EE559 Final Project

Data Set(s): Hand Postures

Zheng Wen, zwen1423@usc.edu

5/7/2020

1. Abstract

In this project, the real-world dataset from UCI machine learning repository, Motion Capture Hand Postures Dataset collected by Louisiana Tech University[1], is processed to recognize the postures of hand with the coordinates of several makers and the class of the posture. Some useful features are designed from the original dataset, then the features are normalized and selected by backward feature selection. The dimension of features are further reduced by dimension reduction algorithm. Naïve Bayes, perceptron algorithms, support vector machine and k-nearest neighbors are used to classify and the parameters are selected by cross validation. The best result on test set is 96.85% and the best result on training set is 99.69%.

2. Introduction

2.1. Problem Statement and Goals

In this problem, the Hand Posture dataset is chosen as the target dataset. This dataset contains the coordinates of several unlabeled markers on a glove when doing some gestures. There are 5 kind of postures to be recognized by 12 users, the data of 9 users are offered as training set and the rest 3 users are regarded as test set. In this problem, a pattern recognition system is developed, containing feature extraction changing the data to a useful form, preprocessing, feature selection, dimension adjustment and classification to figure out what posture is made in the test set. To achieve a better performance, cross validation is used in this problem to select parameters and choose features.

2.2. Literature Review

From the tutorial, because all of the markers are unlabeled when obtaining these data, so the coordinates of these data points are meaningless in terms of pattern recognition if they are directly feed into the system, some useful features which have nothing to do with the labels of the markers must be developed as the useful data. Besides, because this dataset is highly correlated by user, so the dataset must be split by user when cross validation is used, which means the validation set could not contain the data from the

user contained in the training set, or the cross validation will become meaningless, the leave-one-user-out technique should be used.

3. Approach and Implementation

In this part, some permutation invariant features are designed in the begining to make the data useful, and the features are preprocessed by normalizing and data balancing, both of which will be explored in 3.1. Then, Phi machine is tested and the backward feature selection technique is also used to further adjust the dimension, which will be covered in 3.2. The usage of data such as how the training set and validation set are split, the number in each set is explained in 3.3, the description of each classification is shown in 3.4.

3.1. Preprocessing and Feature Engineering

The data in the original dataset denote the coordinates captured by cameras, the number of markers captured by the cameras differs between different postures, and the markers are unlabeled, the data points with the same sublabel in different samples may denote the position of different markers, so the data could not be compared simply by its original form. To deal with this problem, some permutation invariant features are designed.

Permutation invariance is defined below,

$$f(\boldsymbol{x}_0, \boldsymbol{x}_1, \dots, \boldsymbol{x}_d) = f\big(\sigma(\boldsymbol{x}_0, \boldsymbol{x}_1, \dots, \boldsymbol{x}_d)\big)$$

where f denotes a function, whose inputs are the coordinates of data points with values in the original data set, $\boldsymbol{x}_i$ denote the data points in the original data set, σ denotes a random permutation for the existing data points. The missing data points are omitted because there are a lot of missing data points as the sublabel increases, and because the markers are unlabeled, some common data filling methods like mean filling and mode filling are meaningless. In my implementation, the extracted features are listed in Table 1. A name is assigned to each of them in convenience. XMean, XStd, XMax, XMin stands for the mean, standard deviation, maximum value and minimum value for the X axis for a single sample, which are similar in the features beginning with Y and Z. SR stands for square root, because this feature is designed as the square root of the sum of square of the three coordinates for a single data point. Mul stands for multiply, because the feature is a simple multiply of the three coordinates for a single data point. CM stands for cross multiply, because the feature is the square root of the multiply of the combination of the three coordinates for a single point. Num stands for number, because this feature count the number of non-empty

coordinates of a sample. In total there are 16 dimension features are designed to classify the data set.

XMean: mean(x_i)	XStd: std(x_i)	XMax: max(x_i)	XMin: min(x_i)
YMean: mean(y_i)	YStd: std(y_i)	YMax: max(y_i)	YMin: min(y_i)
ZMean: mean(z_i)	ZStd: std(z_i)	ZMax: max(z_i)	ZMin: min(z_i)
SR: mean$\left(\sqrt{x_i^2 + y_i^2 + z_i^2}\right)$		Mul: mean$\left(\sqrt[3]{x_i y_i z_i}\right)$	
CM: mean$\sqrt{(x_i y_i + y_i z_i + x_i z_i)}$		Num: number of non-empty points	

Table 1. Extracted Feature and Name

Motivation: these features are designed by 3D intuitions of the unlabeled markers. For each sample, mean of each single coordinate denotes the centroid along a single axis, the standard variation of each single coordinate denotes the deviation along a single axis. The maximum and minimum value of each sample denote the range on each axis. These are all useful features because for different kind of postures, these features may vary a lot. The next three features measure the holistic features of each sample, instead of just along a single axis. $x_i^2 + y_i^2 + z_i^2$ denotes square of distance between the measured data point and origin. So, this feature measures the centroid in 3D view. $x_i y_i z_i$ denotes the volume of cubic whose diagonal is the segment with the original and the measured data points as endpoints. So, this feature measures the average volume under different postures. $x_i y_i + y_i z_i + x_i z_i$ denotes half of the surface area of the same cubic described above, so this feature is another kind of measure of spatial distribution of a certain kind of posture.

Note that this procedure is done before data preprocessing step like normalization and standardization, because some data processing method like standardization will hurt the extracted feature, for example, after all data points are scaled into [0, 1], the multiply between different data will make the outcome smaller for the two factors are all smaller than 1. But when multiplying two numbers in the original dataset, this operation will make the outcome larger. Even though the outcome are normalized by root of the corresponding multiply times, the distortion incurred by preprocessing method is still not estimable. Another reason by doing so is that because there are a lot of missing data in the original dataset because of occlude, and as is shown above, this dataset is not a balanced dataset, to get a better performance, we should upsample or downsample the dataset, which is hard when a number of data is missing.

	Class 1	Class 2	Class 3	Class 4	Class 5	Total
Original	2784	2582	2649	2476	3009	13500
Upsample	3341	3341	3341	3341	3341	16705
Downsample	2127	2127	2127	2127	2127	10635

Table 2. Number of Samples in Each Class and in Total

Based on naïve Bayes classifier and leave-one-user-out cross validation, several preprocessing methods are carried out after feature extraction. Because the data is not normalized, so the data should firstly be processed by standardization. Two common standardization methods are tested, one is z-score standardization, and the other is min-max standardization, both of which could be carried out by functions in sklearn library[6].

Besides, as is shown in Table 2, the number of samples in different classes are not balanced, class 5 has the most training samples, in total 3009 samples, while class 4 has the least training samples, in total 2476 samples, the ratio between these two class is about 4:5. To improve the classification accuracy tentatively, the data points are either leave it as the original form, either balanced by upsampling, either balanced by downsampling. Both of these two algorithms are implemented by user because of the correlation of the data within a user. The upsampling algorithm will generate more data in the minority classes by Synthetic Minority Oversampling Technique (SMOTE)[2], this algorithm will increase the data points by finding a nearest points for a random data points in minority class, the new data point will be added onto the line connecting them randomly as is covered in discussion session, which could be implemented by over_sampling.SMOTE function in imblearn library[3]. The downsampling algorithm will randomly drop the data points in the majority classes to balance the whole dataset, which could be implemented by the function under_sampling.RandomUnderSampler function in imblearn library[4]. So, in the preprocessing part, the data are processed by the following 6 strategies:

1. min-max + original data
2. z-score + original data
3. min-max + upsampled data
4. z-score + upsampled data
5. min-max + downsampled data
6. z-score + downsampled data

The outcome is shown in Table 3. From Table 3, we could see that there is no difference between min-max scaling and z-score scaling, which may because the original data distribution is normal, and there is no abnormal data points in the whole dataset after feature selection. Another reason may lies in Naïve Bayes classifier, because these two method do not change the distribution of the original dataset obviously, which Naïve Bayes depends on, so the final result do not change a lot. But different data balancing method

have different results. In this project, Strategy 4, z-score + upsampling is chosen because of the higher validation accuracy, also because the z-score scaler method could make the dataset zero-mean, which will benefit other classifier like support vector machine[5].

	Strategy 1	Strategy 2	Strategy 3	Strategy 4	Strategy 5	Strategy 6
Training Mean Accuracy	0.9056	0.9056	0.9001	0.9001	0.9010	0.9010
Training Standard Deviation	0.0109	0.0109	0.0094	0.0094	0.0095	0.0095
Validation Mean Accuracy	0.7731	0.7731	0.7968	0.7968	0.7901	0.7901
Validation Standard Deviation	0.1229	0.1229	0.1338	0.1338	0.1269	0.1269

Table 3. Classification Effect of Different Preprocessing Method

3.2. Feature dimensionality adjustment

In this part, based on Naïve Bayes classifier and leave-one-user-out cross validation, the effect of Phi machine is tried in the beginning. Then the features are reduced by backward selection.

Using the balanced dataset, a Phi machine with a degree-2 polynomial features is used. The results are shown in Table 4, the Train Accuracy and Val Accuracy stands for the mean accuracy on training set and validation set throughout the cross-validation process.

	Original	All Features	Interaction only
Train Accuracy	0.9001	0.7747	0.7917
Val Accuracy	**0.7968**	0.6146	0.6155

Table 4. Effect of Phi Machine

In Table 4, the column name “All Features” shows the effect of a Phi machine same as the one mentioned in lecture, the features are augmented by the product of every two features selected from the original features. Surprisingly, the augmented features should have received better result than the original features, however, the accuracy of augmented features are much less than the original data both on the validation set and the training set. The reason of this is Naïve Bayes classifier is not a linear classifier, but a

classifier based on the estimated prior probability of each class and the conditional probability of each sample. In Naïve Bayes, when estimating these probability, the classifier will assume that all of the features are independent from each other. However, in the features generated by Phi machine, there are a lot of correlated features, which will cause the performance worse even through the number of features are increased. To justify this point, the Phi machine with interaction features only is used to train the Naïve Bayes classifier, where the features like x_i^2 will be eliminated. So the number of correlated features will decrease, the result is shown in Table 4, the column named "Interaction only" shows the result. Because the accuracy with interaction-only Phi machine is higher than the original Phi machine, so the statements above is correct, and in this project, besides the decreased performance, because if more feature is used, the training process will cost much more time, so the Phi machine is not used.

Feature Name	XMean	XStd	XMax	XMin
Training Accuracy	0.9142	0.8985	0.9007	0.9044
Validation Accuracy	0.7963	0.7899	0.7956	**0.8161**
Feature Name	YMean	YStd	YMax	YMin
Training Accuracy	0.9072	0.8792	0.8729	0.9032
Validation Accuracy	0.7684	0.7552	0.7379	**0.8052**
Feature Name	ZMean	ZStd	ZMax	ZMin
Training Accuracy	0.9110	0.8907	0.9026	0.8971
Validation Accuracy	**0.8062**	0.7966	**0.8127**	**0.8008**
Feature Name	Num	SR	Mul	CM
Training Accuracy	0.8572	0.8825	0.9069	0.9012
Validation Accuracy	0.7471	0.7609	**0.8246**	**0.7988**

Table 5. Accuracy of Ablated Dataset

Then, the features designed above are further selected by backward feature selection to test whether there is a single feature influence the performance

most. A feature is dropped and the mean accuracy by cross validation is tested again, the result is shown in Table 5. The name of dropped feature is illustrated in Table 1.

From Table 5, the improved classification accuracy on validation set is shown in bold. After the "Mul" feature is dropped, the classification accuracy increased the most, so this feature is dropped from the dataset.

3.3. Dataset Usage

In this project, the dataset containing 9 users, 1500 samples per user is used as training set. In all of the feature engineering processes and the data preprocessing processes, cross validation is used to decide the effect of each preprocess. The validation set is constructed by the data of only 1 user, and the data of from the other 8 users is used as training set. After the features are extracted, there are 13500 data points in the training set and 1500 data points in the validation set when trying different data preprocessing methods. The decision is made based on the classification accuracy of the validation set. When testing the effect of method balancing the dataset, the number of data in the training set and validation set vary in each cross-validation process because the number of samples under each user varies after balancing, which is listed in Table 6 and Table 7 respectively.

Cross Validation Step	1	2	3	4	5	6	7	8	9
Training set	15010	14625	14715	14965	14730	14865	14950	14825	14955
Validation set	1695	2080	1990	1740	1975	1840	1755	1880	1750

Table 6. Data Usage in Upsampling Algorithm

Cross Validation Step	1	2	3	4	5	6	7	8	9
Training set	9390	9745	9510	9430	9370	9425	9385	9475	9350
Validation set	1245	890	1125	1205	1265	1210	1250	1160	1285

Table 7. Data Usage in Downsampling Algorithm

As illustrated in data preprocessing part, the upsampling algorithm is used in the following procedure, so the number of samples in the training set and

validation set in feature dimensionality adjustment part, the classifier training and parameter selection part, the number of samples in the training set and validation set is shown as Table 6. In all of the part, the cross validation step contains 9 folds, because the data is not randomly split into training set and validation set, so the cross validation step is only run once in each usage. The cross validation is used for selecting the most suitable data processing method for 6 times, for the feature dimensional adjustment part for 12 times, twice in testing the effect of Phi machine, 10 times in backward feature selection, for the parameter selection part when training the classifier for 170 times, 100 times for SVM parameter selection, 20 times for perceptron parameter selection and 50 times for KNN parameter selection, so in total, the cross validation is used for 188 times.

The test is kept sealed during the whole process until testing the effect of the trained classifiers, so in total, the test set is used for 4 times.

3.4. Training and Classification

In this part, in total 4 classifiers are used.

3.4.1. Naïve Bayes Classifier

In this project, the Naïve Bayes Classifier is implemented by the function GaussianNB in sklearn library[6], in which the likelihood of the features is estimated as Gaussian distribution: $P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} exp\left(-\frac{(x_i-\mu_y)^2}{2\sigma_y^2}\right)$, where the parameter σ_y^2 and μ_y are estimated using maximum likelihood[1]. The prior of each class is set to None, which means the prior of each class is estimated from the training data. This classifier is used as baseline. To further exploit the performance of this classifier, cross validation stated in 3.3 is carried out.

3.4.2. Support Vector Machine

Support Vector Machine in this project is implemented by the svm.SVC in sklearn library[6]. This is a multiclass classifier in which several support vector machines are trained and the decision region is defined by one-versus-rest strategy. The kernel of these support vector machines is Gaussian kernel, the parameter of the penalty of misclassification and the γ in Gaussian kernel is chosen by cross validation. The cross-validation step is stated in 3.3. The mean accuracy of each cross-validation steps is shown in Fig. 1. The range of C and γ is chosen in log space, from $10^{-3}$to 10^{3}, 10 for each parameter in consideration of the processing speed, in total, 100 kinds of combination of these two parameters are testes, the results are shown in Fig. 1, the x axis

stands for the sequence of C, and the y axis stands for the sequence of γ. The parameter is finally decided by the combination of C and γ with the highest validation accuracy, where C=2.1544346, γ =0.00464159.

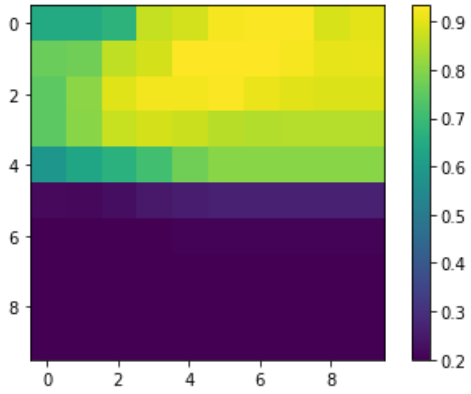


Fig. 1. Accuracy of Cross Validation of SVM

3.4.3. Perceptron Classifier

Perceptron Classifier in this project is implemented by Perceptron function in sklearn.linear_model library[6]. Stochastic gradient descent is used to update the weights of perceptron, the training set is randomly shuffled after each training epoch, the max iteration is 1000, if the difference of the loss in current iteration and the loss in the last iteration is smaller than 10^{-3}, the training procedure will be stopped. The learning rate is chosen by cross validation as stated in 3.3. The mean accuracy of cross-validation steps is shown in Fig. 2. The x axis stands for the learning rate η. The range of η is chosen in log space, from 10^{-5} to 10^{3}, in total 20 value of η is tested. Because after the η exceed a threshold (from Fig. 2, approximately 10), the mean accuracy will not improve again. So, the final η is chosen as the first value achieving this peak, where η =10.

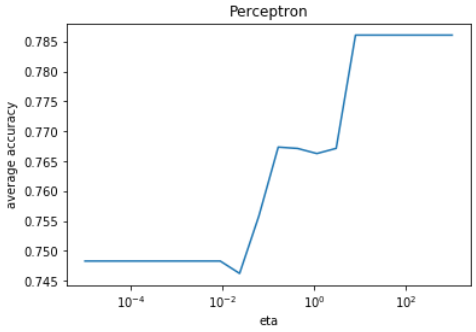


Fig. 2. Accuracy of Cross Validation of Perceptron

3.4.4. K-Nearest Neighbor Classifier

In this project, the K-Nearest Neighbor classifier is implemented by KNeighborsClassifier in sklearn.neighbors library[6]. In this model, the predicted label of each test sample is given by the label of the training data appear in the K neighbor of the test sample most. The number of the neighbor to be detected for each test sample is decided by cross validation as stated in 3.3. The mean accuracy of cross-validation steps is shown in Fig. 3. The x axis stands for number of the selected neighborhood points, K, which is chosen in linear space, from 2 to 100, in total 50 value. The number of neighborhoods with the largest mean validation accuracy is chosen as the final selection, where K=12.

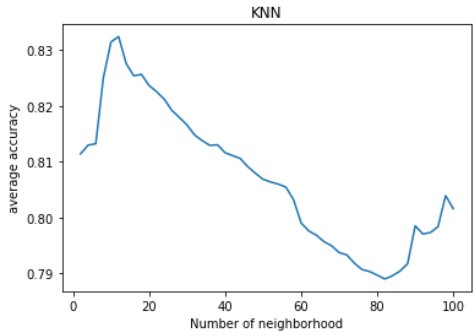


Fig. 3. Accuracy of Cross Validation of KNN

4. Analysis: Comparison of Results, Interpretation

	Naïve Bayes	SVM	Perceptron	KNN
Mean	0.8246	0.9332	0.7860	0.8323
Std	0.1006	0.0755	0.1326	0.1407

Table 8. Mean and Standard Deviation of Accuracy in Cross Validation

The mean and the standard deviation of the classification accuracy of the 4 trained classifiers with the parameters selected in 3.4 using cross validation are shown in Table 8.

From Table 8, we could see that the SVM has the highest mean classification accuracy, 11% more than the baseline, and the standard deviation is also lower than the baseline. because SVM with Gaussian kernels could find the most suitable discriminant boundaries for the dataset which is not linearly separable by minimizing the loss function and maximizing the distance between two decision margins. The one-versus-rest strategy could thoroughly find decision boundary for every class and combine them together to get the final results. So, SVM could get the highest mean

and lowest standard deviation on the accuracy of cross validation. But there is also some drawback of SVM. In this dataset, the parameter selection using the whole training set is very slow, which is because the one-versus-rest strategy, when trying each single choice of the parameters, in total 5 2-class SVM should be trained on the whole training set, in my experiment setting, there are 100 combinations of parameter to choose from, and each is obtained from a 9-fold cross validation, so in total, there are 4500 2-class SVM should be trained. So, it would cost a lot of time.

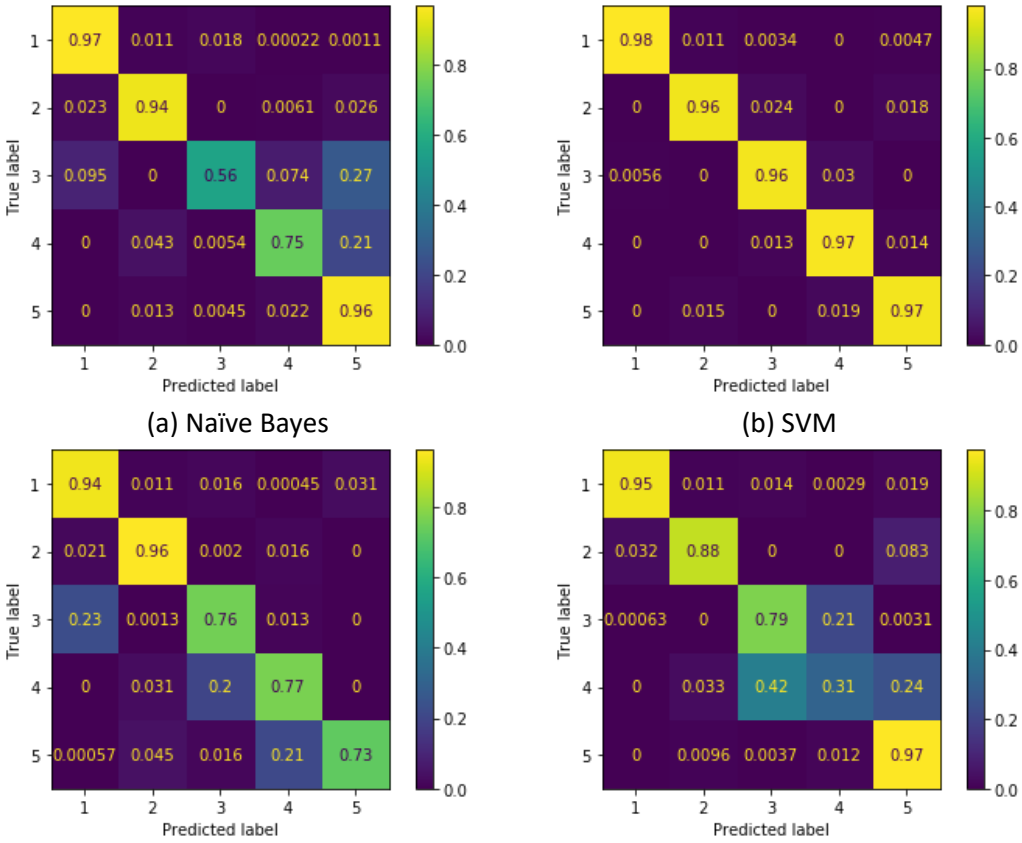
The second highest algorithm in mean of accuracy during cross validation is KNN algorithm. The accuracy is improved for 1%, but the standard deviation is also higher than the baseline. KNN tends to find the most occupied labels of the training data from the neighborhood of a given test sample, and predict the unknown input data as the selected label, which means KNN classifier will only be trained once in each parameter settings. Besides, the training time of KNN depends on the parameter K, if more data points in the neighborhood of a sample should be found, more time should be spent. Because KNN is a statistic based method, the classification accuracy is largely depend on the input data, which means if there are some fluctuation in the training set, the classification accuracy may vary a lot, which is also a reason that the standard deviation of the KNN method is the largest among these 4 algorithm.

The third highest algorithm in mean of accuracy during cross validation is Naïve Bayes algorithm, which is selected as baseline. Naïve Bayes classifier is also a probability-based algorithm, but its standard variation is smaller than KNN, the reason of which is in Naïve Bayes in this project, the likelihood of each feature is selected as Gaussian distribution, which is more stable than just judging the label by the Euclidean distance between input and the training data, but less precise.

The poorest performance is obtained from perceptron algorithm. The mean is lower than the baseline and the standard deviation is higher. Because perceptron algorithm is a simple linear classifier, it could not handle the dataset not linearly separable well. Obviously, the Hand Posture dataset is not a linearly separable one. To prove this, another experiment where the perceptron classifier is trained with the data augmented by a 2-degree Phi machine is carried out, using the same learning rate selected in 3.4.3. In this experiment, the mean of perceptron is 0.8310, the standard deviation is 0.1365, both of them get improved. Besides, perceptron classifier is also sensitive to the training data, because even slightly difference in the data near boundaries could cause a change in updating the weights. The training time of perceptron is quite short because of the simple structure.

	Naïve Bayes	SVM	Perceptron	KNN
Train Accuracy	0.9071	0.9969	0.9627	0.9967
Test Accuracy	0.8291	0.9685	0.8358	0.7848

Table 9. Accuracy on Test Set and Whole Training Set



(a) Naïve Bayes (b) SVM

(c) Perceptron (d) KNN

Fig. 4. Confusion Matrix on Test Set

Next, the classification accuracy on the test set and the whole training set used to train these 4 classifiers is shown in Table 9, and the confusion matrices of them are also shown in Fig. 4.

From Table 9, the algorithm with highest classification accuracy on both training set and test set is still SVM, the accuracy on the test set is 96.85%. From the Fig. 4(b), the confusion matrix also shows that SVM could separate all of these 5 classes best, the classification accuracy on each class is about 96%. With proper parameter, SVM could find proper decision boundaries and avoid overfitting.

KNN algorithm could fit the training set almost as well as SVM, but the accuracy on the test set is the worst. This is because the KNN algorithm is statistic-based, it may over fit the training set. Fig. 4(d) shows that KNN could not recognize class 3 and 4 well, the sample in class 4 are mostly classified as class 3 and class 5, and the most confusing class of class 3 is class 4. This may be caused by the similar distribution between class 3 and class 4.

Perceptron classifier received the third highest training accuracy and the second higher test accuracy, which means perceptron classifier here could fit this training set well. From Fig. 4(c), the performance on class 3, 4, 5 is not so good. Because perceptron is a linear classifier, so the decision boundaries between class 1 and class 3, class 3 and class 4, class 4 and class 5 may not be linear.

The performance of Naïve Bayes is the worst in fitting the training set, and the third in test set performance. Because as illustrated above, the Gaussian distribution spreads wider, so it may not fit the training set well like KNN algorithm, but the wider distribution of Gaussian could also make the performance on the test set better than KNN because of no overfitting.

From all of the confusion matrices above, we could decide that class 1 and class 2 are distinct from the other three classes, in order to separate class 3, 4, 5 apart, non-linear classifier like kernel SVM or some more advanced probability-based algorithm should be used.

	Class 1	Class 2	Class 3	Class 4	Class 5	Difference
Naïve Bayes	3403	3907	3170	3395	2830	1364
SVM	3353	3321	3341	3325	3365	72
Perceptron	3273	3079	3427	3165	3761	1012
KNN	3338	3338	3342	3331	3356	32

Table 10. Classification Result with Random Inputs

Finally, all of the training samples are input into the 4 classifiers without looking at the labels, and the number of each class is shown in Table. 10, the "Difference" column measures the difference between the desired output labels and the real output shown in the previous column where the number of each class should be 3341 because the training set is balanced by upsampling method as stated in 3.1, i.e.

$$Difference = \sum_{i=1}^{5} |3341 - C_i|$$

where C_i is the output label in class i. From Table 10, we could see that all of these 4 classifiers could learn how to fit the training set. KNN fits the training set best, the distribution is closest to be balanced, but the performance on test set is not good, which is because of the overfitting. The second one is SVM, the output is also near balanced. The Naïve Bayes algorithm and perceptron algorithm does not fit the training set as well as SVM and KNN, which could also be inferred from the training accuracy shown in Table. 9.

5. Summary and conclusions

In this project, several permutation invariant features are designed from some unlabeled data for classification, some preprocessing strategy like scaling and data balancing are tried and evaluated by cross validation, the most suitable one is used for further steps. Feature dimension is also tried to be adjusted by Phi machine, backward feature selection and cross validation based on Naïve Bayes classifier, one of the most influential features is dropped to improve the holistic performance. Four classifiers, Naïve Bayes classifier, Support Vector Machine classifier, Perceptron classifier and K-Nearest Neighbor classifier are trained, parameters are selected by cross validation. After the parameter selection section, the whole training set is used to train these 4 classifiers and the test set is evaluated. The performance of these models is analyzed by the mean and standard deviation of accuracy using cross validation, the accuracy and confusion matrices on test set, as well as random Gaussian inputs, the SVM classifier could receive the best result, the accuracy on test set is 96.85%.

References

[1] https://archive.ics.uci.edu/ml/datasets/Motion+Capture+Hand+Postures

[2] Chawla, Nitesh V., et al. "SMOTE: synthetic minority over-sampling technique." Journal of artificial intelligence research 16 (2002): 321-357.

[3] https://imbalanced-learn.readthedocs.io/en/stable/generated/imblearn.over_sampling.SMOTE.html

[4] https://imbalanced-learn.readthedocs.io/en/stable/generated/imblearn.under_sampling.RandomUnderSampler.html

[5] https://towardsdatascience.com/effect-of-feature-standardization-on-linear-support-vector-machines-13213765b812

[6] Pedregosa, Fabian, et al. "Scikit-learn: Machine learning in Python." the Journal of machine Learning research 12 (2011): 2825-2830.