Modules and Datasets

```
import numpy as np
```

```
from sklearn.datasets import fetch_lfw_people
import matplotlib.pyplot as plt
import matplotlib.cm as cm
```

```
people = fetch_lfw_people(min_faces_per_person=20, resize=0.7)
images_shape = people.images[0].shape
```

```
    Downloading LFW metadata: https://ndownloader.figshare.com/files/5976012
    Downloading LFW metadata: https://ndownloader.figshare.com/files/5976009
    Downloading LFW metadata: https://ndownloader.figshare.com/files/5976006
    Downloading LFW data (~200MB): https://ndownloader.figshare.com/files/5976015
```

```
fig, axes = plt.subplots(2, 5, figsize=(15, 8),
                         subplot_kw={'xticks': (), 'yticks': ()})
for target, image, ax in zip(people.target, people.images, axes.ravel()):
  ax.imshow(image, cmap=cm.gray)
  ax.set_title(people.target_names[target])
```

Part 1: Theory

```
# Data
data = np.array([[-2, 1],
                 [-5, -4],
                 [-3, 1],
                 [0, 3],
                 [-8, 11],
                 [-2, 5],
                 [1, 0],
                 [5, -1],
                 [-1, -3],
                 [6, 1]])


# Standardize Data
x1 = data[:, 0]
x2 = data[:, 1]


# Normalize x1

## Calculate mean
x1mean = x1.sum()/len(x1)

## Calculate standard deviation
zeroxmag = ((x1-x1mean)**2).sum()
x1std = np.sqrt((zeroxmag)/(len(x1)-1))

# Normalize
x1norm = (x1 - x1mean)/x1std


# Normalize x2

## Calculate mean
x2mean = x2.sum()/len(x2)

## Calculate standard deviation
zeroxmag = ((x2-x2mean)**2).sum()
x2std = np.sqrt((zeroxmag)/(len(x2)-1))

# Normalize
x2norm = (x2 - x2mean)/x2std


# Covariance Matrix
X = np.array([x1norm, x2norm]).T
covX = X.T @ X
covX = covX/len(X)
```

```
# Characteristic Equation
## l**2 - 1.8l**2 + 0.675

a = 1
b = -1.81
c = 0.675

## Eigenvalues
l1 = (-b+np.sqrt((b**2)-4*a*c))/(2*a)
l2 = (-b-np.sqrt((b**2)-4*a*c))/(2*a)

## Eigenvectors
v1 = np.array([0.70710678, 0.70710678]).T
v2 = np.array([-0.70710678, 0.70710678]).T


# Project data onto largest term
data_projected = data @ v1
print(data_projected)
```

```
[-0.70710678 -6.36396102 -1.41421356  2.12132034  2.12132034  2.12132034
  0.70710678  2.82842712 -2.82842712  4.94974746]
```

Part 2: Dimensionality Reduction with PCA

Setup

```
# Data Check
print("people.images.shape: {}".format(people.images.shape))
print("Number of classes: {}".format(len(people.target_names)))
```

```
people.images.shape: (3023, 87, 65)
Number of classes: 62
```

```
# get target counts
counts = np.bincount(people.target)

# print counts with target names
for i, (count, name) in enumerate(zip(counts, people.target_names)):
  print("{0:25} {1:3}".format(name, count), end='  ')
  if (i+1) % 3 == 0:
    print()
```

```
Alejandro Toledo          39  Alvaro Uribe              35  Amelie Mauresmo
Andre Agassi              36  Angelina Jolie            20  Ariel Sharon
```

```
Arnold Schwarzenegger     42  Atal Bihari Vajpayee      24  Bill Clinton
Carlos Menem              21  Colin Powell             236  David Beckham
Donald Rumsfeld          121  George Robertson          22  George W Bush
Gerhard Schroeder        109  Gloria Macapagal Arroyo   44  Gray Davis
Guillermo Coria           30  Hamid Karzai              22  Hans Blix
Hugo Chavez               71  Igor Ivanov               20  Jack Straw
Jacques Chirac            52  Jean Chretien             55  Jennifer Aniston
Jennifer Capriati         42  Jennifer Lopez            21  Jeremy Greenstock
Jiang Zemin               20  John Ashcroft             53  John Negroponte
Jose Maria Aznar          23  Juan Carlos Ferrero       28  Junichiro Koizumi
Kofi Annan                32  Laura Bush                41  Lindsay Davenport
Lleyton Hewitt            41  Luiz Inacio Lula da Silva 48  Mahmoud Abbas
Megawati Sukarnoputri     33  Michael Bloomberg         20  Naomi Watts
Nestor Kirchner           37  Paul Bremer               20  Pete Sampras
Recep Tayyip Erdogan      30  Ricardo Lagos             27  Roh Moo-hyun
Rudolph Giuliani          26  Saddam Hussein            23  Serena Williams
Silvio Berlusconi         33  Tiger Woods               23  Tom Daschle
Tom Ridge                 33  Tony Blair               144  Vicente Fox
Vladimir Putin            49  Winona Ryder              24
```

```
# Take  up to 50 imagews of each person
mask = np.zeros(people.target.shape, dtype=np.bool)
for target in np.unique(people.target):
  mask[np.where(people.target == target)[0][:50]] = 1

X_people = people.data[mask]
y_people = people.target[mask]

# scale between 0 and 1 for numeric stabiity
X_people =  X_people/255


# Classify with KNN
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_people, y_people, stratify=y_people, ra

# Build and fit classifier
knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(X_train, y_train)
print("Test set score of 1-nn: {:.2f}".format(knn.score(X_test, y_test)))
```

```
Test set score of 1-nn: 0.23
```

KNN

```
def ssd(xtrain, xtest): return ((xtrain - xtest)**2).sum()
```

```
def knn(X_train, y_train, X_test, k=1):
  y_pred = []

  for i in range(len(X_test)):
    neighbors = []

    for j in range(len(X_train)):
      neighbors.append(ssd(X_train[j], X_test[i]))

    nearest_idx = np.argmin(neighbors)
    y_pred.append(y_train[nearest_idx])

  return y_pred

def acc(y_test, y_pred):
  hits = 0

  for i in range(len(y_test)):
    if y_test[i] == y_pred[i]:
      hits = hits+1

  return hits/len(y_test)


y_pred = knn(X_train, y_train, X_test, k=1)


accuracy = acc(y_test, y_pred)
print(accuracy)
```

```
0.23255813953488372
```

PCA

```
# Standardize Data
std = np.std(X_train, axis=0)
mean = np.mean(X_train, axis=0)

X_train_norm = (X_train-mean)/std
X_test_norm = (X_test-mean)/std


# PCA
covX = X_train_norm.T @ X_train_norm
val, vec = np.linalg.eig(covX)


# Get top 100 vectors
indices = (-val).argsort()[:100]
projection = vec[indices]
```

```
# Project Data
X_train_proj = X_train @ projection.T
X_test_proj = X_test @ projection.T


# Predict on Projection
y_pred_proj = knn(X_train_proj, y_train, X_test_proj, k=1)
accuracy = acc(y_test, y_pred_proj)
print(accuracy)
```

```
0.1937984496124031
```

PCA + Whitening

```
# New Eigenvectors
diagval = np.diag(val[indices]**(-1/2))
w_train = (diagval @ projection @ X_train.T).T
w_test = (diagval @ projection @ X_test.T).T


# Predict on Projection
y_pred_white = knn(w_train, y_train, w_test, k=1)
accuracy = acc(y_test, y_pred_white)
print(accuracy)
```

```
0.18023255813953487
```

2D PCA

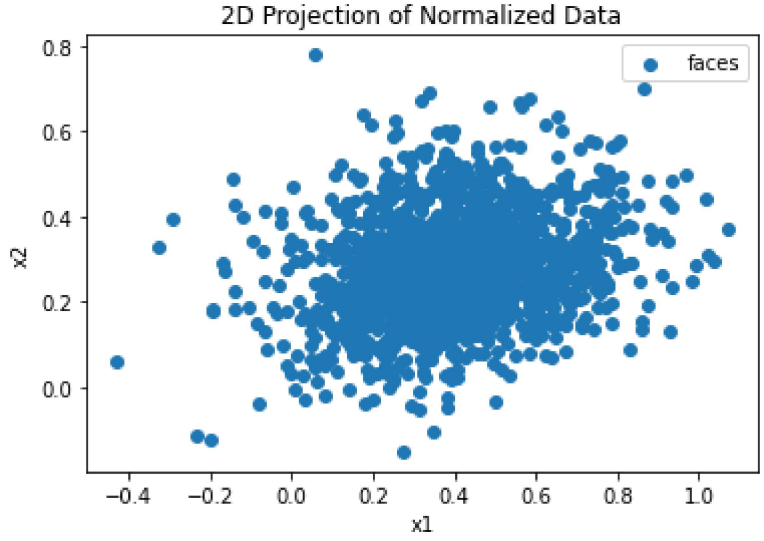
```
# Get top 2 vectors
indices = (-val).argsort()[:2]
projection = vec[indices]


# Project Data
X_train_proj = X_train @ projection.T
X_test_proj = X_test @ projection.T


X_train_proj.shape
```

```
(1547, 2)
```

```
# Plot New Coordinates
plt.scatter(X_train_proj[:, 0], X_train_proj[:, 1], label='faces')
plt.title('2D Projection of Normalized Data')
plt.xlabel('x1')
```

```
plt.ylabel('x2')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7fa401c60b10>
```



Part 3: Eigenfaces

```
# Standardize
std = np.std(X_train, axis=0)
mean = np.mean(X_train, axis=0)

X_train_norm = (X_train-mean)/std
X_test_norm = (X_test-mean)/std

# PCA
covX = X_train_norm.T @ X_train_norm
val, vec = np.linalg.eig(covX)

# Find PC1 and PC2
indices = (-val).argsort()[:1]
projection = vec[indices]

# Project Data
X_train_proj = X_train @ projection.T
```

Max and Min PCA

```
P1max = X_train_proj[:, 0].argmax()
P1min = X_train_proj[:, 0].argmin()

P2max = X_train_proj[:, 1].argmax()
P2min = X_train_proj[:, 1].argmin()
```

```
fig, axes = plt.subplots(1, 2, figsize=(15, 8),
                         subplot_kw={'xticks': (), 'yticks': ()})

axes[0].imshow(X_train[P1max].reshape(87, 65), cmap=cm.gray)
axes[1].imshow(X_train[P1min].reshape(87, 65), cmap=cm.gray)
```

```
<matplotlib.image.AxesImage at 0x7f0860564390>
```

```
fig, axes = plt.subplots(1, 2, figsize=(15, 8),
                         subplot_kw={'xticks': (), 'yticks': ()})

axes[0].imshow(X_train[P2max].reshape(87, 65), cmap=cm.gray)
axes[1].imshow(X_train[P2min].reshape(87, 65), cmap=cm.gray)
```

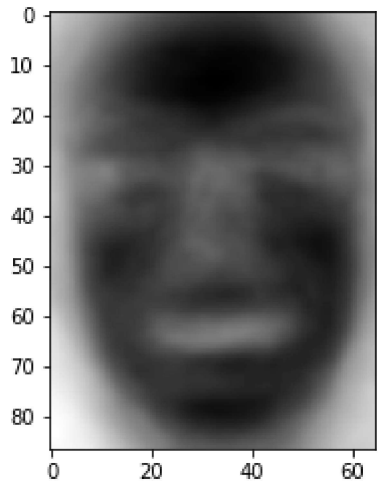
```
Text(0.5, 1.0, 'Jennifer Capriati')
```

Principal 1 Seems to be tracking whether or not there is glasses on the person, while Principal 2 seems to be tracking the mouth position.

1D PCA Visualization


```
# Find PC1 and PC2
indices = (-val).argsort()[:1]
projection = vec[indices]

# Project Data
X_train_proj = X_train @ projection.T


PC1img = (X_train_proj * vec.T[indices])


plt.imshow(PC1img[0].reshape(87, 65), cmap=cm.gray)
```

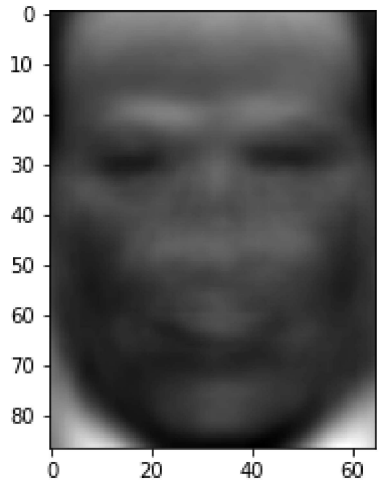
```
<matplotlib.image.AxesImage at 0x7f08600cbd50>
```




```
# Principal Reconstruction
```

```
# Principal Reconstruction
PC1img = ((X_train_proj[0,:] * vec.T[indices])+mean)*std
plt.imshow(PC1img[0].reshape(87, 65), cmap=cm.gray)
```

```
<matplotlib.image.AxesImage at 0x7f08600b3690>
```



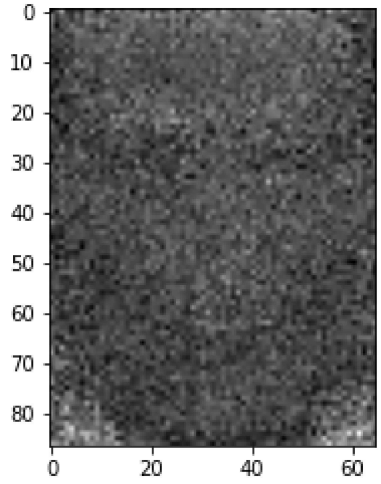
Top K Principal Components (188)

```
totalk = val.sum()
index_all = (-val).argsort()
part = 0

for i, index in enumerate(index_all):
  part = part + val[index]

  if part/totalk >= 0.95:
    print(i)
    break
```

```
188
```

```
topkvec = vec[index_all[:188]]
topkval = val[index_all[:188]]


# Project Data
X_train_proj = X_train @ topkvec.T


# Principal Reconstruction
PCkimg = ((topkvec.T @ X_train_proj[0,:])+mean)*std
plt.imshow(PCkimg.reshape(87, 65), cmap=cm.gray)
```

```
<matplotlib.image.AxesImage at 0x7f086048f0d0>
```



✓ 0s completed at 11:16 PM