Homework 2. Numpy and matplotlib

Double Click here to edit this cell

- Name: 조성현
- Student ID: 201803430
- Submission date: 04/09

Remark. If any kind of loops including for-loop, while-loops, list comprehension, and other loops are found, you get no points (0점).

Use numpy wherever it is possible.

Total: 45 pts

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

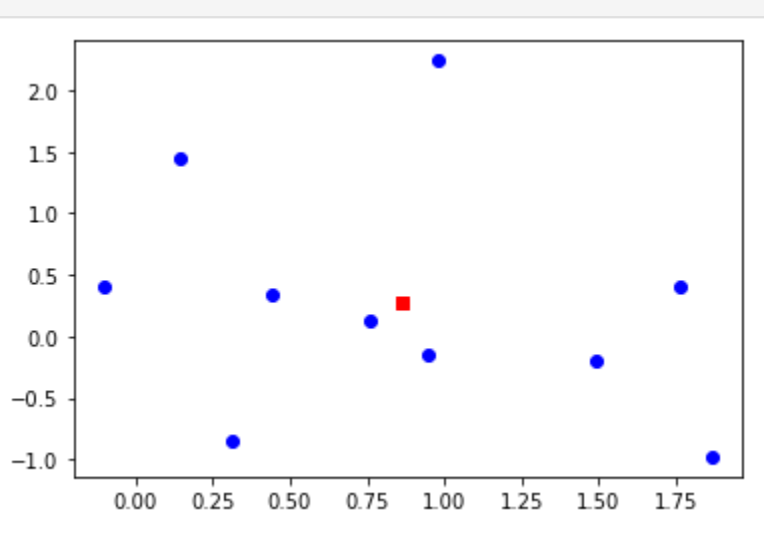
$$\mathbf{C} = \frac{\mathbf{x}_1 + \mathbf{x}_2 + \cdots + \mathbf{x}_k}{k}$$

- This point minimizes the sum of squared Euclidean distances between itself and each point in the set.

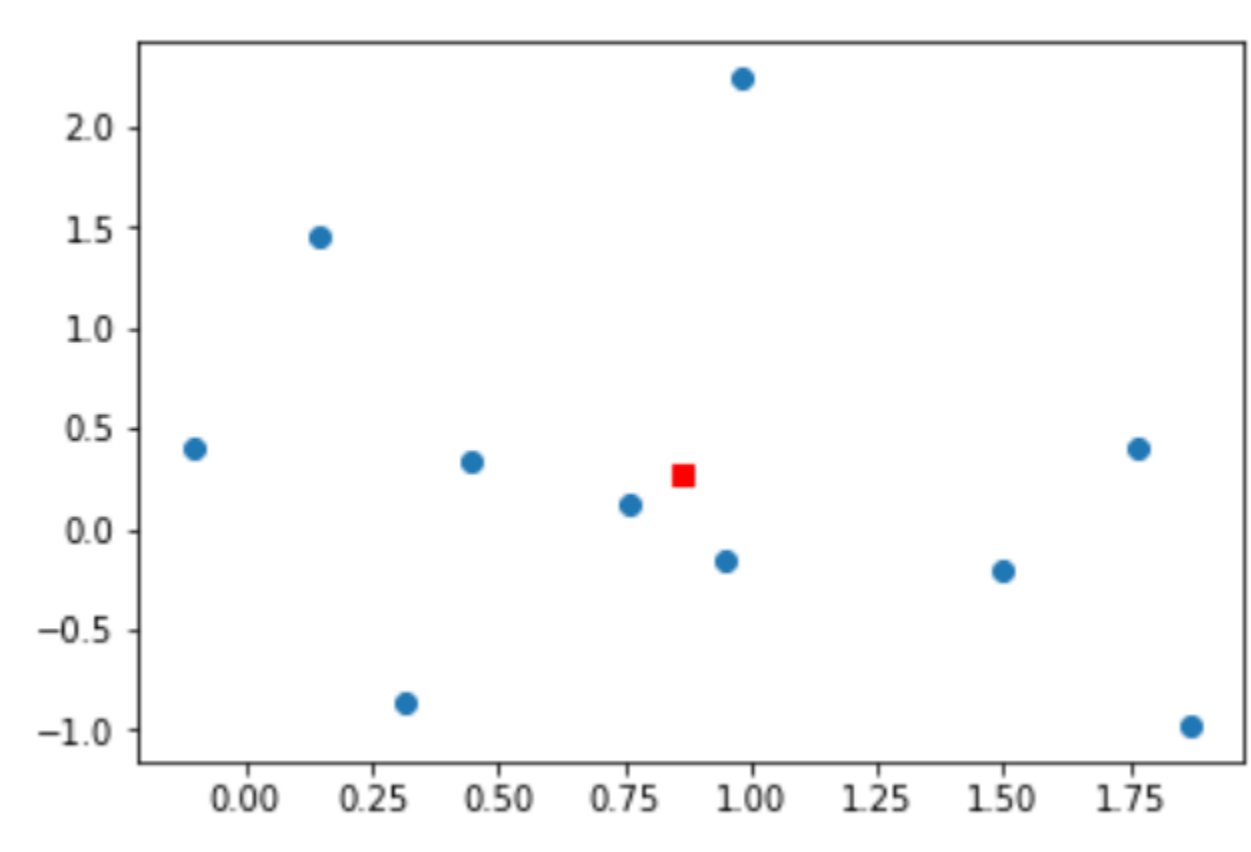
- Compute centroid
- Plot dataset and centroid

In [1]:
```python
%matplotlib inline

import matplotlib.pyplot as plt
import numpy as np

def plot_centroid(data):
    centroid = data.mean(axis=0)
    plt.scatter(data[:,0],data[:,1] ,color="b")
    ax = plt.gca()
    ax.scatter(centroid[0],centroid[1],marker = 's', color="r")
    return plt.show()
```

In [2]:
```python
# 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

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

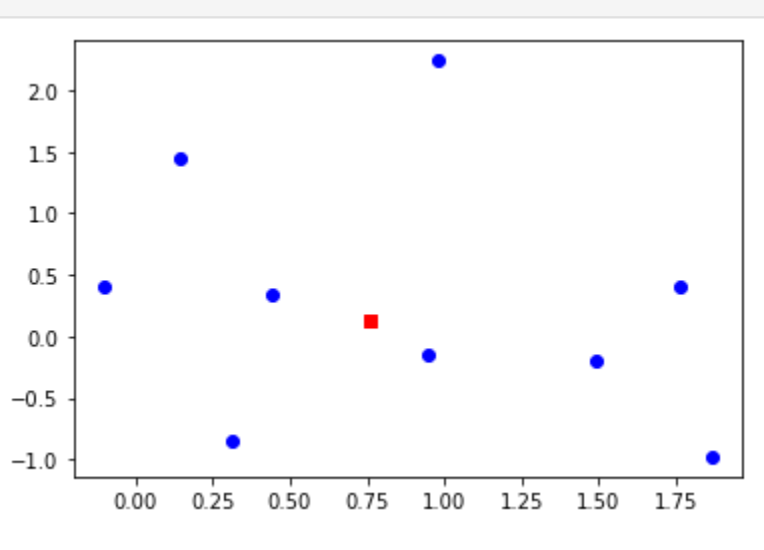
- Compute medoid using Euclidean distance as a distance function.
- Plot dataset and medoid
- *Do not use sklearn, scipy or any module computing distance matrix directly*
- Use numpy functions only

In [3]:
```python
%matplotlib inline

import matplotlib.pyplot as plt
import numpy as np

def plot_medoid(data):
    u = data[:,:,np.newaxis]
    v = data.T[np.newaxis,:,:]
    disall = np.linalg.norm(u-v,axis =1 )
    medoid = data[np.argmin(disall.sum(axis =1))]
    plt.scatter(data[:,0],data[:,1] ,color="b")
    ax = plt.gca()
    ax.scatter(medoid[0],medoid[1],marker = 's', color="r")
    return plt.show()
```

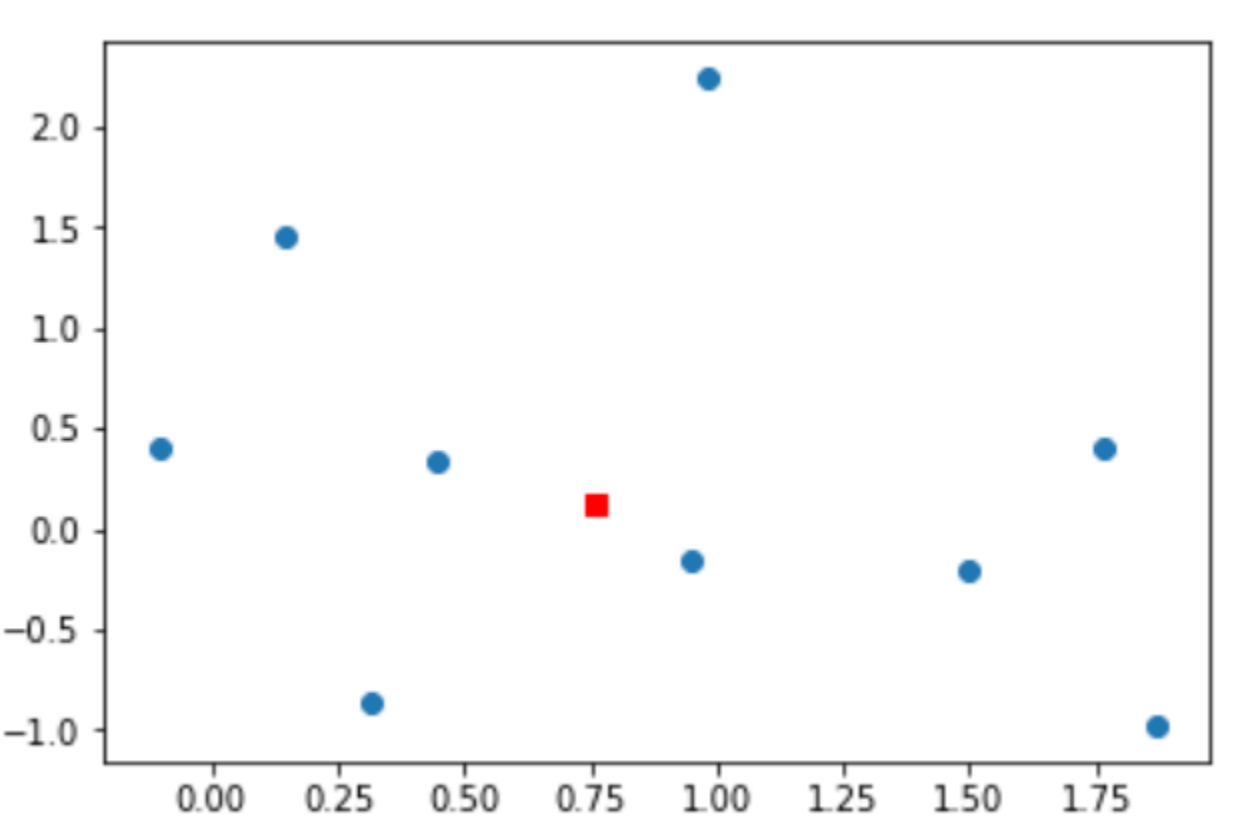
In [4]:
```python
d = np.array([[1,2],[4,3],[5,6],[1,4],[5,2],[1,6]])
u = d[:,:,np.newaxis]
v = d.T[np.newaxis,:,:]
disall = np.linalg.norm(u-v,axis =1 )
print(d[np.argmin(disall.sum(axis =1))])
```

```
[4 3]
```

In [5]:
```python
# DO NOT EDIT THIS CELL
np.random.seed(0)
data = np.random.randn(10,2)
plot_medoid(data)
```

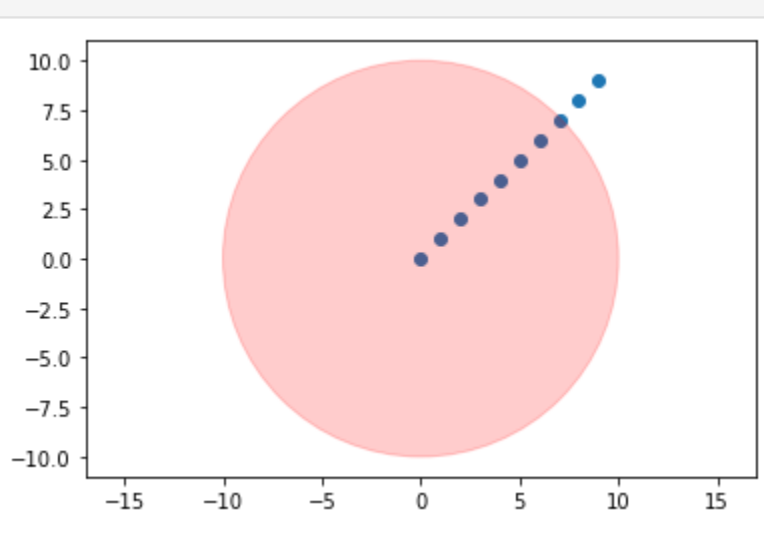


You output must be:



Sample code

In [6]:
```python
%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)

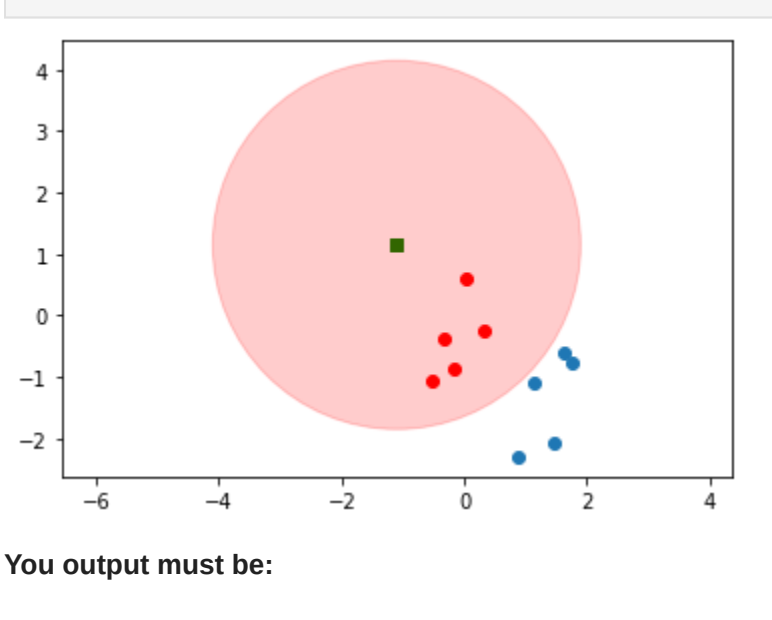
- We want to draw a scatter plot using **data**
- Plot the center using a green square symbol
- Plot points inside **radius** from center using red dots
- Plot points out of the **radius** from center using 'C0' colored dots
- Draw a filled circle centered at **center** using red color and alpha=0.2

In [7]:
```python
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
def points_within_radius(data, center, radius):
    plt.scatter(center[0],center[1],marker = 's', color="g")
    ax = plt.gca()
    ax.add_patch(plt.Circle(center, radius, color='r', alpha=0.2))
    v = center[np.newaxis,:]
    v = np.repeat(v,len(data),0)
    disall = np.linalg.norm(data-v,axis=1)
    bd = data[np.where(disall>radius)]
    rd = data[np.where(disall<=radius)]
    plt.scatter(bd[:,0],bd[:,1], color="C0")
    plt.scatter(rd[:,0],rd[:,1], color="r")
    plt.axis('equal')
    plt.show()
```

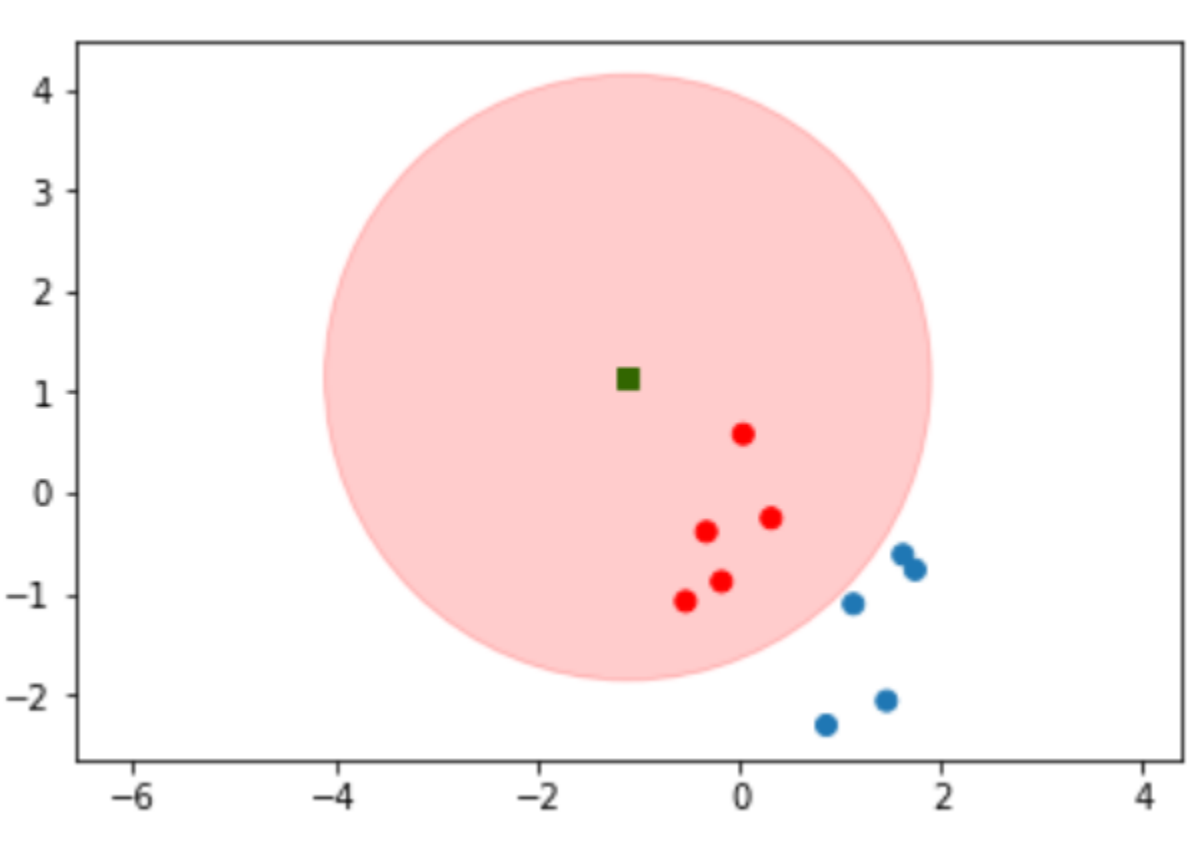
In [8]:
```python
d = np.array([[1,2],[4,3],[5,6],[1,4],[5,2],[1,6]])
radius = 4.0
center = np.random.randn(2)
v = np.repeat(center[np.newaxis,:],len(d),0)
print(v)
disall = np.linalg.norm(d-v,axis=1)
print(disall)
t=np.where(disall <radius)
print(d[t])
```

```
[[-2.55298982  0.6536186 ]
 [-2.55298982  0.6536186 ]
 [-2.55298982  0.6536186 ]
 [-2.55298982  0.6536186 ]
 [-2.55298982  0.6536186 ]
 [-2.55298982  0.6536186 ]]
[3.79953675 6.96040094 9.25372624 4.88077915 7.67205305 6.41930921]
[[1 2]]
```

In [9]:
```python
# 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:

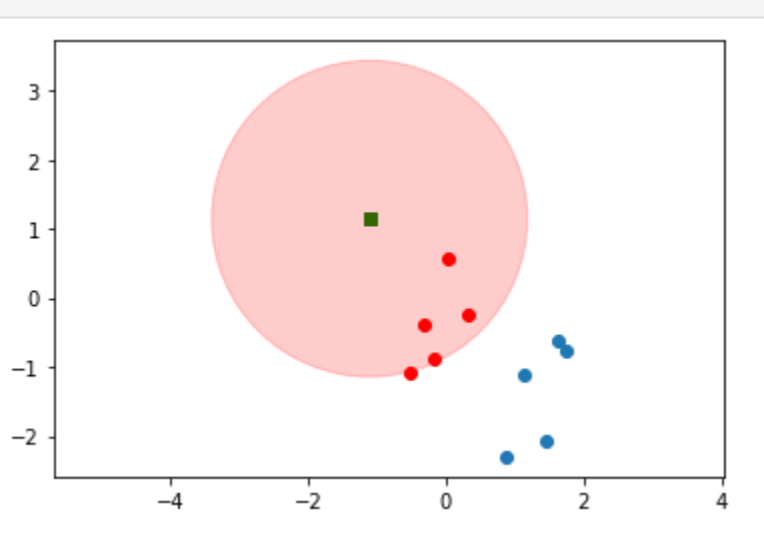


Problem 4 (10 pts)

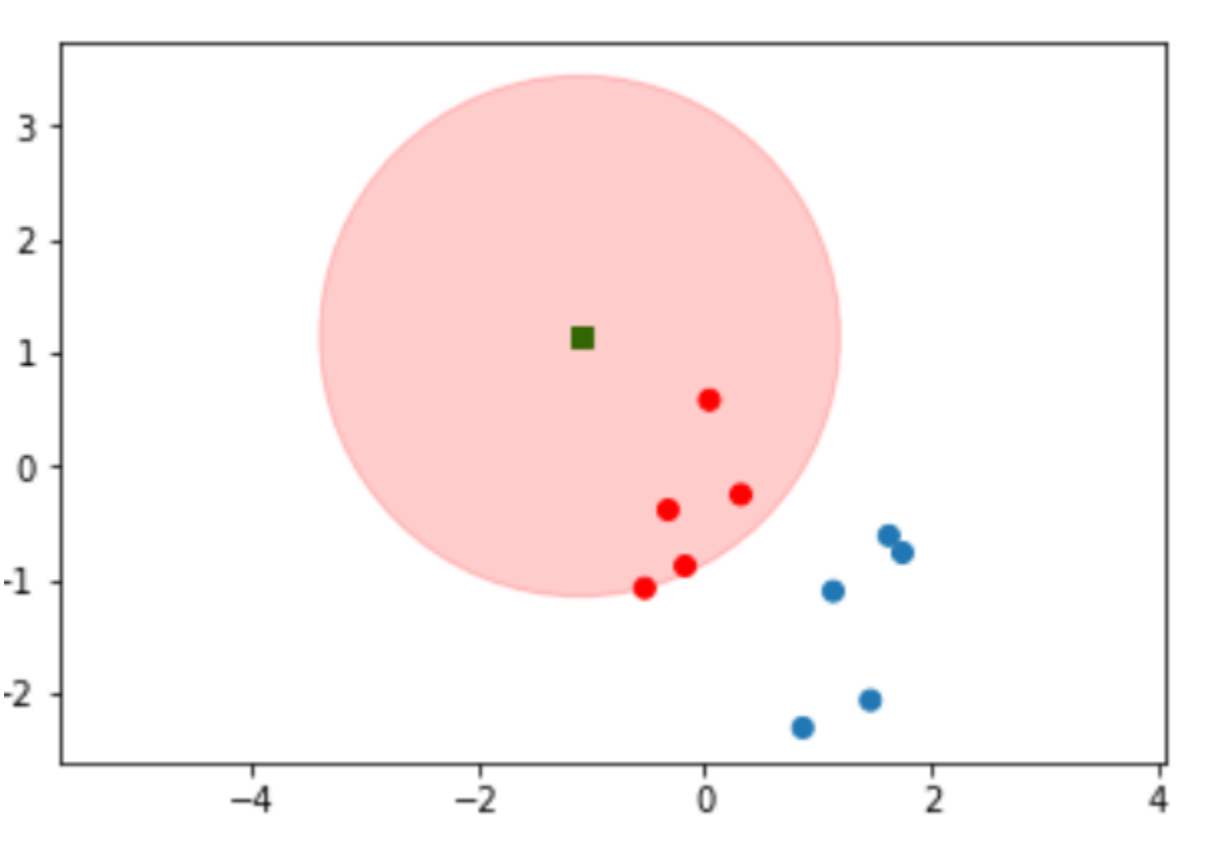
- We want to find k nearest points from the center
- Plot the center using a green square symbol
- Plot k-nearest points from center using red dots
- Plot other points using 'C0' colored dots
- Draw a filled circle centered at **center** using red color and alpha=0.2
- *Do not use sklearn, scipy or any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

In [10]:
```python
%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],marker = 's', color="g")
    v = center[np.newaxis,:]
    v = np.repeat(v,len(data),0)
    disall = np.linalg.norm(data-v,axis=1)
    sortdist=np.argsort(disall)
    radius=disall[sortdist[k-1]]
    ax = plt.gca()
    ax.add_patch(plt.Circle(center, radius, color='r', alpha=0.2))
    bd = data[np.where(disall>radius)]
    rd = data[np.where(disall<=radius)]
    ax.scatter(bd[:,0],bd[:,1], color="C0")
    ax.scatter(rd[:,0],rd[:,1], color="r")
    ax.axis('equal')
    plt.show()
# YOUR CODE MUST BE HERE
```

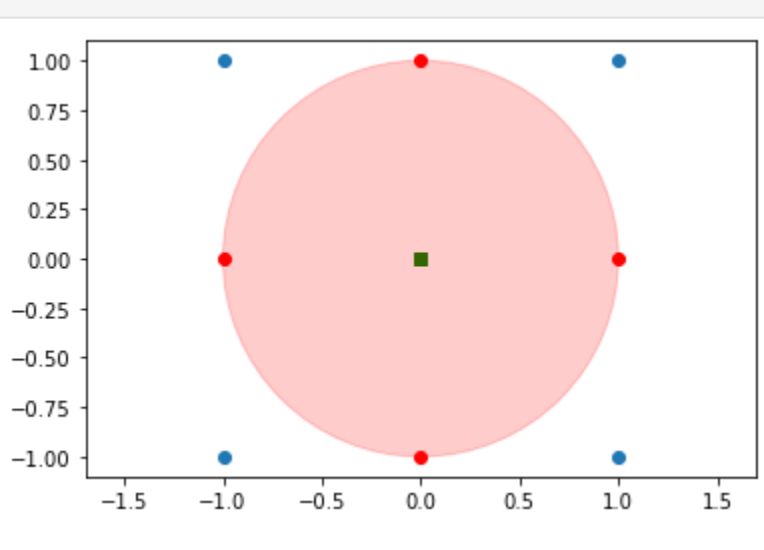
In [11]:
```python
# DO NOT EDIT THIS CELL
np.random.seed(1)
data = np.random.randn(10,2)
k = 5
center = np.random.randn(2)
points_k_nearest(data, center, k)
```



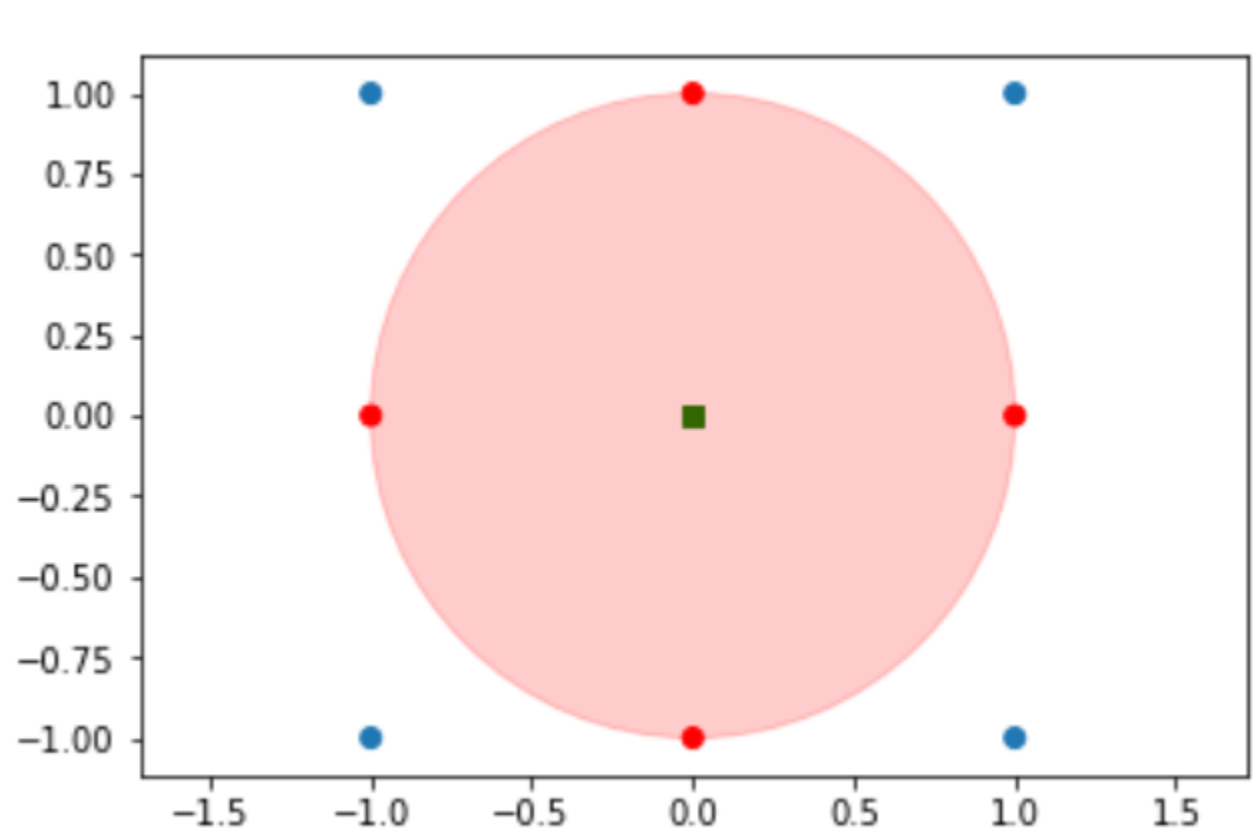
You output must be:



In [12]:
```python
# 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 (15 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
- Points:
  - when mean time $\leq$ 5s: 15 pts
  - when mean time $>$ 5s and $\leq$ 10s: 10 pts
  - when mean time $>$ 10s: 5 pts

In [13]:
```python
import numpy as np

def find_k_nearest_index(data, center, k=1):
    v = np.repeat(center[np.newaxis,:],len(data),0)
    disall = np.linalg.norm(data-v,axis=1)
    radius=disall[np.argsort(disall)[k-1]]
    return np.where(disall<=radius)[0]

# YOUR CODE MUST BE HERE
```

In [14]:
```python
# DO NOT EDIT THIS CELL

np.random.seed(0)
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.])
print(find_k_nearest_index(data, center, k))
```

```
[1 2 4 5]
```

You output must be:

```
[1 2 4 5]
```

In [15]:
```python
# DO NOT EDIT THIS CELL

np.random.seed(100)
data = np.random.randn(10000000,20)   # 10 million data
k = 5
center = np.random.randn(20)
print(find_k_nearest_index(data, center, k))
%timeit find_k_nearest_index(data, center, k)
```

```
[1664998 3042821 3307688 3848441 5351099]
4.27 s ± 11.5 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

You output must be:

```
[1664998 3042821 3307688 3848441 5351099]
```

Your time must be around:

```
1.99 s ± 113 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

If your computer is in poor spec, don't buy new one.

Use google colab! In google colab, I got 1.2s. Google colab outperforms my i5-16GB laptop.

```
1 loop, best of 5: 1.2 s per loop
```

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** after restarting the kernel
- Goto "File -> Print Preview"
- Print the page as pdf
- Pdf file name must be in a form of: homework_2_홍길동_202000001.pdf
- Submit the pdf file in google classroom
- No late homeworks will be accepted
- Your homework will be graded on the basis of correctness and programming skills