599 final Project

Data Set(s) *(specify which)*: Hand Postures

Author: Jiaqi Liu, email contacts: jliu9289@usc.edu

Date 08/05/2020

1. Abstract

In the project, extracted the features and do preprocessing. I explored 6 learning algorithms. I training all models with raw training features, normalized training features and standardization training features. I choose the best performance model based the training accuracy, computation cost and model generality, finally got test accuracy of 90%

2. Introduction

2.1. Problem Statement and Goals

I use hand postures data set. In the experiment, different people wearing a glove and do five different postures for several times. There are markers on the glove, this data set recorded the coordinates of markers when people do postures. My goal is to make a classifier that can predict the posture according to the markers' coordinates. I want to balance between good performance and computation cost and avoid overfitting.

3. Approach and Implementation

3.1. Preprocessing

The origin data set contains the label coordinates, the corresponding user id and class information. Because the markers are not distinguishable, the coordinates of markers can't provide valid information. So, I extracted features recommended from project instruction, including the number of missing markers in each posture, the mean, deviation, maximum, minimum value of x, y, and z coordinates in one posture.

For training data set, specifically, I use pandas, extracting three versions of features. The first is the normal statistic of coordinates mention above, written in 'features.csv'. The second version standardize the features by subtracting the corresponding mean and dividing the standard deviation. The features' value and deviation were scaled inside one. This version was written in 'standardize features'. The third version centered features by subtracting the minimum value and divide by (maximum-minimum). So, the range of every features is between 0 and 1, but the differences in the ranges of features don't change.

I use the similar technique for testing data set. The only difference is the mean and standard deviation used to standardize or normalize testing data set is calculated from training data. Because testing data set represents unknown data, we can't know the mean and deviation of unknown data.

Normalization and standardization make the features have same scale, so, every feature contribute to results equally, and the results become more stable. After center feature, a larger weight means the feature is more important. If don't do this, a large feature value with a small weight may also change the model loss dramatically. This makes the convergence become slow, and even makes model divergent. Also, it is likely that some features have abnormal large or small data, this will also influence the model. However, some data should keep raw, like the binary value data. The mean of binary data is meaningless.

When we know the distribution of the data set in each class is not Gaussian or we don't assume data distribution is Gaussian, then we should not standardize the data.

In conclusion, normalization and standardization will influence the parameters and final performance of model.

3.2. Feature engineering (if applicable)

The collected data has three coordinates, x, y and z. I calculate the range in three axis, (xrange, yrange and zrange). I extracted this feature because different postures may stretch the fingers in different direction, as a result the distance largest distance between makers may be helpful to classify postures. Although we already have the maximum and minimum of coordinates, the range is obscure.

3.3. Feature dimensionality adjustment

I run the same classifier using single feature to classify training data, choose features that provides highest accuracy. The classifier is rbf kernel SVM classifier with penalty parameter C=1, slack variable ζ =1(will explain later). The training data set I use raw features, because classification only use one feature, no need to keep scale same. The train acc in the table was calculate by average the cross validation accuracy, the F1 score was calculated by averaging the F1 score of each class, when average, the number of data points in each class is considered.

Table 1: individual feature and classification performance

Feature	Train acc	F1	Feature	Train acc	F1
X mean	0.22	0.46	X max	0.22	0.56
Y mean	0.30	0.60	Y max	0.43	0.69
Z mean	0.38	0.53	Z max	0.41	0.61
X deviation	0.30	0.51	X min	0.32	0.56
Y deviation	0.42	0.67	Y min	0.24	0.54

Z deviation	0.26	0.52	Z min	0.22	0.45
X range	0.30	0.56	Y range	0.38	0.65
Z range	0.42	0.62	Number of missing markers	0.54	0.50

According to the train accuracy and F1 score, I choose Y deviation, z range, Y max, Z max and number of missing markers as the reduced feature dimension.

3.4. Dataset Usage

3.4.1 feature preprocessing & engineering

The given data set is the coordinates of markers. For 'missing marker' feature, I found the number of nan value in each row and divide by 3, because each missing marker has 3 coordinates. For other features like 'xmean', 'xdev', 'xmin', 'xmax'… I found the mean of x-axis, deviation of x-axis, minimum, maximum value of x-axis in each row. For 'xrange', 'yrange', 'zrange' feature, I calculated the maximum value subtract minimum value. So, finally I got 16 features. I also use each feature's mean and standard deviation to normalize and standardize the training feature. Use the mean and deviation of training features to normalize and standardize the testing features. In code, I use pandas package.

3.4.2 dimensionality adjustment

I use individual feature to classify the training data set and choose features that give good accuracy and F1 score. I use rbf kernel SVM classifier, with C=1 and gamma=1. The reason why I choose rbf kernel SVM is that I don't have prior knowledge about the feature distribution. SVM optimizes the boundary and rbf kernel maps the space to infinite dimension feature space. Using SVM classifier can give relatively good result.

3.4.3 train and model selection

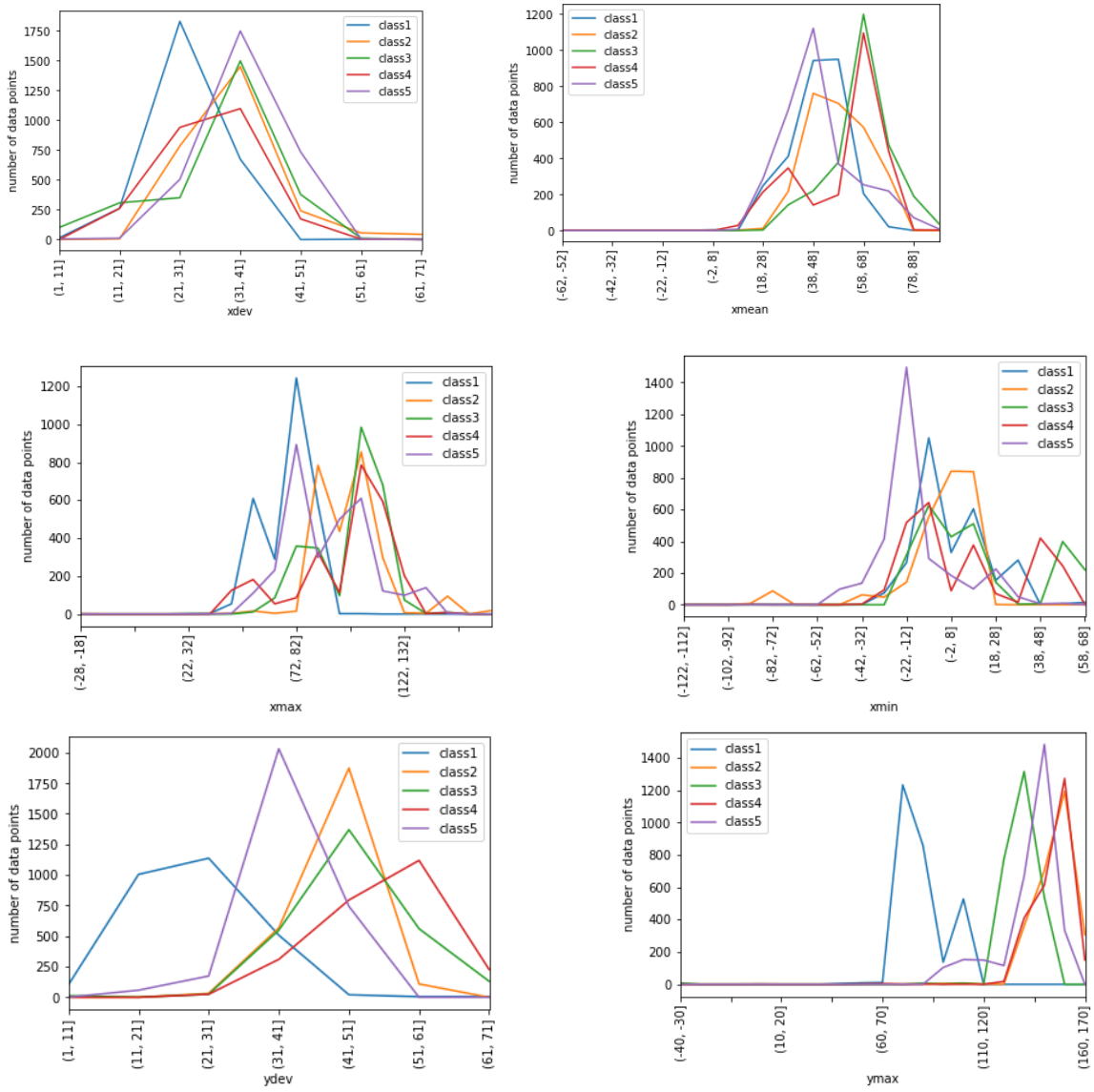
In training dataset, there are 9 users. Because different users may dramatically influence training dataset, I use cross validation based on different user training and evaluating the training accuracy. Specifically, each time use data points from one user for validation and data points from other users for training. For each classifier, run 9 times and each time choose different user for validation. In the code, cross validation training is accomplished by LeaveOneGroupOut() and cross_val_score() in sklearn. Pass the training features, labels and corresponding group criterion into LeaveOneGroupOut(), it will return a iterator. Pass the return value into cross validation process, in each run, the classifier chooses one group of data points as validation and other groups as training data. cross_val_score() can fit the training data and evaluate the validation data. cross_val_score() allows to choose scoring to change the evaluate rule for validation. If not state, I just use the default one, which is the accuracy rate that predictions *exactly* match the corresponding set of true labels. I also use the f1 score and calculate the confusion matrix for some classifier.

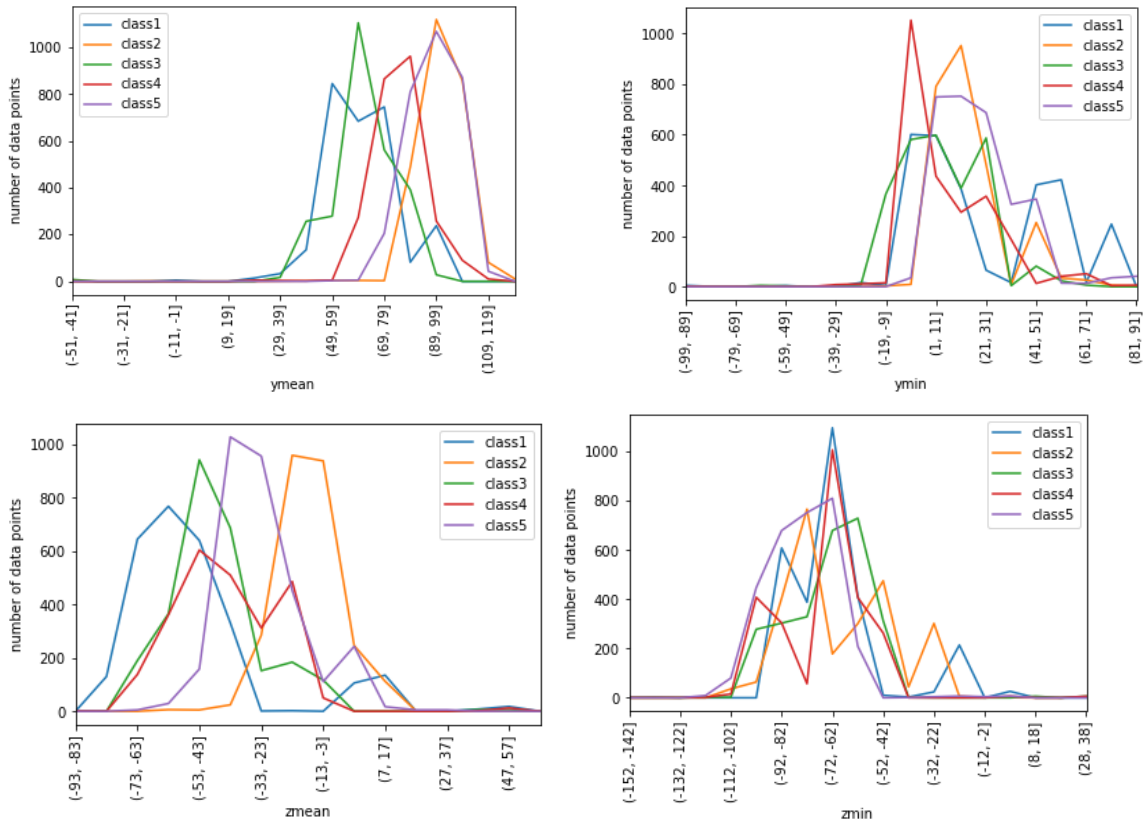
There are totally 13500 training data points. The data points in each class although is not balanced, doesn't have too much difference. The mean of data points number in each class is 2700, deviation is 183 and the gap of maximum and minimum value is 533. The number of data points provided by different user are the same. Except specifically state, the training data set use standardized training features.

In testing data set, there are totally 21099 data points from 3 users, 4865, 8739 and 7495 respectively. The number of data points in each class is also imbalanced. The testing data set will be used once after finally decided the model and parameters.

3.4.4 visualize the feature distribution

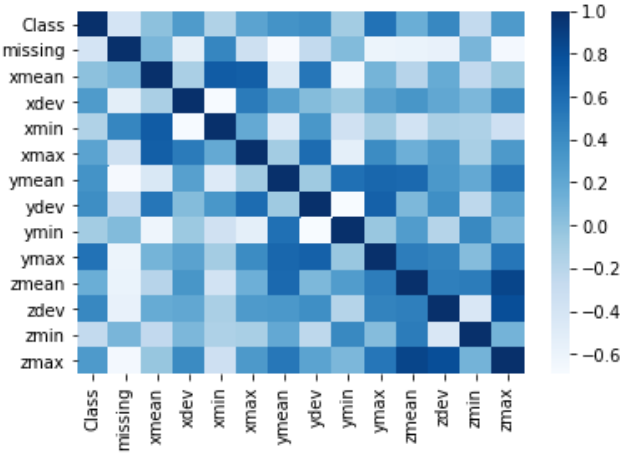
I draw the distribution of some raw features in training data set. Found as assumption, the distribution of features are nearly gaussian distribution.





3.4.5 PCA

I also calculate the covariance matrix of features and targets. According to the figure below, ymax, zdev, zmax etc. have higher correlation with the class label. The feature xmean, xmin, xmax have higher correlation with each other, the same situation happens to y axis and z axis. Because highly correlated features mean that one can actually represent the other, I would avoid choosing two features that are highly correlated, for example choose both xmean and xmax.



I use variance and covariance matrix check features, but I won't reduce features according to them. Because statistic property like variance can show how to map features in lower dimension, but is not closely related to classification. The covariance matrix is related to linear classifier, but in the project I use most nonlinear classifier.

3.4.6 Code

There are Five python file for my project folder. All the preprocess steps are in 'extract feature.py'. The code for visual data distribution is in 'visual.py'. The model selection code is in 'model selection.py'. The core part is in 'util.py' and 'classifier.py'. In 'util.py' you can choose use which data set, which features and use which kind of classifier. You can also set debug=True and run on the debug model, so the training data set would be reduced to 1000 data points. After set 'util.py' then you can run 'classifier.py' to get cross validation accuracy and confusion matrix of training data. When got final model, run 'test.py' to get test accuracy.

3.5. Training and Classification

3.5.1 random classifier:

I first find the accuracy if one doesn't train classifier, just pick the class label randomly. The method is generating random number between 1 and 5. If the class label is the same with the true class label, then this classification is assumed as a right one. I got the accuracy of random classifier is 19.59%. The accuracy matches the intuition, since there are totally 5 classes, each class has fairly same number of data points, the random accuracy should be approximately 1/5.

3.5.2 Naïve Bayes classifier:

Naïve Bayes classifiers assume the distribution of one feature to be independent with another feature. Also, the distribution of feature in one class is independent with data in another class. In the project, I use Gaussian Naïve Bayes classifier. This classifier assumes the data distribution to be Gaussian. I choose to use Gaussian Naïve Bayes, because there is no prior knowledge about features. Normally, when the data distribution is unknown, we use Gaussian.

The average accuracy of using 5 selected features is 82.01%, the standard deviation is 0.091.

Table 2: cross validation accuracy use 5 features

Validation data	User 1	User 2	User 3	User 4	User 5	User 6	User 7	User 8	User 9
accuracy	0.76	0.91	0.79	0.65	0.75	0.78	0.94	0.94	0.86

I tried to use all features to classify, got the average cross validation accuracy of 79.52% which is slightly lower than use selected features. The deviation of cross validation is 0.105. The F1 score is 79.51%.

Table 3: cross validation accuracy use 16 features

Validation data	User 1	User 2	User 3	User 4	User 5	User 6	User 7	User 8	User 9
accuracy	0.72	0.73	0.61	0.77	1	0.85	0.85	0.90	0.75

I also tried to use raw data and normalized data. Using raw data and normalized data got the same average accuracy of 82.01% and deviation 0.091. The training accuracy of Naïve Bayes Gaussian classifier are the same.

3.5.3 SVM, rbf kernel

$$\min_{w,b,\zeta} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \zeta_i$$
$$\text{subject to } y_i(w^T \phi(x_i) + b) \geq 1 - \zeta_i,$$
$$\zeta_i \geq 0, i = 1, \ldots, n$$

Formula 1: SVM with Dual Lagrange(from sklearn user guide)

For SVM classifier, I first choose rbf kernel function, penalty parameter C=1. I use rbf kernel because there is no knowledge if a linear, or quadratic classifier can correctly classify data points. The rbf kernel maps the original feature to infinite dimension feature space, so the training data can always be classified. The problem of rbf kernel SVM is that the model might overfit training data, so the parameter ζ also called slack variable needs to be choose carefully. ζ is a regularization parameter. A small ζ provides little tolerance to misclassified data points and larger ζ can prevent overfit. In my model I first choose default value, which is 1/ (number of features).

Table 4: cross validation accuracy use 5 features

Validation data	User 1	User 2	User 3	User 4	User 5	User 6	User 7	User 8	User 9
accuracy	0.76	0.85	0.88	0.75	1	0.91	0.98	0.87	1

Table 5: cross validation accuracy use 16 features

Validation data	User 1	User 2	User 3	User 4	User 5	User 6	User 7	User 8	User 9
accuracy	0.77	0.93	0.72	0.91	0.97	0.97	1	0.95	0.96

The average cross validation accuracy using 5 features are 88.83%, the standard deviation is 0.087. Using all 16 features are 90.73%, the deviation is 0.091, the F1 score is 90.71%

I tried to use normalized and raw feature for rbf SVM, when use normalized data the average accuracy is 87.82%, the standard deviation is 0.081, when use raw data the average accuracy dropped to 26.54%, the standard deviation is 0.070.

3.5.4 Perceptron classifier

I first tried to use default perceptron classifier in sklearn, which means no l1 or l2 regularization term, apply intercept to linear classifier, shuffle the data set at each iteration and stop fitting after 1000 iteration. The result shows below.

Table 6: cross validation accuracy use 5 features

Validation data	User 1	User 2	User 3	User 4	User 5	User 6	User 7	User 8	User 9
accuracy	0.64	0.72	0.73	0.80	0.70	0.70	0.76	0.81	0.77

Table 7: cross validation accuracy use 16 features

Validation data	User 1	User 2	User 3	User 4	User 5	User 6	User 7	User 8	User 9
accuracy	0.63	0.65	0.94	0.72	0.77	0.93	0.72	0.73	0.88

The average cross validation accuracy using 5 features are 75.28%, the standard deviation is 0.095. Using 16 features are 84.55%, the standard deviation is 0.143. The F1 score is 79.95%.

When use raw training data, the classification accuracy is 64.57% and the standard deviation is 0.127.

3.5.5 SVM with linear kernel

Perceptron algorithm is a linear classifier. To compare different linear classifier, I add the SVM with linear kernel in my project.

Table 8: cross validation accuracy use 5 features

Validation data	User 1	User 2	User 3	User 4	User 5	User 6	User 7	User 8	User 9
accuracy	0.76	0.90	0.79	0.78	0.92	0.88	0.97	0.81	0.77

Table 9: cross validation accuracy use 16 features

Validation data	User 1	User 2	User 3	User 4	User 5	User 6	User 7	User 8	User 9
accuracy	0.78	0.81	0.83	0.72	0.89	0.98	0.91	0.85	0.86

When use 5 features, the linear kernel SVM classifier gets average accuracy of 84.32%, the deviation is 0.072. When use 16 features, the linear kernel SVM classifier gets average accuracy of 84.91%, the deviation is 0.071, the F1 score is 84.89%

Linear kernel SVM and perceptron algorithm are all linear model, linear kernel gets better performance because SVM optimize the boundary by maximum the margin between data set and boundary.

When use raw data I get accuracy: 82.71, deviation: 0.103.

3.5.6 MSE regression and Ridge classifier

In sklearn MSE algorithm called Ordinary Least square, can only be used as a regression model. But there is another classification model in sklearn called Ridge classifier, which is a variant of MSE. This classifier adds a penalty term to prevent weight being too large, gets coefficients by minimizing below formula.

$$\min_{w} ||Xw - y||_2^2 + \alpha||w||_2^2$$

Formula 2: Loss function of Ridge classifier

I use Ridge classifier and default penalty parameter alpha=1, the average cross validation accuracy using 5 features is 73.86%, the deviation of cross validation accuracy is 0.083. the average cross validation accuracy using 16 features is 77.88%, the deviation of cross validation accuracy is 0.117.

Table 10: cross validation accuracy use 5 features

Validation data	User 1	User 2	User 3	User 4	User 5	User 6	User 7	User 8	User 9
accuracy	0.61	0.78	0.92	0.71	0.74	0.68	0.77	0.66	0.76

3.5.6 Nearest Centroid

Nearest Centroid classifier is based on the distance between the class mean and the data points. So, it should also be sensitive to the scale of features. So, I will feed raw, centered only and normalized data into classifier.

the average cross validation accuracy using 5 features is 84.77%, the deviation of cross validation accuracy is 0.085.

Table 11: cross validation accuracy use 5 features

Validation data	User 1	User 2	User 3	User 4	User 5	User 6	User 7	User 8	User 9
accuracy	0.76	0.93	0.79	0.98	0.96	0.76	0.76	0.82	0.85

For nearest centroid algorithm, I also tried use raw and normalized training feature. Using raw feature only got average accuracy of 66.35%, deviation of 0.162.

3.5.7 optimize model

I measure the performance of a model by using the average cross validation accuracy subtract the deviation.

Table 12: model and performance (use five features)

model	performance	model	performance
Naïve Bayes	0.7291	Perceptron	0.6658
Rbf SVM	0.8013	Linear SVM	0.7712
MSE	0.6556	Nearest Centroid	0.7627

I choose two models and further optimize them. Because it is known nearest centroid is a simple model, and is rely only on the data, Naïve Bayes makes assumption on feature distribution and doesn't need parameters, I won't choose these two models. Linear SVM and Rbf SVM give large accuracy the accuracy are close, so, I will choose to optimize Rbf kernel SVM, Linear kernel SVM. Use standardized training data.

I use GridSearchCV() In sklearn optimize the parameters. The function allows to set which kind of estimator you want to optimize, which parameters in the estimator and the range of parameters you want to try. Also, you can choose to use cross validation and you can specify the accuracy matrix you want to use. I choose to optimize Rbf kernel, with 10 gamma values in the log space between $[10^{-1}, 10]$, 10 C values between $[10^{-1}, 10]$, (the parameters were explained before) use cross validation that each time split one user, use accuracy_score matrix and F1 score (explained before) to evaluate the training performance.

Table 13: parameter C & performance of linear kernel(use 5 features)

C	0.1	0.167	0.278	0.464	0.774	1.291	2.154	3.593	5.995	10
Accuracy(%)	84.44	84.08	84.13	84.15	84.24	84.28	84.26	84.25	84.30	84.24
deviation	0.062	0.066	0.069	0.072	0.073	0.073	0.073	0.075	0.075	0.079

From the table above, when change the parameter C, the performance is almost the same. With C increase, the accuracy slightly decrease and the deviation slightly increase. We know large C gives less tolerance to the misclassified data points. But in the table above, the accuracy didn't change a lot. I think this is because the data set is not linear separable. Despite how to penalty the miss-classified data, the classifier won't give a better boundary.

The below table use F1 score to show performance.

Table 13: parameter C & performance of linear kernel (5 features)

C	0.1	0.167	0.278	0.464	0.774	1.291	2.154	3.593	5.995	10
F1 score (%)	84.44	84.09	84.14	84.16	84.24	84.28	84.26	84.25	84.30	84.24

The F1 score is calculated according the precision and recall rate, also considered the imbalance number of each class. In code, use scoring matrix 'f1_micro'.

The F1 score and accuracy of C=0.1 is the largest. This is because small penalty makes the model more general.

Table 14: parameter C & performance of linear kernel (16 features)

C	0.1	0.167	0.278	0.464	0.774	1.291	2.154	3.593	5.995	10
F1 score (%)	84.31	84.11	82.93	83.27	83.87	85.28	84.93	85.46	85.29	85.67

When use 16 features, the result is similar to using 5 features. The accuracy increases as penalty C increase, but the increment is not very large.

So, to guarantee the generality of model, I will still choose the default parameter C=1 for testing. When use linear kernel SVM classifier, the accuracy is about 84% for training 16 features and 5 features.

Below heat map shows the result of optimizing parameters of rbf kernel SVM. The penalty parameter C and regularization parameter gamma are two keys values. C tends to increase the training classification accuracy and gamma tends to make the classifier more general and sacrifice the accuracy. So small gamma and large C gives largest training accuracy rate.

Figure: use 5 features

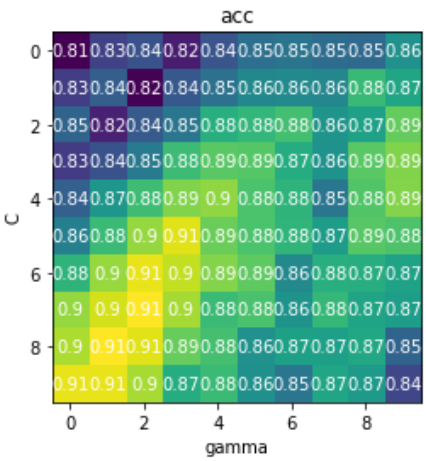
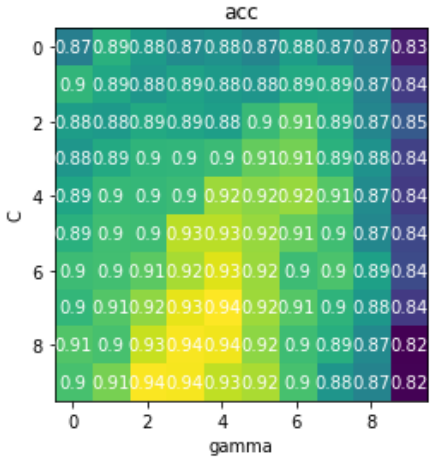


Figure: use 16 features



The common C value is 1, in the heat map above, the common C gives accuracy between 0.86 to 0.91. Because large C will make model overfit, to avoid overfit, I won't choose large C model and only consider to get good training accuracy. In the above figure, I use log space, (C, gamma = [0.1 , 0.16681005, 0.27825594, 0.46415888, 0.77426368, 1.29154967, 2.15443469, 3.59381366, 5.9948425 , 10.]).

The 6th C is 1.29. So, I choose the raw of 5th raw. In the 5th row, the 7th column gets relatively large regularization parameter gamma and 0.89 and 0.9 accuracy. So, I choose C=0.774 and gamma=2.15 to make the model more general.

Finally, when choose between Rbf SVM and linear SVM, because Rbf SVM is a lot more complex, to reduce the computation cost, I choose linear kernel SVM.

I test the model use 16 features of test dataset and linear kernel SVM with C=1. Finally get test accuracy of 90.23 and F1 score is 0.90.

I also test the rbf kernel SVM, surprisely, only got test accuracy of 0.18 and F1 score 0.19

4. Analysis: Comparison of Results, Interpretation

Observation 1: Compare using normalization data set and standardization data set, when use Naïve Bayes classifier and linear kernel classifier, the accuracy doesn't reduce a lot, when use other classifier, the accuracy reduces by more than 50%.

Table 14

Model	Train acc drop (%)	Model	Train acc drop (%)
Naïve Bayes	0	Linear SVM	2
Rbf SVM	62	Perceptron	13
Nearest centroid	22		

Interpretation: Naïve Bayes classifier make assumption of the feature distribution. The classification boundary is only related to the feature distribution, not the scale of features. In other words, the mean and variance of feature is used when substitute data set into assumed feature distribution. So, whether preprocess the training data doesn't influence the performance of Naïve Bayes classifier.

SVM utilizes some data points that close to boundary as support vectors. The scale of features directly influences the classifier. If don't change the features to the same scale, important features that has smaller value would be ignored. So SVM should be influenced by feature scale. However, linear SVM accuracy didn't drop a lot when use raw features. This is because among the five selected features, three of them actually has similar scare. What's more,

Formula 3: rbf kernel

$$K(x,x')=\exp(-\gamma||x-x'||^2)$$

Formula 4: linear kernel

$$K(x, x') = x^T x'$$

the kernel function is exponential of square of support vectors for rbf kernel, linear kernel is inner product of support vector. This cause the scale influence for rbf kernel is a lot more than linear kernel SVM. So, rbf kernel SVM got much bad performance when use raw data but linear kernel SVM accuracy reduced but didn't change a lot.

Perceptron learning influenced by feature scale because the loss function of perceptron also uses the distance between boundary and misclassified data points, so scale is important.

nearest centroid algorithm calculates the distance between feature center. So feature scale will influence the training accuracy.

Observation 2: The running time

The running time of SVM classifier is longest, the running time of Naïve Bayes is shortest. This is because Naïve Bayes makes assumption of the feature distribution. When classification, the classifier only needs to substitute the data points into density function. There is no need to change the parameters. So, training is very fast. Similarly, nearest centroid algorithm also trains very fast.

However, SVM and perceptron need to finetune weight, so training SVM and perceptron classifier is pretty slow.

Observation 3: except perceptron learning, using other classifier, the accuracy won't change if I don't change the parameters.

When using perceptron algorithm, the accuracy changes at each run. This is because perceptron is a linear classifier, for non-linear separable data. By default, it returns the weight at the 1000^{th} iteration, which is unstable.

Table 15: Five runs of perceptron (with same parameters)

Time	Mean accuracy (%)	deviation
1	75.28	0.095
2	74.81	0.179
3	74.52	0.134
4	72.48	0.130
5	73.08	0.078

Observation 4: F1 score and average cross validation accuracy

The F1 score and the average cross validation accuracy are almost the same in the above classifiers. This shows that there is no preference for the classifier to get higher accuracy is some classes.

Observation 5: Training accuracy when use 16 and 5 features

Compared to using 5 features, except Gaussian Naïve Bayes classifier, using 16 features always get better performance in training accuracy or accuracy deviation. Naïve Bayes doesn't get better performance when using more features and the training accuracy even become lower. This is because the assumption of all feature distribution is Gaussian is a very simple assumption. Adding more features is equivalent to use more assumptions. So, for Naïve Bayes classifier, using too much features will lower the performance.

For other classifier, the weight or parameters of classifier changes when fit training data. So, as long as the data is right, and can the feature pattern is the same with both training and testing data set, the more features will give better performance.

5. Contributions of each team member

No teammates.

6. Summary and conclusions

To summarize, I learn how to training different learning algorithm and the running time of algorithm. I observed that the test result of rbf kernel SVM is very unexpected. My future work will continue why this happen. Also, I got the idea that even training with cross validation, the model can still see the data in validation set, so, the overfitting problem is still needed to treat carefully.

References

Use sklearn package, pandas, matplotlib and some basic python package like numpy.