InClass_Lab5

February 15, 2018

In [1]:
```
#1. IMPUTE MISSING VALUES
from numpy import nan
import numpy as np

X_train = ([[nan, 0, 3],
     [3, 7, 9],
     [3, 5, 2],
     [4, nan, 6],
     [8, 8, 1]])
X_test = ([[14, 16, -1],
           [nan, 8, -5]])
```

In [2]:
```
# "Train" imputer
from sklearn.preprocessing import Imputer

imputer = Imputer(strategy="median")
imputer.fit(X_train)
imputer.statistics_
```

Out[2]:
```
array([ 3.5,  6. ,  3. ])
```

In [3]:
```
X_train_fixed = imputer.transform(X_train)
X_train_fixed
```

Out[3]:
```
array([[ 3.5,  0. ,  3. ],
       [ 3. ,  7. ,  9. ],
       [ 3. ,  5. ,  2. ],
       [ 4. ,  6. ,  6. ],
       [ 8. ,  8. ,  1. ]])
```

In [5]:
```
X_test_fixed = imputer.transform(X_test)
X_test_fixed
```

Out[5]:
```
array([[ 14. ,  16. ,  -1. ],
       [  3.5,   8. ,  -5. ]])
```

In [8]:
```
# 2. Scaling numerical features
# Min-max scaling
from sklearn.preprocessing import MinMaxScaler
```

```
        mms = MinMaxScaler()
        mms.fit(X_train_fixed)
        print(mms.scale_)
        print(mms.min_)
```

```
[ 0.2    0.125  0.125]
[-0.6    0.    -0.125]
```

```
In [10]: X_train_norm = mms.transform(X_train_fixed)
         X_train_norm
```

```
Out[10]: array([[ 0.1  ,  0.   ,  0.25 ],
                [ 0.   ,  0.875,  1.   ],
                [ 0.   ,  0.625,  0.125],
                [ 0.2  ,  0.75 ,  0.625],
                [ 1.   ,  1.   ,  0.   ]])
```

```
In [11]: X_test_norm = mms.transform(X_test_fixed)
         X_test_norm
```

```
Out[11]: array([[ 2.2 ,  2.  , -0.25],
                [ 0.1 ,  1.  , -0.75]])
```

```
In [14]: # Standardization
         from sklearn.preprocessing import StandardScaler

         stdsc = StandardScaler()
         stdsc = stdsc.fit(X_train_fixed)
         print(stdsc.mean_)
         print(stdsc.scale_)
```

```
[ 4.3  5.2  4.2]
[ 1.88679623  2.78567766  2.92574777]
```

```
In [16]: X_train_std = stdsc.transform(X_train_fixed)
         X_test_std = stdsc.transform(X_test_fixed)
         X_train_std
```

```
Out[16]: array([[-0.42399915, -1.86669121, -0.41015156],
                [-0.68899862,  0.64616234,  1.64060622],
                [-0.68899862, -0.07179582, -0.75194452],
                [-0.15899968,  0.28718326,  0.61522733],
                [ 1.96099608,  1.00514142, -1.09373748]])
```

```
In [18]: # 3. Convert categorical variables to one-hot encodings
         from sklearn.preprocessing import OneHotEncoder
```

```
        # Train Data
        # 0 0 3
        # 1 1 0
        # 0 2 1
        # 1 0 2

        # Test data
        # 0 1 1

        enc = OneHotEncoder()
        data = [[0, 0, 3], [1, 1, 0], [0, 2, 1], [1, 0, 2]]
        enc.fit(data)
        encoding = enc.transform([[0, 1, 1]]).toarray()
        print("Test data: \n{}".format(encoding))
```

```
Test data:
[[ 1.  0.  0.  1.  0.  0.  1.  0.  0.]]
```

In [19]:
```
# 4. Dimensionality reduction using PCA
from sklearn.datasets import fetch_lfw_people

faces = fetch_lfw_people(min_faces_per_person=60)
```

```
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
```

In [22]:
```
print(faces.target_names)
print(faces.images.shape)
```

```
['Ariel Sharon' 'Colin Powell' 'Donald Rumsfeld' 'George W Bush'
 'Gerhard Schroeder' 'Hugo Chavez' 'Junichiro Koizumi' 'Tony Blair']
(1348, 62, 47)
```

In [24]:
```
from sklearn.decomposition import RandomizedPCA

pca = RandomizedPCA(150).fit(faces.data)
components = pca.transform(faces.data)
projected = pca.inverse_transform(components)
```

```
/home/nbuser/anaconda3_501/lib/python3.6/site-packages/sklearn/utils/deprecation.py:58: Deprecat
  warnings.warn(msg, category=DeprecationWarning)
```

In [29]:
```
# Plot results when we have all ~3000 pixels (top row)
# and when only using 150 components (bottom row)
import matplotlib.pyplot as plt
%matplotlib inline

fix, ax = plt.subplots(2, 10, figsize=(10, 2.5),
                       subplot_kw={'xticks': [], 'yticks': []},
                       gridspec_kw=dict(hspace=0.1, wspace=0.1))
for i in range(10):
    ax[0, i].imshow(faces.data[i].reshape(62,47), cmap='binary_r')
    ax[1, i].imshow(projected[i].reshape(62,47), cmap='binary_r')
```