A6. Principal Component Analysis = PCA

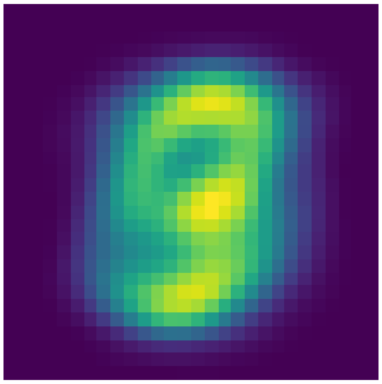
In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import random as rand
import scipy.optimize
from scipy.optimize import minimize
import scipy.linalg as la
from scipy.optimize import LinearConstraint
from scipy.optimize import NonlinearConstraint
```

In [2]:
```
import scipy.linalg as la
from scipy.optimize import LinearConstraint
from mnist import MNIST
mndata = MNIST("./data/")
X_train, labels_train = map(np.array, mndata.load_training())
X_test, labels_test = map(np.array, mndata.load_testing())
X_train = X_train/255.0
X_test = X_test/255.0
```

In [3]:
```
n,d=np.shape(X_train)
one=np.ones(n)
muu=np.matmul(X_train.T,one)/n

new_mu=np.matmul(one.reshape(-1,1),muu.reshape(1,-1))
sigma=np.matmul((X_train-new_mu).T,(X_train-new_mu))/n

n=X_train.shape[0]
mu=(1/n)*X_train.T@np.ones(n)
Big_Sig=((X_train-np.outer(np.ones(n), mu)).T@(X_train-np.outer(np.ones(n), mu)))/n
sigma=Big_Sig
eigenvalue, eigenvector=np.linalg.eig(Big_Sig)


eigvals,eigvecs=np.linalg.eig(Big_Sig)

'''to get rid of coplexity of the number'''
eigvals=np.real(eigvals)
eigvecs=np.real(eigvecs)
muu=mu
```

Average= Mu; one point is that every estimation has this part, so at the low numbers for k we see a picture super close to this (average/mu). Means the best guess is just the average of all pictures. The best signal to sent, if we want to minimize the size of message.

In [98]:
```
x_shape=np.reshape(mu,(28,28))
plt.imshow(x_shape)
plt.axis('off')
plt.show()
```



A6. a)

In [115]:
```
print('The 1st largest eigenvalue',eigvals[1-1])
print('The 2nd largest eigenvalue',eigvals[2-1])
print('The 10th largest eigenvalue',eigvals[10-1])
print('The 30th largest eigenvalue',eigvals[30-1])
print('The 50th largest eigenvalue',eigvals[50-1])
print('The summation of all eigenvalue is',sum(eigvals))
```

```
The 1st largest eigenvalue 5.116787728342082
The 2nd largest eigenvalue 3.7413284788648165
The 10th largest eigenvalue 1.2427293764173324
The 30th largest eigenvalue 0.3642557202788923
The 50th largest eigenvalue 0.16970842700672842
The summation of all eigenvalue is 52.725035495126946
```

In [6]:
```
def k_eigvecs(k):
    return eigvecs[None:k]
```

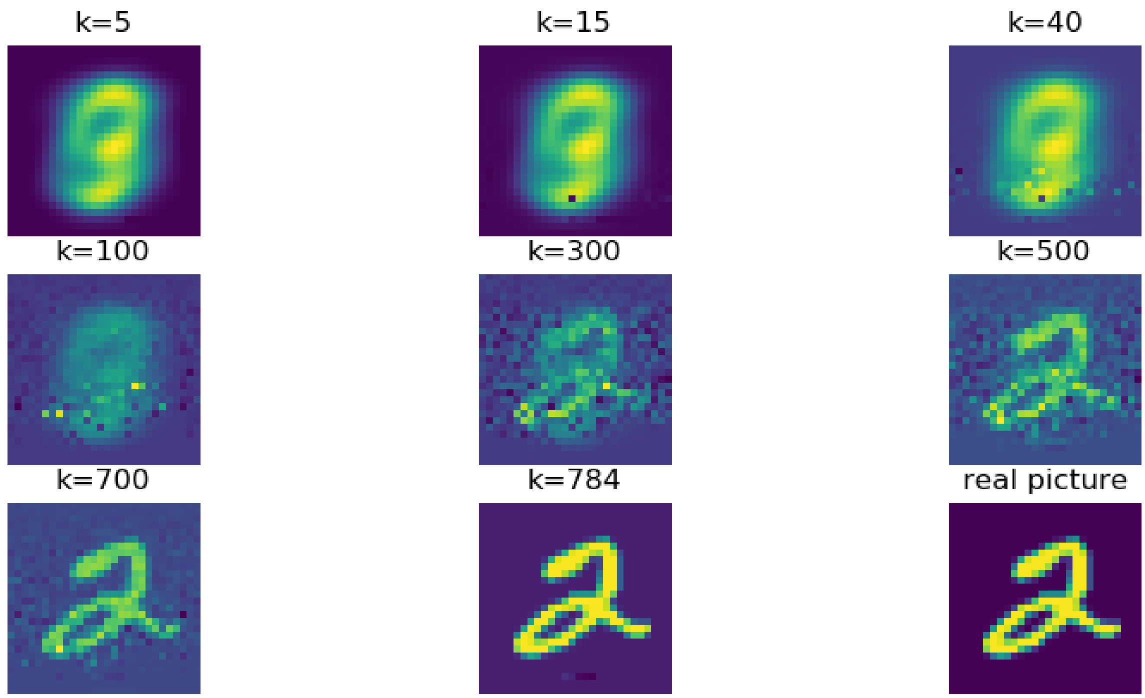
In [7]:
```
def new_PCA_project(x,k):
    return muu+(x-muu)@k_eigvecs(k).T@k_eigvecs(k)
```

A6. e)

approximation imoroves by adding more eigenvectors

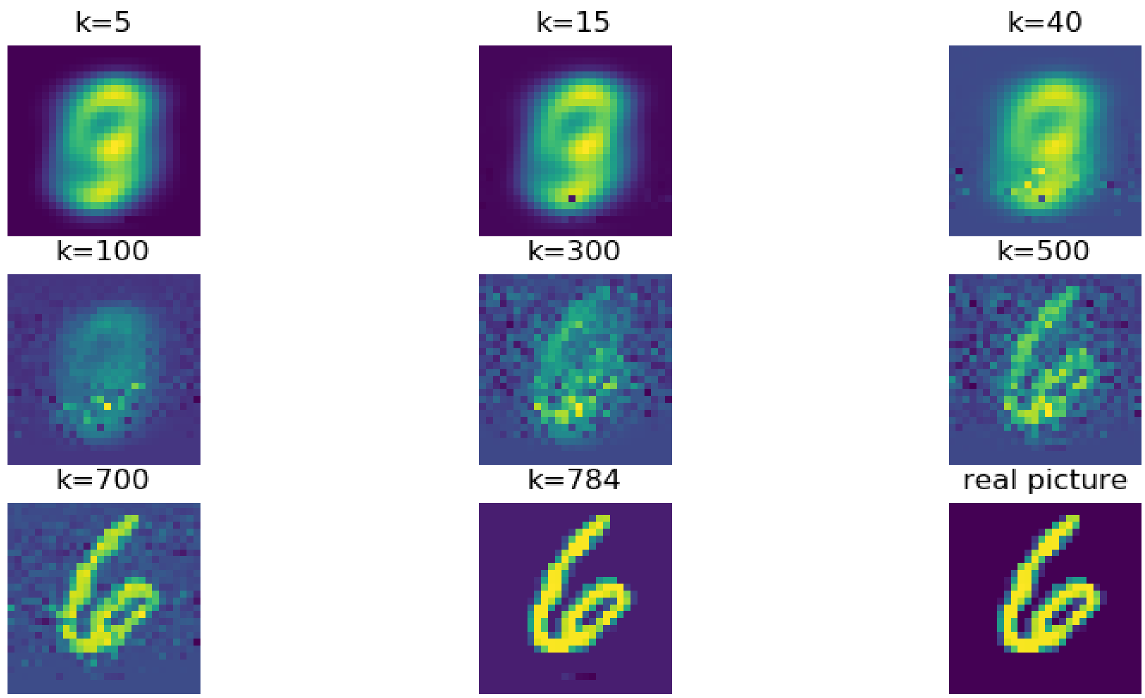
In [146]:

```
j=np.where(labels_train==2)[0][0]
plt.figure(figsize=(10,5),dpi=100)
ii=0
text=['k=5','k=15','k=40','k=100','k=300','k=500','k=700','k=784']
k=[5, 15, 40,100,300,500,700, 28*28]
for i in range(8):
    ii+=1
    plt.subplot(3, 3, ii)
    plt.title(text[i])
    x_shape=np.reshape(new_PCA_project(X_train[j],k[i]),(28,28))
    plt.imshow(x_shape)
    plt.axis('off')

plt.subplot(3, 3, 9)
plt.title('real picture')
x_shape=np.reshape(X_train[j],(28,28))
plt.imshow(x_shape)
plt.axis('off')

plt.show()
```



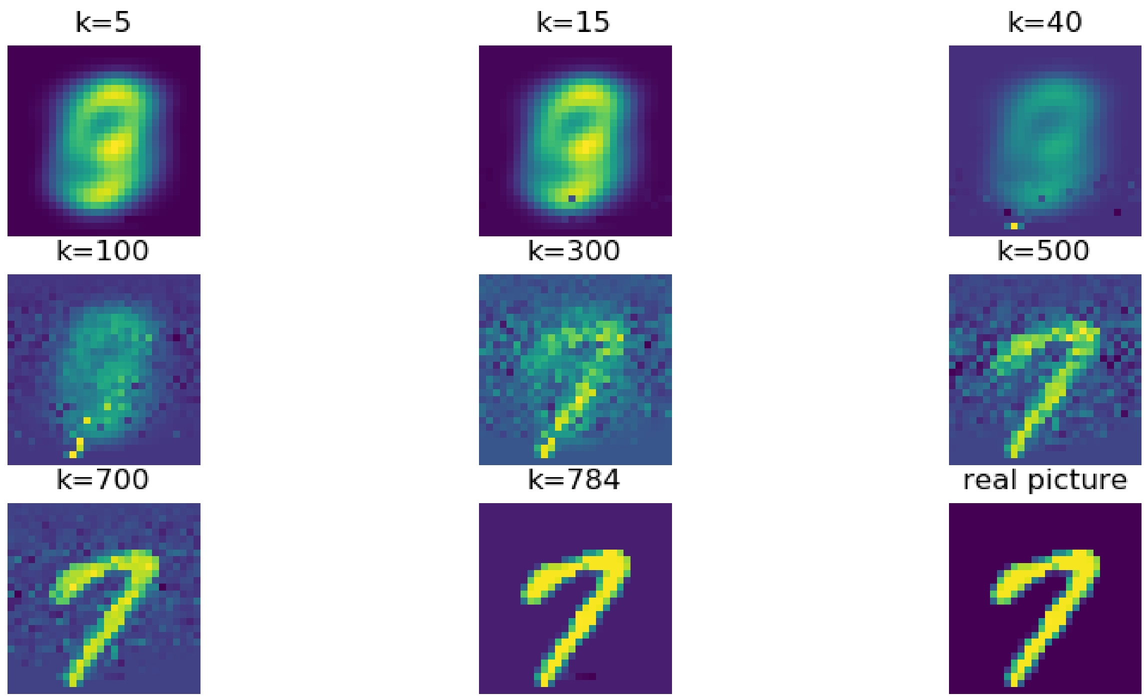
In [147]:

```
j=np.where(labels_train==6)[0][0]
plt.figure(figsize=(10,5),dpi=100)
ii=0
text=['k=5','k=15','k=40','k=100','k=300','k=500','k=700','k=784']
k=[5, 15, 40,100,300,500,700, 28*28]
for i in range(8):
    ii+=1
    plt.subplot(3, 3, ii)
    plt.title(text[i])
    x_shape=np.reshape(new_PCA_project(X_train[j],k[i]),(28,28))
    plt.imshow(x_shape)
    plt.axis('off')

plt.subplot(3, 3, 9)
plt.title('real picture')
x_shape=np.reshape(X_train[j],(28,28))
plt.imshow(x_shape)
plt.axis('off')

plt.show()
```



In [145]:

```python
j=np.where(labels_train==7)[0][0]
plt.figure(figsize=(10,5),dpi=100)
ii=0
text=['k=5','k=15','k=40','k=100','k=300','k=500','k=700','k=784']
k=[5, 15, 40,100,300,500,700, 28*28]
for i in range(8):
    ii+=1
    plt.subplot(3, 3, ii)
    plt.title(text[i])
    x_shape=np.reshape(new_PCA_project(X_train[j],k[i]),(28,28))
    plt.imshow(x_shape)
    plt.axis('off')

plt.subplot(3, 3, 9)
plt.title('real picture')
x_shape=np.reshape(X_train[j],(28,28))
plt.imshow(x_shape)
plt.axis('off')

plt.show()
```



In [148]:

```
def reconstruction_error(k):

    n=1
    train_e=0

    n1=int(np.shape(X_train)[0]/n)
    for i in range(n1):
        train_e+=np.linalg.norm(new_PCA_project(X_train[i],k)-X_train[i])**2

    test_e=0
    n2=int(np.shape(X_test)[0]/n)
    for i in range(n2):
        test_e+=np.linalg.norm(new_PCA_project(X_test[i],k)-X_test[i])**2

    return train_e/n1,test_e/n2
```

In [149]:

```
reconstruction_error(10)
```

Out[149]:

```
(52.66800352142152, 52.84524106822591)
```

In [150]:

```
reconstruction_error(1)
```

Out[150]:

```
(52.7019354788974, 52.87343386035073)
```

reconstruction error goes down by using more dimensions

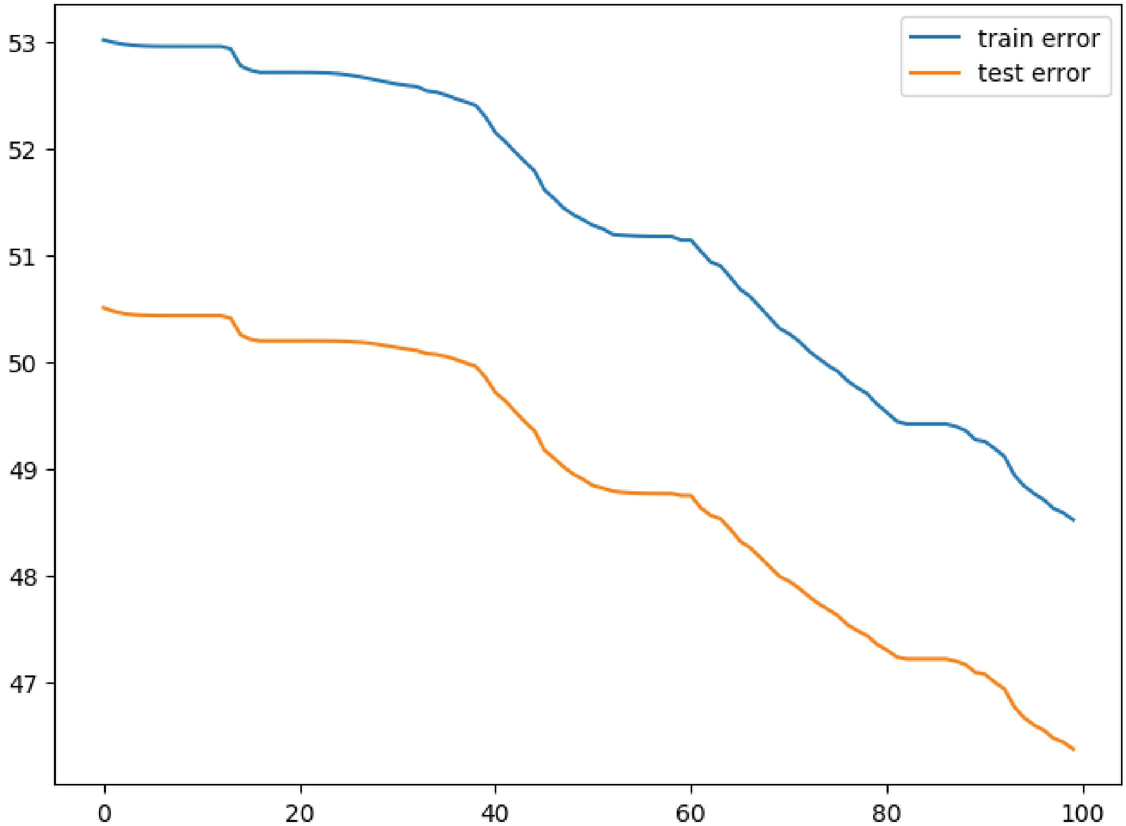
A6. c)

In [102]:

```
n=100
pca_k_range=list(range(0,n))
pca_y=np.zeros(n)
train_e=np.zeros(n)
test_e=np.zeros(n)

for i in range(n):
    pca_y[i]=1-sum(eigvals[None:i])/(sum(eigvals))
    train_e[i],test_e[i]=reconstruction_error(pca_k_range[i])
```

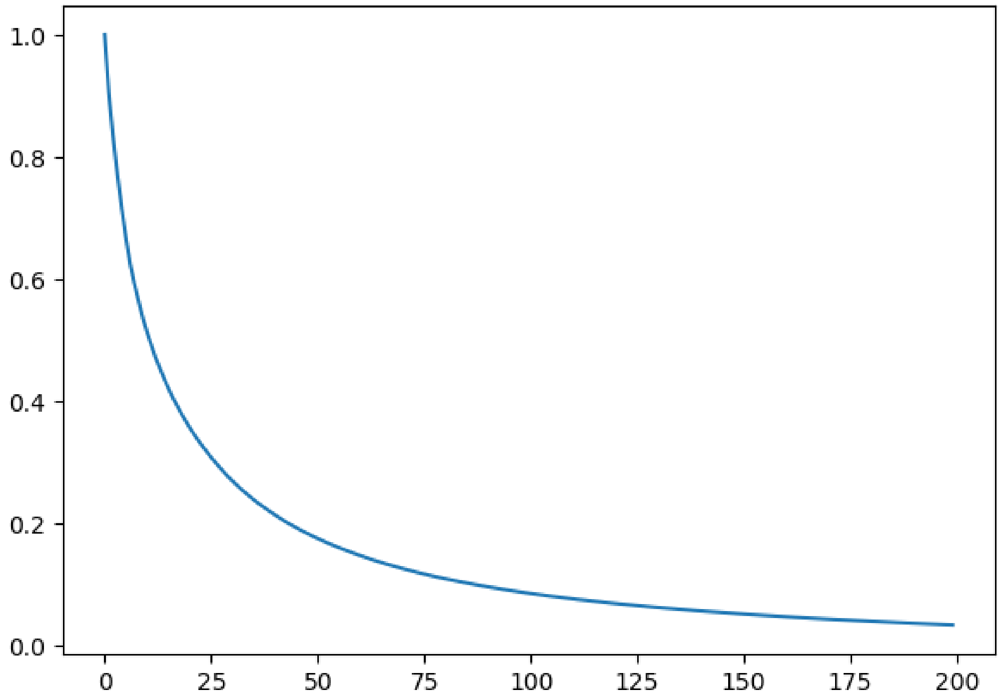
In [93]:

```
plt.figure(figsize=(8,6),dpi=100)
#plt.plot(pca_k_range,pca_y)
plt.plot(pca_k_range,train_e,label='train error')
plt.plot(pca_k_range,test_e,label='test error')
plt.legend(loc="best")
plt.show()
```



In [81]:

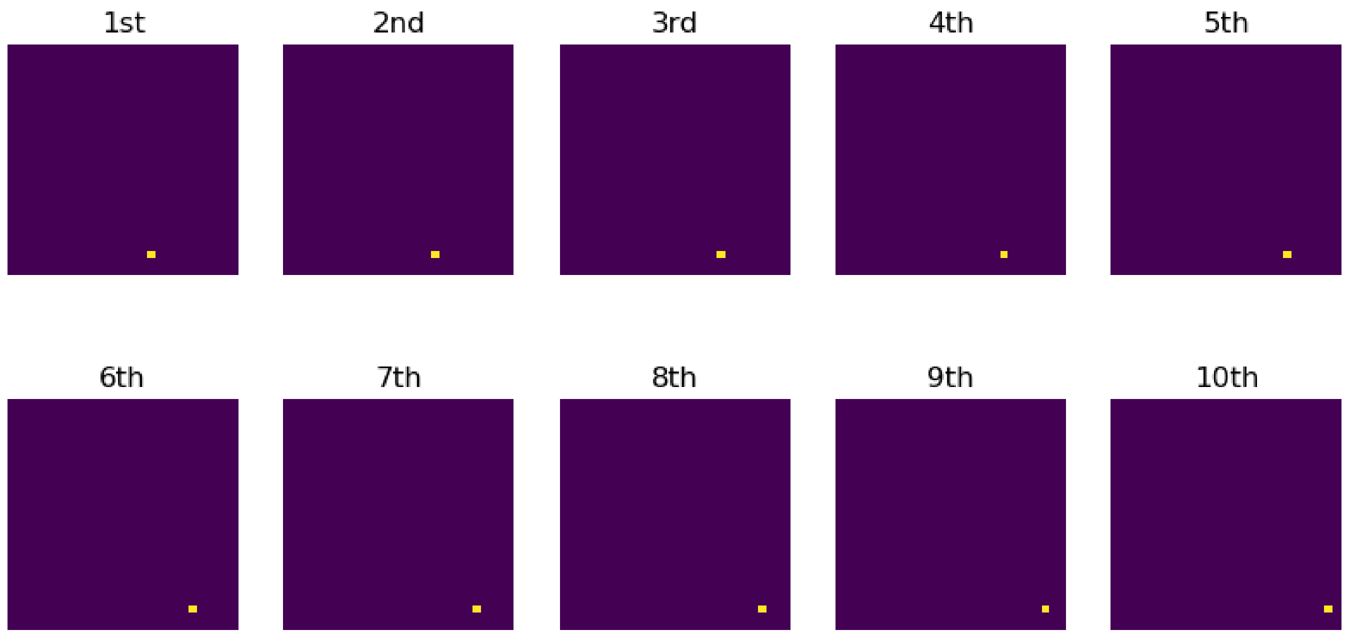
```
plt.figure(figsize=(7,5),dpi=100)
plt.plot(pca_k_range,pca_y)
#plt.legend(loc="best")
plt.show()
```



A6. d) the first 10 eigenvectors

In [114]:

```
plt.figure(figsize=(10,5),dpi=100)
ii=0
text=['1st','2nd','3rd','4th','5th','6th','7th','8th','9th','10th']
for i in range(10):
    ii+=1
    plt.subplot(2, 5, ii)
    plt.title(text[i])
    x_shape=np.reshape(eigvecs[ii-1],(28,28))
    plt.imshow(x_shape)
    plt.axis('off')
plt.show()
```



In [ ]:

```

```