This is code for Midterm questions 11 and 12. It's based on the code provided for one of our classes.

In [12]:
```
%%writefile Kmeans.py
#/Users/ninakuklisova/miniconda2/envs/jupi/bin/python
from numpy import argmin, array, random
from mrjob.job import MRJob
from mrjob.step import MRStep
from itertools import chain

#Calculate find the nearest centroid for data point
def MinDist(datapoint, centroid_points):
    datapoint = array(datapoint)
    centroid_points = array(centroid_points)
    diff = datapoint - centroid_points
    diffsq = diff**2

    distances = (diffsq.sum(axis = 1))**0.5
    # Get the nearest centroid for each instance
    min_idx = argmin(distances)
    return min_idx

#Check whether centroids converge
def stop_criterion(centroid_points_old, centroid_points_new,T):
    oldvalue = list(chain(*centroid_points_old))
    newvalue = list(chain(*centroid_points_new))
    Diff = [abs(x-y) for x, y in zip(oldvalue, newvalue)]
    Flag = True
    for i in Diff:
        if(i>T):
            Flag = False
            break
    return Flag


class MRKmeans(MRJob):
    centroid_points=[]
    k=3
    def steps(self):
        return [
            MRStep(mapper_init = self.mapper_init, mapper=self.mapper,
combiner = self.combiner,reducer=self.reducer)
                ]
    #load centroids info from file
    def mapper_init(self):
        self.centroid_points = [map(float,s.split('\n')[0].split(','))
for s in open('Centroids.txt', 'w+').readlines()]
```

```
        #open('Centroids.txt', 'w').close()
    #load data and output the nearest centroid index and data point
    def mapper(self, _, line):
        D = (map(float,line.split(',')))
        idx = MinDist(D,self.centroid_points)
        yield int(idx), (D[0],D[1],1)
    #Combine sum of data points locally
    def combiner(self, idx, inputdata):
        sumx = sumy = num = 0
        for x,y,n in inputdata:
            num = num + n
            sumx = sumx + x
            sumy = sumy + y
        yield int(idx),(sumx,sumy,num)
    #Aggregate sum for each cluster and then calculate the new centroi
ds
    def reducer(self, idx, inputdata):
        centroids = []
        num = [0]*self.k
        distances = 0
        for i in range(self.k):
            centroids.append([0,0])
        for x, y, n in inputdata:
            num[idx] = num[idx] + n
            centroids[idx][0] = centroids[idx][0] + x
            centroids[idx][1] = centroids[idx][1] + y
        centroids[idx][0] = centroids[idx][0]/num[idx]
        centroids[idx][1] = centroids[idx][1]/num[idx]
        with open('Centroids.txt', 'a') as f:
            f.writelines(str(centroids[idx][0]) + ',' + str(centroids[
idx][1]) + '\n')
        yield idx,(centroids[idx][0],centroids[idx][1])

if __name__ == '__main__':
    MRKmeans.run()
```

```
Overwriting Kmeans.py
```

Driver:

Generate random initial centroids

New Centroids = initial centroids

While(1):

- Cacluate new centroids
- stop if new centroids close to old centroids
- Updates centroids

In [1]:
```
%reload_ext autoreload
%autoreload 2
from numpy import random
from Kmeans import MRKmeans, stop_criterion
mr_job = MRKmeans(args=['Kmeandata.csv', '--file=Centroids.txt'])

#Geneate initial centroids
centroid_points = []
k = 3
for i in range(k):
    centroid_points.append([random.uniform(-3,3),random.uniform(-3,3)]
)
with open('Centroids.txt', 'w+') as f:
        f.writelines(','.join(str(j) for j in i) + '\n' for i in centr
oid_points)

# Update centroids iteratively
i = 0
while(1):
    # save previous centoids to check convergency
    centroid_points_old = centroid_points[:]
    print "iteration"+str(i)+":"
    with mr_job.make_runner() as runner:
        runner.run()
        # stream_output: get access of the output
        for line in runner.stream_output():
            key,value =  mr_job.parse_output_line(line)
            print key, value
            centroid_points[key] = value

        # Update the centroids for the next iteration
        with open('Centroids.txt', 'w') as f:
            f.writelines(','.join(str(j) for j in i) + '\n' for i in c
entroid_points)

    print "\n"
    i = i + 1
    if(stop_criterion(centroid_points_old,centroid_points,0.00001)):
        break
print "Centroids\n"
print centroid_points
```

```
iteration0:
Current path: /private/var/folders/zp/223m_g716ydf9ln0c83zc1jc0000gn
/T/Kmeans.ninakuklisova.20160630.021654.471809/job_local_dir/0/mappe
r/0
Centroids:  [[2.73186590671, -1.38105156974], [-1.45410087909, -2.52
214675805], [-0.0974172557485, -2.87318791569]]
Current path: /private/var/folders/zp/223m_g716ydf9ln0c83zc1jc0000gn
```

```
/T/Kmeans.ninakuklisova.20160630.021654.471809/job_local_dir/0/mappe
r/1
Centroids:  [[2.73186590671, -1.38105156974], [-1.45410087909, -2.52
214675805], [-0.0974172557485, -2.87318791569]]
0 [2.9320774045012343, 2.306289867553679]
1 [-4.538955186248487, 0.6312757218323691]


iteration1:
Current path: /private/var/folders/zp/223m_g716ydf9ln0c83zc1jc0000gn
/T/Kmeans.ninakuklisova.20160630.021654.649426/job_local_dir/0/mappe
r/0
Centroids:  [[2.9320774045, 2.30628986755], [-4.53895518625, 0.63127
5721832], [-0.0974172557485, -2.87318791569]]
Current path: /private/var/folders/zp/223m_g716ydf9ln0c83zc1jc0000gn
/T/Kmeans.ninakuklisova.20160630.021654.649426/job_local_dir/0/mappe
r/1
Centroids:  [[2.9320774045, 2.30628986755], [-4.53895518625, 0.63127
5721832], [-0.0974172557485, -2.87318791569]]
0 [2.78053698516226, 2.4935627536550395]
1 [-4.725681639812464, 0.458625527541844]
2 [2.408602031258859, -2.2217528838341725]


iteration2:
Current path: /private/var/folders/zp/223m_g716ydf9ln0c83zc1jc0000gn
/T/Kmeans.ninakuklisova.20160630.021654.875557/job_local_dir/0/mappe
r/0
Centroids:  [[2.78053698516, 2.49356275366], [-4.72568163981, 0.4586
25527542], [2.40860203126, -2.22175288383]]
Current path: /private/var/folders/zp/223m_g716ydf9ln0c83zc1jc0000gn
/T/Kmeans.ninakuklisova.20160630.021654.875557/job_local_dir/0/mappe
r/1
Centroids:  [[2.78053698516, 2.49356275366], [-4.72568163981, 0.4586
25527542], [2.40860203126, -2.22175288383]]
0 [1.9692539933857425, 3.598004705053755]
1 [-4.8055138050627075, 0.29793236183725796]
2 [4.948850536128022, -1.0855165408361183]


iteration3:
Current path: /private/var/folders/zp/223m_g716ydf9ln0c83zc1jc0000gn
/T/Kmeans.ninakuklisova.20160630.021655.090758/job_local_dir/0/mappe
r/0
Centroids:  [[1.96925399339, 3.59800470505], [-4.80551380506, 0.2979
32361837], [4.94885053613, -1.08551654084]]
Current path: /private/var/folders/zp/223m_g716ydf9ln0c83zc1jc0000gn
/T/Kmeans.ninakuklisova.20160630.021655.090758/job_local_dir/0/mappe
r/1
Centroids:  [[1.96925399339, 3.59800470505], [-4.80551380506, 0.2979
```

```
32361837], [4.94885053613, -1.08551654084]]
0 [0.3288863504993916, 4.840690242232994]
1 [-4.9455168845426245, 0.057607057165595874]
2 [5.1151566040340155, -0.17044764162226675]


iteration4:
Current path: /private/var/folders/zp/223m_g716ydf9ln0c83zc1jc0000gn
/T/Kmeans.ninakuklisova.20160630.021655.285296/job_local_dir/0/mappe
r/0
Centroids:  [[0.328886350499, 4.84069024223], [-4.94551688454, 0.057
6070571656], [5.11515660403, -0.170447641622]]
Current path: /private/var/folders/zp/223m_g716ydf9ln0c83zc1jc0000gn
/T/Kmeans.ninakuklisova.20160630.021655.285296/job_local_dir/0/mappe
r/1
Centroids:  [[0.328886350499, 4.84069024223], [-4.94551688454, 0.057
6070571656], [5.11515660403, -0.170447641622]]
0 [0.05539493412485448, 4.985087518735044]
1 [-4.98580568889943, 0.0009376094363626959]
2 [5.0428936969663996, -0.028603929711745572]


iteration5:
Current path: /private/var/folders/zp/223m_g716ydf9ln0c83zc1jc0000gn
/T/Kmeans.ninakuklisova.20160630.021655.473760/job_local_dir/0/mappe
r/0
Centroids:  [[0.0553949341249, 4.98508751874], [-4.9858056889, 0.000
937609436363], [5.04289369697, -0.0286039297117]]
Current path: /private/var/folders/zp/223m_g716ydf9ln0c83zc1jc0000gn
/T/Kmeans.ninakuklisova.20160630.021655.473760/job_local_dir/0/mappe
r/1
Centroids:  [[0.0553949341249, 4.98508751874], [-4.9858056889, 0.000
937609436363], [5.04289369697, -0.0286039297117]]
0 [0.053065423788147964, 4.987793423944292]
1 [-4.98580568889943, 0.0009376094363626959]
2 [5.0402327160888465, -0.026294229978289455]


iteration6:
Current path: /private/var/folders/zp/223m_g716ydf9ln0c83zc1jc0000gn
/T/Kmeans.ninakuklisova.20160630.021655.674066/job_local_dir/0/mappe
r/0
Centroids:  [[0.0530654237881, 4.98779342394], [-4.9858056889, 0.000
937609436363], [5.04023271609, -0.0262942299783]]
Current path: /private/var/folders/zp/223m_g716ydf9ln0c83zc1jc0000gn
/T/Kmeans.ninakuklisova.20160630.021655.674066/job_local_dir/0/mappe
r/1
Centroids:  [[0.0530654237881, 4.98779342394], [-4.9858056889, 0.000
937609436363], [5.04023271609, -0.0262942299783]]
0 [0.053065423788147964, 4.987793423944292]
```

```
1 [-4.98580568889943, 0.00093760943636269 59]
2 [5.0402327160888465, -0.026294229978289455]


Centroids

[[0.053065423788147964, 4.987793423944292], [-4.98580568889943, 0.00
09376094363626959], [5.0402327160888465, -0.026294229978289455]]
```

MT 11.

In [2]:
```
import pylab
import numpy
```

In [3]:
```
for point in centroid_points:
    pylab.plot(point[0], point[1], '*',color='red',markersize=20)
pylab.show()
```

MT 12.

In [4]:
```
import csv
from Kmeans import MinDist
import math

# average weighted distance:
total_distance = 0

with open('Kmeandata.csv', 'r') as csvfile:
    count = 0
    for row in csvfile:
        point = row.split(',')
        x, y = float(point[0]), float(point[1])
        datapoint = (x, y)
        # weighted distance from the point's assigned centroid:
        weight = math.sqrt(x**2+ y**2)
        dist_weighted = MinDist(datapoint, centroid_points) / weight
        count+=1
        total_distance +=dist_weighted

print 'Average weighted distance is ', total_distance / count


# average weighted distance:
#total_distance = 0
#points = re
```

```
Average weighted distance is  0.205794476594
```

In [ ]:

In [ ]: