['255', '250', '3', '564', '565', '559', '567', '576', '577', '575', '574', '260', '266', '259']
(env) → Code git:(phase2) ✗
```

As we can see for SVD only 2 groups are shown and this is because there are 2 empty clusters/groups.

```
(env) → Code git:(phase2) ✗ python task4cd.py
Enter number of groups p: 4
Choose one of the option to compare :
1. Dot product
2. PCA
3. SVD
4. NMF
5. LDA
6. Edit Distance
7. DTW
Enter your choice: 2
Choose one of clustering technique:
1. KMeans
2. Spectral Clustering
Enter your choice: 1
1 ['569', '265', '254', '255', '269', '257', '252', '253', '18', '251', '250', '6', '7', '5', '4', '1', '3', '583', '568', '2', '564', '571', '565', '559', '573'
, '567', '566', '572', '9', '576', '562', '563', '577', '561', '575', '574', '560', '249', '261', '260', '10', '273', '266', '11', '13', '264', '258', '259', '27
1', '263']
2 ['23', '585', '21', '586', '579', '587', '580', '581', '582', '589', '588', '16', '276', '15', '12']
3 ['31', '24', '17', '22', '20', '19', '25', '26', '27', '14']
4 ['256', '278', '29', '272', '30', '279', '578', '584', '570', '8', '275', '274', '262', '277', '267', '270', '268', '28']
(env) → Code git:(phase2) ✗ python task4cd.py
Enter number of groups p: 4
Choose one of the option to compare :
1. Dot product
2. PCA
3. SVD
4. NMF
5. LDA
6. Edit Distance
7. DTW
Enter your choice: 2
Choose one of clustering technique:
1. KMeans
2. Spectral Clustering
Enter your choice: 2
1 ['256', '31', '24', '278', '17', '29', '272', '254', '22', '20', '19', '25', '253', '30', '26', '279', '27', '1', '274', '262', '273', '267', '270', '258', '27
1', '268', '15', '263', '28', '14']
2 ['275', '276', '277']
3 ['569', '265', '255', '269', '257', '252', '251', '250', '578', '7', '584', '4', '3', '583', '568', '2', '564', '570', '571', '565', '559', '573', '567', '566
', '572', '576', '562', '563', '577', '8', '575', '574', '560', '249', '261', '260', '10', '266', '11', '264', '259']
4 ['23', '585', '21', '18', '6', '586', '579', '587', '5', '580', '581', '582', '9', '589', '588', '561', '16', '13', '12']
(env) → Code git:(phase2) ✗
```

System Requirements/Installation and Execution instructions:

To run the application in your local system, follow these steps:

1. Download python 3.8
2. Create a virtual environment with the command in the application folder:
   python3.8 -m venv env
3. Activate the environment
   source env/bin/activate
4. Install all the requirements from the requirement.txt file to install all the dependencies
   pip install -r requirements.txt
5. Run the task as described in the interface specifications

Related work:

Term frequency and inverse document frequency have been used in document retrieval techniques. These features also work well to find similar gestures.

Earlier techniques for gesture recognition include HMMs[2], FSM and filtering techniques. [4] uses principal component analysis for hand gesture recognition by finding the average normalized vector for each gesture and then performing subtraction on the reduced dimension vectors.

There are many techniques for dimensionality reduction[3] which are divided into data-dependent, data independent and graph based techniques. Data dependent techniques include principal component analysis, multidimensional scaling, auto encoders, linear discriminant analysis, maximum margin criterion. Data independent techniques consist of random projection, compressed sensing, hashing trick, locality sensitive hashing. Graph based techniques consist of isometric mapping, tSNE and UMAP.

With respect to clustering, something that we have not implemented but is highly useful in practise is regarding the choice of k. Specifically for laplacian spectral clustering, we can use something called the eigengap heuristic as outlined in [1]. It describes it as follows - “we can choose the number of clusters k such that all eigenvalues $\lambda 1, \ldots, \lambda k$ are very small, but $\lambda k+1$ is relatively large.”

Conclusion:

In this project, we have used tf-values, tf-idf values combined with dimensionality reduction and similarity metrics to find out the matching gestures. Particularly, we have computed tf and tf-idf values. Then used pca[8], [9]svd, [10]nmf and [7]lda for dimensionality reduction. We compute the gesture-gesture similarity matrix and use this to calculate the score for each gesture.

Using the most important latent semantics and the contributions of each gesture to them, we were able to group gestures based on those contribution scores. This can be viewed as a simple efficient way of clustering gestures together using SVD or NMF based contribution scores. We

also explored more computation intensive clustering techniques like kmeans and spectral clustering, both of which gave differing results. In general spectral clustering performs better than k-means in terms of quality of clusters and this was apparent in the results as we saw similar gestures being in the same cluster in spectral clustering.

Bibliography:

- [1]Technical Report No. TR-149 A Tutorial on Spectral Clustering Ulrike von Luxburg1 August 2006
- [2] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," Proc. IEEE, vol. 77, no. 2, pp. 257–285, Feb. 1989.
- [3]Bahri, Maroua, et al. "Survey on feature transformation techniques for data streams." *International Joint Conference on Artificial Intelligence (IJCAI)*. 2020.
- [4]Kumar, Sunil & Srivastava, Tanu & Singh, Raj. (2017). Hand Gesture Recognition Using Principal Component Analysis. 6. 2249-68.
- [5] 515 Class Notes by Prof. K. Selcuk Candan
- [6]Project Description by Prof. K. Selcuk Candan
- [7]David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation.J. Mach. Learn. Res. 3 (March 2003), 993–1022.
- [8]Jolliffe, Ian T., and Jorge Cadima. "Principal component analysis: a review and recent developments." *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 374.2065 (2016): 20150202.
- [9]Kalman, Dan. "A singularly valuable decomposition: the SVD of a matrix." *The college mathematics journal* 27.1 (1996): 2-23.
- [10]Lee, Daniel D., and H. Sebastian Seung. "Learning the parts of objects by non-negative matrix factorization." *Nature* 401.6755 (1999): 788-791.

Appendix:

The following members worked on the following sections of phase 2. Everyone collaborated and worked on the report.

Adithya

1. Task 0a
2. Task 0b
3. Task 1

Nawendu

1. Task 0a
2. Task 0b
3. Task 1

Vaibhav

1. Task 3a(2, 3)
2. Task 3b(2, 3)
3. Task 4c
4. Refactoring

Zain

1. Task 2(2, 3, 7)
2. Task 3a(2, 3, 7)
3. Task 3b(2, 3, 7)

Varoon

1. Task 2(1, 4, 5, 6)
2. Task 3a(1, 4, 5, 6)
3. Task 3b(1, 4, 5, 6)

Daniel

1. Task 4a
2. Task 4b
3. Task 4d