Hand Posture Dataset recognition

Data Set(s): Hand Postures

Yihang Zhou, Zhiyao Luo
zhouyiha@usc.edu

April.2020

1 Abstract

In this project, we will deal with the recognition problem on the hand postures datasets and aiming to get a high accuracy. There are 7 different methods have been applied to do the classification work, that is support vector machine (SVM), Naïve Bayes classifier, artificial neural network (ANN), MSE method, K-NN method, Random forest classifier and k nearest mean. In our result, the highest accuracy we got for these methods is 96%, 87%, 96%, 86%, 82%, 86%, 94%. The best performance until now is belong to the gaussian kernel SVM classifier. In this project, we develop our own features and designed a new improved k nearest mean algorithm and shared many ideas that helps to choose proper parameters to get a higher accuracy in each method.

2 Introduction

2.1 Problem Statement and Goals

Hand postures (from motion capture) datasets is chosen in this project. Hand postures datasets is a dataset that record the relative location (Cartesian coordinate) of some markers which is stick on some users' hands when they are asked to posture in 5 different ways. There are mainly two properties of this dataset. One is the length of each data can be different. For example, for one posture, 8 markers' data have been recorded while for another posture only 5 markers' data will be recorded; and two is the values recorded in each data is lack of order, which mean any single value in a data is meaningless. The bringing together of these two properties stated above will surely bring more difficulties when we deal with the dataset. Based on the knowledge of this dataset, we believe that it should be a good solution to use the statistical properties of each data as the extracted features. Our goal is designing some feature extracting method and applying 7 different classifiers to deal with the recognition problem and optimize the performance of these classifiers in the sense of accuracy and time consuming.

3 Problem Statement and Goals

3.1 Preprocessing

Preprocessing can be split into 3 steps. Firstly, using the function "csvread(filename, start_row, start_column)" in MATLAB to read the "csv." file into the workspace. Secondly, dropping some data of which the length of data does not reach the threshold of data length. This step is to make sure that the data is long enough to offer a set of statistical features. (for example, if we choose the maximum value of x as a feature, if we only have 3 values of x, then the maximum value will represent less information) The detail method is use a for loop to check within each data, the step

length of each iteration is 3, stop the loop once the value start to be 0 and the iteration number is the number of marker in this data. If the length of data is lower than the threshold, we will not save it into the data matrix that will be used to extract features in the future. The last step is split the labels vector and user vector. It is important to keep the keep correspond labels and users when you keep the data and drop them if you drop the too short data.

It's hard to balance the data in this project because for different postures the length of data will be very different, and the number of data which belongs to a specific class will be smaller than others after the process of data dropping. To alleviate this problem, we should keep a balance when we choose the threshold. If the threshold is too high, then the unbalance of data will get worse; if we choose a too low threshold, the feature we extract from the data may be affected and performs not that well.

3.2 Feature engineering

As we have stated in the introduction part, the values in Hand postures (from motion capture) datasets cannot be used as features directly. In this subsection, 22 features that developed by us will be introduced. (Notation: the statement of "data" in the following part means a row in the dataset matrix which is generated in the processing part)

We extract 22 features from the data set. The first feature is the number of markers in each data. And the follow 6 features are the mean and standard deviation of the x, y, z values of all the markers in a data. The last 6 features are the maximum and minimum value of the x, y, z values of all markers. The choose of these features is implementable. For example, the standard deviation of x, y, z values can be regard as a measure of the density of markers in the space. The maximum and minimum value can stand for the distribution range of markers.

Based on the assumption that the relatively local Cartesian coordinate can only be robust to the scale change but cannot preserve the features in the sense of angle (and for a same posture, the location of markers can be changed into a rotated form), we decide to transfer those markers location into a sphere coordinate. By using the function "cart2sph()" in MATLAB, we can get the location of every marker in sphere coordinate. The new expression of every marker will be in the form (alpha, beta, radius), alpha and beta is two angle and radius are the distance between the marker and the origin. Having transfer the location into a sphere coordinate expression, by calculating the mean and standard deviation of alpha, beta and radius, we can filter out the influence of rotation which was brought in when the data was recorded. Thus, we get 6 new features: 3 means and standard deviation in sphere coordinate. The other 3 features are based on skewness. It presents the degree of asymmetry in the distribution of statistical data which we thought it may also a good feature that can be extract from raw data.

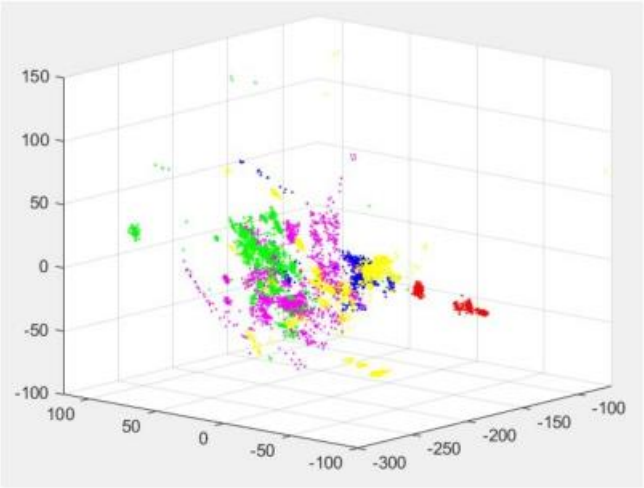

Figure 1: The distribution of 5 different postures in the feature space (standard deviates of 3 coordinates in spherical coordinates)

As shown in Fig. 1, even in a low 3-D feature space, the data from a same class shows a trend to cluster together, which mean these features contains useful information for solving recognition problem.

3.3 Feature dimensionality adjustment

Feature dimensionality adjustment is different when using different classifiers. For method that is more data-driven like support vector machine， k nearest mean, ANN, the feature dimension should be kept as more as possible to get a stronger classification ability. In MSE and random forest, it depends on features quality. However, for method that has strict requirements like Naïve Bayes classifier, the dimensionality adjustment is very important if we want to get a higher accuracy.

3.4 Dataset Usage

The raw dataset that is just read into the workspace of MATLAB is a 13500*38 matrix (the first column which stand for the index of data has been dropped), which means there are 13500 data and at most 38 values in each data. Which is worth to mention is that the dataset has been filtered by the provider before it is sent to us, the data which has less than 3 markers' information have been dropped. The first column of 38 values is the label, which is range from 1 to 5 and stand for 5 different postures. The second column is the index of user, from whom we get the data. There are totally 12 users and the index are ranging from 0 to 11. Following these two columns are 36 columns that represent the location of 12 markers in cartesian coordinate (in x, y, z for each marker). 1 of 12 markers will be pointed to be the origin and the other 11 markers will provide a value that stand for the relative local location. Thus, the last 3 columns of each data are all 0. What's more, not every marker will provide a value, so the length of non-zero column (0 in user index is not considered here) in each data is 2+3n, n is the number of markers that have values.

Once we dropped the data that cannot meet the requirement of length threshold and the last 3 all 0 column and split the vectors of label and user, the data matrix will be reduced to 9616*33. (the threshold of nonzero marker number is 7). Using the method that we have obtained in the feature engineering part, the 9616*33 data matrix will be transfer to a 9616*22 feature matrix. In the dimensionality adjustment part, the feature matrix can be transferred into any lower dimension

matrix by principal components analysis (PCA). To achieve the function of PCA, we can firstly use the multiply of feature matrix X and its transpose X^T (in form of $X^T X$), and then do singular value decomposition (SVD) to the result. We only need to keep the right singular vector matrix . If we want to preserve k dimension of principal feature, we can keep the first k rows of matrix V as V' and the result of XV' is the PCA result. The reason we calculate $X^T X$ first is to save the computation resource. If we do singular value decomposition to X directly, we can get the same result, but the computation complexity will be in the scale of 9616 *22. However, if we calculate $X^T X$ first, the complexity will only be 22*22. Untill now, we can obtain any dimension of feature matrix by PCA as we need.

For cross validation, we developed two methods. The first method is original method that split a percentage of training data to be validation set. For example, if we split the whole feature matrix into 24 groups, in the cross-validation process, 23 of the groups will be used as training data (9216*22) and 400*22 will be used as validation data. Repeating 24 times and using different groups at each time. To achieve this, we can use the function "cvpartition()" in MATLAB, it can help us to split the data every time. The second method is the required cross-user-validation. Things are different here. Firstly, we split the data according to the user and we find that the number of data that belongs to different user is different. Even worse, after the short data dropping procedure, some data from several user are totally dropped. To be more specifically, the data from No. 3, 4, 7 (start from 0) are totally dropped when we choose threshold to be 7. So, we only use 800 data for each user remains (totally 9*800=7200). We build an index matrix that can help us to locate the data of each user in the original feature matrix. At each time, the first 8 rows of index in the index matrix will be used to get the features that come from 8 users and leave the last row in the index matrix to get the feature from the last user as the validate set. After each accuracy test, we only need to use the "circshift()" function in MATLAB to change the position of each row in the index matrix to make sure that we can leave a new user's data to be the validation set. In this procedure, the number of runs is 9 and at each time, the training matrix is 6400*22 and validation matrix is 800*22.

The preprocess, feature extracting, and dimensionality adjustment of test data is like those of training data. I must mention that the arguments that is used in validation data processing should be the totally the same as training data set. For example, when we do standardization on training data, we should save the means and standard deviates of each feature, and use these arguments to standardize the validation data, instead of using the means and standard deviates of validation data to standardize itself. The PCA process on validation data is similar. The number of data that used in validation set after short data dropping is 1856.

The validation data have been used 7 times (one time for each classifier)

3.5 Training and Classification

3.5.1 Naïve Bayes Classifier

The highest accuracy we get from Naïve Bayes Classifier is 87.23% (threshold 6). The parameters in this method is the feature choosing and PCA dimension. We used 16 features (without the mean of x, y, z values and the skewness of these three value). The PCA dimension is 12, which means we adjust the dimension of features from 16 to 12.

Naïve Bayes classifier is a statistical classifier and the fundamental idea is using the covariance matrix and mean vectors to estimate the conditional probability $P(X|S_i)$, and then use the minimum error classification rules to predict the label of testing data. Naïve Bayes classifier has an assumption that the covariance matrix should be a diagonal matrix which means all the features are uncorrelated with each other. In MATLAB, we can use "fitcnb()" function to build the Naïve Bayes classifier model and then use it to generate a predict label vector.

At the first trial, the accuracy of this method is very low like 50~60%. There are two reasons. The first reason is that the Naïve Bayes requires the covariance of features to by a diagonal matrix; and the second reason is that the Bayes classifier is aiming to get the lowest error rate, which means if the feature is not that good and the data is not separable, the performance of Bayes classifier may get affected a lot.

To alleviate this problem, I decide to make some adjustment on the feature matrix. As we all know, we can get a diagonal matrix by doing eigenvalue decomposition. And we can use the multiplication between the original feature matrix and eigenvector matrix to get a new matrix, of which the covariance is a diagonal matrix and thus the features are decorrelated. In MATLAB, we can simply use the function "eig()" to get the eigenvector matrix, and we need to keep this to apply the same procedure on the vadilation data. After this procedure, the validation accuracy will be improved a lot. The accuracy can reach to 70~80%. The covariance matrix is shown below:

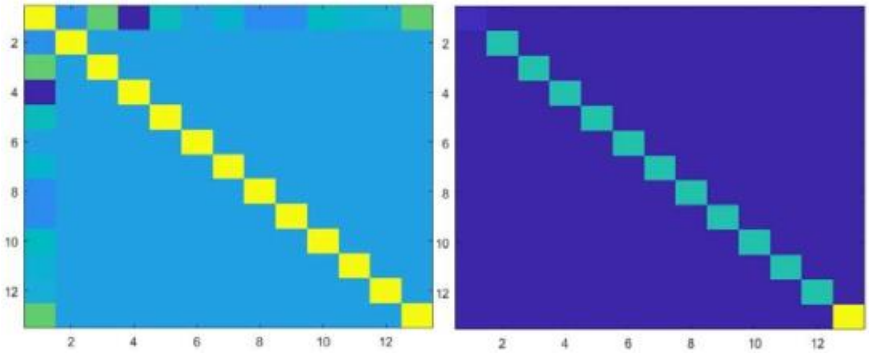

Figure 2: The covariance matrix before decorrelation and after decorrelation

As it is shown is fig. 2, The first feature is highly correlate with other features. After decorrelation, the power of the first feature turns to be very low. Since Naïve Bayes classifier is optimized to get a lowest error rate when all the features are uncorrelated from each other, when the features number get larger, the lower accuracy it may get. Based on this assumption, we drop some features that have very low power in the covariance matrix and only use the discriminant features to train and test. And By dropping the first 4 features after decorrelation, we get the highest validation accuracy as 87.23%.

Moreover, we find out that if we use PCA to get a too small feature space, the accuracy will drop rapidly, we guess the reason is that features will get correlate with each other when we map the feature space onto a lower feature space. The confusion matrix of the best Naïve Bayes classifier is shown below:

Table 1: confusion matrix of Naïve Bayes classifier

	Posture 1	Posture 2	Posture 3	Posture 4	Posture 5
Posture 1	0.9649	0.0206	0	0.0056	0.0090
Posture 2	0.0116	0.9555	0	0.0123	0.0205
Posture 3	0	0	0.7027	0.2889	0.0084
Posture 4	0	0.0544	0.2039	0.7401	0.0015
Posture 5	0	0	0	0.0012	0.9988

The random classifier for this feature space we chose is 20.14%, which mean the performance of Naïve Bayes classifier is much better than a random guess. For the performance of solving lacking data problem, Naïve Bayes shows to be robust to this problem. Even though the number of testing data is even more than that of training data, the accuracy is still very high and if we use the large training dataset to train the classifier, the accuracy gets even worse.

3.5.2 SVM

The highest accuracy we get from SVM is 96.45%. We choose Gaussian kernel SVM in this project because the dataset is not linear separable, and we want to get a stronger classification ability. The parameter we need to define is the cost and the gamma value. Our chose is cost equals $10^{0.55}$ and the gamma should be $10^{-2.61}$. For feature selection, we use all 22 features that we generate before, with the threshold 5 and no PCA used. The random classifier accuracy for this problem is still 20.14%, which means SVM we use has better performance than Naïve Bayes and random classifier of course.

SVM with a gaussian kernel is a method that maps the feature space into a higher feature space and optimize the decision rules in the higher feature space to get a lower loss function value. In this project, we don't need to develop a support vector machine from scratch by ourselves. In MATLAB, by using the open source "libsvm", we can easily build an SVM classifier to solve the problem. By using the function "svmtrain()" we can get the SVM model based on the input training data and using "svmpredict()" we can get the predict vector by inputting the testing data and label.

We use the original cross validation method at first. By split the whole data set into 24 groups, we use one of 24 group data as the validation data and the others as the training data. In MATLAB, we can use the function "cvpartition()" to do this work. We got two 120 by 60 matrices in which each element stands for the mean and standard variance of 24 trials for the specific argument chose. The average accuracy can easily reach to 99% with a very low standard variance. However, when we apply the best arguments into the model and test, the vadidation accuracy is not that high and can only reach to 91% for many trials. At first, we thought it may be due to overfitting problem. But we overturned our conjecture because when we use cross validation, the validate data has never appeared in the training processing, which mean the accuracy of cross-validation should be able to represent the performance of the classifier.

Taking the question, we also tried the cross-user validation. From this part we find some problem. First, the threshold we choose is too high. We choose the threshold as 7 at the first time and we find that nearly all the data that belongs to posture 1 have been dropped and the data generate by user 3, 4, 7 (start from 0) have been totally dropped as well. The reason that we can get a very high accuracy is that the data for posture 1 is so less that cannot influent the accuracy and in fact, nearly all the remain data that belongs to posture 1 is misclassified. What more, as we dropped all short data, if a short testing data being put into this classifier, it is hard for it to classify it correctly. So, we choose a smaller threshold and use the cross-validation method to find a better parameter choice.

By using the parameters that recommended by cross-validation we can get an accuracy as 94%. After that, we made some accommodation of the parameters to avoid overfitting. We choose a little bit smaller cost value and gamma to get a higher accuracy. Thus, we finally get the accuracy that is more than 96%. In this project, even though the number of testing data is twice of the number of training data, the accuracy is still very high. If we use the large dataset to train the classifier, the performance in the sense of accuracy is similar.

Table 2: confusion matrix of SVM

	Posture 1	Posture 2	Posture 3	Posture 4	Posture 5
Posture 1	0.9713	0.0206	0.0056	0	0.0026
Posture 2	0	0.8962	0.0027	0.0048	0.0963
Posture 3	0	0	0.9696	0.0223	0.0081
Posture 4	0	0.0084	0	0.9808	0.0107
Posture 5	0	0.0072	0.0026	0.0089	0.9813

3.5.3 K mean

In our project, the highest accuracy we get from K mean is 94.3%. K mean is a weak supervision classification. The basic idea is using the measure of distance in the feature space to help to decide which cluster a data belongs to. At first, we use the k mean method to divide all data into 5 clusters and each cluster represent a class of posture. Firstly, we use 13 features to train the classifier and get 50~60% accuracy. The accuracy is higher than random classifier, but it is still too low. In MATLAB, we can train this classifier by using function “kmeans()”.

Based on the assumption that the distance in the feature space is the most discriminant property in k nearest mean method, we think the accuracy will get higher if we add more features. As the dimension of feature space get higher, the distance within a same class will be smaller than the distance between classes, and thus, more data will be classified correctly. So, we put all 22 features to train the classifier, and the accuracy changes to 60~70%.

The performance of K nearest mean classifier has been improved a lot, however, we are still not satisfied by its’ accuracy. Since the dataset offered to us is well-labeled, we decide to combine the idea of k mean and the supervised learning method to get a higher accuracy. We designed a new classifier as below:

Firstly, we need to divide all the training data into 5 groups according to its’ label and do k means in each group of data to find k centroids of each class of data. We should keep these centroids as the properties of the model.

Secondly, we input a vadidation data, we need to calculate the distance between the input data and 5 centroids (Euclidean distance). Sum up the top 1/4 largest inverse of distances between the centroids and the input data, we get the measure of the similarity between the input data and the 5 class (posture). The highest similarity class will be chosen to be the prediction of the input data. In MATLAB, we can calculate the distance by the inner product between the distance vector and its' transpose vector.

The performance of this method is not stable. Since the random centroid chosen during the model initialization, the centroids we get can changes at each trial. The accuracy of this method can be in the range from 83% to 94%.

3.5.4 Mean Squared Error (MSE) classifier

We use toolbox "PRTools5" in MATLAB. It is used to get MSE parameter and accuracy.

Principle: MSE classifier is a method which calculates misclassified point distance with boundary and trying to find the minimum sum of all misclassified point distance. Hence the goal of MSE is to reduce misclassified training points distances but not classify them all correctly.

Feature choose: sph (sphere); ori -13(original 13 features)

Table 3: The relationship between parameters and accuracy

	acc	ori-13	ori-norm	ori-std	ori-max	16 (skewness)	19(sphere-std)	22(sph-mean)
threshold=0	train	88.17	48.21	88.17	85.97	89.67	92.02	92.21
	val	80.74	55.74	80.74	80.12	81.32	81.11	82.02
threshold=7	train	88.96.		88.96	90.70	90.52	90.38	90.67
	val	86.14		86.14	85.61	82.47	81.12	86.46
PCA	accuracy decrease heavily							

Best result accuracy for test :86.55%

Final feature chooses: 13 original features + 3 spherical-mean features=16 dimension

Method of choosing features: From table above, we can see if we add some new features, the accuracy will decrease. However, if we add spherical-mean, the accuracy will become higher. Then we combine original 13 features and spherical -mean features which achieve 86.55% accuracy.

Choose Parameters method:

threshold	STD	PCA
6	None	None

We use loop to find best threshold and best PCA dimension. We find that accuracy will decrease heavily if we apply PCA in this method. It means that reduce the degree of freedom is a good choice in this method

Table 4: Confusion matrix of MSE method

	Posture 1	Posture 2	Posture 3	Posture 4	Posture 5
Posture 1	0.9666	0.0206	0	0	0.0129
Posture 2	0.0064	0.9820	0.0011	0.0025	0.0080
Posture 3	0	0.0206	0.7033	0.0027	0.2734
Posture 4	0	0.3031	0	0.6969	0
Posture 5	0	0.0130	0.0012	0.0069	0.9789

3.5.5 Artificial Neural Network (ANN)

we use keras in python. It is used to build ANN network and calculate accuracy.

Principle:

ANN simulates human nerve cell. The structure is input layer, hidden layer and output layer. It uses back propagation to update weight. In this method, we build 6 hidden layers to increase the complexity of neural network. In the network, we use "RELU" activate function to do nonlinear mapping and use dropout method (change random units to zero) and weight decay to ovoid overfitting. In the final, we use "SoftMax" to transform data to probability and do final decision.

Table 5: Parameter chose in ANN

Learning rate	Weight decay	dropout	epoch	momentum	Batch size	threshold
0.0001	0.0001	0.2	200	0.90	64	None

Best result accuracy for validation :96.16%

Method of choosing features: We first choose learning rate, we found that if learning rate is too large, the accuracy is unstable and cannot find highest accuracy. Then we adjust epoch to see where its accuracy upper limit is.

Then we adjust threshold and PCA. we find that if we use threshold and PCA, the accuracy will decrease because number of constrains will decrease.

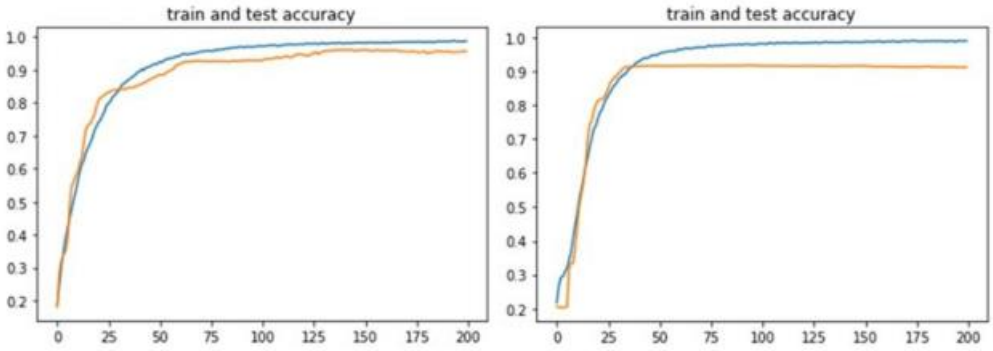

Figure 3: Threshold =None (acc=96.16, left); Threshold =6 (acc=91.15, right)

Table 7: Feature chose in ANN

	Original 13 features	Original 13 features+ sphere mean	Original 13 features+ threshold 6	Ori 13+PCA dimension 10
Test accuracy	96.16	93.36	91.73	0.8435

Table 8: Confusion matrix of ANN

	Posture 1	Posture 2	Posture 3	Posture 4	Posture 5
Posture 1	0.96	0.011	0.026	0.0034	0.0025
Posture 2	0.032	0.94	0.0061	0	0.022
Posture 3	0.016	0	0.96	0.02	0.00063
Posture 4	0	0	0.00026	0.96	0.041
Posture 5	0.0023	0.0073	0	0.024	0.97

3.5.6 Random Forest Classifier

We use MATLAB function Tree Bagger in this method. It is used to build tree structure and calculate accuracy.

Principle: Random forest is consisting of many decision trees. Each decision tree has binary choices. Trainning data features are use here to as threshold to build random forest structure. When Validation data are regarded as input to the decision tree, it will follow the tree's structure and justify flow to which decision note. Final decision is based on majority of decision note's decision.

Table 9: Feature & parameters chose in Random Forest

	Standardization	Threshold=3	Threshold=4	Threshold=5	Threshold=6
train accuracy	1	1	1	1	1
Validation accuracy	0.7053	0.7032	0.6978	0.6761	0.6338

(Threshold=3)	PCA-dimension=10	PCA-dimension=11	PCA-dimension=12	PCA-dimension=13
train accuracy	1	1	1	1
Validation accuracy	0.7266	0.7719	0.7633	0.7812

Best result accuracy for test .0.8150

Best parameter: threshold=3; PCA=13; tree number=170

Method of choosing features: The accuracy of random forest is not stable, so we set loop and each loop run 10 times to calculate mean of accuracy. It can help me choose best threshold. Then, we

apply PCA when threshold equal to 3, we use loop and calculate each PCA dimension 10 times to get mean accuracy to find best PCA dimension.

We use above best parameter and use loop to choose the best trees number (170).

Table 10: Confusion matrix of Random Forest

	Posture 1	Posture 2	Posture 3	Posture 4	Posture 5
Posture 1	0.92	0.010	0.0053	0.012	0.048
Posture 2	0.011	0.9520	0	0.00068	0.035
Posture 3	0.030	0.0010	0.75	0.060	0.16
Posture 4	0	0.39	0.0056	0.40	0.19
Posture 5	0	0.020	0.0045	0.0081	0.97

3.5.7 K Nearest Neighbor (KNN)

We use MATLAB function in this part. The model is built by "fitcknn" and I can adjust the number of neighbors. We use "predict" function to get predict label for calculating test accuracy and confusion matrix.

The principle of KNN is based on distance measure to find its k nearest neighbor. The data which are in these regions are labeled by majority classes. When applying test data, the data is located in which region will be labeled in the region's class.

Table 11: Parameter chose in K-NN

	Threshold=3	Threshold=4	Threshold=5	Threshold=6
Train accuracy	0.9968	0.99680	0.9967	0.9971
Validation accuracy	0.7986	0.7949	0.7848	0.8125

	PCA=8	PCA=9	PCA=10	PCA=11	PCA=12	PCA=13
Train accuracy	0.9769	0.9966	0.9972	0.9975	0.9970	0.9963
Validation accuracy	0.7444	0.7956	0.7883	0.7417	0.7023	0.8226

	K=10	K=50	K=80	K=90	K=150	K=200
Train accuracy	0.9991	0.9963	0.9942	0.9939	0.9888	0.9848
Validation accuracy	0.7184	0.8226	0.8351	0.8484	0.8576	0.8557

Best result accuracy for validaion .0.8576

Best parameter: threshold=6; PCA=13; k neighbors=150

Choose parameter method: We first use loop to find best threshold (6) and use loop to find best PCA dimension (13). In the end, we use loop to search best number of neighbors.

Table 12: confusion matrix K-NN

	Posture 1	Posture 2	Posture 3	Posture 4	Posture 5
Posture 1	0.97	0.021	0	0.0056	0.0077
Posture 2	0.0030	0.96	0	0.00068	0.039
Posture 3	0	0.0010	0.69	0.0033	0.27
Posture 4	0	0.032	0.016	0.77	0.037
Posture 5	0	0.0035	0.0891	0.0110	0.90

4 Analysis: Comparison of Results, Interpretation

4.1 Analysis each classifier

4.1.1 Mean Squared Error (MSE) classifier

In MSE, we find that threshold is very important in accuracy. If threshold is too small, there will contain more features, but some of them is wrong information which will decrease the accuracy. However, if the threshold is too large, some postures number is really small. For example, if we choose threshold equal to 7, the vadidation accuracy can achieve almost 90%. However, there will be no posture 1 in vadidation data. Hence, it is fake test accuracy which make no sense as it cannot distinguish hand posture. Combine the balance, we chose threshold equal to 6.

4.1.2 Artificial Neural Network (ANN)

In ANN, we find that if we choose large threshold or apply PCA in ANN, the test accuracy will decrease. We believe the reason is the feature what we extract is good. If we use threshold and PCA, we will lose some information (number of constrains will decrease) which will lead to accuracy deceasing because ANN need much data to train to get high accuracy.

4.1.3 Random Forest

In this part, we made a mistake in the beginning. We use validation data itself to do standardization which make our random forest classifier easily achieve 93% accuracy for validation data. However, if we apply train data's mean and standard deviation in test data, the accuracy decease heavily (70%). We think the reason is the number of training data is not enough to present vadidation data. We calculate train and test's mean and standard deviation separately, find that they are quite different.

4.1.4 KNN

In this part, we find KNN's result is quite stable. It will not change if we run it again. The reason is it is not a data driven method. It does not need to adjust weight or something else. Once the number of neighbors is decided, the label region is also decided.

4.1.5 Naïve Bayes method

The performance of Naïve Bayes classifier is highly relevant to the dataset quality. The features we use should be uncorrelated with each other and the accuracy will not get higher by simply add more features. The advantages of Naïve Bayes classifier are that the model size is very small, and the result is stable. And Naïve Bayes classifier is flexible to change if more features are used, we only need to extend the covariance matrix.

4.1.6 SVM classifier

SVM classifier in this project performs very well. The accuracy is high and stable, and the model size is small. The short coming of SVM is that the model is easily to overfit. In our project, a small change in the parameters will affect the accuracy of vadidation data a lot while the training accuracy is still very high. That means, we cannot choose the parameters only based on the cross-validation. And SVM is also a classifier that classify data based on the distance measurement in hyper-feature space, the more feature we use, the higher chance we get a stable SVM classifier, which is different from Naïve Bayes classifier.

4.1.7 K nearest mean

K nearest mean is a weak supervised learning algorithm and this method has strict requirements on the distribution of data in the feature space. The larger feature dimension be used, the higher accuracy will be. To use the supervised data that provided to us in this project, we designed a new method that combine the supervised information and the classification rules of K nearest mean together. This method is not stable, and its' accuracy can range from 85%~94% by using a same set of parameters. The reason may come from the initialization of seeds in each cluster of K nearest means.

6 Summary and conclusions

From the aspect of feature engineering, it is surprised to see that the original 13 features can work well enough, the reason is that even any single one of them makes no sense, the combination of some of them can actually represent some important information. And the new 9 features that designed by us can help many classifiers to perform better.

From the aspect of classifier designing, we find that for a specific dataset, some classifier will be more suitable than others. SVM and ANN has the highest accuracy, however, the model size of ANN is too large and SVM is very hard to find a set of parameters that will not suffer from overfitting problem. In general, statistical based classifier like Naïve Bayes and K-NN will have a lower accuracy than other classifiers like SVM and MSE. The reason is that the statistical based classifier has too small model size and only use the statistic properties of a

dataset, which means many detail information of the dataset will be dropped during the statistical processing.

For the following research, we want to find a method that can extract the features from the original dataset automatically and optimize the correspond classifiers with the feature chosen to get a higher accuracy. Now that the hand posture dataset is similar to point cloud problem, maybe we can get inspiration from that theory.

References

[1] Matlab tools for compute the confusion matrix and MSE: PRTOOLS 5.0. [Online]. Available: http://37steps.com

[2] libsvm, MATLAB tools for building SVM classifier,". http://www.csie.ntu.edu.tw/~cjlin/libsvm/.

[3] keras from temsorflow 2.0 used in ANN inplementation.