K-PCA

November 3, 2018

In this file we will study the Kernel-PCA

1 Steps:

1.1 We pick a kernel

1.2 We construct the normalized kernel matrix of the data (dimension mŒm):

1.2.1

$$K_{zeromean} = K - 2 * 1_{\frac{1}{n}} K + 1_{\frac{1}{n}} K 1_{\frac{1}{n}}$$

1.3 We solve an eigenvalue problem:

1.3.1

$$K\alpha_i = \lambda_i \alpha_i$$

1.4 For any data point (new or old), we can represent it as:

1.4.1

$$y_j = \Sigma_i(\alpha_{ij}K(x, x_i)), j = 1, .., d$$

```
In [2]: import matplotlib.pyplot as plt
        from sklearn.datasets import make_moons
        from numpy import ones, exp, loadtxt, tanh
        from numpy.linalg import eig, norm
        from sklearn.preprocessing import normalize, scale
        import numpy as np

        VERBOSE = False
        def __DEBUG(msg):
                if VERBOSE: print(msg)

        fig = 1
        K2_SIGMA = 0.007

        def liner_kernel(X,Y):
                return np.dot(X,Y)
        def polynomial_kernel(X, Y):
                return (X.T.dot(Y) + 1) ** 2

        def gaussian_kernel(X, Y):
                return exp( (-1 * (norm(X - Y) ** 2)) / (2 * (K2_SIGMA ** 2)) )
```

The function that compute the zero mean Garm matrix:

```
In [3]: def k_matrix(A, kernel):

                n = A.shape[0]
                d = A.shape[1]
                K = ones((n, n))
                for i in range(n):
                        for j in range(n):
                                K[i, j] = kernel(A[i], A[j])

                K_SUM = K.sum() / (n ** 2)
                K_SUMROWS = K.sum(axis=1) / n

                K_ = ones((n, n))
                for i in range(n):
                        for j in range(n):
                                K_[i, j] = K[i, j] - K_SUMROWS[i] - K_SUMROWS[j] + K_SUM

                return K_
```

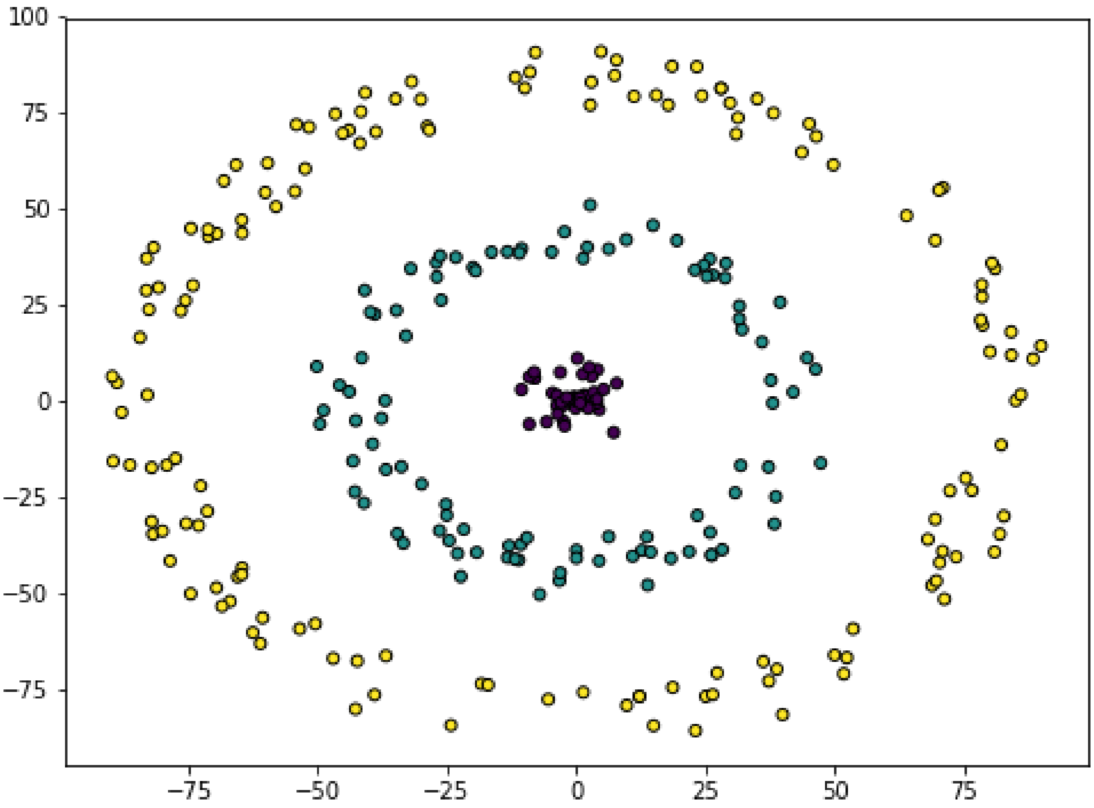
In this version of the K-PCA, we will generate only d component where d is the original dimensionality of the dataset

```
In [6]: def kpca(A, kernel):
                n = A.shape[0]
                d = A.shape[1]
                # calculate the kernelized matrix of data
                K = k_matrix(A, kernel)

                # eigendecoposition of kernelized covariance matrix
                eig_values, eig_vectors = eig(K)
                idx = eig_values.argsort()[::-1]
                eig_values = eig_values[idx]
                eig_vectors = eig_vectors[:,idx]

                # project data (only the first d component) d: the number of features in the o
                sub_eig_vectors = eig_vectors[0:d,:]
                #__DEBUG("sub_eig_vectors = \n" +  str(sub_eig_vectors))

                A_new = ones((n,d))
                for i in range(n):
                        for j in range(d):
```

```
                        temp = 0
                        for z in range(n):
                                temp += eig_vectors[j,z]*kernel(A[i],A[z])
                        A_new[i,j] = temp

        #__DEBUG("A_new = \n" + str(A_new))
        return A_new
```

Read and visulize the data:

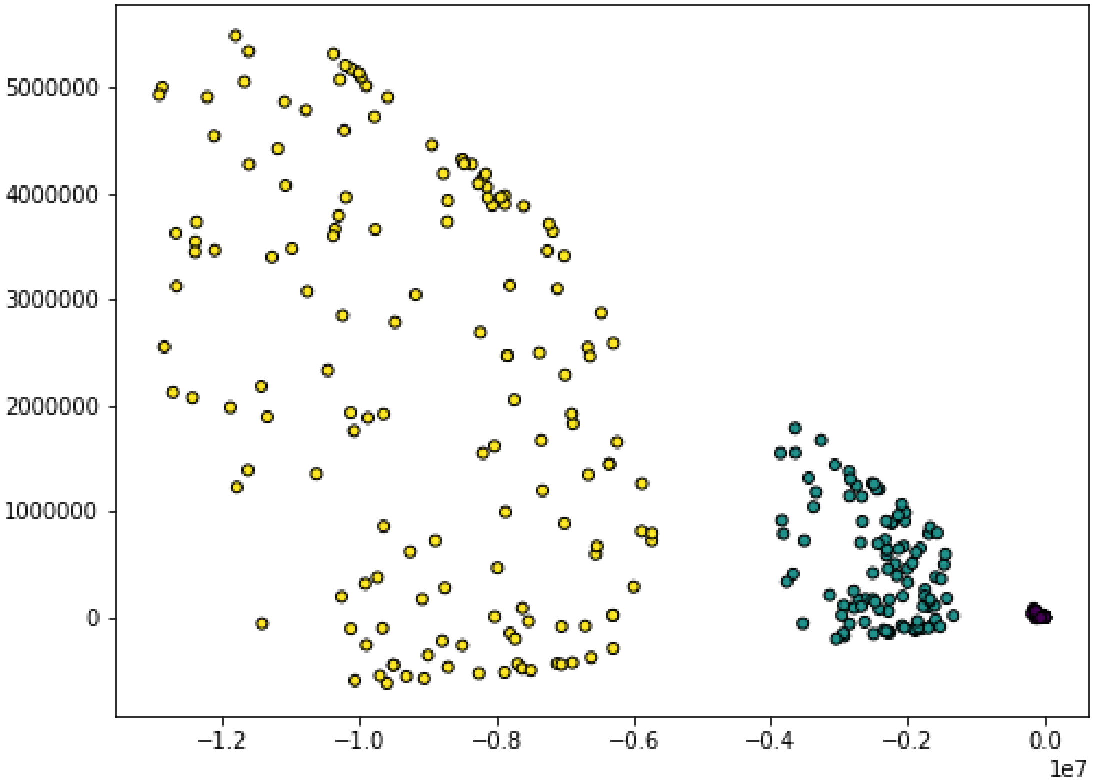
```
In [7]: from numpy import array
        #VERBOSE = True

        data = loadtxt("data.data")
        #__DEBUG("data : \n" + str(data))
        d = data.shape[1]
        A = data[:,0:(d-1)]
        Y = data[:,(d-1):d].T[0]

        plt.figure(fig, figsize=(8, 6))
        plt.clf()
        plt.scatter(A[:, 0], A[:, 1], c=Y,s=25, edgecolor='k')
        plt.show()
        fig += 1
```



Using the polunomial kernel:

```
In [8]: A_new = kpca(A, polynomial_kernel)
        plt.figure(fig, figsize=(8, 6))
        plt.clf()
        plt.scatter(A_new[:, 0], A_new[:, 1], c=Y,s=25, edgecolor='k')
        plt.show()
        fig += 1
```

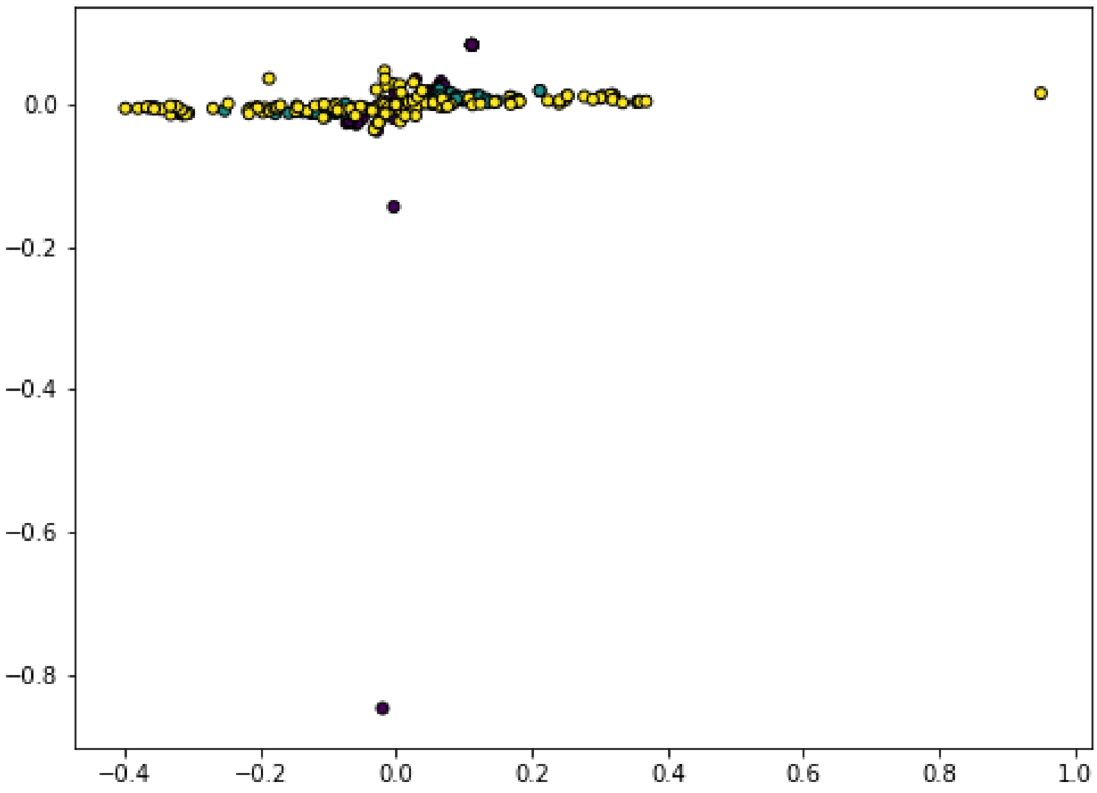
```
C:\Users\Muaz\Anaconda3\lib\site-packages\ipykernel_launcher.py:23: ComplexWarning: Casting co
```

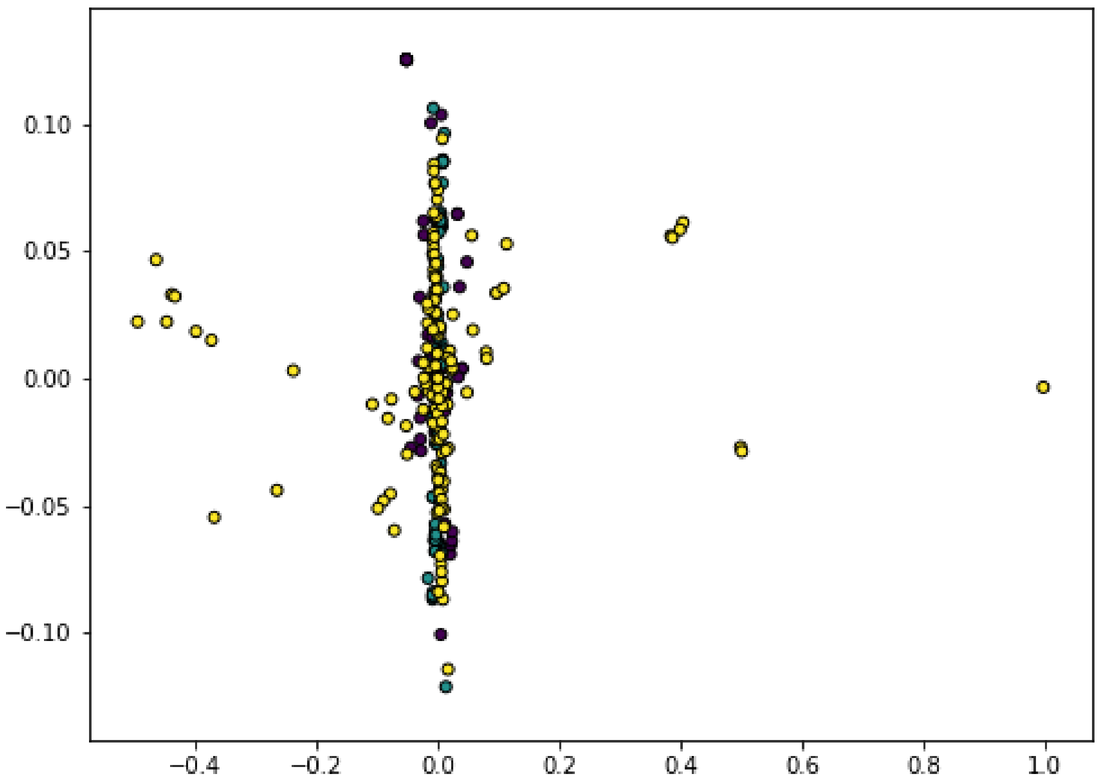


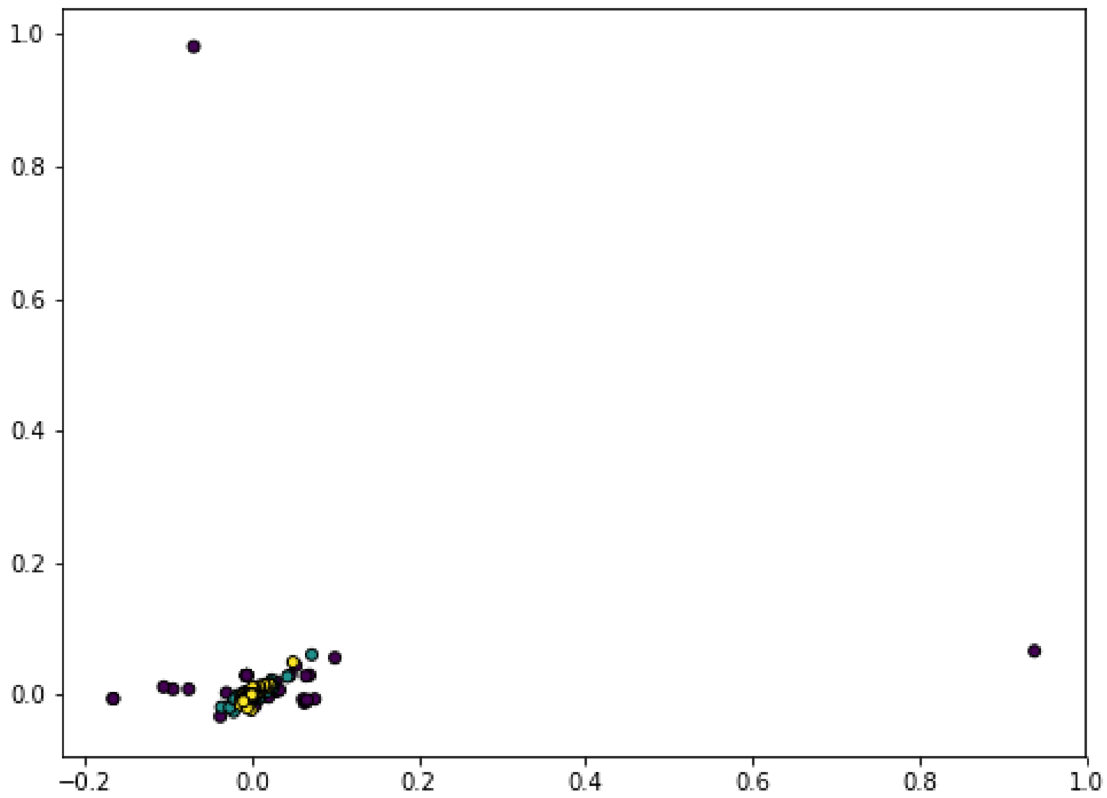
Using rbf kernel:

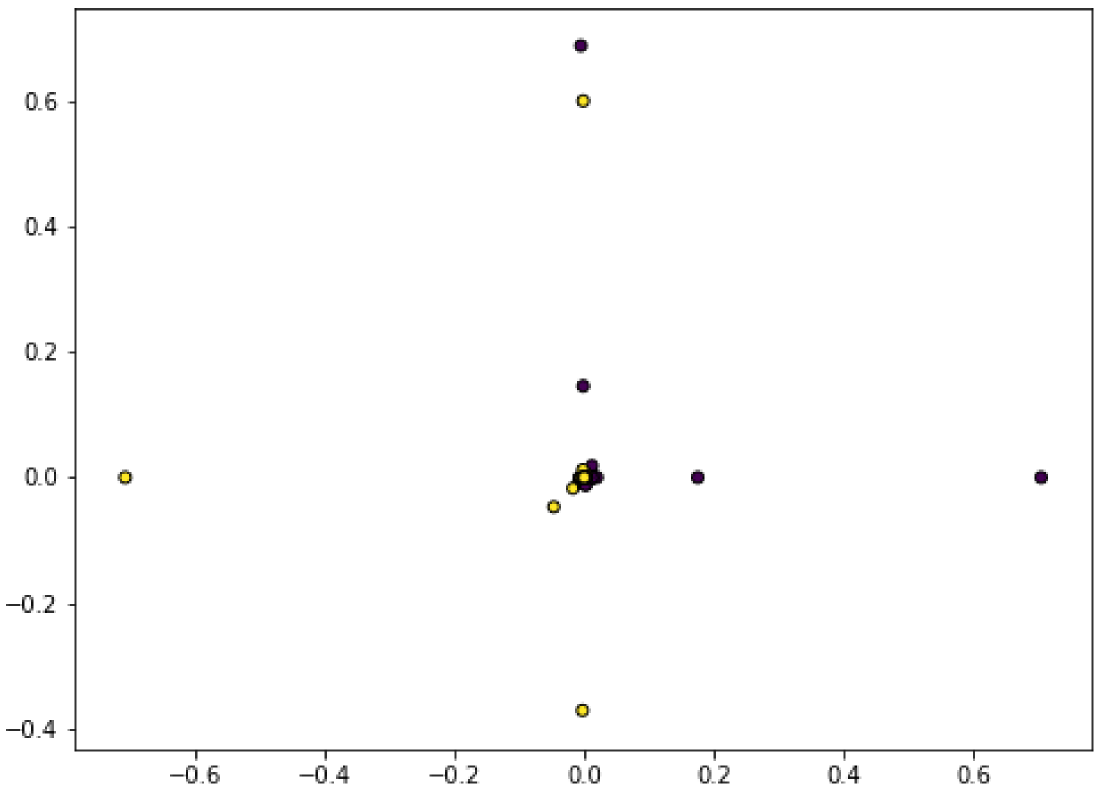
```
In [9]: sigma_array = [0.09,0.01,0.1,0.3,0.5,1,2]

        for sg in sigma_array:
                K2_SIGMA = sg
                A_new = kpca(A, gaussian_kernel)
                plt.figure(fig, figsize=(8, 6))
                plt.clf()
                plt.scatter(A_new[:, 0], A_new[:, 1], c=Y,s=25, edgecolor='k')
                plt.show()
                fig += 1
```

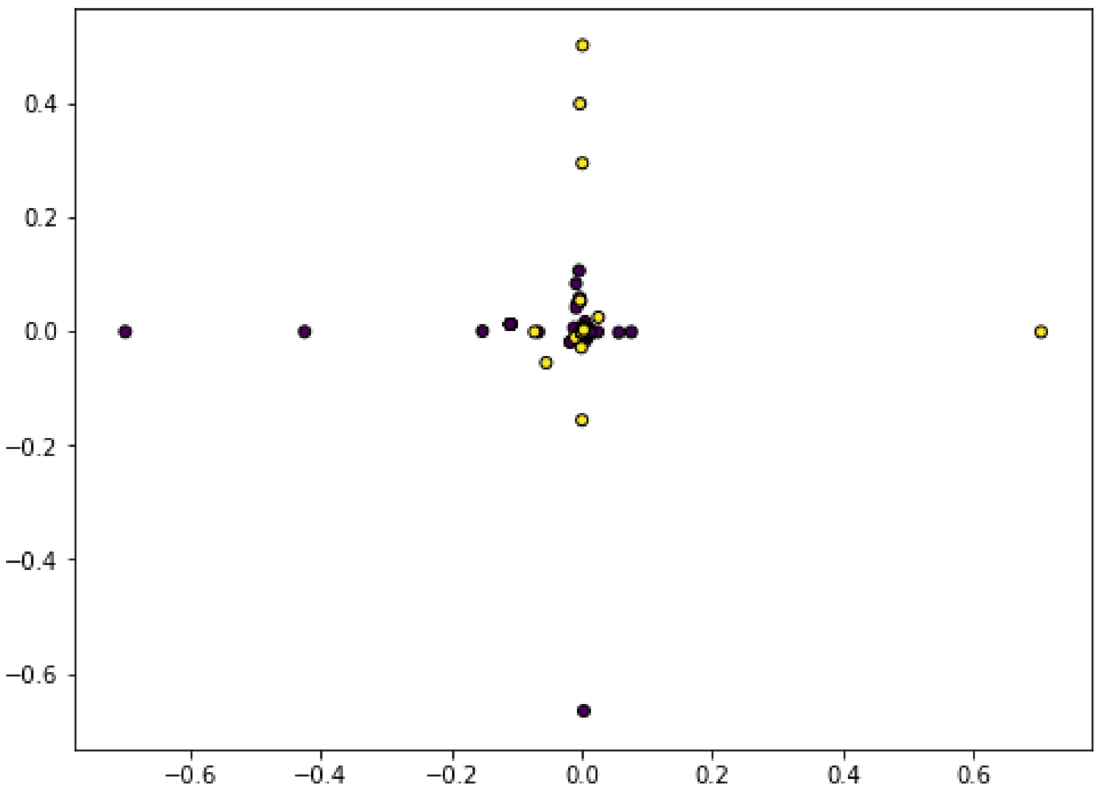
```
C:\Users\Muaz\Anaconda3\lib\site-packages\ipykernel_launcher.py:23: ComplexWarning: Casting co
```

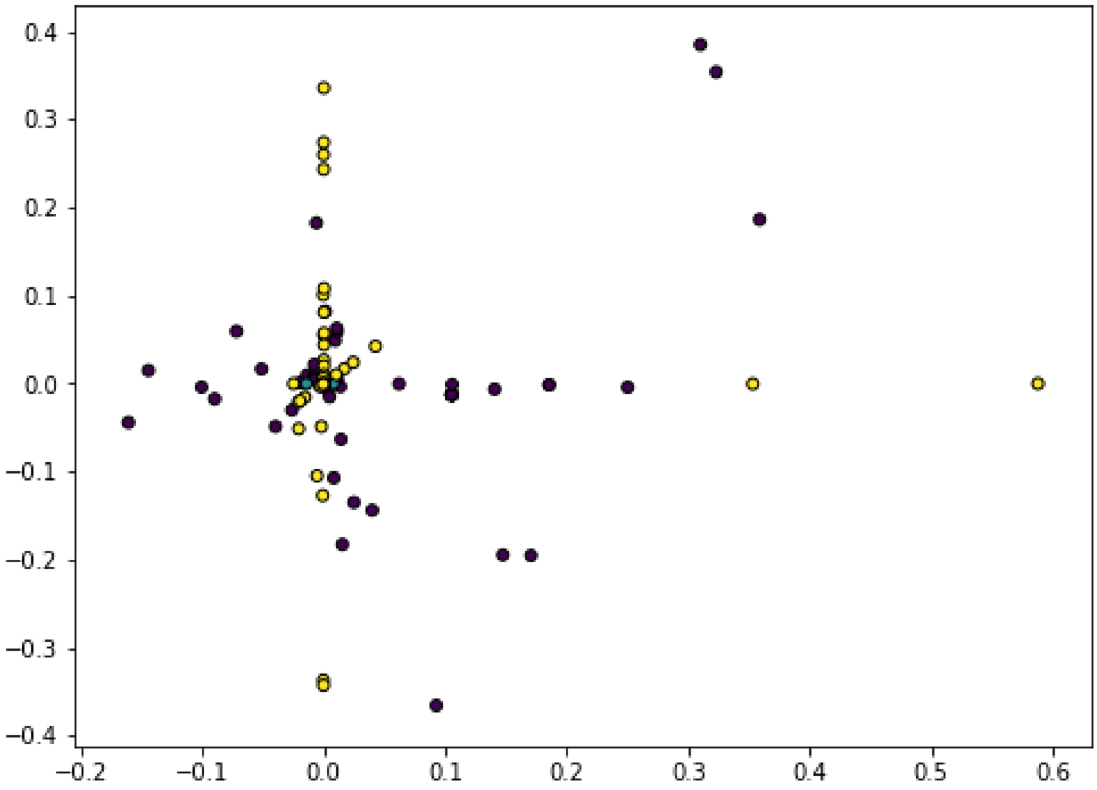


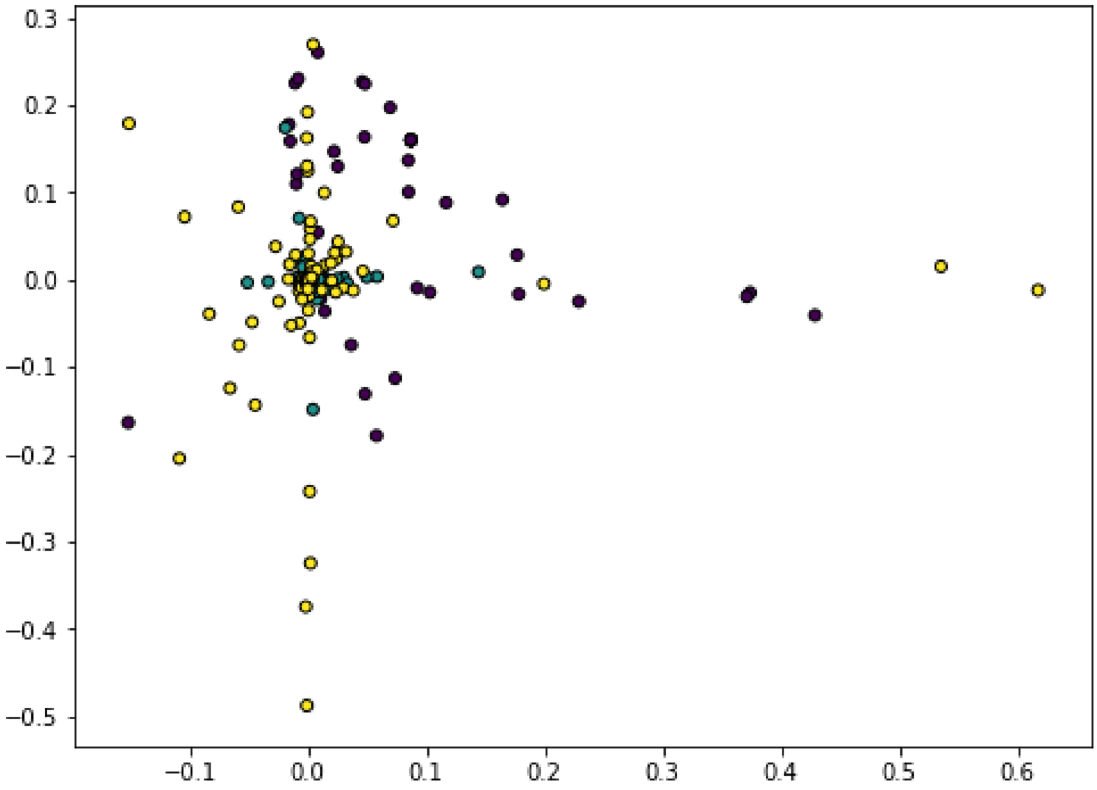

0.10
0.05
0.00
−0.05
−0.10
−0.4
−0.2
0.0
0.2
0.4
0.6
0.8
1.0


1.0
0.8
0.6
0.4
0.2
0.0
−0.2
0.0
0.2
0.4
0.6
0.8
1.0


0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6


0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6


0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
−0.2
−0.1
0.0
0.1
0.2
0.3
0.4
0.5
0.6


0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
−0.5
−0.1
0.0
0.1
0.2
0.3
0.4
0.5
0.6