Homework 11

Lingfeng Zhao

LZ1973

1

(a)

```
mean = X.mean(axis=0)
```

Sample mean : $[1.5, 2.5, 3.5]$

(b)

```
Q = np.cov((X - mean))
```

Covariance matrix Q is:

$$\begin{bmatrix} 1.66666667 & 0.33333333 & -1.66666667 \\ 0.33333333 & 1. & 1. \\ -1.66666667 & 1. & 3.66666667 \end{bmatrix}$$

(c)

```
eigval, eigvec = np.linalg.eig(Q)
```

Eigenvalues are $[4.74888619, 1.56450706, 0.01994008]$.

Eigenvectors are:

$$\begin{bmatrix} -0.45056922 & -0.66677184 & -0.59363515 \\ 0.19247228 & -0.72187235 & 0.66472154 \\ 0.87174641 & -0.18524476 & -0.45358856 \end{bmatrix}$$

(d)

```
a = np.dot(X - mean, eigvec.T)
```

The coefficients are:

$$\begin{bmatrix} 1.14161996 & -1.01215927 & 2.53421339 \\ -2.11589509 & 0.01050993 & -0.52237677 \\ 0.85548811 & -0.06766074 & -0.11645654 \\ 0.11878703 & 1.06931007 & -1.89538007 \end{bmatrix}$$

(e)

```
rec = np.dot(a, eigvec) + mean
```

Reconstructed samples:

$$\begin{matrix} 3.00000000e+00 & 2.00000000e+00 & 1.00000000e+00 \\ 2.00000000e+00 & 4.00000000e+00 & 5.00000000e+00 \\ 1.00000000e+00 & 2.00000000e+00 & 3.00000000e+00 \\ -2.22044605e-16 & 2.00000000e+00 & 5.00000000e+00 \end{matrix}$$

(f)

Here we choose 4.75 and 1.56.

```
two_large_eigvec = eigvec[:, 0:2]  # first two eignvalues are the biggest
a2 = np.dot(X - mean, two_large_eigvec)
rec2 = np.dot(a2, two_large_eigvec.T) + mean
```

The reconstructed samples:

$$\begin{bmatrix} -2.95145599 & -0.17610969 & -0.0888421 \\ 1.37104342 & -1.69406159 & 0.0198819 \\ -0.30682473 & 0.78694448 & 0.19125108 \\ 1.8872373 & 1.0832268 & -0.12229089 \end{bmatrix}$$

(g)

```
error = np.sum((rec2 - X)**2)
```

The error is 0.059820245731225984

2

(a)

$\alpha_1 = (\mathbf{x} - \mu) \cdot \mathbf{v}_1 = 2\sqrt{2}$

$\alpha_2 = (\mathbf{x} - \mu) \cdot \mathbf{v}_2 = -\sqrt{2}$

(b)

$\hat{\mathbf{x}} = \mu + \alpha_1 \mathbf{v}_1 + \alpha_2 \mathbf{v}_2 = [2, 3, 2]$

(c)

$\|\mathbf{x} - \hat{\mathbf{x}}\|^2 = 4$

3

```
Xtr, Xts, ytr, yts = train_test_split(X, y, 0.25)

mu, V = PCA(Xtr)
loss = []
for i in range(X.shape[1]):

    Xtr_tran = (Xtr - mu).dot(V[:,:i+1])
    Xts_tran = (Xtr - mu).dot(V[:,:i+1])
    clf = Classifier()
    clf.fit(X_tran, ytr)
    yhat = clf.predict(Xts_tran)
    loss.append(np.sum((yts-yhat)**2))

print(f"PCA should use {np.argmin(loss)+1} as the componets number")
```

4

```
Y = reshape(X, shape)
pca = PCA(n_components=5)
pca.fit(Y[:500])
Z = pca.transform(Y[500:])
Yhat = pca.inverse_transform(Z)
```

5

```
U, s, Vtr = svd(X, full_matrices=False)
PCs = Vtr
mean = x.mean(axis=0)
lam = s**2
lam_sum = np.sum(lam)
accu = 0
for index, l in enumerate(lam):
    accu += l
    if accu/lam_sum > 0.9:
        break
pc = Vtr[:i]
Xhat = X.dot(pc)

```