EEE 485-585 SPRING 2018 TERM PROJECT PHASE 2 REPORT

Title of the project: Identification of hand postures from Mocap Records
Project Assistant: Mohammad Mohaghegh
Group Members: Section 1, Yamaç Ergiz 21400306, Yunus Umeyr Kılıç 21301404
Abstract: In this project we propose to identify hand posture from a Mocap Dataset. Since mocap recording is a noisy process, there are some missing attributes in the dataset. For that reason we will use more than one approach as both supervised and unsupervised clustering. In addition to that due to possible shuffling in the order of the data during the retrieval(especially in the rows with numerous loss of data), we intent to create a set of attributes using the original dataset. The primary trait of the attributes is independence from the order of the columns in matrix. We will use Leave One out Cross Validation to verify the identification algorithms. In addition to these, considering high loss of data due to errors in retrieval(missing attributes), we will try to find corresponding coordinates for missing attributes using matrix completion algorithms.

Revisions to Phase 1 report:

- Results of Matrice completion added
- Transformation prrocess of point sets in the dataset into order independent component is added with example results.
- Results of K-means Clustering added
- Results of Linear Regression added.
- Due to inefficiency of the current methods(for the reason their performance is below 50%) a new clustering technique proposal composed of 2 layer neural network is added which will be implemented for final demonstration.
- Some ambigious sentences were corrected.

Revisions to Phase 2 report:

- DBSCAN method was replaced with Naive Bayes Classification due to incompatibility with dataset.(Upon investigations from knn graph no suitable epsilon could be found)
- Results for Naive Bayes were added.
- Overall Conclusion was added.
- Matrice Completion Analysis is added.
- A new challenge due to Naive Bayes was added.

Motion capture is a process of recording the movements of a person. This process is performed by capturing an extensive dataset by using specific cameras which tracks markers placed on a character. By this way, data set of points which can be mapped in 3d spaces is collected. In recent years there has been an increase in availability of new mocap technologies which makes it affordable and convenient to use. It has also many applications in military, entertainment and medical fields. For example in biomedical sector data obtained from motion capture can be used to measure the extent of a client's disability or a client's progress with rehabilitation. However, due to the nature of the recording process the data is generally corrupted. Despite the smoothing or removing operations applied to data, at the end only 50-60% is valid to be used .For these reasons, different techniques like machine learning applications or digital filter design are being researched to improve it.

The dataset which we propose to use in our project is a multivariate collection of records with real numbers of the following hand movements: Fist(with thumb out),Stop(hand flat),Point1(point with pointer finger),Point2(point with pointer and middle fingers),Grab(fingers curled as if to grab). The position of the markers can be seen in Figure 1. Our dataset is composed of 12 users. These 12 users performed these 5 hand motion with their left hands while wearing a special glove. After the data was collected it was preprocessed. The data was collected as a csv file and can be reached from the web address [1]. It has 78095 instances and 38 attributes. However, because of the nature of streaming process there have been some problems in collecting data and for this reason there are rows with missing column values. The proposed plans to solve these problems are explained in the methodology section with our explaining how we intend to deal with them.

Challenges

- The first problem is the difference between records of different people. Since our data is recorded from different people, it is possible that there exists randomness in our dataset to a degree.
- The other challenge is duplication of data. There is a risk that for record there is an existing near duplicate record originating from the same user which may shift the balance of our clusters in favor of a user instead of balancing equally which may cause an increment in error.
- The third challenge is missing elements in our dataset. There is a considerable amount of missing data and to classify them we need to adapt our number of attributes. For example we need to cluster with 25 attributes if there are 13 missing elements in the corresponding row.
- The fourth challenge is also related with columns having missing data. Due to missing elements in mocap data, there may be mistakes in ordering data columns in rows with missing data. For example an element in column jth of a particular row may not correspond to same element of the row . So we may have to deal with unlabeled (unsupervised classification) clustering in order to obtain a sufficient accuracy beside the classical methods to handle missing rows.
- The fifth challenge is that due to nature of the Naive Bayes we need to find likelihood functions for our dataset which is impossible with traditional ways due to continuous nature of our dataset.

Solution to Challenge1: To overcome this problem, we intend to initialize clusters by determining an average value of a posture performed by different people for every movement instead of random selection .However we should care about over fitting. So, while training clusters we will try to optimize test error.

Solution to Challenge2: we intend to use validation for verifying our algorithms. By this way we will be able to measure the generalization of the algorithms to users. Another option is to detect linear dependencies between features rows in order to save our resources. This operation also causes another problem. Since the recording process is not ideal, there may be noise in rows which may cause to identify matrices as full rank. To overcome this, we will use numerical ranking to reduce identical rows including unknown noise.

Solution to Challenge3 and Challenge4: We may select additional suitable attributes to clustering our data beside the original raw data including coordinates of postures (as mentioned previously we may have to regard them as unlabeled sets in case of a high error probability). At this stage we propose to use the following attributes that are created using dataset beside the original version. The criterion why we selected them is that they do not depend on the row order. As a result, we will use our user generated attributes for clustering instead of the provided raw. It is because they pose a greater rate of error due to possible shuffling in ordering.

Solution 5: Since likelihood functions which are required for Naive Bayes cannot be found I will assume a distrubution for prior probabilites Upon some research I decided on Normal distrubution. So in order to find likelihood function I will extract mean and values of features for every movement type.

Covariance Matrix [3] we will use Eigen values and vectors of covariance matrix as attributes using built-in commands. Then we will use them as attributes. An example generated in Matlab can be seen in following figure. As it can be clearly seen change in the order of the rows of the matrix do not cause a change in the covariance matrix of our elements in the row. The following steps show how we derive the covariance matrix of our data matrix.

- We will calculate means for each coordinate set.
- We will calculate variance for each coordinate set.
- We will find covariance using the means and variance
- We will place these values to our vectors and use them as attribute.

1) Maximum distance between two points.

The maximum distances between two points for every axis of x,y,z found and use as attribute.

Methodology A (Classification methods)

In our project the dataset is qualitative rather than quantitative taking values of coordinates according to hand posture performed. For this reason we prefer classification over regression in order to predict the responses(in our case the response is identification of hand posture). We will start with building a classifier with a process similar to regressions. Using data, we partitioned as training we will set up clusters one set for each algorithm and using these clusters we will test by clustering our test data. At the final stage, we will compare our results and select which algorithm is the most fitted for our task. Although there are variations of algorithm (like Leader Cluster) which are capable of automatically control number of clusters like incrementing or deleting, we prefer to have a manual control on the number of clusters in our algorithms since we exactly know how many type of movements we have from the dataset description.

Method A1 (K-Means)

We plan to use K-Means in order to partition data into a predetermined number of clusters. The reasons for this selection are the facts that our data is large and multidimensional. K-Means is a clustering algorithm which is used to find groups in a data. It uses distance between points (like Euclidean distance) to cluster data into groups where the groups are represented with their centers. After convergence, all centers should cover some subset of data instances. According to research, it is a good practice to initialize centers according to some data points selected previously instead of selecting random elements from our dataset. The reason for that, K-means algorithm is not convex that means it gives local optima and may result with different optimal values for each time. By selecting initial points instead of random selection, it will help centroids to be far from each other as much as possible.

Method A2(PCA)

PCA (Principal Component Analysis) is an unsupervised learning method. It uses variance minimization as a criterion. The principal component w1 can be spread in such a way that the differences between sample points are distinguished. So, we search for a w1 where the variance is maximized using the constraint transpose(w)*w is 1. By this way, we will be able to project the data into smaller number of dimensions. The challenge is that if the dimensions are not correlated we may not be able to take gain PCA. We may not be able to have a large reduction in dimension.

Method A3(Naive Bayes)

Naive Bayes is a classification Problem based on Bayes Theorem. It assumes that a feature in a random class is completely unrelated to an another feature in the same class. To perform this classification it uses posterior probability which is found easily due to Bayes theorem.

$P(c|x)=P(x|c)*p(c)/p(x)$

Step1: It extracts frequencies of the dataset

Step2: It creates likelihood table.

Step3:Calculates MAP estimate and classify.

Method A4 (Neural Network based solution)

A neuron is an artificial neural network which has a set of input variables x and weights w and lastly an activation function which maps result. Neutrals are organized into 3 types of layers which are input, hidden, output. The first input layer is tasked with feeding data into neural network.

Training Phase: In neural networks training is performed in an iterative manner adjusting weights in architecture iteration. We indent to use back propagation for this task which We plan to use fmincgrad function for training which is a continuous differentiable multivariate function minimizer written by Carl Edward Rasmussen.The code uses conjugate gradients in order to minimize the function.

Structure for the Neural Network

We intend to decide the number of hidden layers of the network using validation. By this way we will set up a threshold for error and our algorithm will iterate until difference in error reaches below the threshold.

Method A5 Linear regression & Logistic Regression

We tested our data with linear regression in order to numerically verify our PCA visualizations results. We used MSE estimation to find linear regression coefficients.

Methodology B (Matrice Completion methods)

Method B1 Low Rank Matrix Completion

In our data set **34.6693 %** of the data is missed.

In this case, there are two ways two go. First one is to use the data set as it is. The second one is to complete the data by using matrix completion algorithm. In the matrix, these missing values are represented by NaN. In our case, we will try to infer these values by using the known values which is called matrix completion.Another problem with dataset is that;

We have 11 markers for each instance but in the given dataset they are unlabeled which means we cannot know which marker corresponds to which part of the hand. Also k-th marker of one instance may not correspond to k-th marker of another instance.

We have 11 markers for each instance. For example, in one instance first marker corresponds to markers above the thumbnail and in another instance first marker may correspond to marker on the knuckle of the thumb.

Therefore, we first put the data in order then we can complete the matrix. It is because matrix completion without order may not result in a good approximation.

Putting the data in order:

In order to put the data in order, we need to find the similar instance of each row. These similar instances are so probable to correspond to same marker in the hand posture. In order to find the similar instances more efficiently, we split our whole dataset in to 70 parts. Since we have 5 different hand postures and 14 different users. Therefore we have one dataset for each user's each hand posture.

Then, we put these 70 groups of data in order separately. That means in each group of data, the instances are in order in itself but as a whole they are not necessarily in order. Since matrix completion will take place in each group of data separately, being in order in each group separately is enough.

After putting the data in order, we implement the matrix completion algorithm.

Matrix Completion Algorithm:

One of the methods of matrix completion is low rank matrix completion. In that way we need to complete the matrix in such a way that it has the lowest possible rank.

The rank of a matrix is the maximum number of linearly independent columns of this matrix.

Let M is the matrix that we have in our hand with some missed values. X will be our completed matrix. E is the all the observed values.

$$\min_{X \quad subject\ to\ X_{ij} = M_{ij\ \forall\ i,j \in E}} \operatorname{rank}(X)$$

With this method, if we have sufficient amount of observed entries, the probability of having a unique solution becomes very high [5].

In phase 1 report, we mentioned two algorithms for low rank matrix completion. These algorithms were Gradient Descent and Alternating Least Squares Minimization. However both of these methods are computationally less effective and hard to implement compared to the one that we implement in phase 2. Also in these methods, it is required to make several assumptions which may not be satisfied by our data set. Therefore we use a computationally more efficient algorithm which is Singular Value Tresholding(SVT) algorithm. This algorithm is easy to implement.

Singular Value Tresholding:

As it is noted above, we need to form the lowest rank matrix. Therefore our problem is a minimization of the completed matrix.

Let M be our observed matrix with known entries. Let the number of observed entries be m.

$\{M_{ij} : (i,j) \in \Omega\}$

Therefore, our problem becomes

$minimize\ subject\ to\ \ rank(X)_{X_{ij} = M_{ij,}(i,j) \in \Omega}$ [6]

where X is the completed matrix. Let P_Ω be an orthogonal projection on Ω. Therefore outside the span of the Ω, all the data will be vanished.Then,

$P_\Omega(X_{ij}) = X_{ij}$ if $(i,j) \in \Omega$

$P_\Omega(X_{ij}) = 0$ otherwise

Then starting with $Y^0 = 0$, the following iterations are made until a stopping criterion.

$$\begin{cases} \boldsymbol{X}^k = \mathcal{D}_\tau(\boldsymbol{Y}^{k-1}), \\ \boldsymbol{Y}^k = \boldsymbol{Y}^{k-1} + \delta_k \mathcal{P}_\Omega(\boldsymbol{M} - \boldsymbol{X}^k) \end{cases}$$

[6]

To open up the shrinkage operatorD_τ;

Singular value decomposition of X is as follows

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*$$

Where the columns of U are left singular vectors and columns of the V are right singular vectors of X.

Also diagonal elements of $\sum$ are singular values of X.

$$\mathcal{D}_\tau(\boldsymbol{X}) := \boldsymbol{U}\mathcal{D}_\tau(\boldsymbol{\Sigma})\boldsymbol{V}^*$$

[6]

Where

$D_\tau(\sum) = diag(\max(0,(\sigma_i - \tau))$

Therefore with enough number of iterations, we make a good approximation of the completed matrix.

Also, the parameter $\sigma\ and\ \delta$ should be tuned well in order to reach a good approximation.

After implementation of matrix completion, the elapsed time for running of the code is **69.637772 seconds** which is acceptable for a matrix with 78095 elements.

MSE error also calculated for matrix completion.

Since we run the matrix completion for each group of data separately, we have different MSE for each of them as follows;

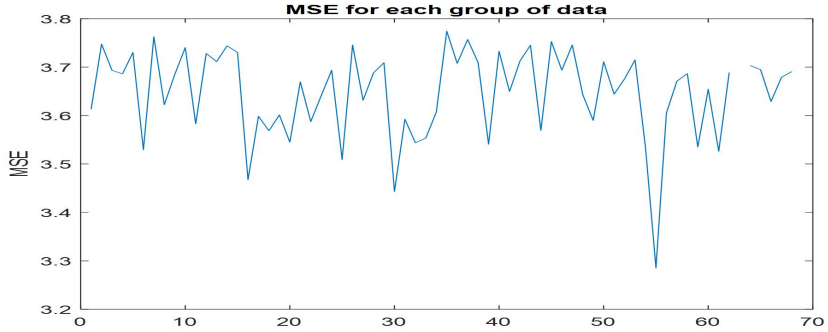


Figure 1: MSE for matrix completion

Then in average our MSE becomes **3.6471**

Validation

For identifying posture as mentioned in the challenges, we will use leave one out cross validation for validating our algorithms due to possible duplications. It uses an entire model to fit all the data by exempting a single point. After that, it makes a prediction at that point. To do this we will partition our database. At first stage, we will partition our dataset. We will do the training for our clusters. We intent to make a partition 70 to 30 % for training and test in respective order.

Similarly for matrix completion algorithms, we will train and test our algorithms with the rows with no missing values. Then we will compare our newly found ones with the original ones and take Euclidian distance between points as reference.

In addition to this to visualize the methods performance we will use confusion matrix concept. Confusion matrix is a table used to compare classification methods results and see the weight of the error. Using confusion in addition to error rate we can calculate different traits like accuracy, misclassification rate, true positive rate, false positive rate, and specificity, precision, prevalance.

Results

Matrice Completion

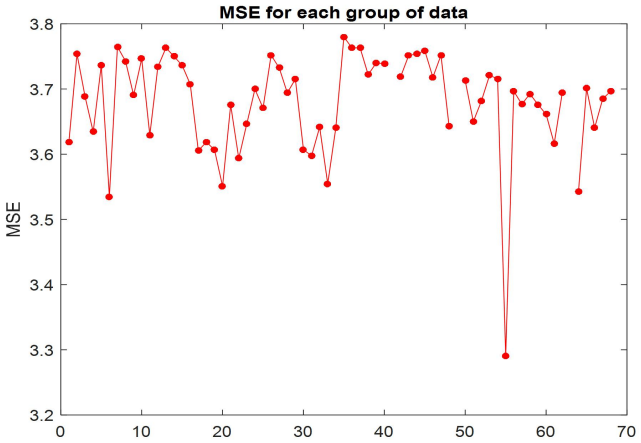


Figure 2:

Then in average of MSE becomes **3.6471**

Also for each of the hand posture, we calculate the MSE differently for matrix completion.

The results are as follows;

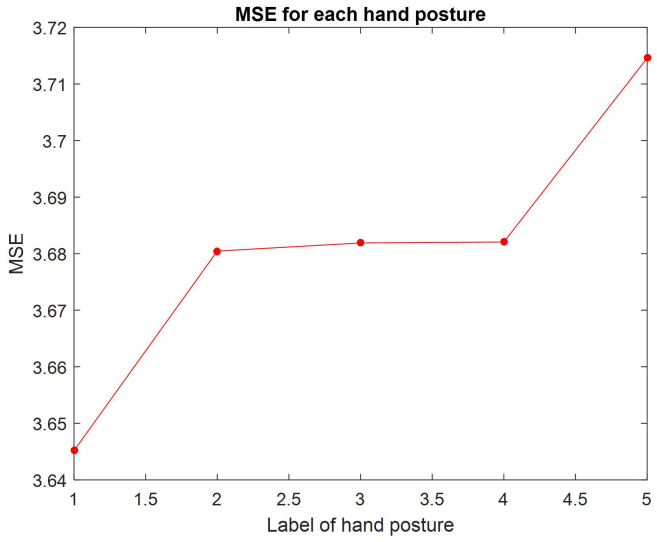


Figure 3:

As in the figure, first hand posture which is Fist(with thumb out) has the lowest error. Since in Fist each data points are so close to each other and the distance between the points are lower, the MSE for Fist is the lowest. Second, third and fourth postures which are Stop(hand flat), Point1(point with pointer finger), Point2(point with pointer and middle fingers) have similar error. Since these hand postures are very similar shape, the errors are similar. The fifth posture which is Grab(fingers curled as if to grab) has the largest error since in this posture, fingers are little bit far away from each other and they are inclined to have a different positions in each record of the data.

Transformation of dataset to order independent form:

As mentioned in methodology we started with creating a new dataset composed of elements independent of orders. An example transformation is shown in the below figures.

To transform these instances we started with finding covariance matrix of the row. For example in row 1 of the dataset is shown below.

1 0 54.2638799540698 71.4667761378817 -64.807708780882 76.8956347751343 42.462499897063 -72.7805451867589 36.6212291601289 81.6805569287795 -52.9192723726971 85.2322638852917 67.7492195028673 -73.68413004183359.1885757027887 10.6789364098231 -71.2977813147725

In the first stage I count the number of missing attributes out of total 38 elements. For the first row given above it is 21. So the mocap data in the first row has 17 attributes. I extract the user column since I features I selected for transformation process is based on covariance matrices of the points. In addition I also extracted movements id labels since we aimed to do movement identification so I extracted y and used them as response variable for our project.

Step2: After we partitioned dataset as described in the previous step now we started to transform we created a covariance matrix using the point sets. The conditions for the covariance matrix was that it should have had three columns since each points set had three attributes which are namely in order x,y and z coordinates of the points in local space where the origin reference also adjusted. Covariance matrix of the row specified in previous step is follows.

368.1206 -142.5101 -150.8513
-142.5101 816.7275 147.3565

-150.8513 147.3565 74.8971

Step3: After this stage I found Eigen values and Eigen vectors of the covariance matrix. For example for the above row Eigen values is

2.3435 0 0
0 357.4173 0
0 0 899.9844

V = 0.3424 -0.8867 0.3106

-0.1089 -0.3658 -0.9243

0.9332 0.2827 -0.2219

So using diagonal entries of the eigen values matrice and vectors of the V I created a new row of data so the transformation process is done.

PCA analysis of our dataset

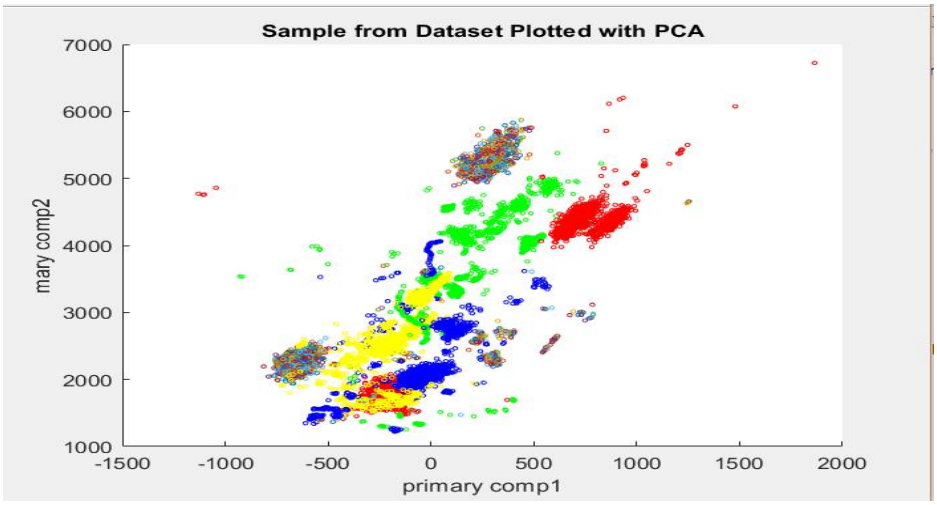


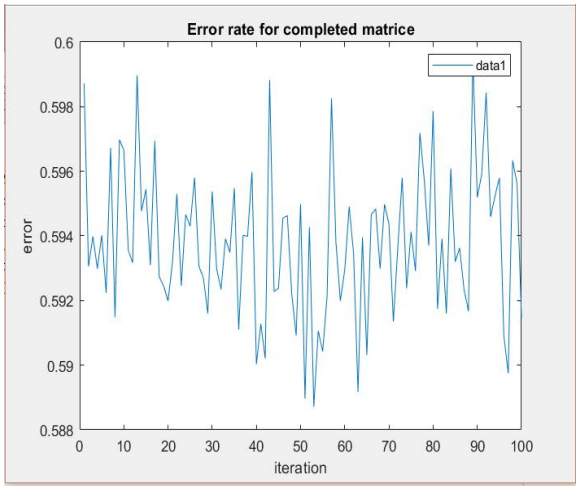
Figure 4

Analysis of PCA:

- It can be said that some of the classes are not linearly separable due to collisions.
- The movement type 4(green) has scattered around the other data. It can be thought that the variance is high. It is expected that weight of errors related with movement 4 be higher than each other.
- It can be also said that the variance of red(movement 1) is high. It can be expected that while data belonging to some users are easily differentiated from the others the other parts will have high error rate due to collisions
- From the graph it can be deduced that the data is highly complex.

D1)Kmeans Results

min	0.5887
max	0.5995



mean	0.5938
median	0.5937
mod	0.593
std	0.002306
range	0.01075

Figure 5:

Test error is 16662 out of 28095. Standardized test error is 0,59

Error for 100 iterations of k means and the plots statistical description.

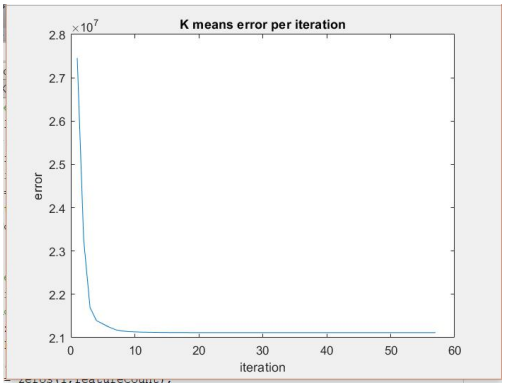
Elapsed time is 5.774381

3665	895	271	660	357
876	1824	2185	1562	1253
373	1846	1846	1266	1877
523	682	401	1355	170
443	110	1153	495	2007

Figure6

3259	541	171	168	281
223	1573	2121	1606	2395
124	1659	1579	1131	1118
66	738	947	1743	15
2208	846	1038	690	1855

Example Error Distribution for completed and uncompleted Matrices using K means



Bit error rate: 0.6433

In side you can see the description statistics table of 100 iteration of completed matrices.

Figure 7:

min	0.5684
max	0.5806
mean	0.5749
median	0.5748
mod	0.5747
std	0.002356
range	0.01221

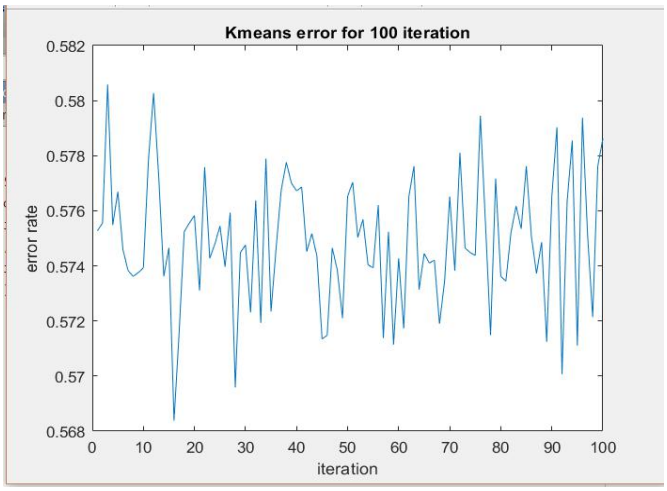
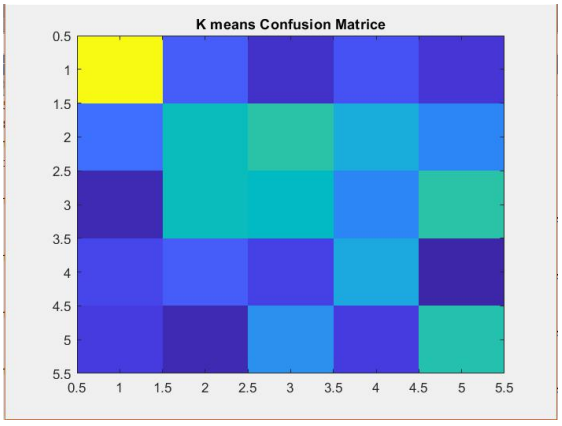


Figure 8

Figure 9

An Example Error Distribution for Completed and Uncompleted Matrices using K means is below



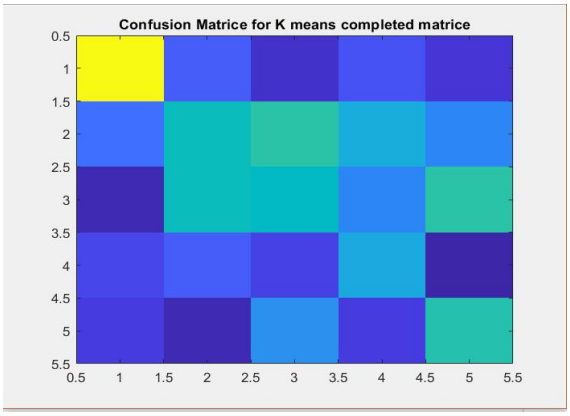


Figure 11

Figure 12

Review of K means

With this algorithm we tested our K means clusterings. Upon that we see that uncompleted matrices gave worse result than not completed matrice so it can be say that completing matrice caused a saturation in covariance matrice of the dataset. But it should be also noted that the error variance in completed matrices performed better than uncompleted matrices. It can be said that confusion matrices verified our PCA algoritm by showing that while the ones who are seperate from the others like type 1 can easily be seperated but the ones with collision have a hard time in seperation. More iteration better results K means gave also.

Results B Linear Regression && Ridge Regression

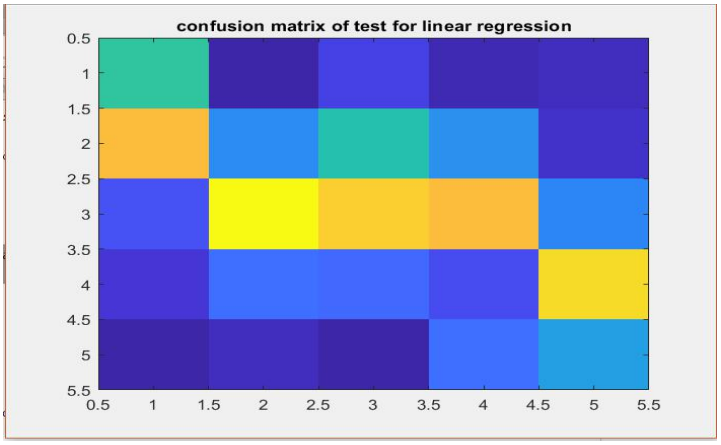


Figure 13

8946 true which becomes 31 percent accuracy. As we have expected from PCA results while some of the data can be seperated easily by linear ways, there are equal error distrubution is some movement types due to collision of data. So since some parts of the dataset is highly complex the linear regression performed better at some areas and worse at other.

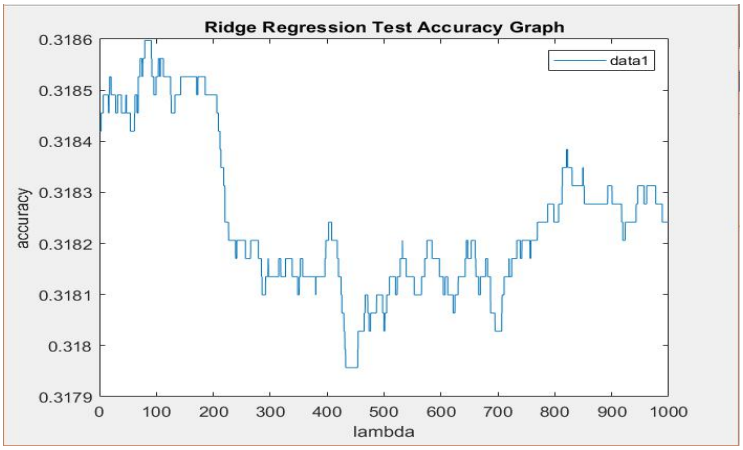


Figure 14

min	0.318
max	0.3186
mean	0.3182
median	0.3182
mod	0.3181
std	0.00015

Figure 15: RidgeRegression for Uncompleted Matrice

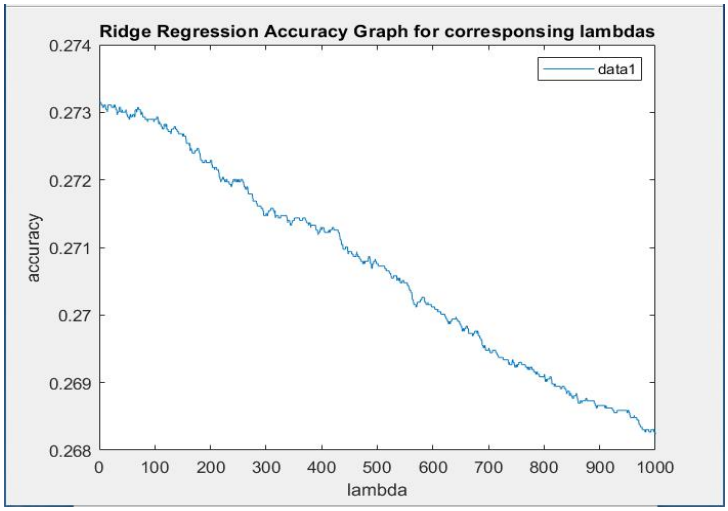


Figure 16 RidgeRegression Completed Matrice

Review of Linear Classification Methods

As it can be seen from the figures above even we take the peak values as reference, the other methods outperformed the linear classifiers&predictors. This verifies our sample PCA graph that our data is not linearly seperable. So linear classifiers is not suitable for our aim.

Results C Naive Bayes

To perform Naive Bayes Classification we started with finding prior estimates which are in

Class	Prior Probabilities
P1	0.209460000000000
P2	0.190840000000000
P3	0.208240000000000
P4	0.189800000000000
P5	0.208240000000000

In addition to this I assumed normal distribution for entries since we deal with continuous data in constrast to traditional usage of Naive Bayes with discrete data. To obtain a possible normal distrubution for data I obtained mean matrices and standard deviation matrices for probabilities as shown in the below table.

24.8052	0.1008	0.1013	-0.3607	0.2932	0.2423	0.2933	-0.0220	-3.2781e-04	0.3833	196.6477	571.2034	1.8470e+03	172.5448
32.9649	-0.2044	0.5330	-0.2410	0.2250	0.3529	0.4247	-0.1774	-0.0246	0.4063	120.6093	937.9044	3.0008e+03	520.7367
27.1470	0.1661	0.3426	-0.1702	-0.0095	0.1013	0.2804	0.2570	0.2042	0.1994	292.1015	1.0069e+03	2.9073e+03	277.0557
28.3851	-0.1110	0.2967	-0.1041	-0.0031	0.0450	0.4508	0.1657	-0.0098	0.2460	254.0041	783.7565	3.1777e+03	329.1497
32.4377	-0.0738	0.3150	-0.4183	0.3212	0.3897	0.3709	0.0826	0.0243	0.3028	524.4236	966.8650	2.7175e+03	404.6317

Figure17 Mean arr

7.5921	0.4917	0.5780	0.5232	0.4850	0.5565	0.4737	0.6530	0.5361	0.3727	247.4787	236.9603	940.0052	85.8830
2.6402	0.3452	0.6295	0.3172	0.3994	0.3752	0.5868	0.7730	0.2319	0.3893	113.3788	377.1921	789.0417	71.2989
7.8246	0.4253	0.6695	0.4438	0.4552	0.4894	0.6815	0.7199	0.3780	0.4375	283.4732	376.1025	837.2925	114.2697
5.8187	0.4782	0.7047	0.4044	0.3606	0.4479	0.6813	0.7756	0.4611	0.3128	212.5190	371.6005	876.4005	81.6343
3.2958	0.4371	0.4825	0.5446	0.4083	0.5103	0.4246	0.7259	0.5051	0.3448	212.5809	288.9108	559.0439	75.6752

Figure 18 Std arr

5586	22	697	514	102
18	5188	9	271	148
176	35	3795	343	223
11	160	1063	3833	177
1	31	368	324	5000

Figure 19 Confusion Matrice for Naive Bayes Uncompleted Matrices

4876	35	951	355	296
0	4291	1156	962	455
773	961	3015	1680	333
0	114	204	2231	259
143	35	606	57	4307

Figure 20 Confusion Matrice for Completed Entries

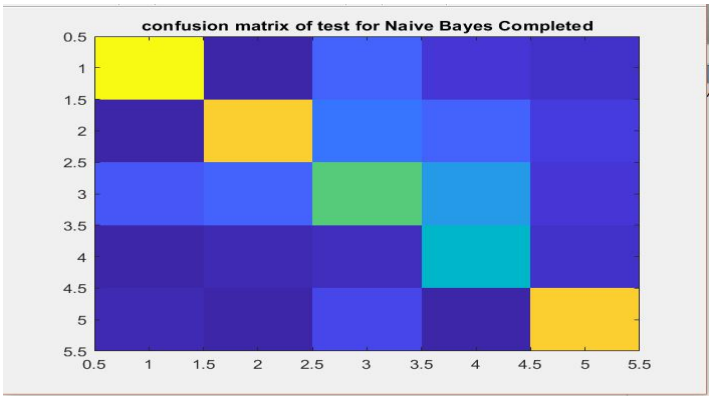


Figure 21 Completed Matrice Confusion Matrice

Review of Naive Bayes Between the methods we used naive Bayes gave the best result. As it can be seen from the confusion matrices the diagonal entries were correctly identified and it resulted with 23402 correct of 28095 test entries. In constrast naive Bayes with completed Matrice resulted with 18720 correct identifications out of 28095 which corresponds to 0.6630 percent accuracy. As a result it can be said that between all the methods probabilistic based classifier gave the best result. Also similar to other ones uncompleted matrice gave worse result.

Important Note: We also made some analysis using neural network by using some libraries and available codes in Internet using which we created a 2 layer model which stuck at accuracy of 57%. We relate the cause with the fact that our data is highly complex so more complex neural classifiers is needed. **Since we use available code we do not put the code in this report.**

Conclusion In this project we tried to classify mocap data in order to do hand movement identification. We tried different methodology for both prepossessing and prediction. So it can said that we used two different approach for every algorithm one with completed data and the other with uncompleted data. The conclusion for matrice completion is that it caused overfitting in data and caused our accuracy rate to fall. On the other hand we tested three machine learning algorithms which are in order a linear regression predictor,a clustering predictor and a probabilistic predictor. Upon analysis we concluded that our data is highly complex and for this reason the first two predictions operations failed. On the other hand probabilistic predictor(Naive Bayes) outperformed the other two operators.

References

[1]UCI Machine Learning Repository: Data Set, archive.ics.uci.edu/ml/datasets/Motion Capture Hand Postures.

[**2**]"Variance: Simple Definition, Step by Step Examples." *Statistics How* statisticshowto.com/probability-and-statistics/variance

[3]Packer,Ben "What-is-an-eigenvector-of-a-covariance-matrix"https://www.quora.com/What-is-an-eigenvector-of-a-covariance-matrix

[4] An introduction to statistical learning, James G, Witten D, Hastie T, Tibshirani R., 2013, Springer

[5]Candès, E. J.; Recht, B. (2009). "Exact Matrix Completion via Convex Optimization". Foundations of Computational Mathematics. 9 (6): 717–772.

[6] Candès, E. J.; Cai Jian-Feng, Shenn Zuowei(2010). " A Singular Value Thresholding Algorithm For

Matrix Completion". SIAM Journal on Optimization . 20(4):1956-1982.

[7] Matrix Completion[Online]. Available: https://en.wikipedia.org/wiki/Matrix_completion

Appendix:

Matlab code for all parts are as follows

Order Mvm

```
function [ output ] = MatrixCompletion( input, zeronumber, whole)
%UNTÝTLED5 Summary of this function goes here
%   Detailed explanation goes here

average = zeros(1,36);

%averaging

    for k = 1:36
        average(1,k) = sum(input(:,k)) /
sum(sign(abs(input(:,k))));
    end

%completion
for i = 1:size(input,1)
    if zeronumber(i,1) == 0
        for j = 1:36
            if input(i,j) == 0
                input(i,j) = average(1,j) + 3*randn;
            end
        end
    else
        input(i,:) = whole(i,:);
    end
end

output = input;




%create the whole matrix
whole = [X0 Y0 Z0 X1 Y1 Z1 X2 Y2 Z2 X3 Y3 Z3 X4 Y4 Z4 X5 Y5 Z5 X6
Y6 Z6 X7 Y7 Z7 X8 Y8 Z8 X9 Y9 Z9 X10 Y10 Z10 X11 Y11 Z11];


%loop for 70 different index values

%start and end values
stend = zeros(2,70);

%current index value
current = 0;
```

```
%to get the start and end values
for i = 1:14
   for j = 1 : 5

   stend(1,(i-1)*5 + j) = current + 1;
   stend(2,(i-1)*5 + j) = current + index(i,j);

   current = current + index(i,j);

   end
end


stend = [stend(:,1:19),stend(:,21:70)];

stend = [stend(:,1:33),stend(:,35:69)];



%fully completed whole matrix
orderedWhole = zeros( 78095,36);
%SVT
orderedWholeSvt = zeros( 78095,36);

NotOrder = zeros(70,3);

%total not observed initially
missed = 0;
%total not observed last
missedlast = 0;


% to iterate the ordering for each parts

for i = 1:68

   tempwhole = whole( stend(1,i):stend(2,i) ,:);
   [ orderedMat, zeronumber, missedInt,missedLast ] =
OrderMat( tempwhole);



   missed = missed + missedInt;
   missedlast = missedlast + missedLast;

   NotOrder(i,1) = sum(sign(zeronumber));
   NotOrder(i,2) = size(orderedMat,1);
```

```
    NotOrder(i,3) = sum(sign(zeronumber))/size(orderedMat,1);

    %construction of the whole notcompleted matrix
    NotorderWhole(stend(1,i):stend(2,i),:)  =  orderedMat;

    %completion  of the matrix
    %orderedMat = MatrixCompletion( orderedMat,zeronumber,
tempwhole);
    %SVT
    orderedMatSvt = SVT( orderedMat,zeronumber, tempwhole);

    %construction of the whole completed matrix
    %+orderedWhole(stend(1,i):stend(2,i),:)  =  orderedMat;
    %SVT
    orderedWholeSvt(stend(1,i):stend(2,i),:)  =  orderedMatSvt;
end
```

Main code for matrice completion

```
%percentage order success
disp('Percentage of the ordered rows:')
100 - (sum(NotOrder(:,1))/78095)*100

%initial missed percent
disp('Initial percentage of the observed samples:')
100 - missed/(78095*12) * 100

%completed missed percent
disp('Last percentage of the  observed samples:')
100 - missedlast/(78095*12) * 100




%write a csv file
%csvwrite('CompletedPosture.csv',orderWhole);
%csvfile = [Class, User, orderWhole];
```

Indexing

```
%Use it only once then do not use it again
%Class = Class(3:size(Class,1),1);
%User = User(3:size(User,1),1);


index = zeros(15,5);
m = 1;
n = 1;
```

```
index(m,n) = 1;


for i = 2:size(User,1)

    prev = [User(i-1,1) Class(i-1,1)];
    current = [User(i,1) Class(i,1)];

    if current(1,1) ~= prev(1,1) || current(1,2) ~= prev(1,2)

        n = current(1, 2);
        m = current(1,1)+1;
    end

    index(m,n) = index(m,n)+ 1;
end

  index = [ index(1:3,:); index(5:15,:)];
```

SVT

```
function [ output ] = SVT( input, zeronumber, whole)
%UNTÝTLED5 Summary of this function goes here
%   Detailed explanation goes here



%only takes the fully ordered rows

fullinput = [];

for i = 1:size(input,1)
    if zeronumber(i,1) == 0
        fullinput = [fullinput;input(i,:)];
    end
end

%calculate the number of observed values
observed = 0;
for i = 1:size(fullinput,1)
    for j  = 1:size(fullinput,2)
        if fullinput(i,j) ~= 0
            observed = observed + 1;
        end
    end
end
```

```
M = fullinput;
Mzero = sign(abs(M));


%input M
tao = 30;
delta = 1.2*(size(M,1)*size(M,2))/observed;


Y = delta * M;
%%%%%%%%%
for j = 1:10


%singular value shrinkage

[U,S,V] = svd(Y);

Str = S;

for i = 1:min(size(S,2),size(S,1))

    Str(i,i) = max((S(i,i)-tao),0);

end

Ytr = U*Str*transpose(V);
%%%%%%%%%%%%%%%%%%

X = Ytr;

Y = X + delta*(M-X).*Mzero;

end


output = X;

 if zeronumber(1,1) ~= 0
       output = [whole(1,:);output];
 end


for i = 2:size(whole,1)
    if zeronumber(i,1) ~= 0
```

```
       if size(whole,1) <= size(output,1)
           output =
[output(1:i-1,:);whole(i,:);output(i:size(output,1),:)];
       else
          output = [output;whole(i,:)];
       end
    end
end


end
```

Matrice Completion

```
function [ output ] = MatrixCompletion( input, zeronumber, whole)
%UNTÝTLED5 Summary of this function goes here
%   Detailed explanation goes here

average = zeros(1,36);

%averaging

    for k = 1:36
       average(1,k) = sum(input(:,k)) /
sum(sign(abs(input(:,k))));
    end

%completion
for i = 1:size(input,1)
    if zeronumber(i,1) == 0
       for j = 1:36
          if input(i,j) == 0
             input(i,j) = average(1,j) + 3*randn;
          end
       end
    else
       input(i,:) = whole(i,:);
    end
end

output = input;
```

Linear regression

```
testdata=new_dataset(testindices,:);

y=movement_y(trainindices,:);

regression=inv(new_dataset(trainindices,:)'*new_dataset(traini
ndices,:))*new_dataset(trainindices,:)'*y;

prediction=round((testdata*regression));
```

Data partitioner

```
n=1:78095;

trainindices=intersect(n,randsample(n,50000)) ;

testindices=n(~ismember(n,trainindices));
```

Pca graph

```
colors=['r''g ''b''y''c' ];

for i=1:79085

    scatter(ans(i,1), ans(i,2), 5, colors(movement_y(i))); hold
on;

End

Naive Bayes

traindataset=new_dataset(trainindices,:);

movement1arr=traindataset(find(movement_y(trainindices)==1),:)
;

movement2arr=traindataset(find(movement_y(trainindices)==2),:)
;

movement3arr=traindataset(find(movement_y(trainindices)==3),:)
;

movement4arr=traindataset(find(movement_y(trainindices)==4),:)
;

movement5arr=traindataset(find(movement_y(trainindices)==5),:)
;

normalizemean=mean(new_dataset);

normalizestd=std(new_dataset);



%Prior probabilities

pclass1=length(find(movement_y(trainindices)==1))/50000;

pclass2=length(find(movement_y(trainindices)==2))/50000;

pclass3=length(find(movement_y(trainindices)==3))/50000;
```

```
pclass4=length(find(movement_y(trainindices)==4))/50000;

pclass5=length(find(movement_y(trainindices)==3))/50000;


meanarr=[mean(movement1arr);mean(movement2arr);mean(movement3a
rr);mean(movement4arr);mean(movement5arr)];

stdarr=[std(movement1arr);std(movement2arr);std(movement3arr);
std(movement4arr);std(movement5arr)];

testdataset=new_dataset(testindices,:);

testlabel=movement_y(testindices,:);

predicted_labels=zeros(1,28095);

count=0;

for i=1:28095

    x=testdataset(i,:);

    is1=prod(normpdf(x,meanarr(1,:),stdarr(1,:)),2)*pclass1;

    is2=prod(normpdf(x,meanarr(2,:),stdarr(2,:)),2)*pclass2;

    is3=prod(normpdf(x,meanarr(3,:),stdarr(3,:)),2)*pclass3;

    is4=prod(normpdf(x,meanarr(4,:),stdarr(4,:)),2)*pclass4;

    is5=prod(normpdf(x,meanarr(5,:),stdarr(5,:)),2)*pclass5;

    normvector=sum([is1 is2 is3 is4 is5]);

    arr=[is1 is2 is3 is4 is5]./normvector;

    [M,I] = max([is1 is2 is3 is4 is5]);

    predicted_labels(i)=I;

    if(testlabel(i)==predicted_labels(i))

        count=count+1;

    end

end

count
```

K means counter

```
counts = zeros(5,5);

for i=1:28095

if(label(i)==1)

       counts(1,floor( allf(i)) ) = counts(1, floor( allf(i))) +
1;

end

if(label(i)==2)

       counts(2, floor( allf(i)) ) = counts(2, floor( allf(i)) )
+ 1;

end

if(label(i)==3)

       counts(3, floor( allf(i)) ) = counts(3, floor( allf(i)) )
+ 1;

end

if(label(i)==4)

       counts(4, floor( allf(i)) ) = counts(4, floor( allf(i)) )
+ 1;

end

if(label(i)==5)

       counts(5, floor( allf(i)) ) = counts(5, floor( allf(i)) )
+ 1;

end


end
```

Ridge Regression

```
testdata=new_dataset(testindices,:);

y=movement_y(trainindices,:);

countarr=zeros(1,1000);

index=1;

for lambda=0.01:0.1:1000
```

```
regression=inv(new_dataset(trainindices,:)'*new_dataset(traini
ndices,:)+lambda*eye(14))*new_dataset(trainindices,:)'*y;
    prediction=round((testdata*regression));
    count=0;
    for i=1:28095
    if(testlabel(i)==prediction(i))
        count=count+1;
    end
    end
    countarr(index)=count;
    index=index+1
end
```

Kmeans

```
trainx=new_dataset(trainindices,:);
trainy=movement_y(trainindices,:);
error = zeros(1,100);
counti = 0 ;
dataSize=50000;
featureCount = 14;
predictions = zeros(50000, 1);
next_predict = ones(50000, 1);
tempprediction = zeros(k,1);
while norm(predictions - next_predict) > 0
    predictions = next_predict;
    errornow = 0;
    for index1 = 1:50000
        for index2 = 1:5
```

```
          tempprediction(index2) = norm( cluster_cen(:,index2) -
transpose(trainx(index1,:)) );
       end
       next_predict(index1) = 1;
       for index3 = 2:5
          if tempprediction(index3) <
tempprediction( next_predict(index1) )
             next_predict(index1) = index3;
          end
       end
       errornow = errornow +
tempprediction( next_predict(index1) );
   end
   for index1 = 1:5
       tam = zeros(1,14);
       element_count = 0;
       for index2 = 1:50000
          if next_predict(index2) == index1
             tam = tam + trainx(index2,:);
             element_count = element_count + 1;
          end
       end
       cluster_cen(:,index1) = tam / element_count;
   end
   counti = counti + 1
   error(counti)=errornow;
end
```

Kmeans tester

```
function [predictor] = kmtest(centers, testarray)

temp = zeros(5,1);

for index2 = 1:28095

    for index3 = 1:5

        temp(index3) = norm( centers(:,index3) -
testarray(index2,:)' );

    end

    predictor(index2) = 1;

    for i2 = 2:5

        if temp(i2) < temp( predictor(index2) )

            predictor(index2) = i2;

        end

    end

end

end
```