In [1]:

```
import matplotlib
import matplotlib.pyplot as plt
matplotlib.style.use('ggplot')

from IPython.core.display import display, HTML
display(HTML("<style>.container { width:100% !important; }</style>"))

import numpy as np
import pandas as pd
```

Location problem

We want to find the smallest circle such that n points are all contained in it.

In [2]:

```
# pick a bunch of random points
pos = np.random.randn(1000,2)

import matplotlib.pyplot as plt
plt.scatter(pos[:,0], pos[:,1],s=50,marker=u'x')
plt.xlabel('x'), plt.ylabel('y')
plt.axis('equal')
plt.show()
```

In [3]:

```
# solution with cvxpy
from cvx.util import cvx, minimize

def location(pos):
    R,x = cvx.Variable(1), cvx.Variable(2)
    minimize(objective=R, constraints = [cvx.norm(row-x,2) <= R for row in pos])
    return R.value, x.value

print(location(pos))
```

```
(array([3.78584482]), array([-0.55701664, -0.5040435 ]))
```

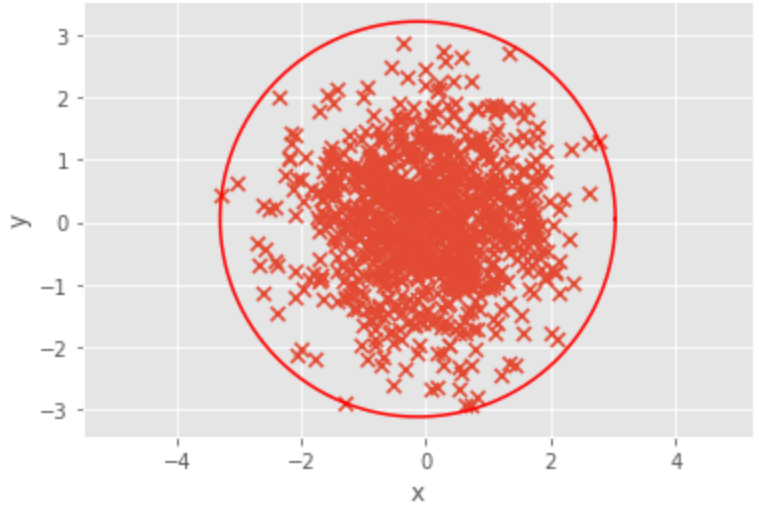
In [4]:

```
# pick a bunch of random points
pos = np.random.randn(1000,2)

plt.scatter(pos[:,0], pos[:,1],s=50,marker=u'x')
plt.xlabel('x'), plt.ylabel('y')
plt.axis('equal')

radius, midpoint = location(pos)

import math
c=np.array([[radius*np.cos(a)+midpoint[0], radius*np.sin(a)+midpoint[1]] for a in np
plt.plot(c[:,0],c[:,1],'r')
plt.show()
```



Summary

- Each constraint $\|\mathbf{x} - \mathbf{c}\|_2 < R$ represents a cone. Feasible domain is the intersection of all cones.
- It is trivial to generalize (but not to plot) for points in higher dimensional spaces.
- However, all of this fails once we can construct multiple circles.