Experiment 3: PCA Dimensionality Reduction Analysis and Clustering Experiment

I. Principal Components Analysis (PCA)

1. Learn http://scikit-learn.org/stable/modules/decomposition.html#pca

a) Generate three crowds of data;

b) Calculate the mean value of the generated data by using numpy.mean();

c) Calculate the covariance matrix by using numpy.cov(), understand the meaning of parameter rowvar;

d) Calculate the eigenvalues and eigenvectors by using numpy.linlag.eig();

e) Rearrange the eigenvalues and respective eigenvectors, find the largest n vectors and use them to reduce the dimensionality of original data;

f) write the corresponding codes.

The codes are shown below:

```
#Task-a Data
cluster1=np.random.uniform(0.5,1.5,(2,200))
cluster2=np.random.uniform(3.5,4.5,(2,200))
cluster31=np.random.uniform(3.5,4.5,(1,200))
cluster32=np.random.uniform(0.5,1.5,(1,200))
cluster3=np.concatenate((cluster31,cluster32))
X=np.concatenate((cluster1,cluster2,cluster3),1).T
# ##################################################
#Task-b mean
mean0 = np.mean(X,0)
mean1 = np.mean(cluster1,1)
mean2 = np.mean(cluster2,1)
mean3 = np.mean(cluster3,1)
X0 = X - mean0
# ##################################################
#Task-c covariance
cov0 = np.cov(X,rowvar=False)
print('Covariance Matrix:')
print(cov0)
print('————————')
# ##################################################
# ##################################################
#Task-d eigs
w,v = la.eig(cov0)
print("Eigenvalues:")
print(w)
print('————————')
print("Eigenvectors:")
print(v)
print('————————')
# ##################################################
#Task-e sort
eig_pairs = [(np.abs(w[i]), v[:,i]) for i in range(len(w))]
eig_pairs.sort(reverse=True)
new_feature=eig_pairs[0][1]
print(np.shape(eig_pairs))
new_data = np.dot(new_feature,np.transpose(X0))
print(np.shape(X0))
plt.figure(1)
plt.plot(X[:,0],X[:,1],'r.')
plt.plot(new_data,[3]*len(new_data),'b.')
plt.legend(["Original Data","Dim-reduced data"])
plt.grid(True)
plt.show()
```

Fig.1 Codes of PCA

From above description and codes we can get the main procedure of PCA. PCA is used to reduce the dimensionality of original data by calculate the eigenvalues and corresponding eigenvectors of the covariance matrix and choosing some

rearranged eigenvectors to form the new basis vectors and representing the original data in the frame of the space supported by chosen eigenvectors. The covariance matrix and its eigenvalues, eigenvectors are shown below:

```
Covariance Matrix:
[[2.11925043 1.03698292]
 [1.03698292 2.11743273]]
————————
Eigenvalues:
[3.1553249  1.08135826]
————————
Eigenvectors:
[[ 0.70741658 -0.70679685]
 [ 0.70679685  0.70741658]]
————————
```

Fig.2 Covariance matrix and eigenvalues, eigenvectors

Then, we put the processed data after PCA and original data in the same plot as shown in Fig.3.

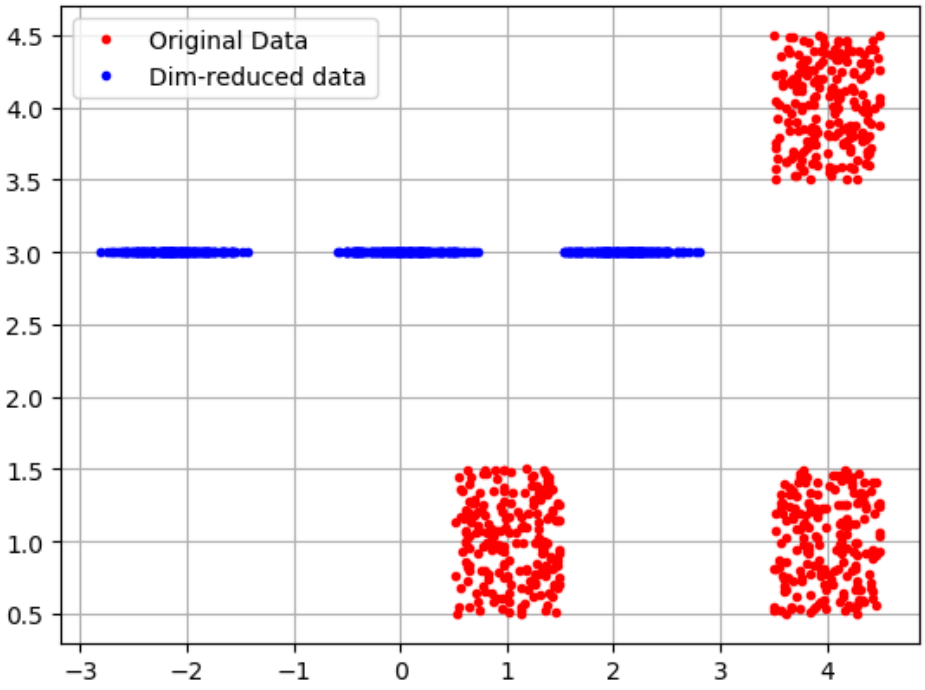

Fig.3 Data comparison

It is easy to find out that the original 2-D data is deduced to 1-D data and is more separate in some sense and this is exactly what PCA is used for.

2. Refer to http://scikit-learn.org/stable/auto_examples/applications/plot_face_recognition.html#sphx-glr-auto-examples-applications-plot-face-recognition-py and rearrange the eigenvalues. Choose the vectors corresponding to top 80% an 98% variance to reform the eigenfaces constructed by PCA.

First, we choose the top 500 faces out of 966 faces. It turns out that 500 eigenfaces takes more than 99% variance. Then we find the position corresponding to 80% and 98% variance. The core codes are shown below:

```
79  var_sum = np.cumsum(pca.explained_variance_ratio_)
80  # print('Explained Variance Ratio:',var_sum)
81  var80 = np.where(var_sum >= 0.8)
82  p80 = var80[0][0]
83  var98 = np.where(var_sum >= 0.98)
84  p98 = var98[0][0]
85  print('Position of Sum Variance 80%:',p80)
86  print('Position of Sum Variance 98%:',p98)
```

Fig.4 Codes and results of finding the specified position

As is shown in Fig.4 First we compute the cumulative sum of the explained variance ratio of the eigenfaces. Then we use numpy.where() to determine the position which meets the demands of 80% and 98 variance.
We can see that the top 28 faces can explain 80% information and the top 238 faces can explain 98% information. The rest part of the codes is rather simple, just make a duplicate procedure of PCA computation and SVM classifier training.

```
Total dataset size:
n_samples: 1288
n_features: 1850
n_classes: 7
Extracting the top 500 eigenfaces from 966 faces
Position of Sum Variance 80%: 28
Position of Sum Variance 98%: 238
done in 0.311s
_____________________________________
Projecting the input data on the eigenfaces orthonormal basis
done in 0.035s
The shape of eigenfaces with 80% variance: (28, 50, 37)
The shape of eigenfaces with 98% variance: (238, 50, 37)
Fitting the classifier to the training set
_____________________________________
```

Fig.5 Details of codes in Fig.4

```
Best estimator found by grid search(80%):
SVC(C=1000.0, cache_size=200, class_weight='balanced', coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma=0.01, kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

```
Best estimator found by grid search(98%):
SVC(C=1000.0, cache_size=200, class_weight='balanced', coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma=0.001, kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

Fig.6 Best estimators of 80% and 98% variance

Predicting people's names on the test set

80%

	precision	recall	f1-score	support
Ariel Sharon	0.50	0.69	0.58	13
Colin Powell	0.71	0.85	0.77	60
Donald Rumsfeld	0.52	0.59	0.55	27
George W Bush	0.89	0.84	0.86	146
Gerhard Schroeder	0.65	0.60	0.63	25
Hugo Chavez	0.91	0.67	0.77	15
Tony Blair	0.73	0.61	0.67	36
accuracy			0.76	322
macro avg	0.70	0.69	0.69	322
weighted avg	0.77	0.76	0.76	322

```
[[ 9  2  2  0  0  0  0]
 [ 3 51  3  2  1  0  0]
 [ 4  2 16  4  1  0  0]
 [ 1 11  7 122  2  0  3]
 [ 0  1  2  4 15  0  3]
 [ 0  2  0  1  0 10  2]
 [ 1  3  1  4  4  1 22]]
```

98%

	precision	recall	f1-score	support
Ariel Sharon	0.53	0.69	0.60	13
Colin Powell	0.78	0.88	0.83	60
Donald Rumsfeld	0.70	0.70	0.70	27
George W Bush	0.93	0.90	0.92	146
Gerhard Schroeder	0.83	0.80	0.82	25
Hugo Chavez	0.80	0.53	0.64	15
Tony Blair	0.85	0.81	0.83	36
accuracy			0.84	322
macro avg	0.78	0.76	0.76	322
weighted avg	0.84	0.84	0.84	322

```
[[ 9  1  1  2  0  0  0]
 [ 2 53  2  2  0  1  0]
 [ 3  2 19  2  0  0  1]
 [ 2  5  3 132  1  1  2]
 [ 0  2  0  2 20  0  1]
 [ 0  4  0  1  1  8  1]
 [ 1  1  2  1  2  0 29]]
```

Fig.7 The prediction results of SVM with 80% and 98% variance

As for eigenfaces, they are generated just like the last task we did to dots data. We put all faces together and can get a “mean face”, then we can calculate the covariance matrix of the “difference face”, which is the results of original faces minus mean faces. Then, we calculate the eigenvalues and eigenvectors of the covariance matrix and multiply original faces with the new basis of selected eigenvectors, we can get the following eigenfaces.

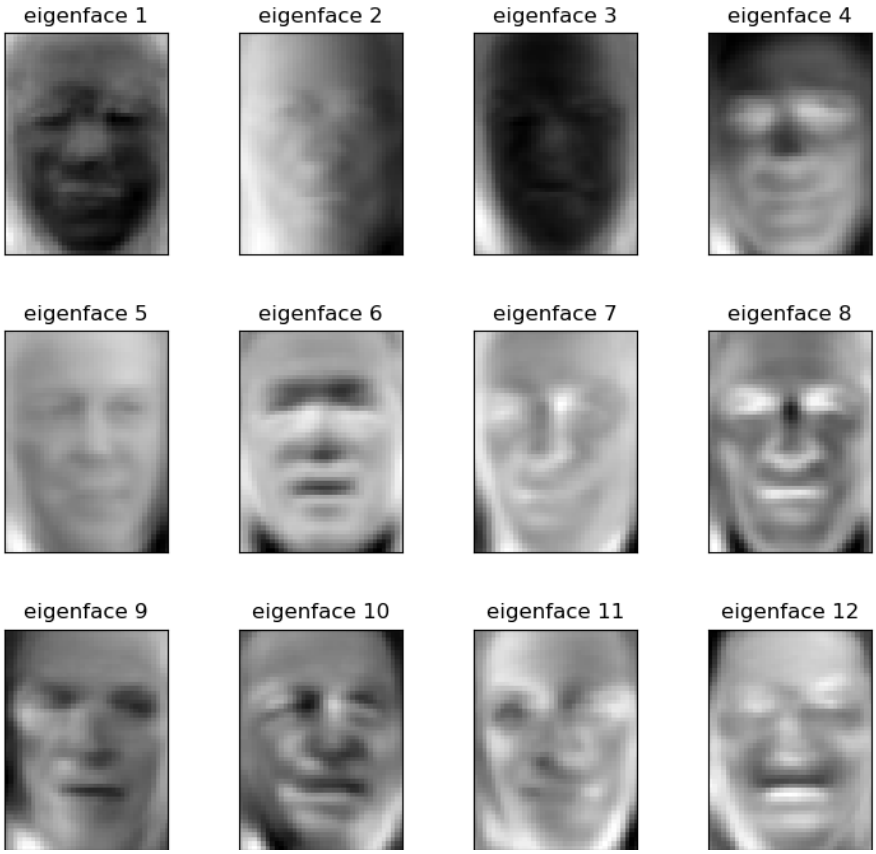

Fig.8 The top 12 eigenfaces

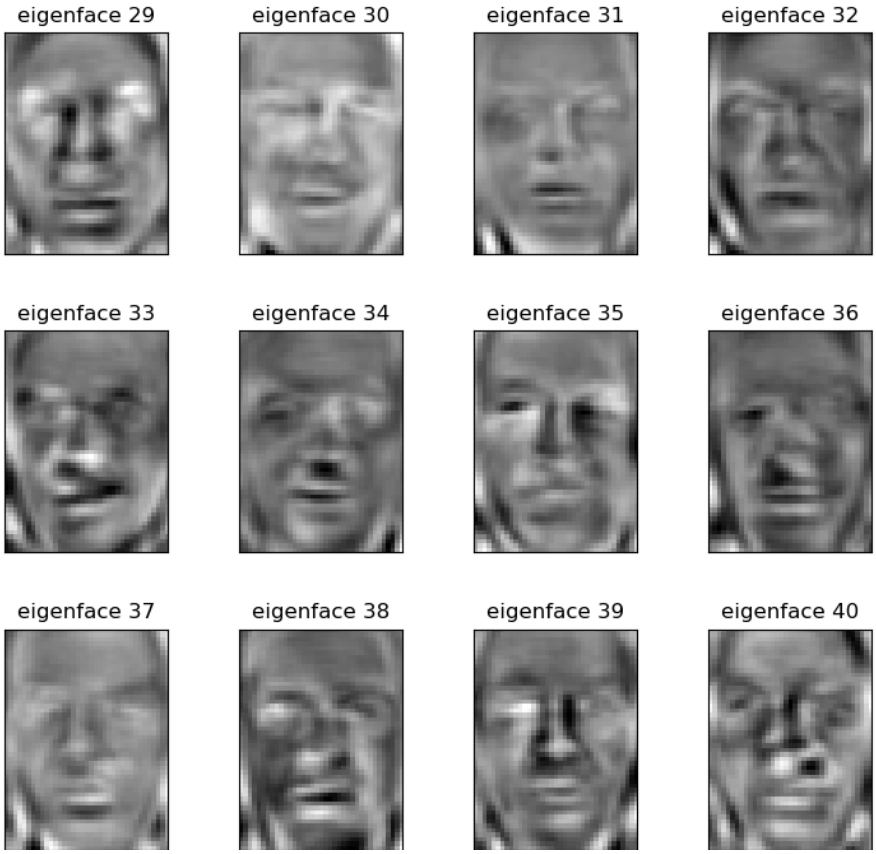

Fig.9 The top 29~40 faces

II. Clustering Techniques Analysis

1. K-means Clustering

a) Understand the procedure of K-means clustering:
http://scikit-learn.org/stable/auto_examples/cluster/plot_mini_batch_kmeans.html#sphx-glr-auto-examples-cluster-plot-mini-batch-kmeans-py

b) write codes to show the influence of different k value for data in last task and visualize the influence. Explain the reasonable k value and why it is reasonable.

The procedure of this task is rather like that of task I.1: data generation, mean value calculation, covariance calculation, eigenvalues and eigenvectors calculation, eigenvalues sorting, PCA dimensionality reduction and utilize sklearn.cluster.KMeans() to cluster the processed data.

Since data in task I.1 has a rather small size, here we generate data of larger scale. The original data and dimensionality-reduced data along with the clustering results of different k value are shown below.

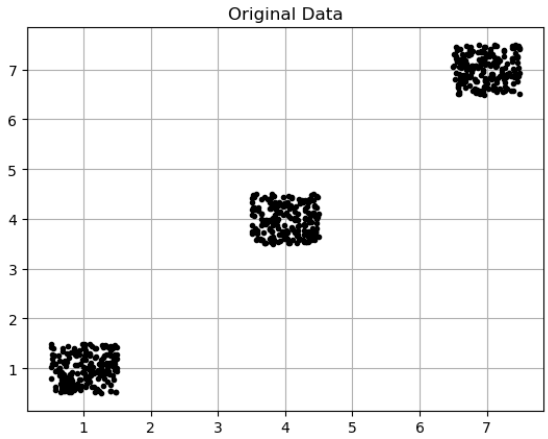

Fig.10 Original data

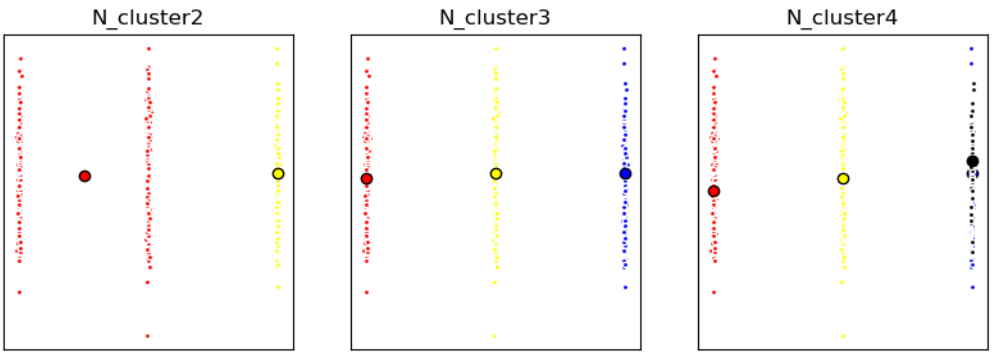

Fig.11 K-means clustering

From above we can find that k = 3 is the optimal value since the data after PCA is clearly divided into three crowds. This shed a light on us that specifying a proper k value is surely important in clustering when using K-means method. Because the clustering result is strongly influenced by the initial k value.

2. hierarchical Clustering

1) Understand influence of different linkage criterion.
http://scikit-learn.org/stable/auto_examples/cluster/plot_digits_linkage.html#sphx-glr-auto-examples-cluster-plot-digits-linkage-py

2) Utilize AgglomerativeClustering algorithm to cluster iris data into 3 crowds. Plot the clustering results and print the number of each crowd. Give the evaluation results of confusion matrix.
According to the codes in the reference, we can give the results of this task as below.

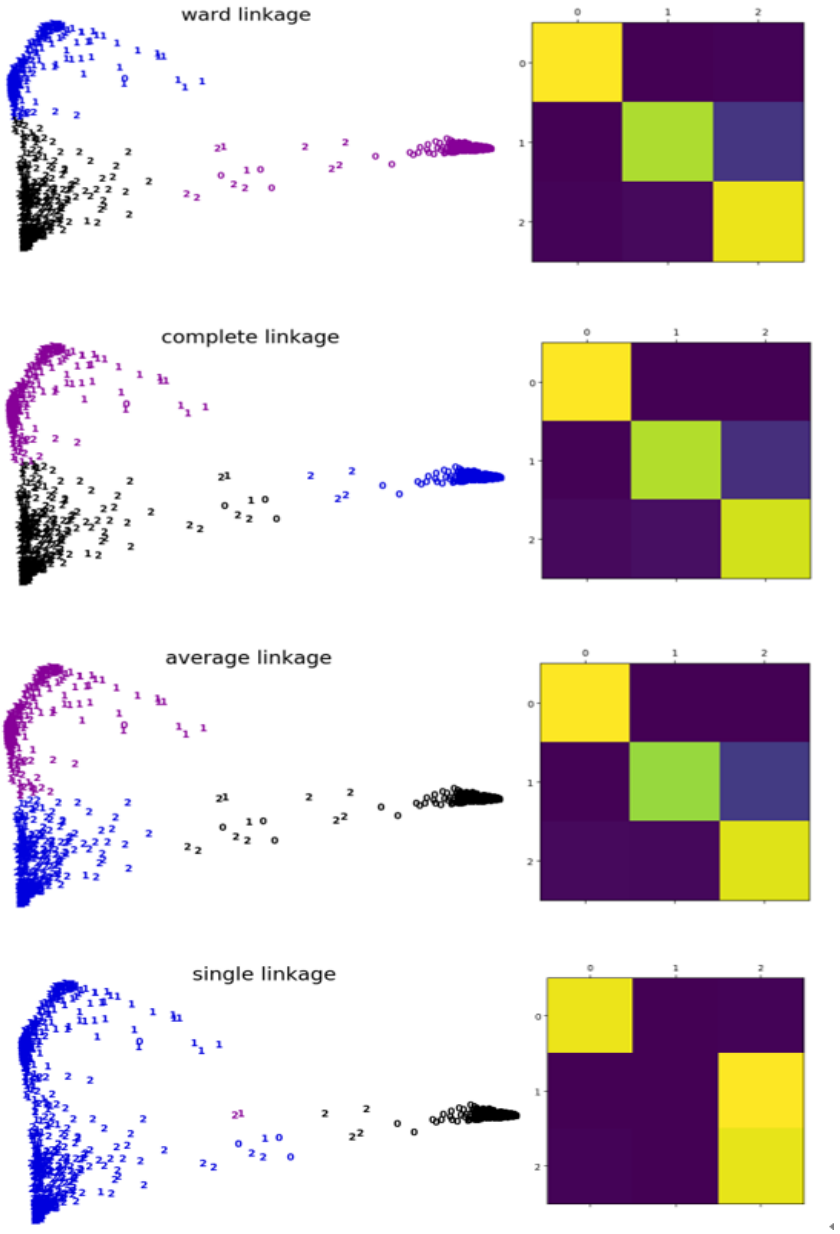

Fig.12 Clustering with different linkage criterion

From Fig.12 we can see that ward/complete/average linkage show approximately alike results while single linkage shows rather poor performance. From this we

can learn that we should consider which linkage criterion is better suited for the data in the task we meet.

Sometimes, before we choose some fancy algorithms or models, we might better check up the dataset and dig into the feature and characteristics of the dataset. This would do much benefit to our project.

3. Spectral Clustering

1) Understand the principle of spectral clustering.
http://scikit-learn.org/stable/auto_examples/cluster/plot_segmentation_toy.html#

2) Realize and decipher codes:
https://scikit-learn.org/stable/auto_examples/cluster/plot_coin_segmentation.html#sphx-glr-auto-examples-cluster-plot-coin-segmentation-py

The word "spectral" in spectral clustering means all the eigenvalues of the matrix. Spectral clustering method is to maximize the sum of weights within the subplots itself and minimize the weights of different subplots.

First, compute the similarity matrix W and adjacency matrix D. Then, compute the corresponding Laplace matrix L = D – W. Calculate the eigenvalues of L and resort them from largest to smallest. Then, choose the top k eigenvectors to form matrix U to process the new data. Finally, cluster the new data with K-means method and output the resulting crowds.

The codes show K-means spectral clustering and discrete spectral clustering and the difference of which is that the method to get the projected embedding space.

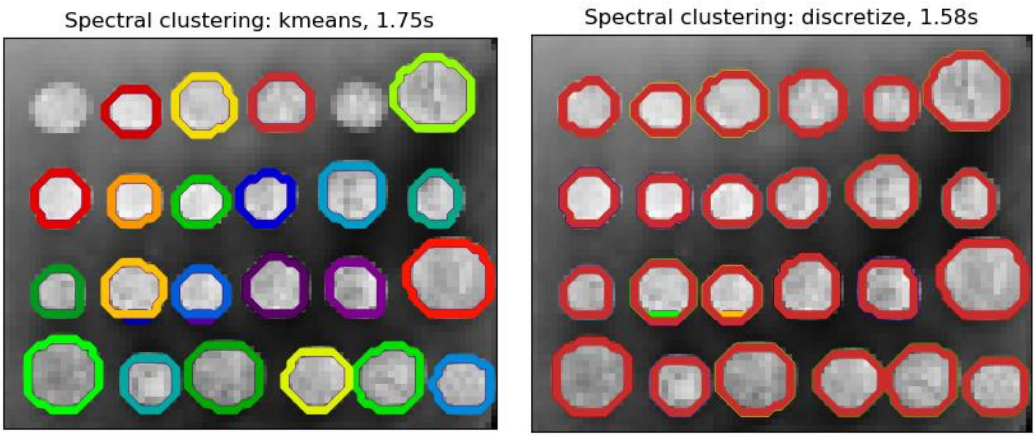

Fig.13 Comparison of 'K-means' and 'discretize' spectral clustering

Conclusion

In this experiment I learned the basic dimensionality reduction method PCA and some clustering methods such as K-means, hierarchical clustering and spectral clustering. It is easy to implement these methods with sklearn. However, it is more important to understand the principles and fundamental logic behind it. By coding and debugging, I gained deeper understanding of these clustering methods. Especially the thoughts of dimensionality reduction and the principle of spectral clustering do interest me. I think I might spend some time to read more materials about it.