Programming task 10: Dimensionality Reduction

In [2]:
```
import numpy as np
import matplotlib.pyplot as plt

%matplotlib inline
```

Exporting the results to PDF

Once you complete the assignments, export the entire notebook as PDF and attach it to your homework solutions. The best way of doing that is

1. Run all the cells of the notebook.
2. Export/download the notebook as PDF (File -> Download as -> PDF via LaTeX (.pdf)).
3. Concatenate your solutions for other tasks with the output of Step 2. On a Linux machine you can simply use `pdfunite`, there are similar tools for other platforms too. You can only upload a single PDF file to Moodle.

Make sure you are using `nbconvert` Version 5.5 or later by running `jupyter nbconvert --version`. Older versions clip lines that exceed page width, which makes your code harder to grade.

PCA

Given the data in the matrix X your tasks is to:

- Calculate the covariance matrix Σ.
- Calculate eigenvalues and eigenvectors of Σ.
- Plot the original data X and the eigenvectors to a single diagram. What do you observe? Which eigenvector corresponds to the smallest eigenvalue?
- Determine the smallest eigenvalue and remove its corresponding eigenvector. The remaining eigenvector is the basis of a new subspace.
- Transform all vectors in X in this new subspace by expressing all vectors in X in this new basis.

The given data X

In [3]:
```
X = np.array([(-3,-2),(-2,-1),(-1,0),(0,1),
              (1,2),(2,3),(-2,-2),(-1,-1),
              (0,0),(1,1),(2,2), (-2,-3),
              (-1,-2),(0,-1),(1,0), (2,1),(3,2)])
```

Task 1: Calculate the covariance matrix Σ

In [6]:
```
def get_covariance(X):
    """Calculates the covariance matrix of the input data.

    Parameters
    ----------
    X : array, shape [N, D]
        Data matrix.

    Returns
    -------
    Sigma : array, shape [D, D]
        Covariance matrix

    """
    # TODO
    mean = np.mean(X, axis=0)
    X_centered = X - mean
    N = X.shape[0]
    Sigma = (X_centered.T @ X_centered) / N
    return Sigma
```

Task 2: Calculate eigenvalues and eigenvectors of Σ.

In [7]:

```python
def get_eigen(S):
    """Calculates the eigenvalues and eigenvectors of the input matrix.

    Parameters
    ----------
    S : array, shape [D, D]
        Square symmetric positive definite matrix.

    Returns
    -------
    L : array, shape [D]
        Eigenvalues of S
    U : array, shape [D, D]
        Eigenvectors of S

    """
    L, U = np.linalg.eigh(S)

    idx = L.argsort()[::-1]
    L = L[idx]
    U = U[:, idx]

    return L, U
```

Task 3: Plot the original data X and the eigenvectors to a single diagram.

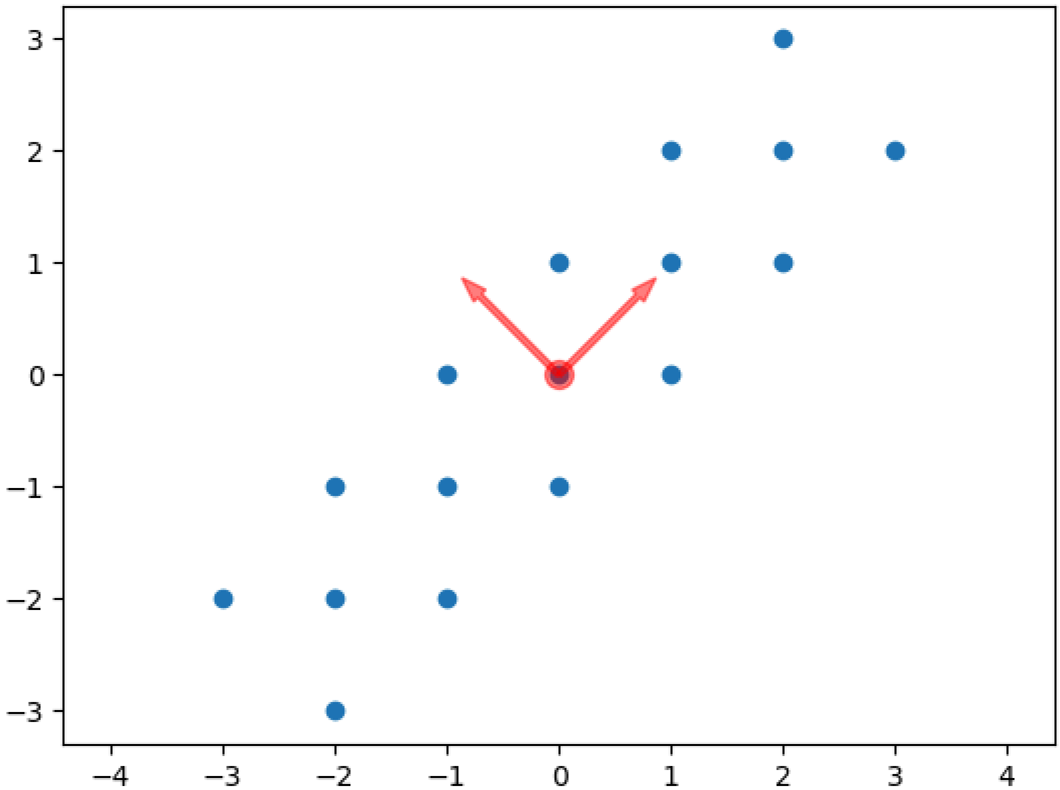
In [8]:

```python
# plot the original data
plt.scatter(X[:, 0], X[:, 1])
plt.axis('equal')

# plot the mean of the data
mean_d1, mean_d2 = X.mean(0)
plt.plot(mean_d1, mean_d2, 'o', markersize=10, color='red', alpha=0.5)

# calculate the covariance matrix
Sigma = get_covariance(X)
# calculate the eigenvector and eigenvalues of Sigma
L, U = get_eigen(Sigma)

plt.arrow(mean_d1, mean_d2, U[0, 0], U[1, 0], width=0.05, color='red', alpha=0.5)
plt.arrow(mean_d1, mean_d2, U[0, 1], U[1, 1], width=0.05, color='red', alpha=0.5);
```



What do you observe in the above plot? Which eigenvector corresponds to the smallest eigenvalue?

Write your answer here:

```
we can see the data can be represented in 2 directions(axis).
The smallest eigenvalue coresponds to the smallest variance in the
dataset, its direction is towards upper left, we need to reduce
this dimension if we want to do PCA.
```

Task 4: Transform the data

Determine the smallest eigenvalue and remove its corresponding eigenvector. The remaining eigenvector is the basis of a new subspace. Transform all vectors in X in this new subspace by expressing all vectors in X in this new basis.

In [11]:
```
def transform(X, U, L):
    """Transforms the data in the new subspace spanned by the eigenvector correspon

    Parameters
    ----------
    X : array, shape [N, D]
        Data matrix.
    L : array, shape [D]
        Eigenvalues of Sigma_X
    U : array, shape [D, D]
        Eigenvectors of Sigma_X
```

```python
    Returns
    -------
    X_t : array, shape [N, 1]
        Transformed data

    """
    mean = np.mean(X, axis=0)
    X_centered = X - mean
    U_reduced = U[:, 0:1]
    X_t = X_centered @ U_reduced
    return X_t
```

In [12]:
```python
X_t = transform(X, U, L)
```

SVD

Task 5: Given the matrix M find its SVD decomposition $M = U \cdot \Sigma \cdot V$ and reduce it to one dimension using the approach described in the lecture.

In [13]:
```python
M = np.array([[1, 2], [6, 3],[0, 2]])
```

In [14]:
```python
def reduce_to_one_dimension(M):
    """Reduces the input matrix to one dimension using its SVD decomposition.

    Parameters
    ----------
    M : array, shape [N, D]
        Input matrix.

    Returns
    -------
    M_t: array, shape [N, 1]
        Reduce matrix.

    """
    U, S, Vt = np.linalg.svd(M, full_matrices=False)

    U_reduced = U[:, 0:1]   #[N,1]
    s_reduced = S[0]    #scalar

    # 计算降维后的矩阵
    M_t = U_reduced * s_reduced

    return M_t
```

In [10]:
```python
M_t = reduce_to_one_dimension(M)
```