How To Choose Classifier

Data Set: Indoor Wireless Localization & Hand Postures

Po Yu, Yang, poyuyang@usc.edu

05/08/2020

1. Abstract

I used both the Wireless Localization and Hand Postures datasets. The classifier I used in this project are Naïve Bayes, SVM, Perceptron, OneVsRest, KNN, DecisionTree, RandomForest. For Wireless dataset, I got 97.75% of accuracy while using SVM. For Hand dataset, I got 96.2 % of accuracy while using SVM classifier.

2. Introduction

2.1. Problem Statement and Goals

I used both the Wireless Localization and Hand Postures datasets. For the first dataset, all the value seems to be in the same range, and the goal I want to achieve is to predict the location of the given data. For the second one, most of the features are irregular, and all of them are from different user. The goal is to predict the postures by the given marker positions provided by different users.

3. Approach and Implementation

3.1-3.5 for Wireless Localization

3.6-3.10 for Hand Postures

3.1. Preprocessing

Before I do any preprocessing, I trained the classifier with the raw data, and the accuracy can already come out with 90% or more. Then I decided to try **Normalization** and **Standardization** to see if I can get any better results.

3.2. Feature engineering (if applicable)

Not applicable

3.3. Feature dimensionality adjustment

There are 7 features in this dataset, so I used **PCA(n_components=2)** to reduce the dimensions to 2 features, the way I checked if it is feasible is to check the variance ratio in all the feature. It shows up that 2 of the feature have [0.65935397 0.19528077] up to 85% of energy.

3.4. Dataset Usage

The total of the dataset for train and validation are 1600, and I split 1/5 of them to validation data, so there are 1280 training data and 320 validation data. Only 400 of data are available for testing.

I used **PCA** to reduce the dimension of the dataset to 2 first, and used **Standardization** to change the value of all the features. Then, I applied both of the preprocessing to the testing data.

For cross validation, I used **StratifieldKFold (K=5)** for every classifiers.

I split the data 5 times to see what is the average of the accuracy of validation data.

3.5. Training and Classification

- **3.5.1 Naïve Bayes --- GuassianNB()**

  Using Gaussian distribution to estimate the mean and standard deviation from the training data.

- **3.5.2 SVM --- SVC(C=100)**

  I have tried C = 10,100,1000, and the results are quite similar, so this dataset have no overfitting issue.

- **3.5.3 Perceptron --- Perceptron()**

  We can see from the classifier if the dataset is linearly separable or not.

- **3.5.4 OneVsRest --- OneVsRestClassifier(SVC())**

  Fitting one classifier per class, for each classifier, the class is fitted against all the other classes. I choose the SVC estimator to get a better result.

- **3.5.5 KNN --- KNeighborsClassifier(n_neighbors = 5)**

  It can predict the label by choosing the nearest data point, so it can also prove that the data in these data set are clustered or not. I chose n = 5 because it already performed very well.

- 3.**5.6 Decision Tree --- DecisionTreeClassifier()**

  The algorithm of this classifier is pretty simple, it checks through all the feature and make it become a subtree, so that when new data is coming, it can predict the results by going through the whole tree.

- **3.5.7 Random Forest --- RandomForestClassifier(max_depth =50)**

It is similar to decision tree, but more powerful if the dataset is more complicated. Max_depth = 50 can improve the accuracy the best from my observation.

3.6. Preprocessing

First, I tried **normalization** and **standardization** to preprocess the original data, but I can get only 60 % of accuracy, so defining new features is necessary for this dataset.

3.7. Feature engineering (if applicable)

If using raw data to train, any of the classifier I choose cannot get good accuracy, so I decided to create new feature such as **mean** of the x, y, z values and **standard deviation** of the x, y, z values.

After that, I added 7 more features, which are **maximum** of the x, y, z values , **minimum** of x, y, z values and the **total count** for x, y, z values.

3.8. Feature dimensionality adjustment

I tried to use **PCA** to the original dataset which has 36 features, but it's not useful because of the features are not labeled. Then, I decided to **replace** all 36 features by 13 new features mentioned above.

3.9. Dataset Usage

The total of the dataset for train and validation are 13500, and I split 1/9 of them to validation data, so there are 12000 training data and 1500 validation data. The total amount of testing data is 21099.

I used **standardization** to the new features, because of the range of the features are quite large. Then, I applied the preprocessing to the testing data.

For cross validation, I leave one user's data out for validation every time for every classifier.

I split the data 9 times to see what is the average of the accuracy of validation data. Moreover, I chose the best accuracy from the 9 set, and used it to train all of the classifier to get the best result.

The test data are used after the classifier trained by the best parameter.

3.10. Training and Classification

- **3.10.1 Naïve Bayes --- GuassianNB()**

  Using Gaussian distribution to estimate the mean and standard deviation from the training data.

- **3.10.2 SVM --- SVC(C=10,gamma =0.02)**

  I used grid search to find the best parameter for SVM classifier.

- **3.10.3 Perceptron --- Perceptron()**

  We can see from the classifier if the dataset is linearly separable or not.

- **3.10.4 OneVsRest --- OneVsRestClassifier(SVC())**

  Fitting one classifier per class, for each classifier, the class is fitted against all the other classes. I choose the SVC estimator to get a better result.

- **3.10.5 KNN --- KNeighborsClassifier(n_neighbors= 15)**

  It can predict the label by choosing the nearest data point, so it can also prove that the data in these data set are clustered or not. I chose n = 15 because it already performed very well.

- **3.10.6 Decision Tree --- DecisionTreeClassifier()**

  The algorithm of this classifier is pretty simple, it checks through all the feature and make it become a subtree, so that when new data is coming, it can predict the results by going through the whole tree.

- **3.10.7 Random Forest --- RandomForestClassifier(max_depth =50)**

  It is similar to decision tree, but more powerful if the dataset is more complicated. Max_depth = 50 can improve the accuracy the best from my observation.

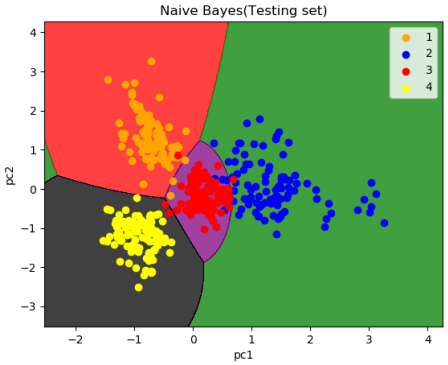
4. Analysis: Comparison of Results, Interpretation

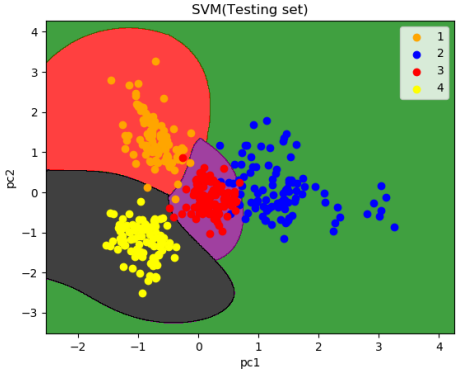
- 4.1. Indoor Wireless Localization

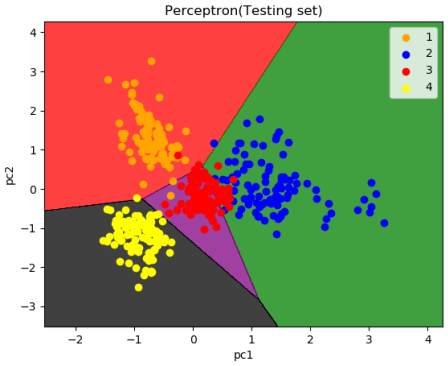
	cross_val_acc mean	cross_val_acc std	Test Accuracy	Confusion Matrix
Naïve Bayes	0.981	0.005	97.25 %	[[97 0 3 0] [0 96 4 0] [1 1 96 2] [0 0 0 100]]
SVM	0.98	0.011	97.75 %	[[99 0 1 0] [0 95 5 0] [1 1 97 1] [0 0 0 100]]
Perceptron	0.959	0.029	95.75 %	[[98 0 2 0] [0 88 12 0] [1 0 97 2] [0 0 0 100]]
OneVsRest	0.979	0.005	97.0 %	[[99 0 1 0] [0 93 7 0] [1 1 96 2] [0 0 0 100]]
KNN	0.979	0.013	97.25 %	[[98 0 2 0] [0 95 5 0] [1 1 96 2] [0 0 0 100]]
Decision Tree	0.966	0.009	97.5 %	[[96 0 3 1] [0 96 4 0] [1 1 98 0] [0 0 0 100]]
Random Forest	0.975	0.003	97.0 %	[[97 0 2 1] [0 93 7 0] [1 1 98 0] [0 0 0 100]]

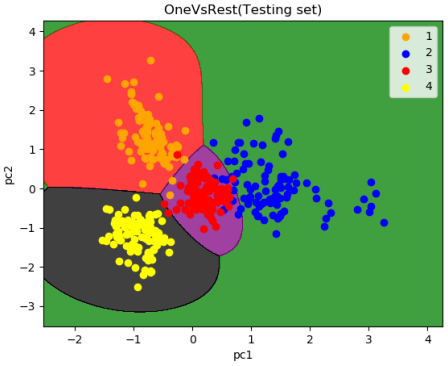
All the classifier can predict class 4 in 100%, so I think the data in class 4 must be very clustered and is separated from other classes.

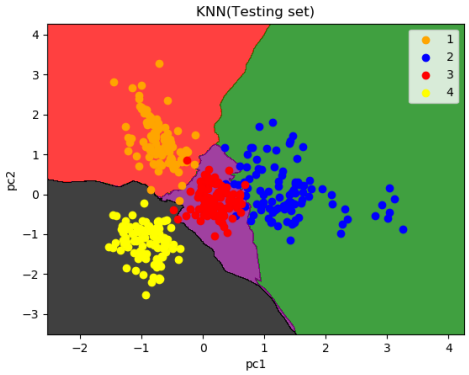
It can be proved by the visualization of testing results below.

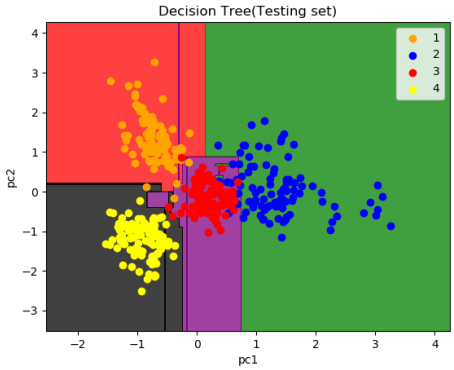

Naive Bayes(Testing set)
pc1
pc2
1
2
3
4

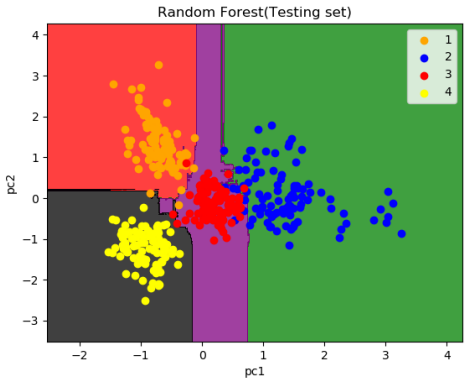

SVM(Testing set)
pc1
pc2
1
2
3
4


Perceptron(Testing set)
pc1
pc2
1
2
3
4


OneVsRest(Testing set)
pc1
pc2
1
2
3
4


KNN(Testing set)
pc1
pc2
1
2
3
4


Decision Tree(Testing set)
pc1
pc2
1
2
3
4


Random Forest(Testing set)
pc1
pc2
1
2
3
4

- 4.2. Hand Postures

Using only 6 features (mean of x,y,z and std of x,y,z)

	cross_val_acc mean	cross_val_acc std	Test Accuracy	Confusion Matrix
Naïve Bayes	0.752	0.124	55.73 %	[[3772 48 256 320 70] [520 2327 25 0 1530] [385 203 1426 642 2123] [0 2281 0 1022 611] [280 28 19 0 3211]]
SVM	0.845	0.071	64.76 %	[[4346 48 3 0 69] [31 1072 1155 10 2134] [185 0 3273 35 1286] [0 1209 7 1769 929] [0 84 3 247 3204]]
Perceptron	0.74	0.113	61.51 %	[[4225 48 0 0 193] [430 1882 2015 0 75] [1639 2 2357 24 757] [0 1827 972 1008 107] [0 0 29 2 3507]]
OneVsRest	0.769	0.072	65.8 %	[[4273 48 67 2 76] [24 1891 733 0 1754] [6 3 3181 6 1583] [0 4 1644 1336 930] [0 30 0 305 3203]]
KNN	0.685	0.117	67.99 %	[[3031 48 94 1103 190] [157 3893 0 1 351] [1 1238 2691 102 747] [0 2510 45 1267 92] [0 32 12 30 3464]]
Decision Tree	0.637	0.15	56.19 %	[[3424 635 39 257 111] [0 3120 381 41 860] [82 930 1641 831 1295] [0 1042 0 648 2224] [0 424 33 58 3023]]
Random Forest	0.708	0.107	53.21 %	[[3394 0 958 2 112] [0 1729 42 0 2631] [112 74 1276 96 3221] [0 477 0 1372 2065] [0 53 14 15 3456]]

Using 13 features(mean of x,y,z & std of x,y,z & maximum of x,y,z & minimum of x,y,z & total count of x,y,z)

	cross_val_acc mean	cross_val_acc std	Test Accuracy	Confusion Matrix
Naïve Bayes	0.781	0.139	80.2 %	[[4330 48 72 15 1] [116 4174 0 0 112] [455 61 2169 336 1758] [0 913 76 2880 45] [0 115 19 35 3369]]
SVM	0.927	0.05	96.25 %	[[4346 48 56 0 16] [39 4175 75 0 113] [153 0 4511 112 3] [0 5 84 3825 0] [0 11 0 77 3450]]
Perceptron	0.799	0.089	83.78 %	[[4238 48 148 0 32] [83 4286 32 1 0] [159 61 4435 121 3] [1 1878 251 1784 0] [310 125 99 71 2933]]
OneVsRest	0.91	0.079	88.68 %	[[4342 48 18 2 56] [28 4218 13 0 143] [191 0 3979 67 542] [0 40 2 2704 1168] [0 49 0 21 3468]]
KNN	0.783	0.123	80.54 %	[[4261 48 65 5 87] [93 3918 0 0 391] [7 6 3588 1031 147] [0 80 1778 1811 245] [0 87 16 20 3415]]
Decision Tree	0.758	0.138	71.24 %	[[4329 12 48 20 57] [7 2032 0 0 2363] [0 11 3839 380 549] [0 4 1352 1367 1191] [0 42 32 1 3463]]
Random Forest	0.84	0.094	73.8 %	[[4346 10 5 0 105] [0 1824 7 0 2571] [0 48 4169 62 500] [0 50 241 1805 1818] [0 31 72 7 3428]]

We can see from above, all of the test accuracy in 13 features are better than only 6 features. For this dataset, Decision Tree and Random Forest are not performing well compared to all the others classifier. I think it might because of the overfitting problem. The best I can get is the SVM classifier with C = 10 and gamma = 0.02.

5. Contributions of each team member

All the works are done by Po Yu, Yang.

6. Summary and conclusions

For Indoor Wireless Localization dataset, the dataset is quite easy and simple, most of the test data can be predicted correctly, especially for class 4, we got 100 % of accuracy. From the visualization of results, we can also see how every classifier performs in different way.

For Hand Postures dataset, I think this might be the one we will face in the future. Some dataset that we cannot easily classify it because of lack of information. Especially when the features dimension is huge, it will take more preprocess to achieve a good result.

After doing this project, I know that features selection is an important part for machine learning. If we can reduce the feature dimension to a smaller size such as using PCA, we can get rid of many unimportant information in the raw data. We can not only shorten the time for training but also get a better accuracy for testing data. The SVM classifier is a powerful classifier, because it is easy to tune and it also performed well in these projects. Every classifier has different characteristic. When facing different data, we have to analyze the data first, then tune the classifier to get a better result.

References

[1] "Decision Trees and Random Forests". Available: https://towardsdatascience.com/decision-trees-and-random-forests-df0c3123f991