1.

(a) Xmean = np.mean(X,0) = [1.5 2.5 3]

(b) Xs = X – Xmean

$$Q = \frac{1}{4} * \text{np.dot(Xs.T, Xs)} = \begin{matrix} 1.25 & 0.25 & -1.25 \\ 0.25 & 0.75 & 0.75 \\ -1.25 & 0.75 & 2.75 \end{matrix}$$

(c)w, v = np.linalg.eig(Q)

w = [3.56166464, 1.1733803 , 0.01495506]

v = [[-0.45056922, -0.66677184, -0.59363515],

[0.19247228, -0.72187235, 0.66472154],

[0.87174641, -0.18524476, -0.45358856]]

(d) coefficient = np.dot(Xs, v)

[[-2.95145599, -0.17610969, -0.0888421],

[1.37104342, -1.69406159, 0.0198819],

[-0.30682473, 0.78694448, 0.19125108],

[1.8872373 , 1.0832268 , -0.12229089]])

(e) Xhat = X_mean + np.dot(coefficient, v.T)

[3.0000000e+00, 2.0000000e+00, 1.0000000e+00],

[2.0000000e+00, 4.0000000e+00, 5.0000000e+00],

[1.0000000e+00, 2.0000000e+00, 3.0000000e+00],

[8.8817842e-16, 2.0000000e+00, 5.0000000e+00]]

(f) Xhat = X_mean + np.dot(coefficient[:,:2], v[:,:2].T)

[[2.94726021, 2.05905526, 0.95970224],

[2.0118026 , 3.98678407, 5.0090182],

[1.11353336, 1.87287129, 3.0867493],

[-0.07259617, 2.08128939, 4.94453025]]

(g) np.sum(coefficient[:,2]**2) == np.sum(np.sum((X-Xhat)**2, 1))

2.

(a) $\alpha_1 = v_1^T(x-\mu) = \frac{1}{\sqrt{2}}(1,1,0)(1,3,2) = 2\sqrt{2}$

$\alpha_2 = v_2^T(x-\mu) = \frac{1}{\sqrt{2}}(1,-1,0)(1,3,2) = -\sqrt{2}$

(b)$\hat{x} = \mu + 2\sqrt{2}\alpha_1 - \sqrt{2}\,\alpha_2$ = (2,3,0)

(c) approximation error = 4

3.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)
mu, V = PCA(X_train)
ncomp = V.shape[0]
acc = []
for n in range(n_components):
    clf = Classifier()
    Z = V[:n]
    clf.fit(Z,y)
    yhat = clf.predit(X_test)
    acc.append(np.mean((y_test - yhat)**2))
index = np.argmax(acc)
opt_num = index + 1
```

4.

```
Y = reshape(X, (1000, 784))
pca = PCA(n_components = 5)
pca.fit(Y[:500,:])
Z = pca.transform(Y[500:,:])
yhat = pca.inverse_transform(Z)
```

5.

```
Xmean = np.mean(X,0)
Xs = X - Xmean[None,:]
U,S, Vtr = svd(Xs, full_matrices = False)
lam = S**2
pov = np.cumsum(lam) / np.sum(lam)
numbers = np.where(pov > 0.9)[0][0] + 1
Xhat = (U[:,:numbers] * S[None, :numbers]).dot(Vtr[:numbers,:]) + Xmean
```