Homework 2. Numpy and matplotlib

Double Click here to edit this cell

- Name: 전기범
- Student ID: 201703091
- Submission date: 2019/03/28

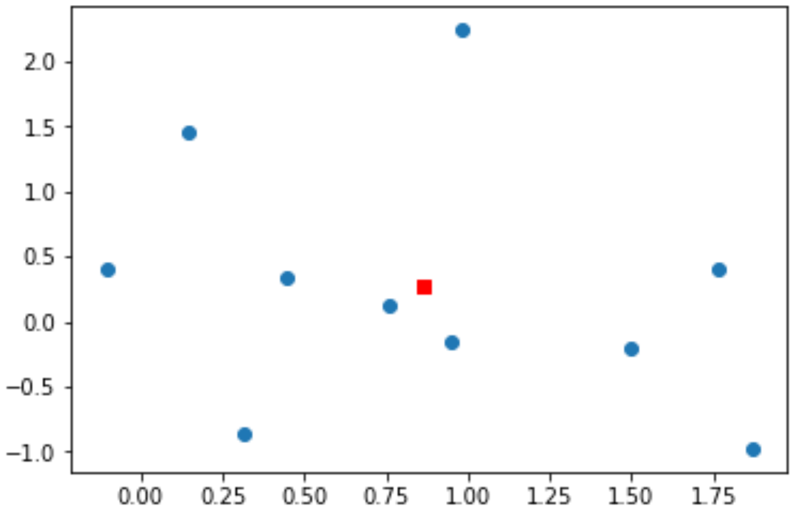
Remark. Use numpy wherever it is possible.

Problem 1 (5 pts)

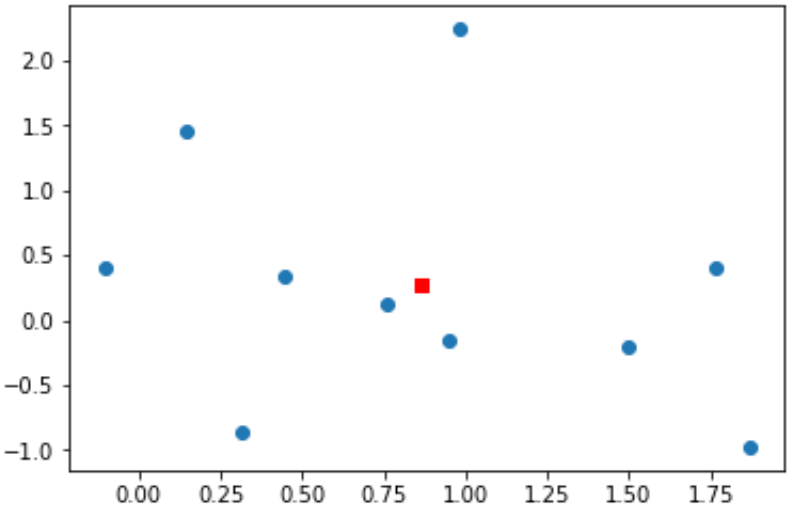
- The centroid of a finite set of k points $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k$ in $\mathbb{R}^n$ is
$$\mathbf{C} = \frac{\mathbf{x}_1 + \mathbf{x}_2 + \cdots + \mathbf{x}_k}{k}$$
- This point minimizes the sum of squared Euclidean distances between itself and each point in the set.
- Compute centroid
- Plot dataset and centroid

In [1]:
```
%matplotlib inline

import matplotlib.pyplot as plt
import numpy as np

def plot_centroid(data):
    plt.scatter(data[:,0], data[:,1])
    plt.scatter(sum(data[:,0])/10, sum(data[:,1])/10,color='red', m
arker='s')
    plt.show()
# YOUR CODE MUST BE HERE
```

```
In [2]: # DO NOT EDIT THIS CELL
        np.random.seed(0)
        data = np.random.randn(10,2)
        plot_centroid(data)
```



You output must be:

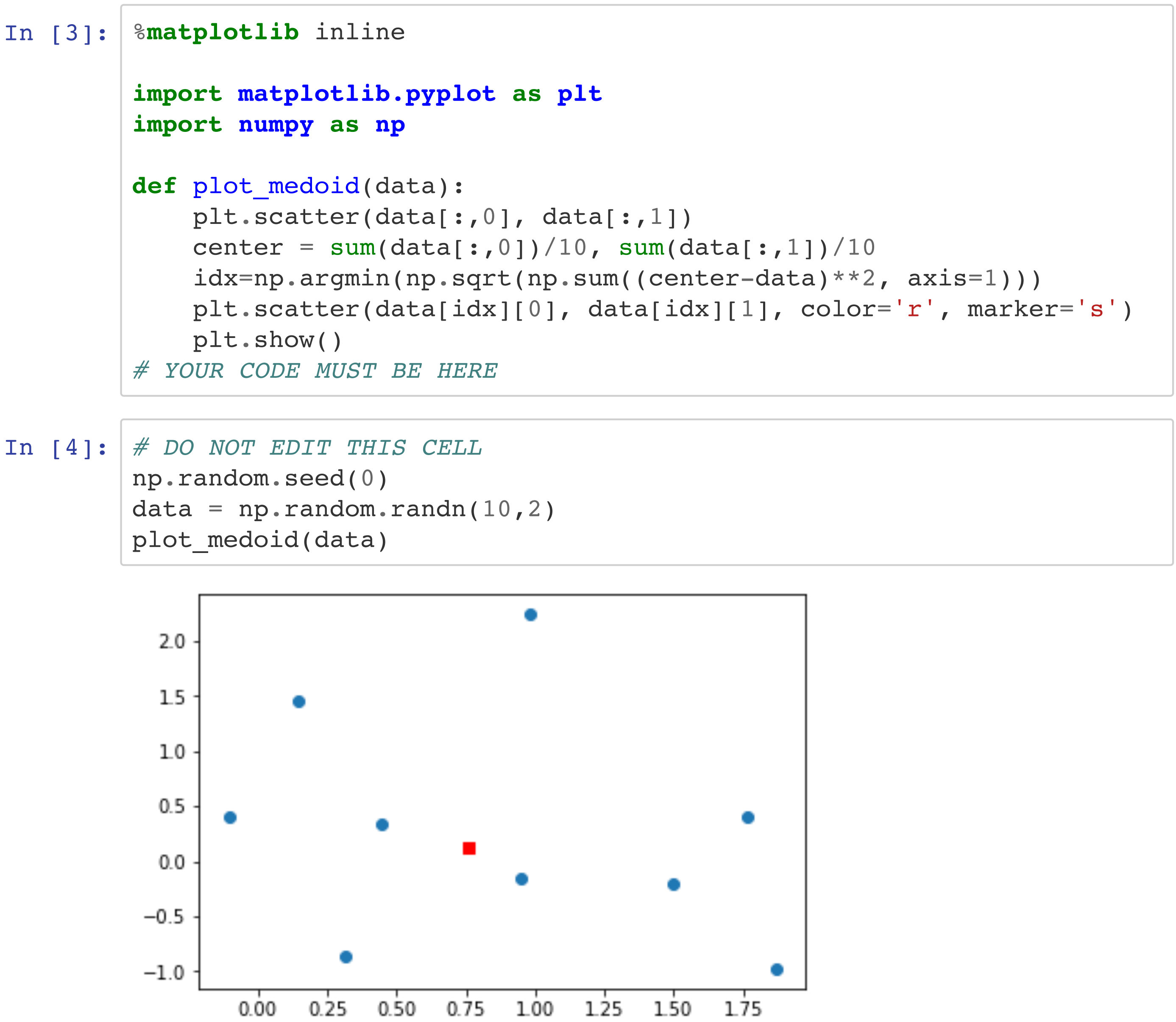


Problem 2 (10 pts)

- Let $x_1, x_2, \cdots, x_n$ be a set of n points in a space with a distance function d.
- Medoid is defined as

$$x_{\text{medoid}} = \text{argmin}_{y \in \{x_1, x_2, \cdots, x_n\}} \sum_{i=1}^{n} d(y, x_i)$$

- Compute medoid using Euclidean distance as a distance function.
- Plot dataset and medoid
- *Do not use sklearn, scipy or any module computing distance matrix directly*
- Use numpy functions only



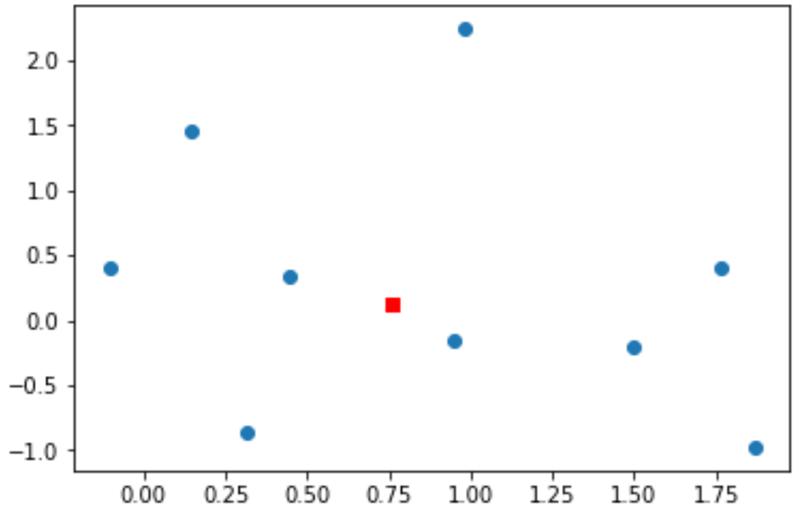
In [3]:

```
%matplotlib inline

import matplotlib.pyplot as plt
import numpy as np

def plot_medoid(data):
    plt.scatter(data[:,0], data[:,1])
    center = sum(data[:,0])/10, sum(data[:,1])/10
    idx=np.argmin(np.sqrt(np.sum((center-data)**2, axis=1)))
    plt.scatter(data[idx][0], data[idx][1], color='r', marker='s')
    plt.show()
# YOUR CODE MUST BE HERE
```

In [4]:

```
# DO NOT EDIT THIS CELL
np.random.seed(0)
data = np.random.randn(10,2)
plot_medoid(data)
```

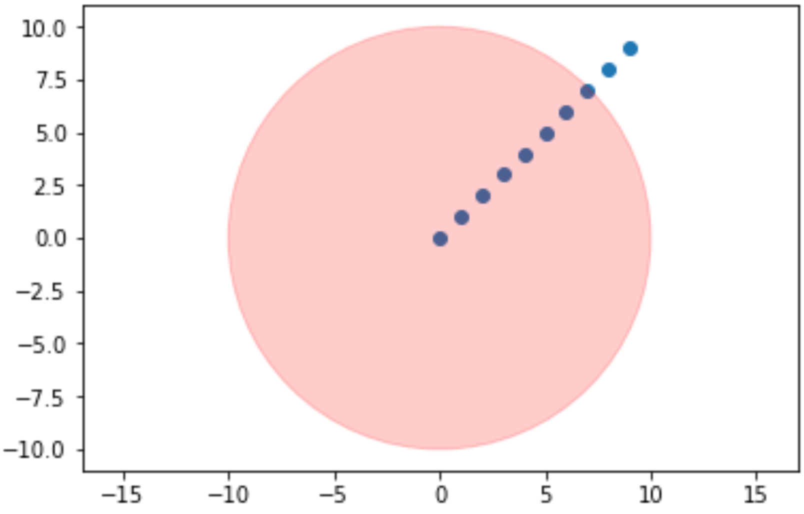
You output must be:



Sample code

In [5]:

```
%matplotlib inline

import matplotlib.pyplot as plt
import numpy as np

def sample_code():
    x = np.arange(10)
    y = np.arange(10)
    center = (0, 0)
    radius = 10
    plt.scatter(x, y)
    ax = plt.gca()
    ax.add_patch(plt.Circle(center, radius, color='r', alpha=0.2))
    plt.axis('equal')
    plt.show()

sample_code()
```



Problem 3 (5 pts)

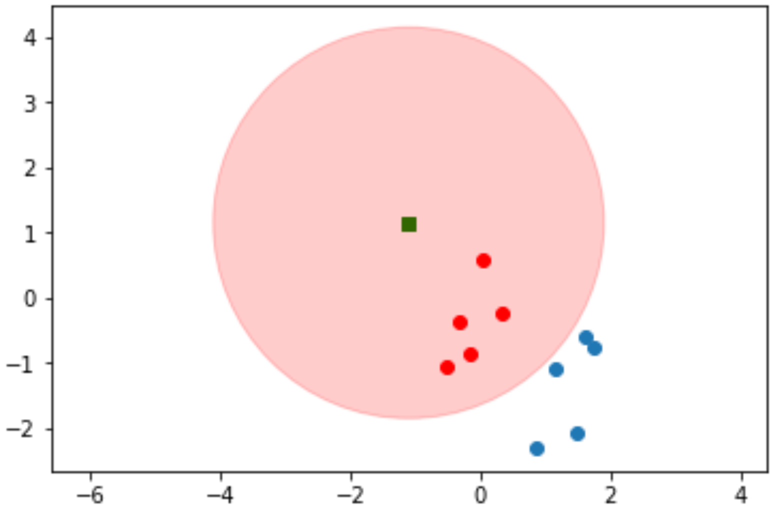
- We want to draw a scatter plot using **data**
- Plot the center using a green square symbol
- Plot points inside **radius** from center using red dots
- If you can't use a color printer, use marker '*' for red dots
- Plot points out of the **radius** from center using 'C0' colored dots
- Draw a filled circle centered at **center** using red color and alpha=0.2

In [6]:

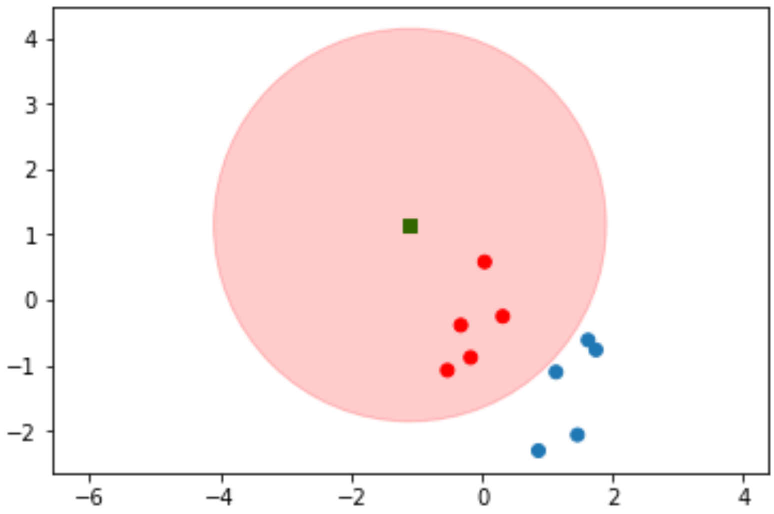
```
%matplotlib inline

import matplotlib.pyplot as plt
import numpy as np
from functools import reduce

def points_within_radius(data, center, radius):
    plt.scatter(center[0], center[1], color = 'g', marker = 's')
    arr1, arr2 = [], []
    for i in range(len(data)):
        if np.sqrt(np.sum((center-data[i])**2))<=radius: arr1.append(data[i])
        elif np.sqrt(np.sum((center-data[i])**2))>radius: arr2.append(data[i])
    arr1, arr2=np.array(arr1), np.array(arr2)
    plt.scatter(arr1[:,0], arr1[:,1],color='r')
    plt.scatter(arr2[:,0], arr2[:,1], color='C0')
    ax = plt.gca()
    ax.add_patch(plt.Circle(center, radius, color='r', alpha=0.2))
    plt.axis('equal')
    plt.show()
# YOUR CODE MUST BE HERE
```

In [7]:

```
# DO NOT EDIT THIS CELL
np.random.seed(1)
data = np.random.randn(10,2)
radius = 3.0
center = np.random.randn(2)
points_within_radius(data, center, radius)
```



You output must be:



Problem 4 (10 pts)

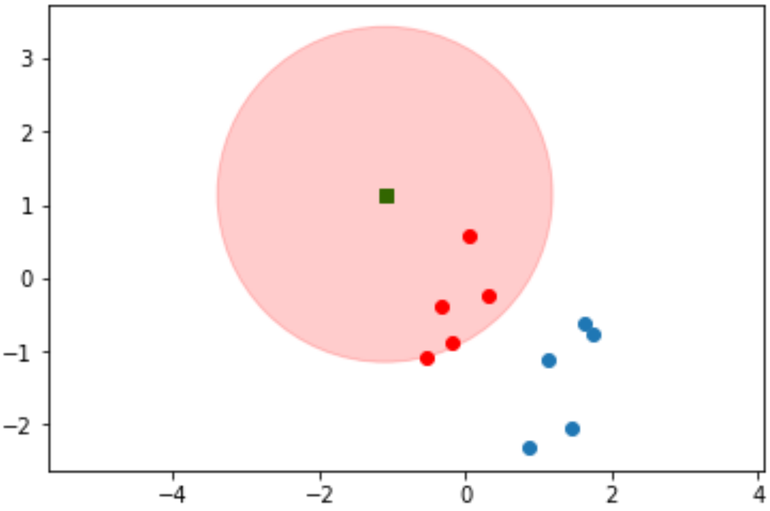
- We want to find k nearest points from the center
- Plot the center using a green square symbol
- Plot k-nearest points from center using red dots
- If you can't use a color printer, use marker '*' for red dots
- Plot other points using 'C0' colored dots
- Draw a filled circle centered at **center** using red color and alpha=0.2
- *Do not use sklearn, scipy or any module computing k-nearest points directly*
- Use numpy functions only

In [8]:

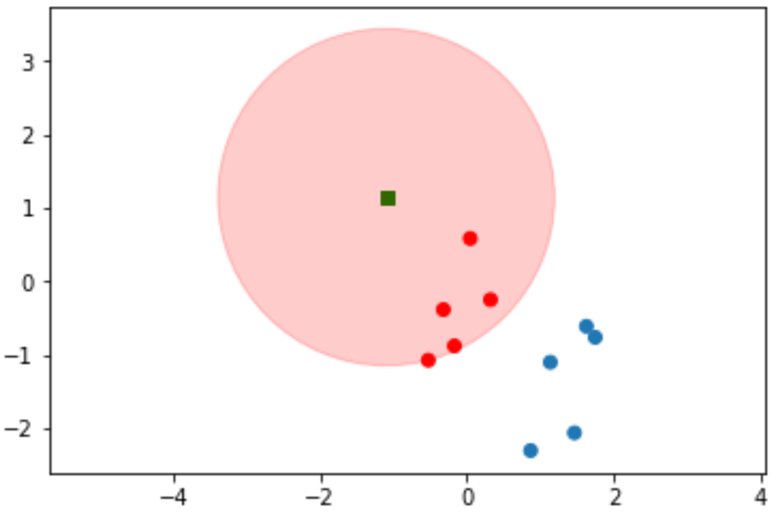
```
%matplotlib inline

import matplotlib.pyplot as plt
import numpy as np

def points_k_nearest(data, center, k=1):
    plt.scatter(center[0], center[1], color = 'g', marker = 's')
    radius=np.sort(np.sqrt(np.sum((center-data)**2, axis=1)))[k-1]
    arr1, arr2 = [], []
    for i in range(len(data)):
        if np.sqrt(np.sum((center-data[i])**2))<=radius: arr1.append(data[i])
        elif np.sqrt(np.sum((center-data[i])**2))>radius: arr2.append(data[i])
    arr1, arr2=np.array(arr1), np.array(arr2)
    plt.scatter(arr1[:,0], arr1[:,1],color='r')
    plt.scatter(arr2[:,0], arr2[:,1], color='C0')
    ax = plt.gca()
    ax.add_patch(plt.Circle(center, radius, color='r', alpha=0.2))
    plt.axis('equal')
    plt.show()
# YOUR CODE MUST BE HERE
```

In [9]:

```
# DO NOT EDIT THIS CELL
np.random.seed(1)
data = np.random.randn(10,2)
k = 5
center = np.random.randn(2)
points_k_nearest(data, center, k)
```

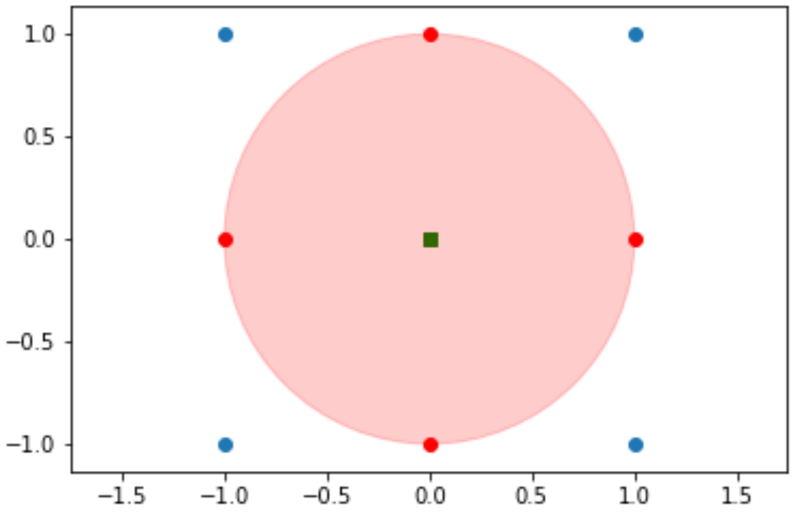


You output must be:

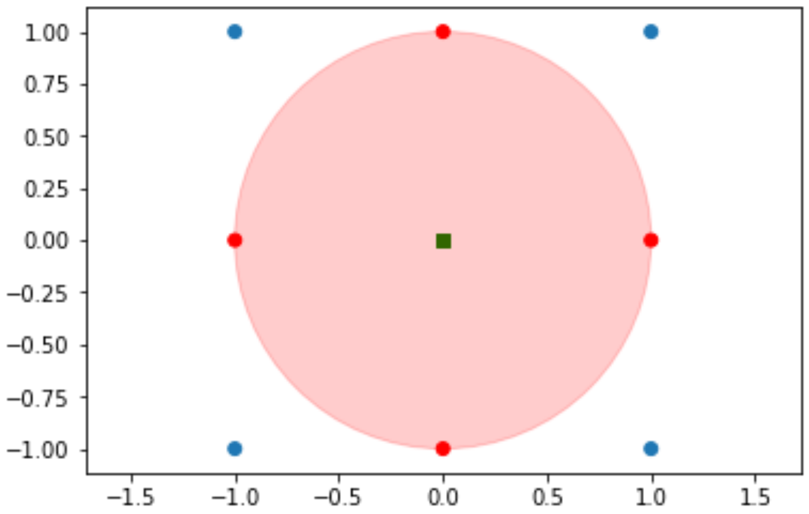


In [10]:

```
# DO NOT EDIT THIS CELL
np.random.seed(1)
data = np.array([[1.,0.],[0.,1.],[-1.,0.],[0.,-1.],[1.,1.],[1.,-1.]
,[-1.,1.],[-1.,-1.]])
np.random.shuffle(data)
k = 1
center = np.array([0.,0.])
points_k_nearest(data, center, k)
```



You output must be:



Problem 5 (5 pts)

- **find_k_nearest_index** returns the index of the k-nearest
- We want to time the execution
- *Do not use sklearn, scipy or any module computing k-nearest points directly*
- Use numpy functions only

In [11]:

```
import numpy as np

def find_k_nearest_index(data, center, k=1):
    r=np.linalg.norm(center-data, axis=1)
    radius=np.partition(r,k-1)[k-1]
    print(np.where(r<=radius))     # np.argwhere can use
    # YOUR CODE MUST BE HERE
```

In [12]:

```
# DO NOT EDIT THIS CELL
np.random.seed(1)
data = np.random.randn(10000000,2)    # 10 million data
k = 5
center = np.random.randn(2)
%time find_k_nearest_index(data, center, k)
```

```
(array([3146213, 4362536, 6716705, 6845205, 7607470]),)
CPU times: user 1.03 s, sys: 131 ms, total: 1.16 s
Wall time: 395 ms
```

Your time must be around:

```
CPU times: user 384 ms, sys: 232 ms, total: 616 ms
Wall time: 619 ms
```

You output must be:

```
array([3146213, 4362536, 6716705, 6845205, 7607470])
```

Ethics:

If you cheat, you will get negatgive of the total points. If the homework total is 22 and you cheat, you get -22.

What to submit

- Run all cells
- Goto "File -> Print Preview"
- Print the page
- Submit in class
- No late homeworks accepted
- Your homework will be graded on the basis of correctness and programming skills

Deadline: 3/28