Dimensionality reduction

PCA with NumPy

In [1]:

```
# Load NumPy package
import numpy as np
```

In [2]:

```
# Define data
x1 = np.array([1, 2, 4, 4, 6, 8])
x2 = np.array([2, 3, 3, 5, 5, 7])
x = np.column_stack([x1, x2])
x
```

Out[2]:

```
array([[1, 2],
       [2, 3],
       [4, 3],
       [4, 5],
       [6, 5],
       [8, 7]])
```

In [3]:

```
# Compute centered data
x_ = x - x.mean(axis=0)
x_
```

Out[3]:

```
array([[-3.16666667, -2.16666667],
       [-2.16666667, -1.16666667],
       [-0.16666667, -1.16666667],
       [-0.16666667,  0.83333333],
       [ 1.83333333,  0.83333333],
       [ 3.83333333,  2.83333333]])
```

In [4]:

```
# Compute covariance matrix C
C = np.cov(x_.T)
C
```

Out[4]:

```
array([[6.56666667, 4.36666667],
       [4.36666667, 3.36666667]])
```

In [5]:

```
# Compute eigenvalues S and eigenvectors U
[S, U] = np.linalg.eig(C)
```

In [6]:

```
# Sort eigenvalues and eigenvectors by descending eigenvalues
idx = S.argsort()[::-1]
S = S[idx]
U = U[:, idx]
```

In [7]:

```
# Get U_reduced
U_reduced = U[:, :1]
U_reduced
```

Out[7]:

```
array([[0.81976949],
       [0.57269362]])
```

In [8]:

```
# Compute q
q = np.dot(x_, U_reduced)
q
```

Out[8]:

```
array([[-3.8367729 ],
       [-2.44430979],
       [-0.8047708 ],
       [ 0.34061643],
       [ 1.98015542],
       [ 4.76508164]])
```

In [9]:

```
# Reconstruct objects
x_approx = np.dot(q, U_reduced.T)
x_approx
```

Out[9]:

```
array([[-3.14526936, -2.19729536],
       [-2.00377059, -1.39984062],
       [-0.65972655, -0.46088711],
       [ 0.27922696,  0.19506886],
       [ 1.623271  ,  1.13402237],
       [ 3.90626855,  2.72893185]])
```

In [10]:

```
# Compute average projection error
projection_error = np.sum((x_ - x_approx) ** 2) / x_.shape[0]
```

In [11]:

```
# Compute total variation in data
total_variation = np.sum(x_**2) / x_.shape[0]
```

In [12]:

```
# Fraction of variance explained by projection
frac_explained = 1 - (projection_error / total_variation)
frac_explained
```

Out[12]:

```
0.968177933056478
```

PCA with sklearn

In [13]:

```
# Import PCA algorithm from sklearn package
from sklearn.decomposition import PCA
```

In [14]:

```
# Define number of principal components
pca = PCA(n_components=1)
```

In [15]:

```
# Compute q
q = pca.fit_transform(x)
q
```

Out[15]:

```
array([[-3.8367729 ],
       [-2.44430979],
       [-0.8047708 ],
       [ 0.34061643],
       [ 1.98015542],
       [ 4.76508164]])
```

In [16]:

```
# Compute U_reduced
U_reduced = pca.components_.T
U_reduced
```

Out[16]:

```
array([[0.81976949],
       [0.57269362]])
```

In [17]:

```
# Reconstruct objects
x_approx = pca.inverse_transform(q)
x_approx
```

Out[17]:

```
array([[1.0213973 , 1.96937131],
       [2.16289608, 2.76682605],
       [3.50694011, 3.70577956],
       [4.44589363, 4.36173553],
       [5.78993766, 5.30068904],
       [8.07293521, 6.89559852]])
```

In [18]:

```
# Fraction of variance explained by projection
pca.explained_variance_ratio_[0]
```

Out[18]:

```
0.9681779330564779
```