Übung TUDaExercise
LastName, FirstName: _____ EnrollmentID: ␣␣␣␣␣␣␣␣

Task 3: Principal Component Analysis

3a)

Normalizing can convert the original data into pure,dimensionless values and remove the unit restriction on data, then it is easy to compare and weight indicator among data with different units or scales. After Normalizing the data is easy to handle. One Advantage is that the convergence speed of the model improves. For example, if there is only two features, and the range of component 1is 1000 and the range of component 2 is 5. The speed of iteration is very low during the optimization. But after normalizing the speed will be faster. Another advantage is that the accuracy of the model improves. In the last example, when we need to calculate about the distance, the second feature make much less contribute to the result than the first one.

Code:

```
import numpy as np
import matplotlib.pyplot as plt
# get data
iris = np.loadtxt("./dataSets/iris.txt", delimiter=',')
iris_set = iris[:, 0:4]
iris_label = iris[:, 4]
# Normalizing
iris_norm = (iris_set-np.mean(iris_set, axis=0))/np.std(iris_set, axis=0)
print('mean:', np.mean(iris_norm, axis=0))
print('var:', np.var(iris_norm, axis=0))
```

3b)

We need only two components to in order to explain at least 95% of the dataset variance. From the picture below, we can know that the cumulative variance of first two components is 95.8010%.

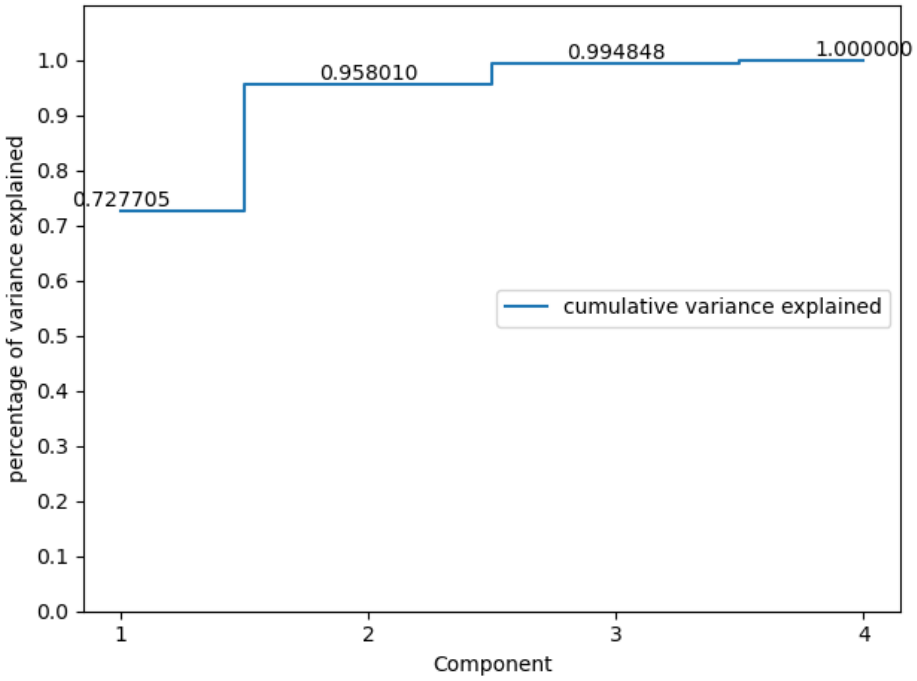


Figure 1: cumulative variance explained

Code:

```
#PCA
cov_matrix = np.cov(iris_norm, rowvar=0)
eigenvalues, eigenvectors = np.linalg.eig(cov_matrix)
eigenvalues_sorted = np.sort(eigenvalues)[-1::-1]
total = np.sum(eigenvalues_sorted)
var_explained = np.zeros(4)
for i, item in zip(range(4), eigenvalues_sorted):
   var_explained[i] = item/total
cum_var_explained = np.cumsum(var_explained)
x = ['1','2','3','4']
plt.step(x, cum_var_explained, where='mid', label='cumulative variance explained')
for a, b in zip(x, cum_var_explained):
   plt.text(a, b, '%f' % b, ha='center', va='bottom')
plt.ylim((0, 1.1))
plt.yticks(np.arange(0,1.1,0.1))
plt.xlabel('Component')
plt.ylabel('percentage of variance explained')
plt.legend(loc='center right')
plt.show()
```

3c)

From the picture below, we can know that the data can be clearly distinguished by using 2 components, specially, through component1 the data is distinguished. Two components can explain 95% of the dataset variance. After PCA we spared two feature, i.e., 50% data, but the data can be still good distinguished and only 5% dataset variance is lost.

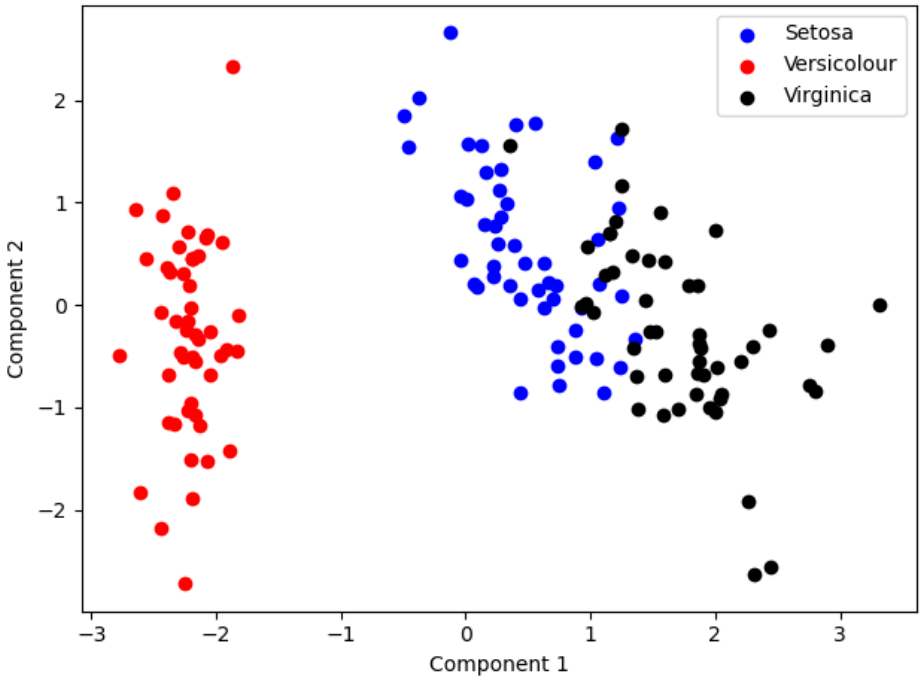


Figure 2: lower dimensional projection

Code:

```
vector_n = eigenvectors[:,0:2]
iris_lowDim = iris_norm@vector_n
name =['Setosa', 'Versicolour', 'Virginica']
print(iris_lowDim.shape)
for lab, col in zip(range(3), ('blue', 'red', 'black')):
    plt.scatter(iris_lowDim[iris_label==lab, 0], iris_lowDim[iris_label==lab, 1],
        label=name[lab], color=col)
plt.xlabel('Component 1')
plt.ylabel('Component 2')
plt.legend()
plt.show()
```

3d)

N. of components	x1	x2	x3	x4
1	0.22925036	0.18006108	0.29805579	0.31692192
2	0.11261513	0.19281695	0.13314032	0.12717403
3	0.07919279	0.23615959	0.13302896	0.1257637
4	0.04925365	0.23658978	0.1322316	0.10227491

Code:

```
# reconstruct
def nrmse(eigvec,n):
   vec_n = eigvec[:, 0:n]
   lowDim = iris_norm @ vec_n
   std = np.std(iris_set, axis=0)
```

```
    iris_reconstruct = lowDim@vec_n.T+np.mean(iris_set, axis=0)
    k = np.max(iris_set, axis=0)-np.min(iris_set, axis=0)
    return np.sqrt(np.mean(np.square(iris_reconstruct-iris_set), axis=0))/k
for n in range(4):
    print(nrmse(eigenvectors, n))
```

3e)

1. Explain the difference between PCA and ZCA whitening.

PCA whitening: It ensures the variance of each dimension of the data is 1.
ZCA whitening: It ensures that the variance of each dimension of the data is the same. It does a rotation operation on the basis of PCA whitening to make the data after whitening closer to the original data

2. State the equation(s) to compute the ZCA whitening parameters, given the data.

There are m samples, and the feature dimension of each sample is n. We have a sample set feature matrix X of n rows and m columns. And then we calculate Zero-average each row of X. The covariance matrix of X:

$$\Sigma = \frac{1}{m} X X^T \tag{1}$$

Calculate the eigenvalues and corresponding eigenvectors of the covariance matrix:

$$\frac{1}{m} X X^T = U A U^T \tag{2}$$

Rotate data

$$X_{rot} = U^T X \tag{3}$$

Then we get the PCA whitening

$$X_{PCAwhite,i} = \frac{X_{rot,i}}{\sqrt{\lambda_i + e}}, e = 1.0e^{-5} \tag{4}$$

ZCA whitening:

$$X_{ZCAwhite,i} = U X_{PCAwhite,i} \tag{5}$$

3. State the equation(s) to whiten a (new) data example x, given the ZCA parameters.
4. Compute and report the ZCA whitening parameters for the unnormalized IRIS data (including numerical values!).

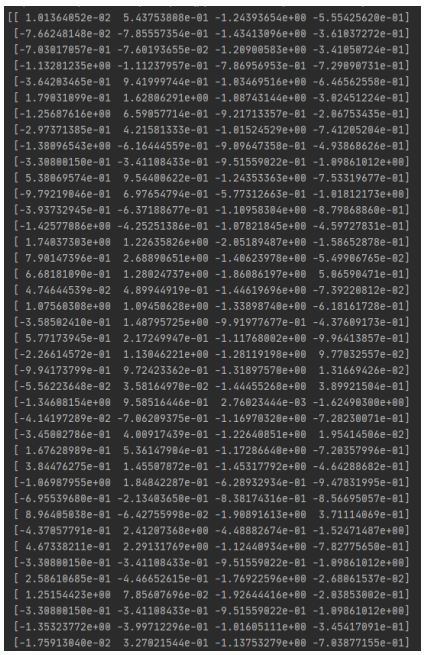
```
import numpy as np
# get data
iris = np.loadtxt("./dataSets/iris.txt", delimiter=',')
iris_set = iris[:, 0:4]
iris_label = iris[:, 4]


def ZCA_whitening(x):
    avg = np.mean(x, axis=0)
    x = x - avg
    sigma = np.cov(x, rowvar=0)
    u,s,v = np.linalg.svd(sigma)
    xRot = x@u
    xPCAwhite = xRot / np.sqrt(s + 1e-5)
    xZCAwhite = (u@xPCAwhite.T).T
```

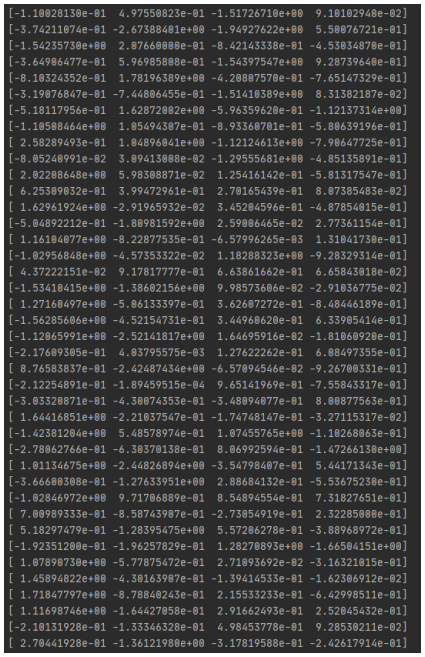
```
    return xPCAwhite,xZCAwhite


xPCAwhite, xZCAwhite = ZCA_whitening(iris_set)
print(xZCAwhite)
```

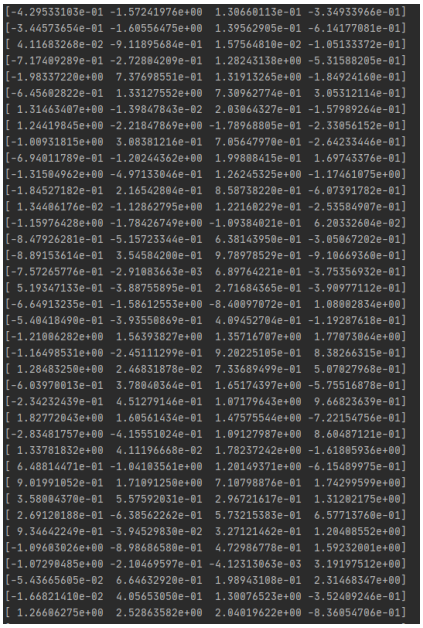
result:



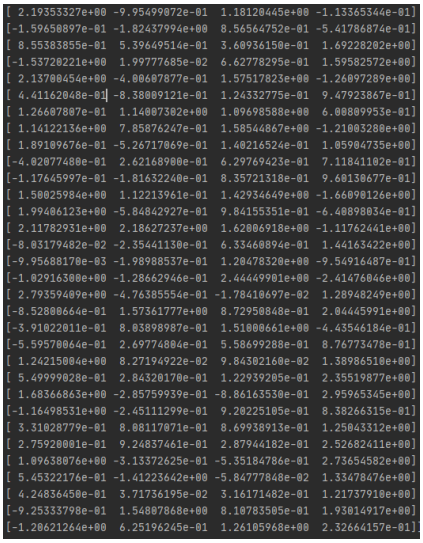
(a) data1.



(b) data2.



(c) data3.



(d) data4.

Figure 3: ZCA

3f)

In general, PCA is suitable for linear dimensionality reduction of the data. Kernel Principal Component Analysis enables non-linear dimensionality reduction of data and is used to deal with linear distinguishable data set.The basic idea of KPCA is that for a input matrix X, by using a nonlinear mapping we map all the samples in X to a high or even infinite dimensional space to making it linearly distinguishable, and then process the data in this high-dimensional space using PCA.

These are the steps for the implementation of the kernel PCA algorithm:
1. Choose a kernel mapping $k(\mathbf{x}_m, \mathbf{x}_n)$.
2. Obtain$\mathbf{K}$ based on training data $\{\mathbf{x}_n, (n = 1, \cdots, N)\}$.
3. Solve eigenvalue problem of $\mathbf{K}$ to get λ_i and $\mathbf{a}_i$.
4. For each given data point $\mathbf{x}$, obtain its principal components in the feature space: $(f(\mathbf{x}) \cdot \phi_i) = \sum_{n=1}^{N} a_n^{(i)} k(\mathbf{x}, \mathbf{x}_n)$
5. Do whatever processing (e.g., feature selection, classification) in the feature space.
(Reference: http://fourier.eng.hmc.edu/e161/lectures/kernelPCA/node4.html)

Limits: KPCA coats more time than PCA