Report of Project 3

6364 Machine Learning

Linsheng Liu

1. Introduction

In this project, 2 parts are involved focusing on relation and comparison of k-means clustering and PCA on Human Activity Recognition Dataset as well as comparison of Random forests and KNN on MNIST dataset. In the second part, a KNN based on the results of PCA is also available. The code for both parts are given as Jupyter notebook and some brief changes are made for better visualization purposes.
For part 1, the code and data link: *https://www.kaggle.com/ruslankl/k-means-clustering-pca*
For part 2, the code and data link: *https://www.kaggle.com/sflender/comparing-random-forest-pca-and-knn*

2. K-Means Clustering vs PCA

2.1 Dataset Overview and Preprocessing

The Simplified Human Activity Recognition w/ Smartphone dataset contains 3609 instances among 563 variables containing 1 id, 1 label (activity) and other 561 features.

	rn	activity	tBodyAcc.mean.X	tBodyAcc.mean.Y	tBodyAcc.mean.Z	tBodyAcc.std.X	tBodyAcc.std.Y	tBodyAcc.std.Z	tBodyAcc.mad.X	tBo
2883	8250	WALKING_UPSTAIRS	0.322	-0.04320	-0.1580	-0.2150	-0.360	0.00255	-0.241	
2852	8143	SITTING	0.221	-0.08730	0.0441	-0.8100	-0.306	-0.03280	-0.813	
3492	9941	STANDING	0.279	-0.01920	-0.1050	-0.9950	-0.929	-0.93900	-0.996	
908	2592	SITTING	0.279	-0.01680	-0.1070	-0.9980	-0.987	-0.98200	-0.998	
1382	3928	WALKING	0.335	0.00837	-0.1380	-0.0779	0.270	-0.03720	-0.163	

5 rows × 563 columns

Figure 1. Sample of Simplified Human Activity Recognition w/Smartphone Dataset

Statistical analysis shows that all the values of each features are stored as real number and about to follow a normal distribution. Due to too much dimension of the dataset, the histogram is not presented here. To perform k means as well as PCA, normalizing is required as following:

```
#normalize the dataset
scaler = StandardScaler()
Data = scaler.fit_transform(Data)
```

Figure 2. Codes for normalizing the data

2.2 K-Means

As an unsupervised classifier, we could use the Inflection point method to find the best k value as following:

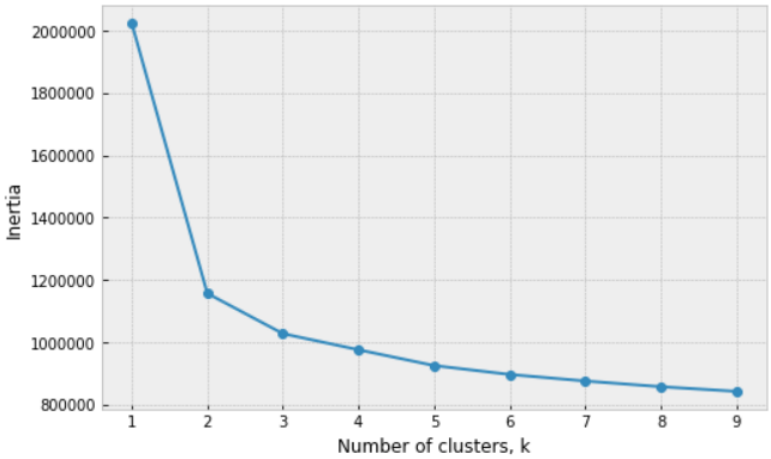

Figure 3. Finding the best k

From figure 3, we can see that k = 2 is the best choice for clustering. However, since we have six activity labels of the original data, it is natural to try 6 clusters as well. The following is the result of K-Mean s for k from 2 to 7.

```
k_means(n_clust=2, data_frame=Data, true_labels=Labels)
```

orig_label clust_label	LAYING	SITTING	STANDING	WALKING	WALKING_DOWNSTAIRS	WALKING_UPSTAIRS
0	680	622	668	0	0	6
1	1	1	0	603	493	535

```
k_means(n_clust=3, data_frame=Data, true_labels=Labels)
```

orig_label clust_label	LAYING	SITTING	STANDING	WALKING	WALKING_DOWNSTAIRS	WALKING_UPSTAIRS
0	6	3	1	442	198	503
1	675	620	667	0	0	0
2	0	0	0	161	295	38

```
k_means(n_clust=4, data_frame=Data, true_labels=Labels)
```

orig_label clust_label	LAYING	SITTING	STANDING	WALKING	WALKING_DOWNSTAIRS	WALKING_UPSTAIRS
0	194	225	290	0	0	2
1	0	0	0	159	293	38
2	486	398	378	0	0	0
3	1	0	0	444	200	501

```
k_means(n_clust=5, data_frame=Data, true_labels=Labels)
```

orig_label clust_label	LAYING	SITTING	STANDING	WALKING	WALKING_DOWNSTAIRS	WALKING_UPSTAIRS
0	0	0	0	245	310	97
1	194	225	292	0	0	1
2	1	1	0	333	109	440
3	486	397	376	0	0	0
4	0	0	0	25	74	3

```
k_means(n_clust=6, data_frame=Data, true_labels=Labels)
```

orig_label clust_label	LAYING	SITTING	STANDING	WALKING	WALKING_DOWNSTAIRS	WALKING_UPSTAIRS
0	554	21	0	0	0	0
1	0	0	0	248	311	97
2	1	0	0	329	107	438
3	20	445	479	0	0	0
4	0	0	0	26	75	4
5	106	157	189	0	0	2

```
k_means(n_clust=7, data_frame=Data, true_labels=Labels)
```

orig_label clust_label	LAYING	SITTING	STANDING	WALKING	WALKING_DOWNSTAIRS	WALKING_UPSTAIRS
0	106	156	190	0	0	0
1	0	0	0	314	189	125
2	1	1	0	179	53	392
3	0	0	0	97	205	23
4	554	23	0	0	0	0
5	0	0	0	13	46	1
6	20	443	478	0	0	0

Figure 4. Clustering result for k from 2 to 7

We can see directly that when k is 2, it seems already very good performance. When k gets larger, it is just like a "force split" to some clusters. There is a strong dependent among "laying", "sitting", and "standing" three activity labels; as well as a similar dependent among "walking", "walking_downstairs", and "walking_upstairs". (It is also a hint that PCA would also be a good choice here.) The sklearn package also calculates several performance index, such as silhouette coefficient, which evaluates how well define of clusters on average. The following figure 5 shows the silhouette in terms of k, we can then conclude that k = 2 is the best choice here.

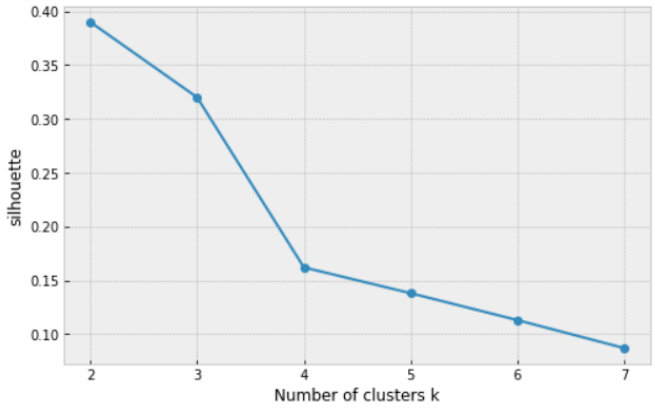

Figure 5. Silhouette coefficient vs number of clusters k

orig_label clust_label	0	1
0	1970	6
1	2	1631

The runtime of clustering is
0:00:01.702866

Figure 6. Result of clustering and runtime

In summary, we will accept the following cluster result above shown in figure 6 as well as the runtime, which is 1.703 seconds. The overall accuracy is 3601/3609=0.9978.

2.3 PCA

The Human Activity Recognition Dataset has 561 features and it is natural to apply PCA to try to reduce the dimension for better result as well as easier computation. With the convenient tool in sklearn, we can easily perform a PCA. The following figure 7 shows the variance explained by each principle component of the first 15 most important PCs (Left) as well as the cumulative variance explained by the first 100 most important PCs (Right).

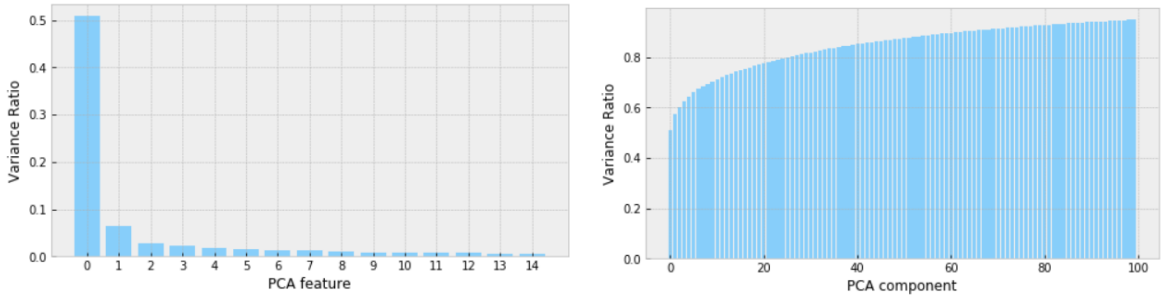

Figure 7. Variance explained by Principle Components

The first component is very important with over 50% variance explained and it is also significantly larger than other PCs. However, in order to explain more than 90% of the variance (which is common in statistical studies), it seems that at least 60 PCs are required. The following figure 8 shows the result of the PCA with different number of principle components settings (number of PCs: 1, 2, 4, 8, 16, 32):

Shape of the new Data df: (3609, 1)

orig_label	0	1
clust_label		
0	1971	8
1	1	1629

Shape of the new Data df: (3609, 2)

orig_label	0	1
clust_label		
0	1969	6
1	3	1631

Shape of the new Data df: (3609, 4)

orig_label	0	1
clust_label		
0	1970	6
1	2	1631

Shape of the new Data df: (3609, 8)

orig_label	0	1
clust_label		
0	1970	6
1	2	1631

Shape of the new Data df: (3609, 16)

orig_label	0	1
clust_label		
0	1970	6
1	2	1631

Shape of the new Data df: (3609, 32)

orig_label	0	1
clust_label		
0	1970	6
1	2	1631

Figure 8. PCA results for different number of PCs

It is interesting that only one principle component with around 50% variance explained will perform so well. There might be some intrinsic dependency among all the features. The silhouette coefficient also supports that 1-PC is the best choice:

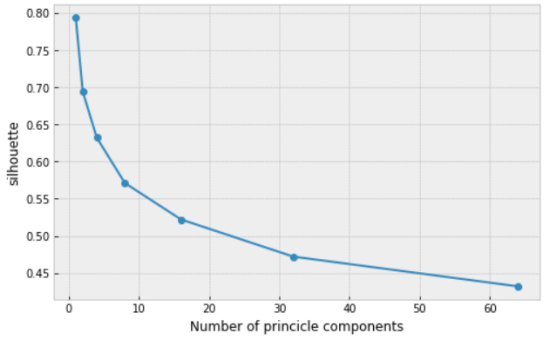

Figure 9. Silhouette coefficient vs number of PC

orig_label	0	1
clust_label		
0	1971	8
1	1	1629

The runtime of PCA is
0:00:00.266983

Figure 10. Result of PCA and runtime

The figure 10 shows the accepted setting's PCA result as well as the runtime. The overall accuracy is 3600/3609=0.9975.

2.4 Comparison

From the above results, in summary, the accepted classifiers are k means with k = 2 and PCA with only 1 principle components. The following table shows the results of them:

Method:	Accuracy	Runtime
2-means	0.9978	1.703 seconds
PCA of 1 component	0.9975	0.267 seconds

We can conclude that both methods perform well for almost identical accuracy and the PCA is around 10 times faster due to dimension reduction.

3. Random Forest vs KNN vs PCA then KNN

3.1 Dataset Overview and Preprocessing

The MNIST dataset contains 42000 labeled 2-D handwriting digit instances, each one has 28x28 pixels varies from 0 to 255. The frequency of each digit class is shown in figure11. Some examples are shown in figure 12.

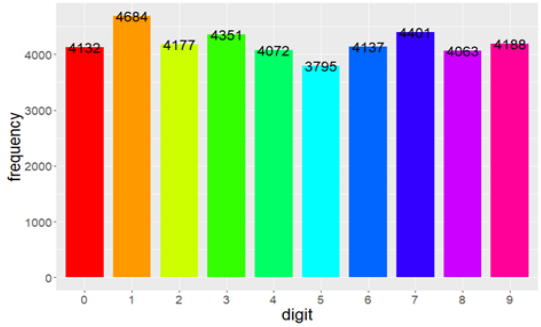

Figure 11. The Overall distribution of MNIST dataset

Figure 12. Examples of some digits

Rather than the previous project 1, here I took 2000 data for train to compare the performance of different classifiers. The split ratio is 0.8 for training and testing data. The data is stored as pixels and is not necessary to do any normalizations. Following 3.2~3.3 applies random forest, KNN, KNN after PCA on a subset of MNIST of size 2000 to finetune the hyper parameter and then applying on the whole dataset.

3.2 Random Forest

With the help of sklearn package, it is relatively easy to implement the random forest. One key hyper parameter of random forest is the number of estimators. Several models are trained for number of estimators equals to 1, 5, 10, 50, 100, 200, 500. It looks like that 100 estimators will offer best performance. The best accuracy is close to 90%.

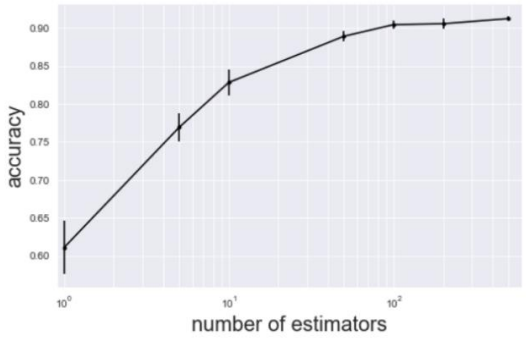

Figure 13. Accuracy for different number of estimators

Since the MNIST dataset is already clean and all the digits are concentrating in the center of each image, it seems the center pixels are more important than surroundings pixels. However, since there are 28 x 28 = 784 features for each image, it is not a good idea to measure the importance of variables. The figure 14 shows the top 10 important features.

```
Feature ranking:
1. feature 378 (0.009662)
2. feature 409 (0.008761)
3. feature 433 (0.007280)
4. feature 406 (0.007171)
5. feature 155 (0.006951)
6. feature 461 (0.006842)
7. feature 405 (0.006724)
8. feature 350 (0.006527)
9. feature 437 (0.006369)
10. feature 597 (0.006240)
```

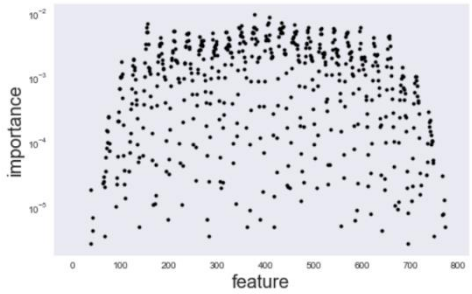

Figure 14. top features of MNIST dataset

Based on the result above, the accepted random forest model is with 100 estimators and the result is shown in figure 15. The overall accuracy is 0.91 and the runtime is 0.4748 seconds.

```
The runtime of random forest is
0:00:00.474770
Accuracy is
0.91
```

Figure 15. Result of random forest (100 estimators) on 2000 subset

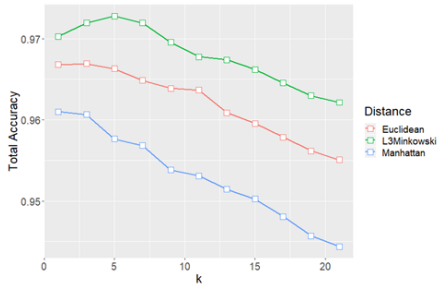

Figure 16. Total Accuracy in terms of k for three distances on MNIS

3.3 KNN and PCA_then_KNN

The KNN method for MNIST was implemented in the second part of project 1. The figure 16 above is the results of accuracy for different k for three types distances.

The best accuracy is 0.9728 when k = 5 for L3 Minkowski distance.

Since there are 784 pixels for each digit. It is extremely computation expensive to apply KNN directly. A PCA experiment is performed to see if it is possible to reduce the dimension of MNIST dataset as well.

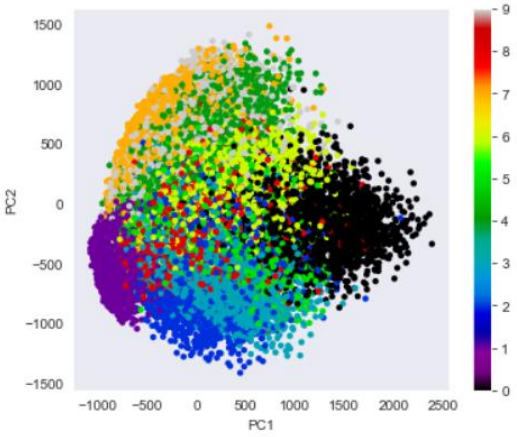

Figure 17. A visualization of PCA on MNIST dataset for 2 PCs

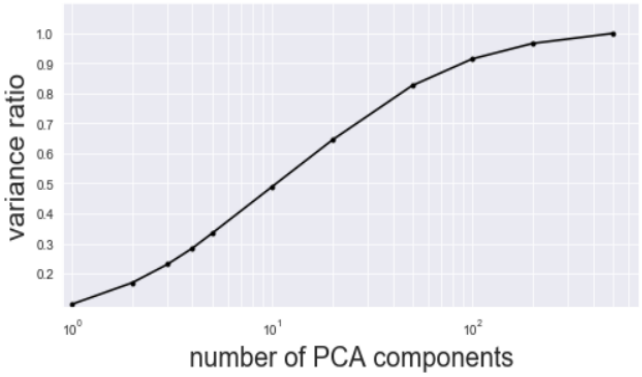

Figure 18. Variance explained for different number of PCA components

The figure 17 shows a trend of clustering when the original MNIST dataset only mapping to a 2-dimension principle components space. It is a hint that we should try more components. Like part 2, the figure 18 shows the variance explained accumulative ratio in terms of different number of PCA components. Statistically, we should choose around 100 PCs to explain over 90% variance. Like part 2, several tests are performed to find the best number of PCs. The results are shown in figure 19.

We can see that 20~50 PCs is the best setting to bobtain best accuracy. The accepted result is shown in figure 20 with 20 PCs for 2000 data points and split ratio of 0.8. The accuracy is 0.905 and the runtime in total is 0.9595 seconds.

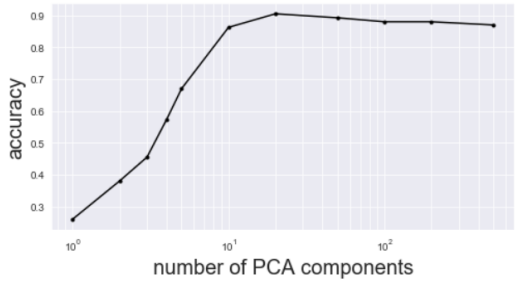

Figure 19. KNN accuracy for different number of PCA components

```
The runtime of KNN after PCA is
0:00:00.959458
Accuracy is
0.905
```

Figure 20. Results and runtime of KNN after PCA on 2000 subset

To compare the runtime and accuracy, a directly applied KNN is also performed. And the following figure 21 shows the result and runtime also for 2000 data subsets. The overall accuracy is 0.87 and the runtime is 0.6433 seconds.

```
The runtime of KNN is
0:00:00.624429
Accuracy is
0.87
```

Figure 21. Results and runtime of KNN on 2000 subset

Now with the hyper parameter we got, we can perform all three tests on the whole dataset with split ratio still 0.8. Figure 22 shows the results and runtime. I will compare and discuss in the next section.

```
The runtime of random forest is
0:00:16.318564
Accuracy is
0.9648809523809524

The runtime of KNN is
0:05:04.203420
Accuracy is
0.9680952380952381

The runtime of KNN after PCA is
0:00:04.474487
Accuracy is
0.9652380952380952
```

Figure 22. Results (validation accuracy) and runtime of all three methods for the whole dataset with split ratio of 0.8

3.4 Comparison

From the above results, in summary, the overall performance and runtime is shown in the following table:

Method:	Accuracy	Runtime
Random Forest	0.9649	16.32 seconds
KNN	0.9681	304.20 seconds
KNN after PCA	0.9652	4.47 seconds

The overall validation accuracy (performance) is similar among all three methods. KNN is slightly better but is significantly computation expensive than random forests, which is close to 20 times slower. However, if apply PCA first and reduce the dimension from 784 into 20, the accuracy is even still slightly better than random forest but is about 70 times faster. For MNIST dataset, we can conclude that KNN after PCA would be the best classifier. If not considering PCA, random forest is better for much faster runtime but negligible difference of accuracy.

4. Conclusion

For the Simplified Human Activity Recognition w/ Smartphone dataset, both K-Mean clustering and PCA methods perform well for almost identical accuracy and the PCA is around 10 times faster due to dimension reduction. For the MNIST dataset, both random forest and KNN obtains almost identical performance on accuracy but random forest with 100 estimators are significantly faster, unless apply KNN after a PCA dimension reduction.

5. Codes

All the codes or Jupyter notebook files are available on my GitHub:

https://github.com/lls1129/MachineLearningProject3

For this project, codes are available as well as packages are free to use. Based on given codes and datasets, some changes are made for better results and comparisons. The bonus part is also available on my Github.

6. Reference

Ruslan Klymentiev (2018, July 21). K-Means Clustering and PCA of Human Activity Recognition
https://www.kaggle.com/ruslankl/k-means-clustering-pca
Samuel (2013). Comparing random forest, PCA and kNN
https://www.kaggle.com/sflender/comparing-random-forest-pca-and-knn