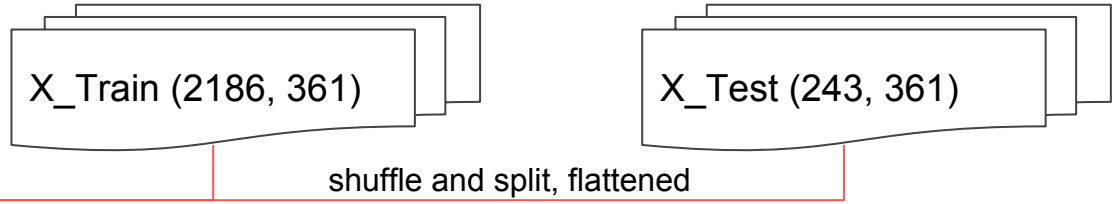
Eigenfaces

Image processing, Retrieval, and Analysis II
Project 02

Split up Train / Test Data

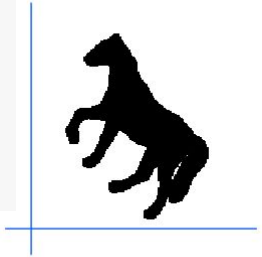
2429 cbcl faces
19 x 19 (361)

```
allSetIndex = np.arange(1, 2430)
np.random.shuffle(allSetIndex)

trainSetIndex = allSetIndex[243:]
testSetIndex = allSetIndex[:243]

x_train = np.zeros((trainSetIndex.size, 19*19))
for idx, val in enumerate(trainSetIndex):
    filename = 'cbcl-faces/face' + format(val, '05d') + '.pgm'
    x_train[idx] = readImage(filename).flatten()
```

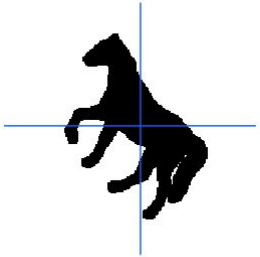


Zero Mean Dataset

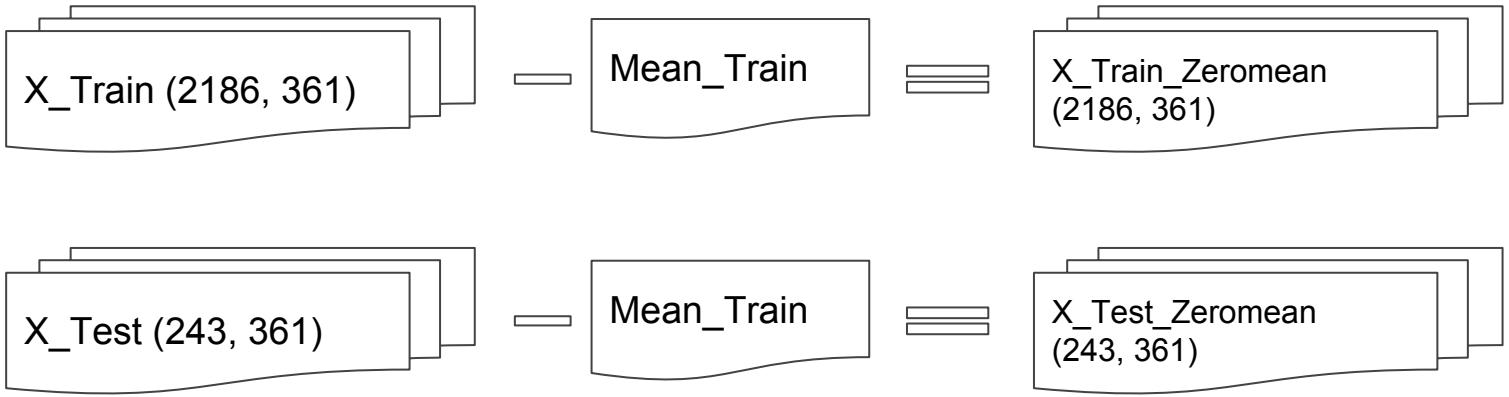
```
mean_train = np.mean(x_train, axis=0)

x_train_zeromean = np.subtract(x_train, mean_train)
x_test_zeromean = np.subtract(x_test, mean_train)
```



original data



subtract μ



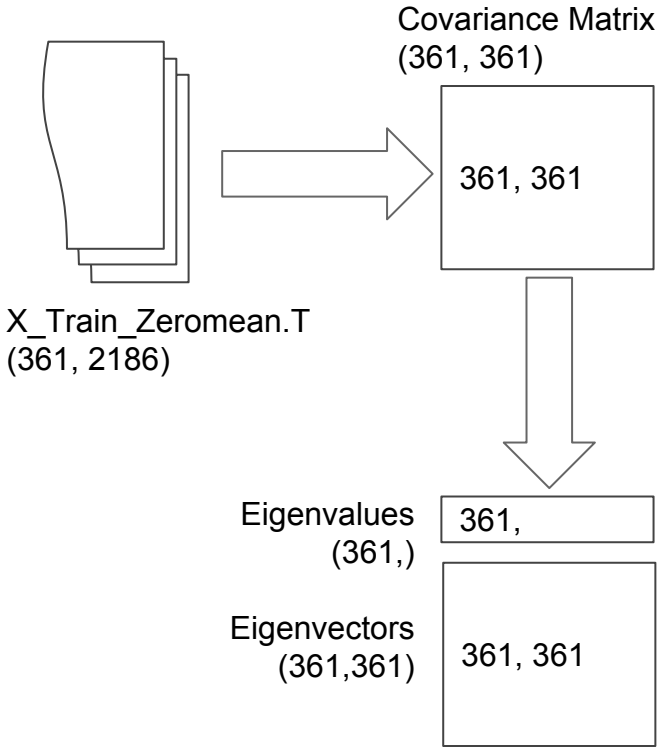
Zero Mean Dataset

X_Train[:16]

X_Train_Zeromean[:16]

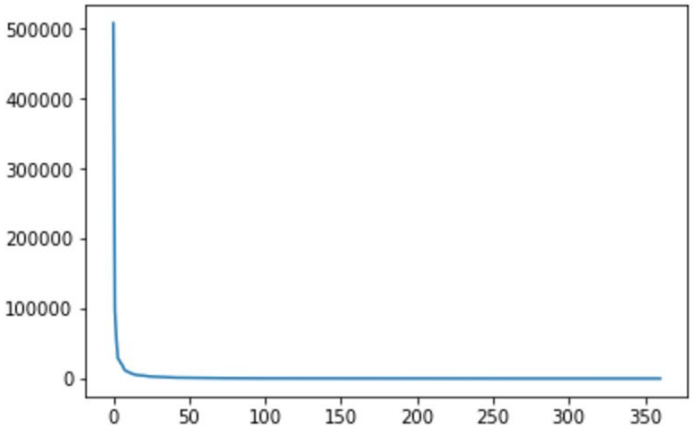
Covariance Matrix

```
# compute sample covariance matrix
C = np.cov(x_train_zeromean.T)

# compute eigenvalues/eigenvectors using eigh
evalsh, evectsh = la.eigh(C)

inds = np.argsort(evalsh)[::-1]
evalsh = evalsh[inds]
evectsh = evectsh[:,inds]
```



Covariance Matrix Spectrum

Plot eigenvalues of C in desceding order

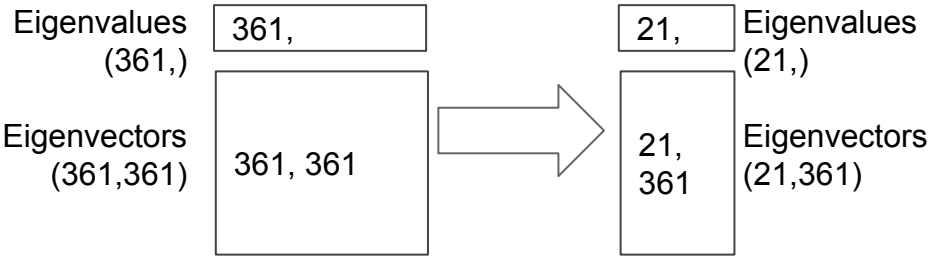


Determine the smallest eigenvalue s.t.

$$\frac{\sum_{j=1}^{k} \lambda_j}{\sum_{j=1}^{m} \lambda_j} \geqslant 0.9$$

```
evalshTotal = np.sum(evalsh)
partialSum = 0
k = 0
for idx, val in enumerate(evalsh):
    partialSum += val
    if (partialSum/evalshTotal >= 0.9):
        k = idx
        break
evalsh_k = evalsh[:k+1]
evectsh_k = evectsh.T[:k+1]
```

k = 21

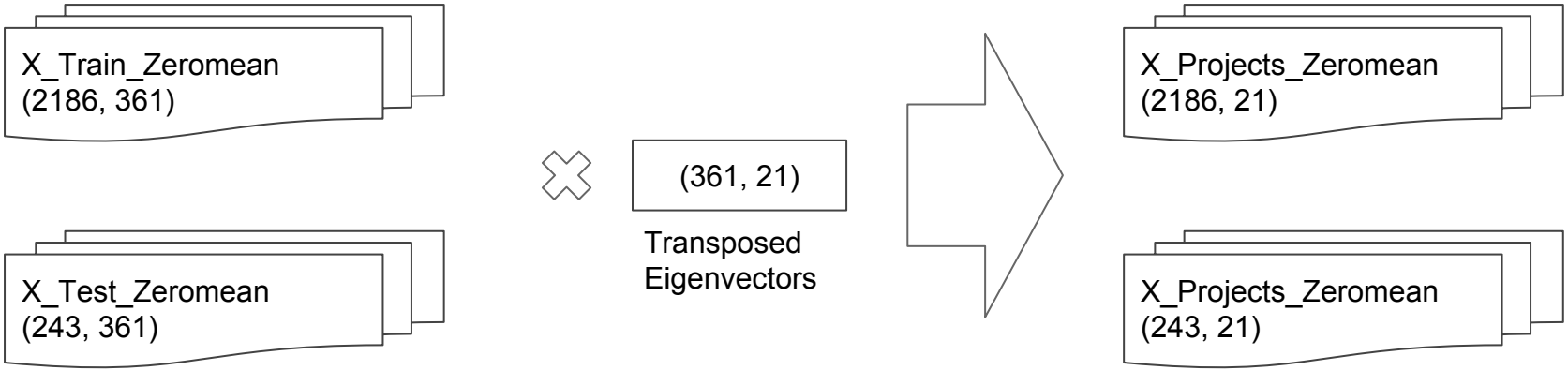


First K eigenvectors

```
for i in range(len(evectsh_k)):
    if (i % 9 == 0) :
        plt.figure()
    plt.subplot(3, 3, (i%9)+1)
    plt.axis('off')
    plt.imshow(evectsh_k[i].reshape((19,19)), cmap='gray')

plt.show()
```

Project data into subspace

```
x_projects_train = np.dot(x_train_zeromean, evectsh_k.T)
x_projects_test = np.dot(x_test_zeromean, evectsh_k.T)
```



Compute Distances

```
def euclideanDistance(p,q):
    p = np.asarray(p).flatten()
    q = np.asarray(q).flatten()
    return np.sqrt(np.sum(np.power((p - q), 2)))
```

```
distances_original = np.zeros((10, trainSetIndex.size))
nn_original = np.zeros((10))
for i in range(10):
    dist = np.zeros((trainSetIndex.size))
    for j in range(trainSetIndex.size):
        dist[j] = euclideanDistance(x_test[i], x_train[j])
    inds = np.argsort(dist)[::-1]
    dist = dist[inds]
    distances_original[i] = dist
    nn_original[i] = inds[-1]
```

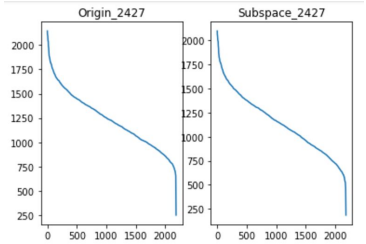
- For 10 sampled test images,
  compute distances to all training images.
  Distances are sorted in descending order.

```
distances_subspace = np.zeros((10, trainSetIndex.size))
nn_subspace = np.zeros((10))
for i in range(10):
    dist = np.zeros((trainSetIndex.size))
    for j in range(trainSetIndex.size):
        dist[j] = euclideanDistance(x_projects_test[i], x_projects_train[j])
    inds = np.argsort(dist)[::-1]
    distances_subspace[i] = dist[inds]
    nn_subspace[i] = inds[-1]
```

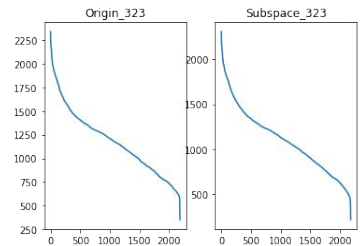
- For same 10 test images,
  compute distances to all training images in the subspace.
  Distances are sorted in descending order.

Distances and Nearest neighborhood

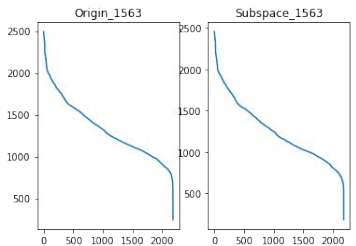
Comparing computed distances.
Left - in Original space, Right - in Subspace



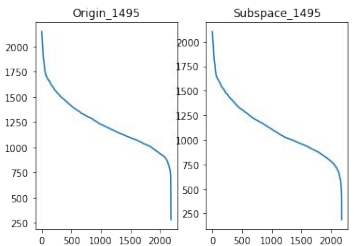
NN: 1626 1626



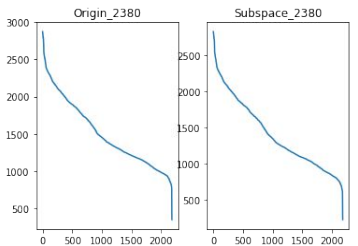
1959 1959



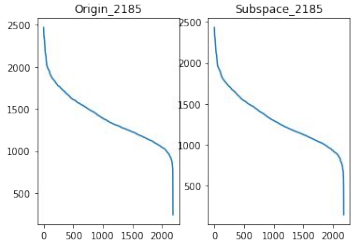
993 993



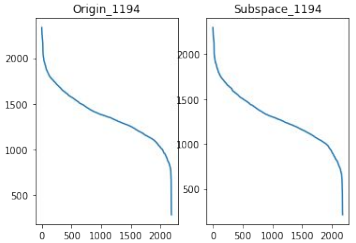
784 784



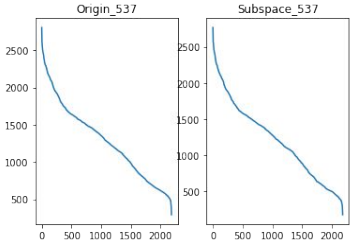
1167 1167



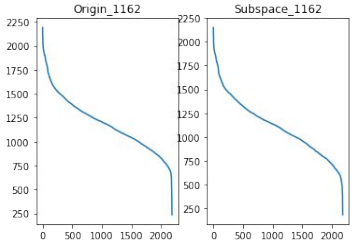
NN: 1310 1310



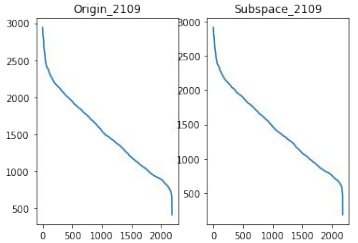
19 495



14 14



750 750



292 292