

MULTIMEDIA AND WEB DATABASES PROJECT

(Phase #3: indexing, classification and relevance feedback)

ABSTRACT

In phase 1, for a given multi-variate data set of gesture files, each uni-variate time series in a gesture file is associated with gesture vectors namely TF, TF-IDF and TF-IDF2. In phase 2, we mainly experimented with dimensionality reduction and unsupervised learning. In this phase, we implemented indexing, classification and relevance feedback techniques on the deliverables of phase 2 where we performed dimensionality reduction on Phase 1 deliverables. We have indexed the data by implementing Locality Sensitive Hashing (LSH) and used this index structure for similarity search. We have classified the data into different classes based on the given training data by using the KNN, Decision tree and n-ary SVM classifiers. We have also implemented relevance feedback which allows the user to provide feedback and improve the search results with relative importance of different features in the query.

Keywords:

Clustering, indexing, classification, relevance feedback, Locality Sensitive Hashing, K-Nearest Neighbor, Decision Tree and Support vector machine.

Group #7

1. Varun Siruvoru
2. Satya Swaroop Boddu
3. Mohanraj Balumuri
4. Raghuveer Nanduri

INTRODUCTION

Terminology:

- **Clustering:** Grouping a set of objects in such a way that objects in the same group are more similar to each other than to those in other groups is called clustering.
- **Indexing:** Indexing collects, parses and stores the data to enable fast and accurate information retrieval.
- **Classification:** Based on the training set of labeled data, the new data will be classified into one of the labels (categories) of training set labeled data.
- **Relevance feedback:** In Relevance Feedback, we initially compute the results for a given query and to use that information for determining whether or not those results are relevant to perform a new query.
- **Locality Sensitive Hashing:** In Locality Sensitive Hashing, we hash the input data, so that similar data items are mapped to the same buckets with high probability.
- **K-Nearest Neighbor:** K-Nearest Neighbor Algorithm is a method of classification that predicts class memberships of new data based on the class memberships of K closest training data that was provided.
- **Decision Tree:** Decision Tree based classification algorithm is a predictive model which maps observations about an item to conclusions about the item's target value.
- **Support vector machine:** SVM takes a set of input data and predicts, for each given input, which of two possible classes forms the output, making it a non-probabilistic binary linear classifier. Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples into one category or the other.

Locality Sensitive Hashing (LSH):

The multidimensional index structures like grid files, KD trees, R trees, SR trees, X trees, TV trees etc. can be seen as locality sensitive hashes but are deterministic, which tends to map the closer points to the same or nearby hash value. LSH is also a locality sensitive hashing based technique but the hash functions used are random or nondeterministic. Hash functions used in LSH should satisfy the constraint

$$\text{Prob}(h(O1)=h(O2)) = \text{Sim}(O1,O2)$$

where $h()$ is a hash function, $\text{Sim}()$ is the similarity function of a vector space containing the objects $O1$ and $O2$. Thus the hash functions used in LSH map the similar objects to the same hash value with very high probability. Since LSH is a random hash based technique, there are chances of mapping dissimilar objects also to same hash value. To reduce these false positives, instead of using a single hash function, LSH employs a family of hash functions H . Let $P1$ be the probability that a hash function in H , maps similar objects to same hash value. Let ' k ' be the number of hash functions used. Thus the probability of false positives will be reduced by ' $P1^k$ '. At the same time

the probability of mapping similar objects to same hash value is also reduced by 'P1k'. This might result in misses. To reduce these misses, more than one layer of hash functions are introduced instead of single layer. Let H be a family of hash functions, then

$$gj(O) = (h_{1,j}(O), h_{2,j}(O), h_{3,j}(O), \dots, h_{k,j}(O))$$

Where $h_{i,j}$ belongs to H and $gj()$ is a composite hash function composed of 'k' different hash functions from H. Each $gj()$ represents a layer of hash functions. LSH contains 'L' layers of such hash functions each again containing 'k' hash functions, thus resulting in a total of 'kL' hash functions. Thus if two similar objects miss, going into a different hash value in a layer, they might result in same hash value in one of the 'L' layers, thus again pulling up the probability of similar objects going to same hash value linearly. Let O be an object in a vector space, then $Sign(gj(O))$ determines the bucket to which O goes in each layer.

Given a dataset, these family of hash functions are executed on each object in the dataset and are placed into the corresponding buckets. Now for a query object Q all the hash functions are again executed, $Sign(gj(Q))$ determines all the buckets that the query object goes and all the data objects that already exist in these buckets are retrieved, which probably indicate the objects similar to query Q.

Decision Tree Classifier:

Decision tree is a supervised machine learning technique which partitions the dataset recursively until all the partitions/leaf nodes consists of a single class. This does a better classification then KNN and also is quicker. Each non-leaf node contains a "split point" for a given feature based on which test data is classified. A decision tree classifier is developed in two phases. 1. Growth phase and 2. Prune Phase.

In Growth phase, we form the decision rules and tree. Rules can be developed in two ways.

1. Entropy:

In this, we calculate the entropy after splitting and before splitting and calculate information gain at that step.

2. Fishers Discriminant Ratio:

In this we calculate the means and variance and take the features which have low variance and far means for splitting at node level. It works better for two class labels.

$$FSR(f_i) = \frac{(\mu_{i,1} - \mu_{i,2})^2}{\sigma_{i,1}^2 + \sigma_{i,2}^2}$$

As we are having multiple classes and more features in this dataset, we considered entropy for building the tree.

Support Vector Machines (SVM):

Support Vector Machines is a binary classification algorithm i.e. that classifies the objects into two classes only. Consider there are N training data points, x_i $1 \leq i \leq N$, with D dimensionality and are one of the two classes. The training data can be represented in the below form.

$$\{x_i, y_i\} \text{ where } i = 1, 2, 3, \dots, N ; y_i \text{ belongs to } \{-1, 1\}$$

Consider the case where this data is linearly separable. Thus if $D=2$, a line can be drawn that separates the points of these two classes of the training data. For $D > 2$, a hyperplane can be drawn that separates the objects of the two classes.

The data points in the training data set that are closest to this separating plane from both the sides are termed as Support Vectors. Two planes can be considered parallel to this separating plane one on each side, passing through these support vectors. The plane on one side can be described by $w \cdot x + b = 1$ (H_1) and the plane on other side can be described by $w \cdot x + b = -1$ (H_2). All the training data points that belongs to the class of point described by first equation satisfy the constraint $w \cdot x + b \geq 1$ and the data points of the later satisfy $w \cdot x + b \leq -1$. When both combined, all the points satisfy $y(w \cdot x + b) - 1 \geq 0$.

The plane that is farthest from both the planes H_1 and H_2 is chosen as the separating plane, which helps in the correct classification results in cases of small errors in calculation. This separating hyperplane can be described by $w \cdot x + b = 0$ where w is normal to the hyperplane and $b/(||w||)$ is the perpendicular distance of the origin from the hyperplane. Let d_1, d_2 be the distances of H_1 and H_2 from the separating hyperplane. In this case $d_1 = d_2$ and $d_1 + d_2$ is called as the SVM's margin, which is mathematically equal to $1/(||w||)$. Since a big margin is desired, SVM can be seen as an optimization problem that either maximizes $1/(||w||)$ or minimizes $||w||$. This optimization problem when solved returns the values of w and x .

To find out the label or class of an unlabeled data point Q , $\text{sign}(w \cdot Q + b)$ will decide the class of Q i.e. if this value is ≥ 1 , then Q belongs to first class denoted by H_1 and if this value is ≤ -1 , then Q belongs to the other class.

If the number of classes is > 2 , then multiple binary SVM's are executed, each with a unique combination of classes. Hence if there are n classes, $nC_2 = n(n-1)/2$ binary SVM's are executed.

Goal Description:

We have implemented 4 tasks (2, 3, 5, and 6) in this phase:

The Goal of Task 2: The goal of this task is to implement the Locality Sensitive Hashing (LSH) algorithm, which takes following inputs - the number of layers, L , the number of hashes per layer, k , and a set of vectors as input and create an in-memory index structure containing the given set

of vectors. For a given gesture, we also need to search a similar gesture using this index structure formed by using LSH and outputs the t most similar gestures.

The Goal of Task 3: In this task we need to implement 3 classification algorithms which are k-nearest neighbor based classification algorithm, a decision tree based classification algorithm and an n-ary SVM based classification algorithm. These algorithms take a CSV file which contains the file numbers with corresponding classes and associates a class to the rest of the gestures in the database.

The Goal of Task 5: In this task, we need to implement a decision-tree based relevance feedback algorithm to improve nearest neighbor matches based on the relative importance of sensors or X,Y,Z, and W components or the relative importance of the different words. For a given query, the system has to return k similar files. And the user gives his feedback based on those results and then the system returns revised outputs and this process continues every time the user enters his feedback.

The Goal of Task 6: In this task, we need to implement an interface for Task 5 where the user is allowed to provide a query, relevant query parameters and number of output files to be shown based on their rank (i.e., decreasing order of matching). Based the results shown, the user should be able to enter his feedback to the system (i.e., which files are relevant and which are not). Now based on the user's feedback, the system returns a new set of revised output. And this process can continue iteratively.

Assumptions:

1. As a part of preprocessing, the system computes tf-idf values for the given gesture files.
2. In the next step of preprocessing, the system implements dimensionality reduction using PCA technique on the given data and then performs indexing, classification and relevance feedback on that data.

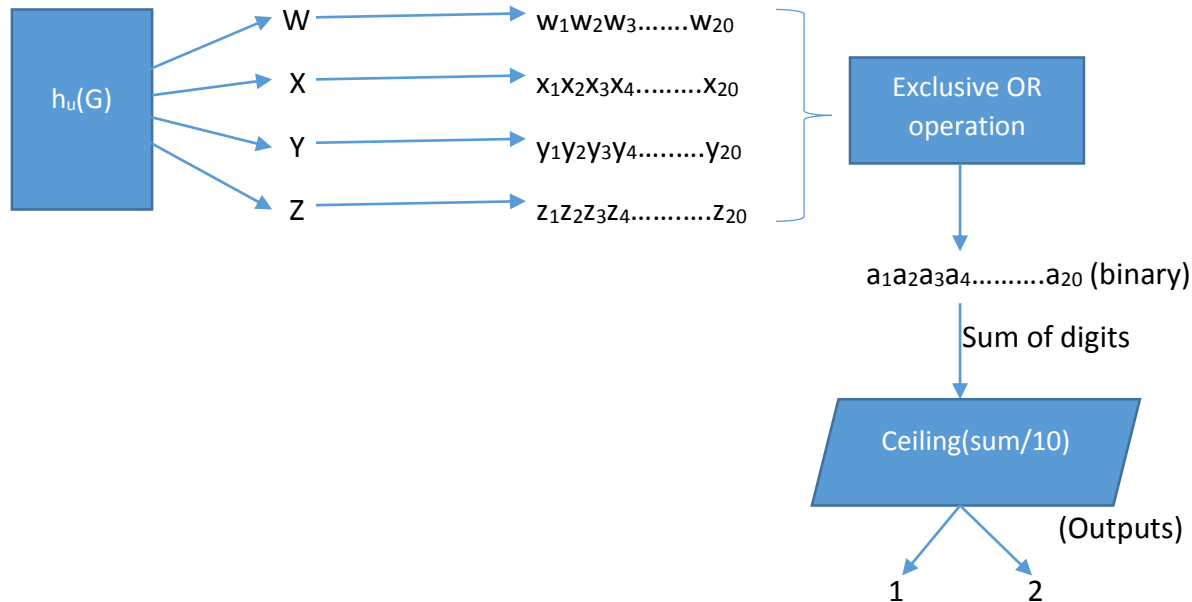
IMPLEMENTATION

Task 2: Locality Sensitive Hashing (LSH):

In part 1 of this task, an in-memory index structure based on LSH technique is to be developed, for the gestures in the given dataset. We are using the data that is projected into the vector space that is formed by the top 3 principal components of each sensor. Since each sensor is projected on to its corresponding top 3 PC's, if there are 20 sensors in a data object, the data object in the projected space will be a 20 X 3 matrix. The crux of this task is the formation of the hash functions. We are constructing the hash functions based on the logic of "Arccos", which uses the cosine similarity as the distance metric. The hash functions can be seen as hyperplanes that divide the space into two half spaces. To form a hash function, we are using a random unit length vector where each coordinate of this vector is picked from Gaussian distribution. Let u be a unit vector and p be a data point vector. The hash function represented by u is defined as

$$h_u(p) = \begin{cases} 1 & \text{if } u \cdot p \geq 0 \\ 0 & \text{if } u \cdot p < 0 \end{cases}$$

In our case, as a gesture is a 20 X 3 matrix, the unit length vector is a 1X3 matrix. The inner product is performed between this unit vector and each sensor data of a gesture, thus resulting in a 20 digit binary string. As we have 4 different components namely W, X, Y and Z, performing inner product of this unit vector with a gesture in each component will result in 4 binary strings of 20 digits. We are performing an exclusive OR operation among these 4 binary string, thus resulting in a single binary string.



To output a single digit hash value for a gesture, we are adding up all the digits of this binary string and then taking the Ceiling value of the quotient when the sum of digits is divided by 10. Thus each hash function either outputs 1 or 2. Thus in each layer, if there are 'k' hash functions, the output of each function is among {1, 2}. All the outputs of 'k' functions are concatenated and is the hash value of the composite hash function $g_j(G) = (h_{1,j}(G), h_{2,j}(G), \dots, h_{k,j}(G))$. In order to differentiate the hash value of one layer with other, the layer number is appended in the beginning of this concatenated string of hash function outputs. This string denotes the hash value of a layer. A sample hash value is shown below

9 1 2 1 2 1 2 1 2
 ↓ ←————→
 Layer No outputs of 'k' Hash functions

Corresponding to each unique hash value in a layer, a bucket is maintained with the list of all gestures that map to this same unique hash value.

In part 2 of this task, the index structure constructed above is used for the similar gesture search. For a given input query gesture and a 't', the LSH index structure returns 't' gestures that are probably more similar to the query gesture. The search technique is executed as follows. First the same family of hash functions, that are used above to construct the index structure, are executed on the query gesture. All the gestures from the buckets that are corresponding to the hash values of the query gesture are retrieved. Then the distance between the query gesture and each of the retrieved gesture is calculated using cosine similarity. The top 't' gestures that are closer to the query are returned as the result.

If the number of gestures retrieved from the buckets is $< t$, then the gestures in the buckets whose corresponding hash values are almost similar to the query hash values are retrieved. The number of mismatches allowed start from 1 until t files are retrieved.

Task 3: Gesture classification

- a. **K-nearest neighbor based classification algorithm:** In this task 3a, we have implemented a k-nearest neighbor based classification algorithm. It predicts the class memberships of unclassified data based on the classified data which is provided as training data set. For this function, the system takes the database path and the k value as input. Here k is the number of nearest neighbors that you want to consider for voting. You also need to provide labels.csv file which contains the gesture file numbers and their corresponding classes. These gestures are considered as classified and the rest of the gestures as unclassified. Now for each unclassified gesture, we compute its distance to all the classified gestures. We have use **cosine similarity** as our distance function. The output of cosine similarity between 2 gestures is computed as a summation of cosine similarity between the 2 files of all components. The cosine similarity of 2 files in a component is

the summation of the cosine similarity between the tfidf values of all the sensors. Therefore cosine similarity between 2 files will be 20 if they are perfectly same. And hence, when 2 gestures are perfectly same then their cosine similarity will be 80 (4 components, 20 sensors, cosine similarity value: 1). Now we sort the matrix of gestures based on their corresponding cosine similarity values with the unclassified gesture and take the top k similar gestures. And now we take the votes of each gesture i.e., we consider their corresponding classes as their vote for the unclassified gesture to be in that class. Based on the majority, a class will be assigned to the unclassified gesture. In case of a tie, the unclassified gesture will assigned the class of a gesture which is closest (Most Similar) to it. In this way, all the unclassified gestures will be classified based on the classified data (Training data) and the output will be shown to the user which consists of unclassified gesture names with their corresponding assigned classes.

b. Decision tree based classification algorithm:

The tree is built by recursively partitioning the given dataset until each non-leaf node is either small (user defined parameter) or “pure”. Split point plays an important role in building a good classifier. This is usually chosen based on the type of training data provided. We considered “Binary split” as it usually lead to more accurate trees.

- *Finding Split points:* Goal of this phase is to determine the “best split” at each node that best divides the provided training dataset. Value of the split condition depends on how well it separates the classes. We implemented this using Entropy and information gain.
- *Information Gain:* We calculate information gain for each feature and set the feature with maximum information gain as the node at this particular layer. As we have continuous data we check each and every split point for feature to find which will have the maximum information gain. For every feature we form an attribute list and associate it with this node. For every feature, we find a split point at which it has maximum information gain. Process is explained below:

Attribute List:

Feature1	Class	Object ID
17	A	1
20	A	2
23	B	3
24	B	4
25	B	5

For this attribute list, we verify information gain at each point to find the split with maximum information gain. And we get maximum information gain at 23. We repeat this process for all the attribute lists and find the correct split with maximum information gain and finally assign the node with the attribute with

maximum information gain at the corresponding split. In this scenario, the split will be $\text{feature1} \geq 23$ will right sub tree while $\text{feature1} < 23$ will be left sub tree.

- Information Gain formula:

We calculate entropy before and after the division.

$$\text{Entropy} = -p(a) \cdot \log(p(a)) - p(b) \cdot \log(p(b))$$

$$\text{Information Gain} = (\text{Entropy before}) - (\text{Entropy after})$$

- General Tree-Growth Algorithm:

Partition (Data S)

 If (all points in S are of the same class) then

 Return;

 For each attribute A do

 Evaluate splits on attribute A ;

 Use best split found to partition S into $S1$ and $S2$;

 Partition ($S1$);

 Partition ($S2$);

- Initial call: Partition (Training Data)

Steps taken to avoid over fitting:

In pruning phase, the tree is pruned to remove any dependencies that are specific to the training dataset. This increases the accuracy of the test data. Growth phase is usually more expensive than the Prune phase since the data is scanned multiple times while building the classifier. We did not use any pruning for the tree but we used supporting number of objects in each leaf node. After forming the tree, we check the supporting number of objects for each leaf node i.e. label. If there are less than three objects, then we don't consider that node as classifier. The leaf node will be ignored and parent feature node will be ignored while the right sub tree will be moved a layer up. We also made mark a node as leaf node it has 80% of the labels as homogenous.

Implementation:

We used java to implement the decision trees. We used Hashmaps and custom java classes to implement decision tree in java. We used recursive function calls to form and parse the tree.

Advantages:

Main advantages of a decision tree based classifiers are a) Simple and easy to understand and interpret. b) Doesn't require much data preparation. c) Can handle Numerical, Categorical and Continuous data. d) Follows White box model i.e., Unlike Black box model, any observation can be expressed in the form of simple conditions and Boolean logics. e) Performs well with large datasets.

c. **n-ary SVM based classification algorithm**

In part 3 of this task, an n -ary SVM based classification algorithm is to be implemented. The data given can be divided into two parts namely trained data and unlabeled data. If there are n classes in the given trained data, then nC_2 binary SVM's are executed to decide the class of unlabeled gestures. As mentioned in the description, an SVM algorithm boils down to an optimization problem that tries to minimize $\|w\|$ where each point satisfies $y_i(w \cdot x_i + b) - 1 \geq 0$, where $0 \leq i \leq L$, $y_i \in \{-1, 1\}$ and x_i is data point. As minimizing $\|w\|$ is same as $1/2(\|w\|^2)$, this can be seen as a quadratic programming problem. To satisfy the constraints Lagrange multiplier is used and the equation comes as

$$\sum_{i=1}^L (\alpha_i) - \left(\frac{1}{2}\right) (\alpha^T H \alpha) \text{ such that } \alpha_i \geq 0 \text{ and } \sum_{i=1}^L (\alpha_i y_i) = 0$$

where L is the number of points. Keeping derivations aside, w can be found out from α using $w = \sum_{i=1}^L \alpha_i y_i x_i$ and b can be found by substituting a point in the equation after w value is known. The value of H is chosen as $H_{ij} = y_i y_j (x_i \cdot x_j)$ since our data is linearly separable. For each combination of the classes, H is calculated using each of the data points that belongs to these both classes. Thus it will form a symmetric matrix. We are using Matlab function "quadprog" to solve the equation.

$$\alpha = \text{quadprog}(H, f, A, b, Aeq, beq, lb, ub, \alpha_0, options)$$

The equation that this function solves is of form $(1/2) * x' * H * x + f' * x$. By multiplying our equation with -1 and comparing it with this equation, f is a column vector of ones. As there are no inequalities in constraints A and $b = []$. To satisfy the equality constraint $\sum_{i=1}^L (\alpha_i y_i) = 0$, $Aeq = \text{transpose}(y)$ where $y = [y_i]$ $beq = 0$. As $\alpha_i \geq 0$, $lb = \text{zeros}()$ matrix and $ub = []$. In options we are using `interior-point-convex` algorithm.

Once this function returns α , the values of w and b are found out. After finding the values of w and b , the $\text{sign}(w \cdot q_i + b)$ decides the class/label of each unlabeled point q_i i.e. if this value is ≥ 1 the point belongs to class denoted by $w \cdot x + b = 1$, if this value is ≤ -1 the point belongs to class of $w \cdot x + b = -1$.

Like this, we construct each binary SVM based on the training data and decide the classes of each of the unlabeled gestures. The class that is determined in majority of the binary classifiers, will be likely the class of an unlabeled gesture. So we are considering the class that is determined in majority of binary classifiers as the class of an unlabeled gesture. There are two corner cases to be handled in this kind of classification. One is the class of a query point is determined by neither of the classifiers i.e. the value of $-1 < \text{sign}(w \cdot q_i + b) < 1$ and the other is when each class gets equal majority. In these cases,

as mentioned in the class, we are calculating the distance of the query point to each of the separating hyperplanes. The hyper plane that is far from the query point is selected, then if the value is less than zero we are assigning the class of $w \cdot x + b = -1$ else if the value is greater than zero we are assigning the class of $w \cdot x + b = 1$.

Task 5 and 6: Classifier-based relevance feedback with interface:

In this task, we implemented a decision-tree based relevance feedback algorithm to improve nearest neighbor matches. For this function, the system takes the following input from the user – number of results to be returned, the path of the query gesture and the path of the database. As the input query gesture will be raw data, first tfidf values will be computed and that data will be projected into the new dimensions which are the principal components of PCA algorithm. Then cosine similarities will be computed between the query gesture and all the gestures in the database. Now, the top k most similar gestures will be shown to the user where k is the initial user input for number of results to be returned.

If the user is not satisfied with the results, then the user enters the relevant and irrelevant gestures from the top k similar gestures, into the system. Here the relevant gestures that the user have given will be assigned the class 'R' (Relevant) and the irrelevant gestures given by the user will be assigned the class 'I' (Irrelevant). Now this classified data will be given to the Decision tree based classification algorithm which produces a decision tree by using this training data set. Now all the unclassified gestures in the database will be parsed through this decision tree and will get some class assigned. Now all the gestures in the database will be classified as either relevant 'R' or irrelevant 'I' classes. Now again cosine similarities will be computed between the query gesture and all the relevant 'R' class gestures in the database that are identified by the decision tree. Now, the top k most similar gestures will be shown to the user.

If the user is not satisfied with the results, then the user will again enter the relevant and irrelevant gestures from the top k similar gestures, into the system. Here again the gestures will be classified into relevant and irrelevant classes and sent to the Decision tree based classification algorithm to train the decision tree again. The relevant gestures of the previous decision tree which are classified as irrelevant by the user are noted and the features that classified them as relevant in the previous decision tree are now ignored in the new decision tree training. In this way we are giving relative importance to sensors. Now a new decision tree will be formed from the new training data and the remaining gestures will be classified into relevant and irrelevant classes again by parsing them through the decision tree. This produces a new set of relevant and irrelevant gestures. Now again cosine similarities will be computed between the query gesture and all the new relevant gestures in the database. Now, the top k most similar gestures will be shown to the user. If the user is still not satisfied, then he will again give the relevant and irrelevant gestures to the system and the process continues iteratively until the user is satisfied.

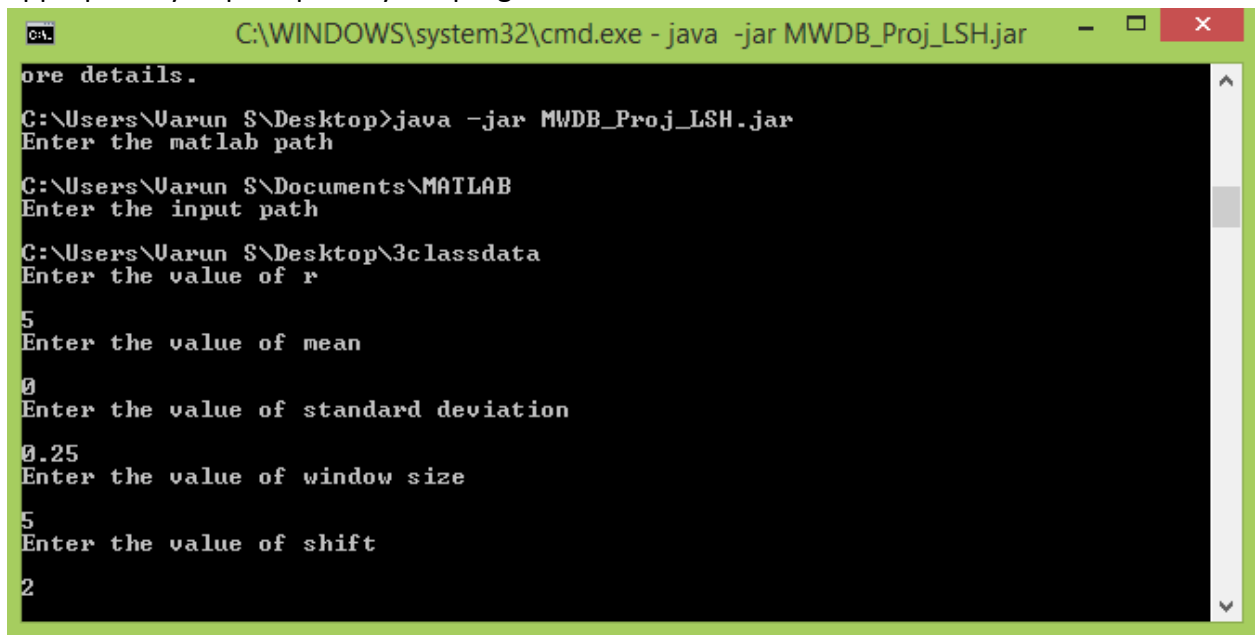
INTERFACE SPECIFICATIONS

1. Execute the jar file in the command prompt
2. When asked for "matlab path", please enter the path where matlab code is placed.
3. When asked for "input path", please enter the path where input gesture files are placed.
4. When asked for "r" (resolution), "mean", "standard deviation", "window size" and "shift value", please enter the corresponding values.
5. Each task is present in each jar file and all the tasks should be executed in sequence to get desired output.
6. Labels should be given in labels.csv file in the database path
7. The query gesture should be given in a folder called 'test' in the database path.

EXECUTION INSTRUCTIONS

Task 2

1. Run the MWDB_Proj_LSH.jar from the command prompt. Then give the inputs appropriately as prompted by the program.



```
C:\WINDOWS\system32\cmd.exe - java -jar MWDB_Proj_LSH.jar
ore details.
C:\Users\Varun S\Desktop>java -jar MWDB_Proj_LSH.jar
Enter the matlab path
C:\Users\Varun S\Documents\MATLAB
Enter the input path
C:\Users\Varun S\Desktop\3classdata
Enter the value of r
5
Enter the value of mean
0
Enter the value of standard deviation
0.25
Enter the value of window size
5
Enter the value of shift
2
```

2. After entering the values shown in the above screenshot, the program forms the database by calculated TF, TF-IDF2, TF-IDF2, PCA on the sensors, projecting the data etc.
3. Then the program prompts the user to enter the number of layers and number of hash functions per layer for LSH.

```

5
Enter the value of shift
2
Enter the number of layers
15
Enter the number of hash functions per layer
10

```

4. Then the user is prompted for entering the test gesture name that exists in the “test” folder at data “input path”. Also the number of similar gestures required is also asked. The results are displayed as shown in the below screenshot. Similar gestures for other gestures can be extracted by repeating this step, else ‘quit’ can be entered to exit the program.

```

C:\WINDOWS\system32\cmd.exe - java -jar MWDB_Proj_LSH.jar
568.csv
15.csv
Enter the test file name<enter 'quit' to exit>
270
Enter the number of similar files required
15
Overall no of gestures considered: 44
No of unique gestures considered: 26
270.csv
271.csv
272.csv
276.csv
275.csv
273.csv
279.csv
278.csv
30.csv
277.csv
571.csv
15.csv
581.csv
578.csv
31.csv
Enter the test file name<enter 'quit' to exit>

```

Task 3-1

1. Open Cmd prompt and go to the jar files folder path and run Task3-1.jar by using the command: Java -jar Task3-1.jar

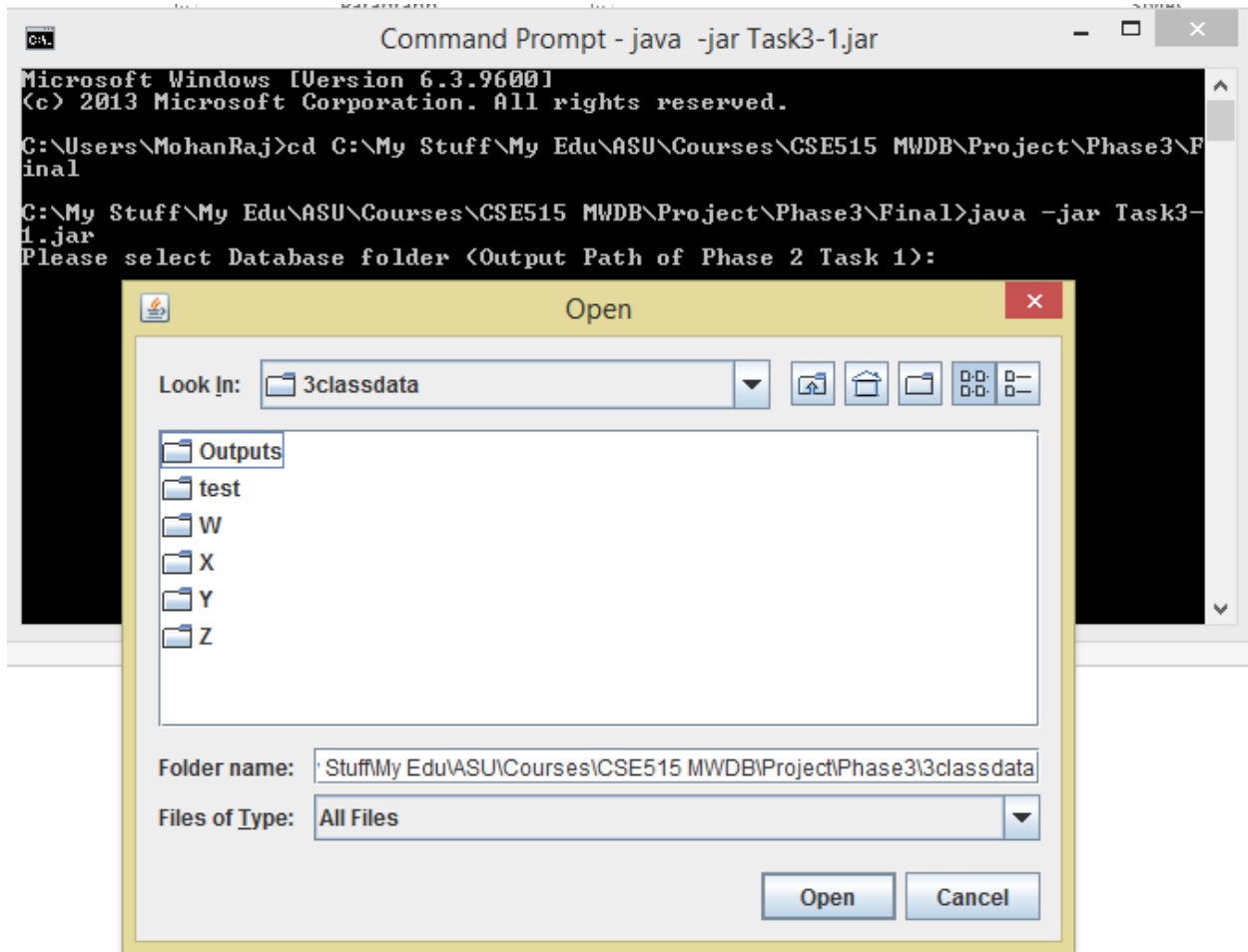
```

Command Prompt
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\MohanRaj>cd C:\My Stuff\My Edu\ASU\Courses\CSE515 MWDB\Project\Phase3\Final
C:\My Stuff\My Edu\ASU\Courses\CSE515 MWDB\Project\Phase3\Final>java -jar Task3-1.jar

```

2. Select the input folder that consists of database



3. Now enter k value.

```
C:\My Stuff\My Edu\ASU\Courses\CSE515 MWDB\Project\Phase3\Final>java -jar Task3-1.jar
Please select Database folder <Output Path of Phase 2 Task 1>:
Folder selected:C:\My Stuff\My Edu\ASU\Courses\CSE515 MWDB\Project\Phase3\3classdata
Enter K value:
5
```

4. Now MatLab pops up, enter matlab path

```
C:\My Stuff\My Edu\ASU\Courses\CSE515 MWDB\Project\Phase3\Final>java -jar Task3-1.jar
Please select Database folder <Output Path of Phase 2 Task 1>:
Folder selected:C:\My Stuff\My Edu\ASU\Courses\CSE515 MWDB\Project\Phase3\3classdata
Enter K value:
10
Enter Matlab Path:
C:\My Stuff\My Edu\ASU\Courses\CSE515 MWDB\Project\Phase3\Final\Code\MatlabCode
```

5. Now MatLab computes and generates the output.

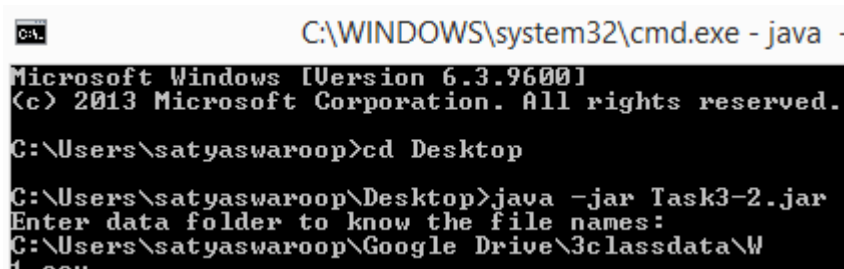
output =

```
'26.csv'      'A'
'27.csv'      'A'
'274.csv'     'B'
'275.csv'     'B'
'276.csv'     'B'
'277.csv'     'B'
'278.csv'     'B'
'279.csv'     'B'
'28.csv'      'A'
'29.csv'      'A'
'30.csv'      'A'
'31.csv'      'A'
'584.csv'     'C'
'585.csv'     'C'
'586.csv'     'C'
'587.csv'     'A'
'588.csv'     'C'
'589.csv'     'C'
```

Note: labels.csv file should be in database path.

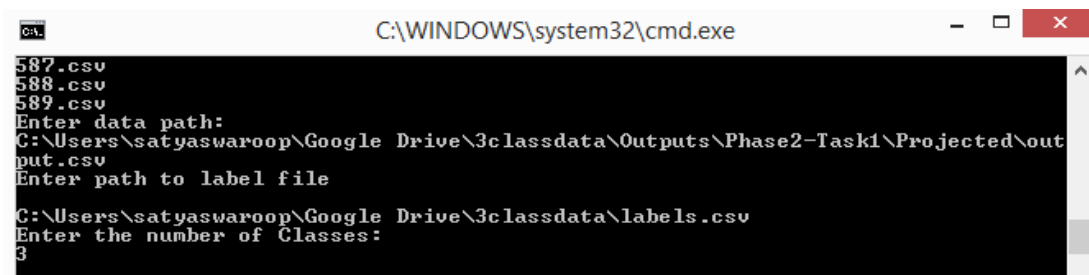
Task 3-2

1. Open Cmd prompt and go to the jar files folder path and run Task3-2.jar by using the command: Java -jar Task3-2.jar



```
C:\WINDOWS\system32\cmd.exe - java -
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.
C:\Users\satyaswaroop>cd Desktop
C:\Users\satyaswaroop\Desktop>java -jar Task3-2.jar
Enter data folder to know the file names:
C:\Users\satyaswaroop\Google Drive\3classdata\W
t.csv
```

2. Enter require inputs as prompted for.



```
C:\WINDOWS\system32\cmd.exe
587.csv
588.csv
589.csv
Enter data path:
C:\Users\satyaswaroop\Google Drive\3classdata\Outputs\Phase2-Task1\Projected\out
put.csv
Enter path to label file
C:\Users\satyaswaroop\Google Drive\3classdata\labels.csv
Enter the number of Classes:
3
```

3. This is the tree formed by algorithm on console.

```
Value: 0.75429 Children: Right:130 Children: Left:86
Feature: 90 Value: 0.18045 Children: Right:0 Children: Left:5
Class Label:A Parent:130
Feature: 5 Value: 0.88086 Children: Right:0 Children: Left:0
Class Label:C Parent:5
Class Label:B Parent:5
Feature: 86 Value: 0.03635 Children: Right:0 Children: Left:106
Class Label:B Parent:86
Feature: 106 Value: -0.012127 Children: Right:0 Children: Left:5
Class Label:A Parent:106
Feature: 5 Value: 0.25777 Children: Right: null Children: Left:0
Class Label:A Parent:5
```

4. Following are the contents of output file.

589.csv:A
21.csv:A
30.csv:A
22.csv:B
585.csv:C
270.csv:B
579.csv:A
588.csv:C
274.csv:C
31.csv:A
23.csv:A
586.csv:C
26.csv:A
278.csv:B
580.csv:C
581.csv:C
583.csv:C
29.csv:C
587.csv:C
275.csv:B
24.csv:A
272.csv:B
277.csv:B
279.csv:B
582.csv:C
584.csv:C
273.csv:B
271.csv:B
27.csv:C
276.csv:B
28.csv:C
269.csv:B
25.csv:B

Task 3-3

1. Run MWDB_Proj_SVM.jar from the command prompt. Then give the inputs appropriately as prompted by the program, as shown in the below screenshot.

```
C:\Users\Varun S\Desktop>java -jar MWDB_Proj_SVM.jar
Enter the matlab path

C:\Users\Varun S\Documents\MATLAB
Enter the input path

C:\Users\Varun S\Desktop\3classdata
```

2. The program considers the labels.csv as training data, then it finds out the classes or labels of the gestures missing in the labels.csv i.e. unlabeled gestures and then outputs it in "Phase3-Task3\SVM" under "Outputs" directory. A screenshot is shown below.

	A	B	C	D	E	F	G
1	27	Class 1					
2	28	Class 1					
3	29	Class 1					
4	30	Class 1					
5	31	Class 1					
6	275	Class 2					
7	276	Class 2					
8	277	Class 2					
9	278	Class 2					
10	279	Class 2					
11	585	Class 1					
12	586	Class 3					
13	587	Class 2					
14	588	Class 3					
15	589	Class 3					
16							

Task 5-6

1. Open Cmd prompt and go to the jar files folder path and run Task5and6.jar by using the command: Java -jar Task5and6.jar

```
Command Prompt - java -jar Task5and6.jar

C:\My Stuff\My Edu\ASU\Courses\CSE515 MWDB\Project\Phase3\Final>java -jar Task5a
nd6.jar
```

2. A java interface will popup and we need to enter required input such as k value, database path, query path and matlab path.

MWDB - Phase 3

Input details

Enter K value:

Database Input Path:

Query Input Path:

Matlab Path:

Relevant file no.s with comas:

Irrelevant file no.s with comas:

3. A MatLab will pop up and computes the top k similar gestures for the query gesture and prints them on the right side of the java interface.

MWDB - Phase 3

Input details

Enter K value:

Database Input Path:

Query Input Path:

Matlab Path:

Query Path is: C:\My Stuff\My Edu\ASU\Courses\CSE515 MWDB\Project\Phase3\3classdata\test

K value is: 15

Database Path is: C:\My Stuff\My Edu\ASU\Courses\CSE515 MWDB\Project\Phase3\3classdata

Query Path is: C:\My Stuff\My Edu\ASU\Courses\CSE515 MWDB\Project\Phase3\3classdata\test

-----Top K Similar Files-----

- [1] 270.csv
- [2] 271.csv
- [3] 272.csv
- [4] 276.csv
- [5] 275.csv
- [6] 273.csv
- [7] 269.csv
- [8] 274.csv
- [9] 263.csv
- [10] 264.csv
- [11] 279.csv
- [12] 267.csv
- [13] 589.csv
- [14] 278.csv
- [15] 30.csv

4. If the user is not satisfied with the result he can give the relevant and irrelevant files on the left side of the java interface. Here my query gesture is 270. The first result shown to

the user shows that first 13 files belong to the class of 270. Therefore, the user enters 589 and 30 file numbers in irrelevant field and rest in relevant field and submits the data to the system to get improved results.

The screenshot shows the 'MWDB - Phase 3' application window. On the left, the 'Input details' section contains the following fields and values:

- Enter K value: 15
- Database Input Path: E515 MWDB\Project\Phase3\classdata
- Query Input Path: 5 MWDB\Project\Phase3\classdata\test
- Matlab Path: \Project\Phase3\Final\Code\MatlabCode

Below these fields are two 'Submit' buttons. The bottom section of the input details contains two text boxes for file numbers:

- Relevant file no.s with comas: 263.csv,264.csv,279.csv,267.csv,278.csv
- Irrelevant file no.s with comas: 589.csv,30.csv

On the right side of the window, the output area displays the following information:

- Query Path is: C:\My Stuff\My Edu\ASU\Courses\CSE515 MWDB\Project\Phase3\classdata
- K value is: 15
- Database Path is: C:\My Stuff\My Edu\ASU\Courses\CSE515 MWDB\Project\Phase3\classdata
- Query Path is: C:\My Stuff\My Edu\ASU\Courses\CSE515 MWDB\Project\Phase3\classdata
- Top K Similar Files-----
- [1] 270.csv
- [2] 271.csv
- [3] 272.csv
- [4] 276.csv
- [5] 275.csv
- [6] 273.csv
- [7] 269.csv
- [8] 274.csv
- [9] 263.csv
- [10] 264.csv
- [11] 279.csv
- [12] 267.csv
- [13] 589.csv
- [14] 278.csv
- [15] 30.csv
-
- Relevant Files are: 270.csv,271.csv,272.csv,276.csv,275.csv,273.csv,269.csv,274.csv
- Irrelevant Files are: 589.csv,30.csv

- Now the Matlab pops up again and computes the top k similar gestures and shows it to the user. As you can see, all the files belong to the class of 270 which is an improvised result when compared to the previous result.

This screenshot shows the same 'MWDB - Phase 3' application window after a second iteration. The input details on the left are identical to the previous screenshot. The output area on the right shows the results of the second iteration:

- Query Path is: C:\My Stuff\My Edu\ASU\Courses\CSE515 MWDB\Project\Phase3\classdata
- K value is: 15
- Database Path is: C:\My Stuff\My Edu\ASU\Courses\CSE515 MWDB\Project\Phase3\classdata
- Query Path is: C:\My Stuff\My Edu\ASU\Courses\CSE515 MWDB\Project\Phase3\classdata
- Top K Similar Files-----
- [1] 270.csv
- [2] 271.csv
- [3] 272.csv
- [4] 276.csv
- [5] 275.csv
- [6] 273.csv
- [7] 269.csv
- [8] 274.csv
- [9] 263.csv
- [10] 264.csv
- [11] 279.csv
- [12] 267.csv
- [13] 589.csv
- [14] 278.csv
- [15] 30.csv
-
- Relevant Files are: 270.csv,271.csv,272.csv,276.csv,275.csv,273.csv,269.csv,274.csv
- Irrelevant Files are: 589.csv,30.csv
- Top K Similar Files-----
- [1] 270.csv
- [2] 271.csv
- [3] 272.csv
- [4] 276.csv
- [5] 275.csv
- [6] 273.csv
- [7] 269.csv
- [8] 274.csv
- [9] 263.csv
- [10] 264.csv
- [11] 279.csv
- [12] 267.csv
- [13] 278.csv
- [14] 277.csv
- [15] 253.csv

Note: We need to delete files in the query folder for every iteration.

CONCLUSION

In phase 2, we have inferred that dimensionality reduction improves the performance of similarity search due to reduction in the number of dimensions for the given data space and in this phase, we have implemented indexing, classification and relevance feedback techniques on that data. We have used LSH algorithm for indexing the data and used it to find t most similar gestures. Because of indexing, the gesture search was fast when compared to previous gestures searches as we need to compute distances to only the gestures that fall into the buckets corresponding to the hash values of the query. We have used KNN, Decision tree and n-ary SVM classifiers for classifying the gestures into different classes based on the given training data. We observed that the results vary from classifier to classifier and the execution time of KNN is more when compared to other classifiers. A decision-tree based relevance feedback algorithm was implemented to improve nearest neighbor matches. We have seen a good improvement in the resulting output after every user feedback.

REFERENCES

- John Shafer, "SPRINT: A Scalable Parallel Classifier for Data Mining", Web, Retrieved from <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.104.152&rep=rep1&type=pdf>, 2012
- Wikipedia, "Decision tree learning", Web, Retrieved from http://en.wikipedia.org/wiki/Decision_tree_learning, 11 September 2013
- Wikipedia, "k-nearest neighbors algorithm", Web, Retrieved from http://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm, 13 September 2013
- Gerard Salton, "Improving Retrieval Performance by Relevance Feedback", Web, Retrieved from <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.92.3553&rep=rep1&type=pdf>, 2012
- Sean D. MacArthur, "Interactive Content-Based Image Retrieval Using Relevance Feedback", Web, Retrieved from <https://engineering.purdue.edu/RVL/Publications/MacArthur02Interactive.pdf>, 2011.
- "Support Vector Machines Explained. "by Tristan Fletcher <http://www.tristanfletcher.co.uk/SVM%20Explained.pdf>
- "Efficient Approximate Similarity Search Using Random Projection Learning" by Peisen Yuan, Chaofeng Sha, Xiaoling Wang, Bin Yang and Aoying Zhou

APPENDIX

Task 2 – Varun, Swaroop

Task 3a – MohanRaj, Raghuveer

Task 3b – SatyaSwaroop, MohanRaj

Task 3c – Varun, Raghuveer

Task 5 – Varun, MohanRaj

Task 6 – SatyaSwaroop, Raghuveer