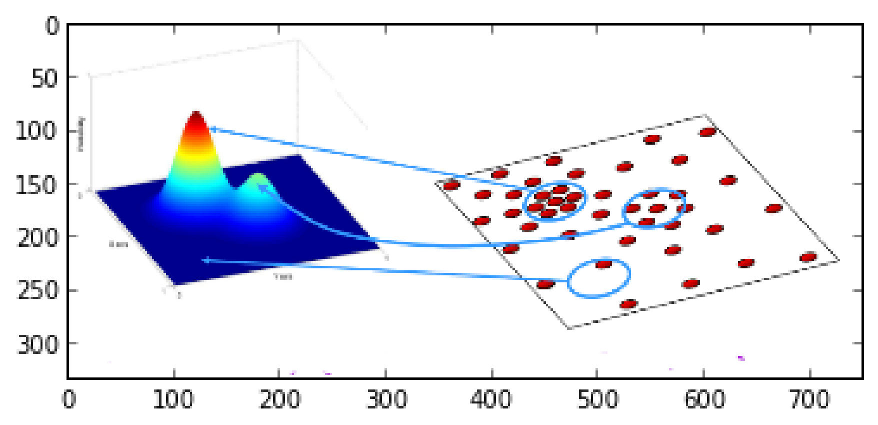
Ortalama Kaydirma ile Kumeleme (Mean Shift Clustering)

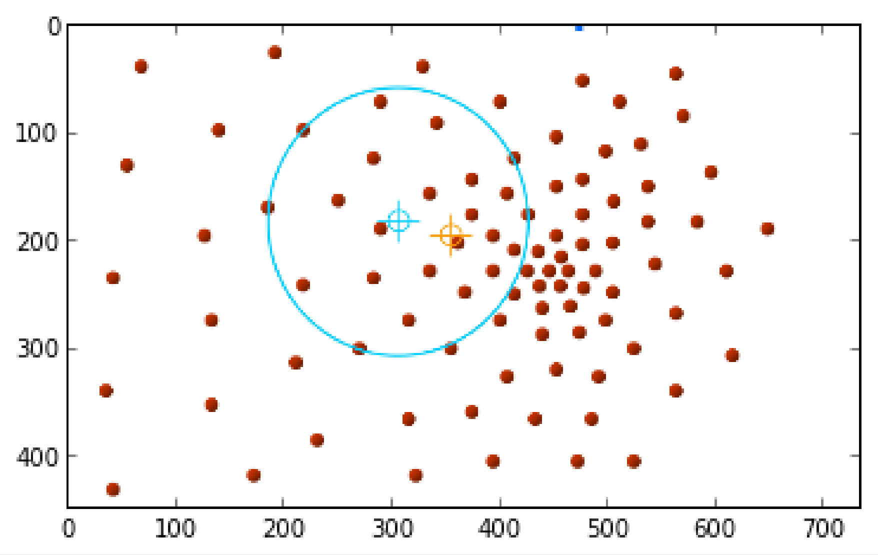
```
im=imread("dist.png"); imshow(im)
```

```
<matplotlib.image.AxesImage at 0xa3f3c4c>
```



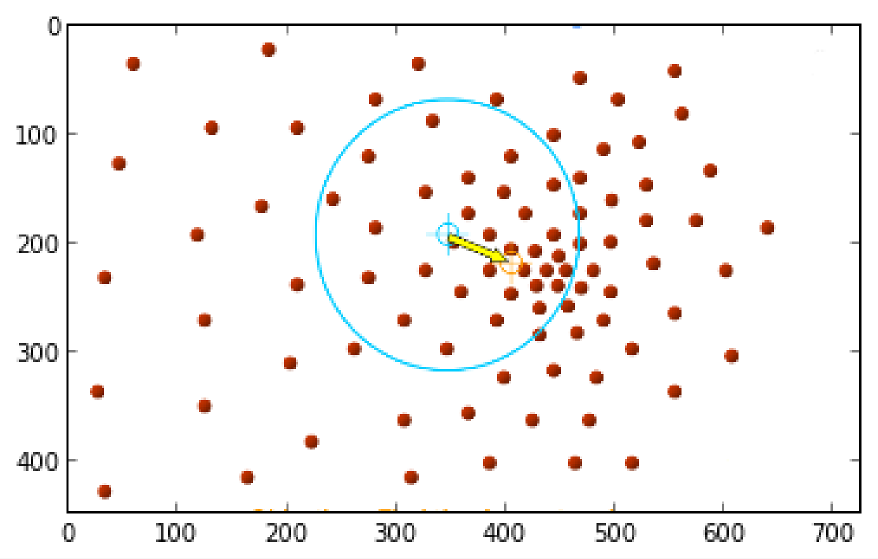
```
im=imread("mean_2.png"); imshow(im)
```

```
<matplotlib.image.AxesImage at 0x9b966ac>
```



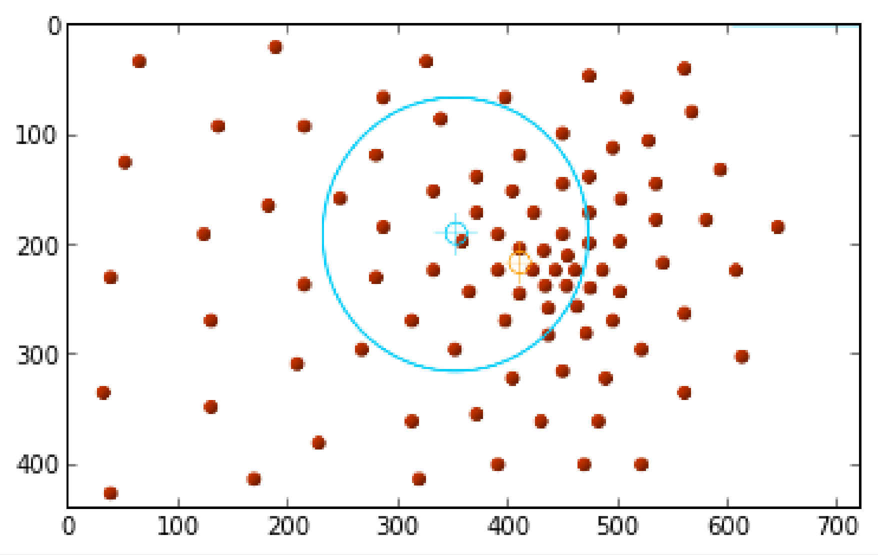
```
im=imread("mean_3.png"); imshow(im)
```

```
<matplotlib.image.AxesImage at 0x9cd99ec>
```



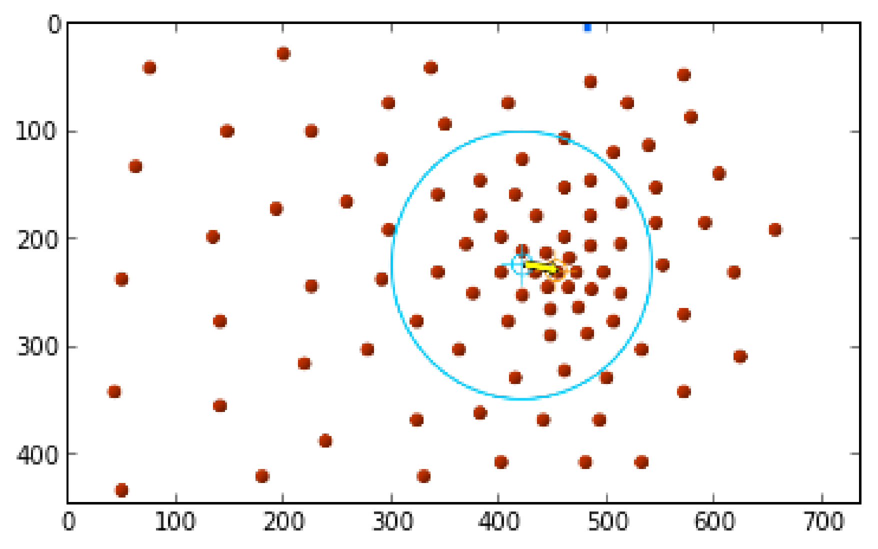
```
im=imread("mean_4.png"); imshow(im)
```

```
<matplotlib.image.AxesImage at 0x9e3cfac>
```



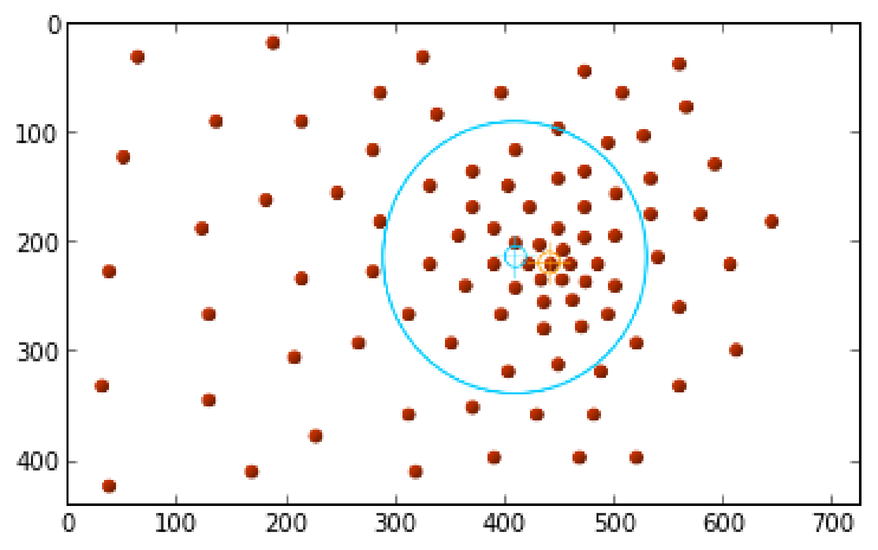
```
im=imread("mean_5.png"); imshow(im)
```

```
<matplotlib.image.AxesImage at 0x9f9b5ec>
```



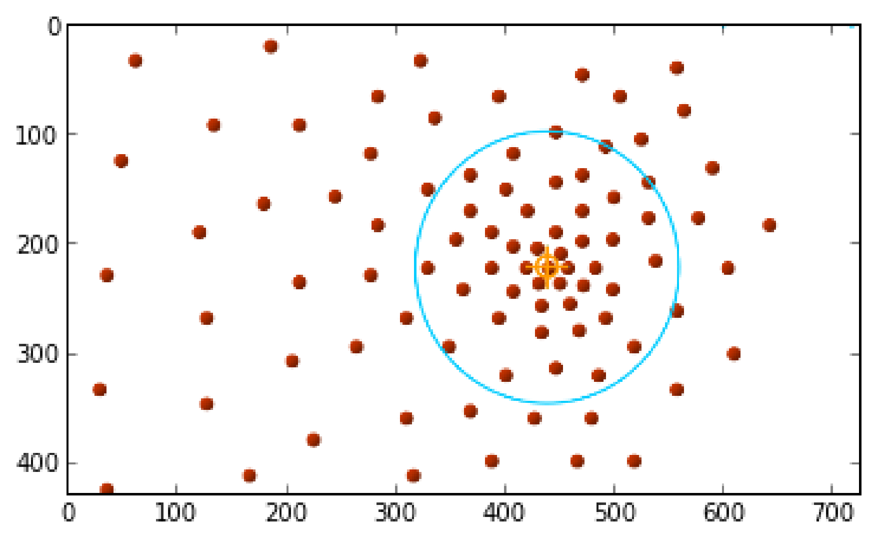
```
im=imread("mean_6.png"); imshow(im)
```

```
<matplotlib.image.AxesImage at 0xa13cd0c>
```



```
im=imread("mean_7.png"); imshow(im)
```

```
<matplotlib.image.AxesImage at 0xa2a132c>
```

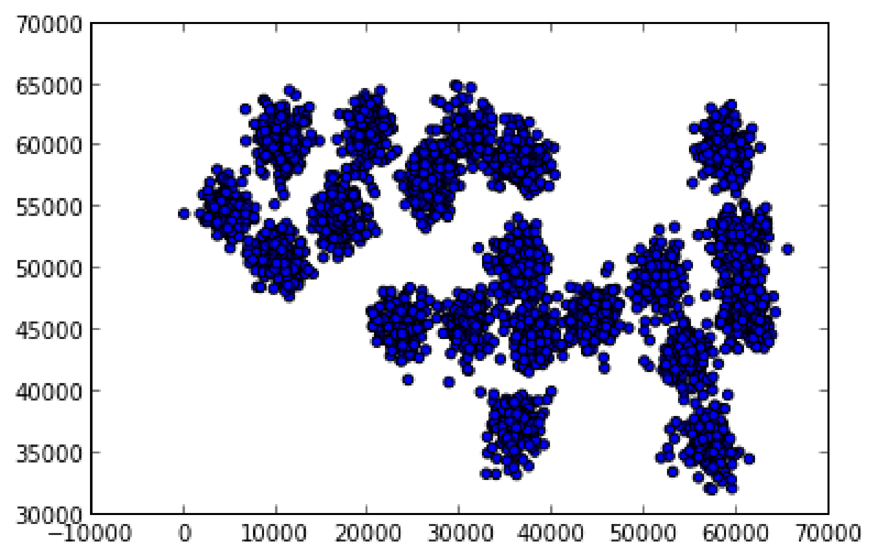


```
from pandas import *
data = read_csv("synthetic.txt",names=['a','b'],sep=" ")
print data.shape
data = np.array(data)
```

```
(3000, 2)
```

```
scatter(data[:,0],data[:,1])
```

```
<matplotlib.collections.PathCollection at 0x9f85d6c>
```



```
from numpy import *
from numpy import linalg as la

def mean_shift(dataPts, bandWidth):
    dataPts = asarray( dataPts )
    bandWidth = float( bandWidth )
    plotFlag = False

    numDim, numPts = dataPts.shape
    numClust = 0
    bandSq = bandWidth**2
    initPtInds = arange( numPts )
    #biggest size in each dimension
    maxPos = dataPts.max(0)
    #smallest size in each dimension
    minPos = dataPts.min(0)
    #bounding box size
    boundBox = maxPos-minPos
    #indicator of size of data space
    sizeSpace = la.norm(boundBox)
    #when mean has converged
    stopThresh = 1e-3*bandWidth
    #center of clust
    clustCent = []
    #track if a points been seen already
    beenVisitedFlag = zeros( numPts, dtype = uint8 )
    #number of points to possibly use as initilization points
    numInitPts = numPts
    #used to resolve conflicts on cluster membership
    clusterVotes = []

    while numInitPts:

        rand = random.rand()
        #pick a random seed point
        tempInd = int(floor( (numInitPts-1e-6)*rand ))
        #use this point as start of mean
        stInd = initPtInds[ tempInd ]
        # intilize mean to this points location
        myMean = dataPts[ :, stInd ]
        # points that will get added to this cluster
        myMembers = []
        #used to resolve conflicts on cluster membership
        thisClusterVotes = zeros( numPts, dtype = uint16 )

        while True:
            #dist squared from mean to all points still active
            sqDistToAll = (( myMean[:,newaxis] - dataPts )**2).sum(0)
            #points within bandWidth
            inInds = where(sqDistToAll < bandSq)
            #add a vote for all the in points belonging to this cluster
            thisClusterVotes[ inInds ] = thisClusterVotes[ inInds ]+1

            #save the old mean
            myOldMean = myMean
            #compute the new mean
            myMean = mean( dataPts[ :, inInds[0] ], 1 )
            #add any point within bandWidth to the cluster
            myMembers.extend( inInds[0] )
            #mark that these points have been visited
            beenVisitedFlag[myMembers] = 1

            if la.norm(myMean-myOldMean) < stopThresh:
                #check for merge posibilities
                mergeWith = None
                for cN in xrange( numClust ):
                    #distance from possible new clust max to old clust max
                    distToOther = la.norm( myMean - clustCent[ cN ] )
                    #if its within bandwidth/2 merge new and old
                    if distToOther < bandWidth/2:
                        mergeWith = cN
                        break

                # something to merge
                if mergeWith is not None:
                    #record the max as the mean of the two merged (I know biased twoards
                        new ones)
                    clustCent[ mergeWith ] = 0.5*( myMean + clustCent[ mergeWith ] )
                    #add these votes to the merged cluster
                    clusterVotes[ mergeWith ] += thisClusterVotes
                else:
                    #increment clusters
                    numClust = numClust+1
                    #record the mean
                    clustCent.append( myMean )
                    clusterVotes.append( thisClusterVotes )

                break

        initPtInds = where(beenVisitedFlag == 0)[0]
        numInitPts = len(initPtInds)

    data2cluster = asarray( clusterVotes ).argmax(0)

    return clustCent, data2cluster
```

```
dataPts = asarray([[1,1],[2,2],[3,3],[9,9],[9,9],[9,9],[10,10]]).T
print dataPts
print dataPts.shape
bandwidth = 2
print 'data points:', dataPts
print 'bandwidth:', bandwidth
```

```
clustCent, data2cluster = mean_shift(dataPts, 2)
print 'cluster centers:', sorted( asarray( clustCent ).squeeze().tolist() )
print 'data2cluster:', data2cluster
print len( clustCent )
print sorted( asarray( clustCent ).squeeze().tolist() )
```

```
[[ 1  2  3  9  9  9 10]
 [ 1  2  3  9  9  9 10]]
(2, 7)
data points: [[ 1  2  3  9  9  9 10]
 [ 1  2  3  9  9  9 10]]
bandwidth: 2
cluster centers: [[1.5, 1.5], [2.5, 2.5], [9.25, 9.25]]
data2cluster: [2 0 0 1 1 1 1]
3
[[1.5, 1.5], [2.5, 2.5], [9.25, 9.25]]
```

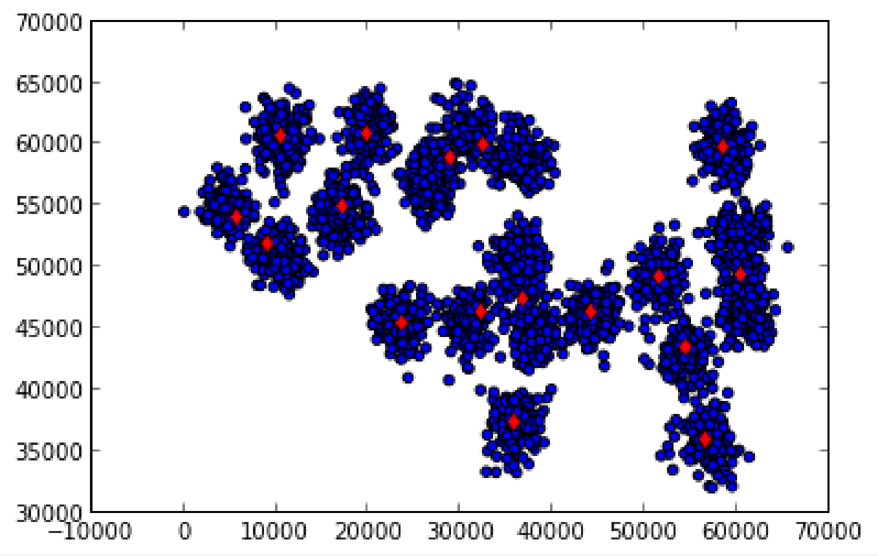
```
print asarray(data.T)[:30]
clustCent, data2cluster = mean_shift(asarray(data.T), 5000)
```

```
[[54620 52694 53253 ...,  8828  8879 10002]
 [43523 42750 43024 ..., 59102 59244 61399]]
```

```
print asarray(clustCent)[:6]
print asarray(clustCent).shape
```

```
[[ 36612.3541253   47646.45712961]
 [ 51485.64457831  49092.07831325]
 [ 44379.26285714  46182.84571429]
 [ 58673.90322581  59703.51612903]
 [ 17186.66049383  54836.2345679 ]
 [ 32502.45283019  59820.74339623]]
(17, 2)
```

```
scatter(data[:,0],data[:,1])
plt.hold(True)
for x in asarray(clustCent): plot(x[0],x[1],'rd')
```


70000
65000
60000
55000
50000
45000
40000
35000
30000
−10000
0
10000
20000
30000
40000
50000
60000
70000