2017 Edition

Anomaly Detection in Network Traffic with K-means clustering

We can categorize machine learning algorithms into two main groups: **supervised learning** and **unsupervised learning**. With supervised learning algorithms, in order to predict unknown values for new data, we have to know the target value for many previously-seen examples. In contrast, unsupervised learning algorithms explore the data which has no target attribute to find some intrinsic structures in them.

Clustering is a technique for finding similar groups in data, called **clusters**. Clustering is often called an unsupervised learning task as no class values denoting an a priori grouping of the data instances are given.

In this notebook, we will use K-means, a very well-known clustering algorithm to detect anomaly network connections based on statistics about each of them. A thorough overview of K-means clustering, from a research perspective, can be found in the following wonderful tutorial.

Goals

We expect students to:

- Learn (or revise) and understand the K-means algorithm
- Implement a simple K-means algorithm
- Use K-means to detect anomalies network connection data

Steps

1. In section 1, we will have an overview about K-means then implement a simple version of it.
2. In section 2, we build models with and without categorical features.
3. Finally, in the last section, using our models, we will detect unusual connections.

1. K-means

1.1. Introduction

Clustering is a typical and well-known type of unsupervised learning. Clustering algorithms try to find natural groupings in data. Similar data points (according to some notion of similarity) are considered in the same group. We call these groups **clusters**.

K-Means clustering is a simple and widely-used clustering algorithm. Given value of $k$, it tries to build $k$ clusters from samples in the dataset. Therefore, $k$ is an hyperparameter of the model. The right value of $k$ is not easy to determine, as it highly depends on the data set and the way that data is featurized.

To measure the similarity between any two data points, K-means requires the definition of a distance function between data points. What is a distance? It is a value that indicates how close two data points are in their space. In particular, when data points lie in a d-dimensional space, the Euclidean distance is a good choice of a distance function, and is supported by MLLIB.

In K-means, a cluster is a group of points, with a representative entity called a centroid. A centroid is also a point in the data space: the center of all the points that make up the cluster. It's defined to be the arithmetic mean of the points. In general, when working with K-means, each data sample is represented in a d-dimensional numeric vector, for which it is easier to define an appropriate distance function. As a consequence, in some applications, the original data must be transformed into a different representation, to fit the requirements of K-means.

1.2. How does it work?

Given k, the K-means algorithm works as follows:

1. Randomly choose k data points (seeds) to be the initial centroids
2. Assign each data point to the **closest centroid**
3. Re-compute (update) the centroids using the current cluster memberships
4. If a convergence criterion is not met, go to step 2

We can also terminate the algorithm when it reaches an iteration budget, which yields an approximate result. From the pseudo-code of the algorithm, we can see that K-means clustering results can be sensitive to the order in which data samples in the data set are explored. A sensible practice would be to run the analysis several times, randomizing objects order; then, average the cluster centers of those runs and input the centers as initial ones for one final run of the analysis.

1.3. Illustrative example

One of the best ways to study an algorithm is trying implement it. In this section, we will go step by step to implement a simple K-means algorithm.

Question 1

Question 1.1

Complete the below function to calculate an Euclidean distance between any two points in d-dimensional data space

In [1]:

```
import numpy as np
```

```python
# calculate distance between two d-dimensional points
def euclidean_distance(p1, p2):
    return np.linalg.norm(np.array(p1)-np.array(p2))

# test our function
assert (round(euclidean_distance([1,2,3] , [10,18,12]), 2) == 20.45), "Funct
ion's wrong"
```

Question 1.2

Given a data point and the current set of centroids, complete the function below to find the index of the closest centroid for that data point.

In [3]:

```python
def find_closest_centroid(datapoint, centroids):
    # find the index of the closest centroid of the given data point.
    L=list()
    for i in range(0,len(centroids)):
        L=L+[euclidean_distance(datapoint, centroids[i])]
    return L.index(min(L))

assert(find_closest_centroid( [1,1,1], [ [2,1,2], [1,2,1], [3,1,2] ] ) ==
1), "Function's wrong"
```

Question 1.3

Write a function to randomize `k` initial centroids.

In [4]:

```python
np.random.seed(22324)
import random
# randomize initial centroids
def randomize_centroids(data, k):
    centroids =random.sample(list(data), k)
    return centroids

assert(len(
    randomize_centroids(
        np.array([
            np.array([2,1,2]),
            np.array([1,2,1]),
            np.array([3,1,2])
             ]),
        2)) == 2), "Wrong function"
```

Question 1.4

Write function `check_converge` to check the stop criteria of the algorithm.

In [5]:

```
MAX_ITERATIONS = 100

# return True if clusters have converged , otherwise, return False
def check_converge(centroids, old_centroids, num_iterations, threshold=0):
    # if it reaches an iteration budget
    if num_iterations>=MAX_ITERATIONS:
        return True
    # check if the centroids don't move (or very slightly)
    distance=list(map(lambda x,y: euclidean_distance(x,y), centroids, old_ce
ntroids))
    difference=np.linalg.norm(np.array(distance))
    if (difference<=threshold):
        return True
    return False
    # check if the centroids don't move (or very slightly)
```

Question 1.5

Write function `update_centroid` to update the new positions for the current centroids based on the position of their members.

In [6]:

```
# centroids: a list of centers
# cluster: a list of k elements. Each element i-th is a list of data points
that are assigned to center i-th
def update_centroids(centroids, cluster):

    for i in range(0,len(centroids)):
        for j in range(0,len(cluster[i])):
            centroids[i]=list(map(lambda x,y: x+y,centroids[i],cluster[i][j]
))
        centroids[i]=[x*(1/len(cluster[i])) for x in centroids[i]]
    return centroids
```

Question 1.6

Complete the K-means algorithm skeleton below, with the functions you wrote above.

In [6]:

```
# data : set of data points
# k : number of clusters
# centroids: initial list of centroids
def kmeans(data, k=2, centroids=None):
    x=True
    # randomize the centroids if they are not given
    if not centroids:
        centroids = randomize_centroids(data, k)

    old_centroids = centroids[:]

    iterations = 0
    while x==True:
        iterations += 1

        # init empty clusters
        clusters = [[] for i in range(k)]

        # assign each data point to the closest ccentroidsentroid
        for i in range(0,len(data)):
            # find the closest center of each data point
            centroid_idx = find_closest_centroid(data[i], centroids)

            # assign datapoint to the closest cluster
            clusters[centroid_idx].append(data[i])

        # keep the current position of centroids before changing them
        old_centroids = centroids

        # update centroids
        centroids = update_centroids(centroids, clusters)

        # if the stop criteria are met, stop the algorithm
        if check_converge(centroids, old_centroids, iterations, threshold=0
)==True:
            x=False

    return centroids
```

Next, we will test our algorithm on Fisher's Iris dataset, and plot the resulting clusters in 3D.

Question 1.7

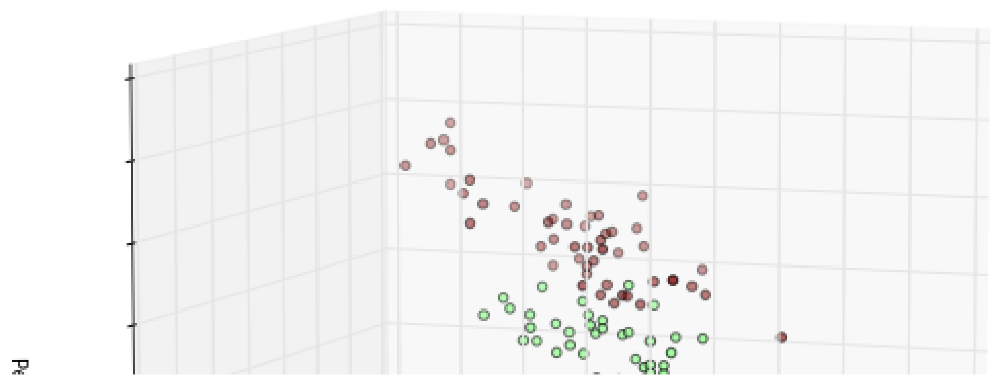
The code below can be used to test your algorithm with three different datasets: `Iris`, `Moon` and `Blob`. Run your algorithm to cluster datapoints in these datasets, plot the results and discuss about them. Do you think that our algorithm works well? Why?

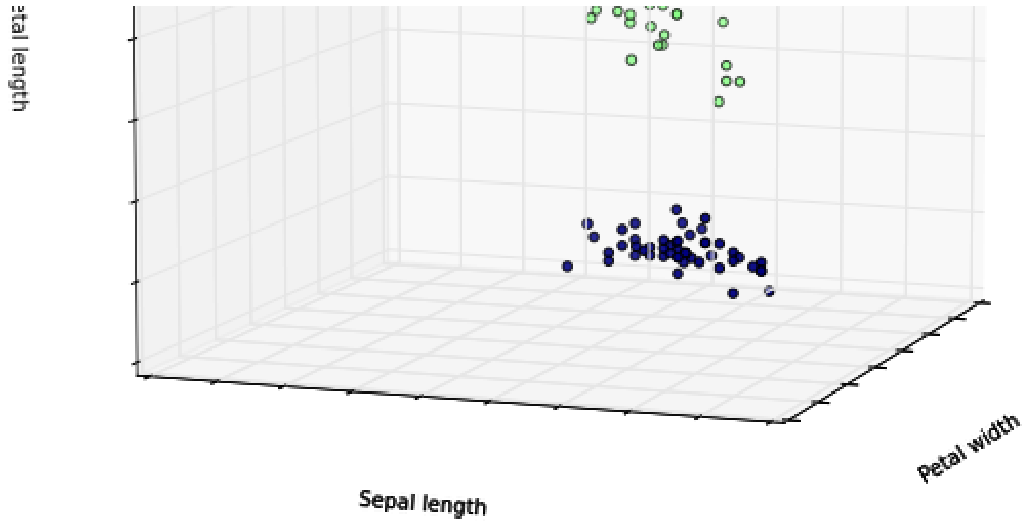
In [7]:

```
# the sourcecode in this cell is inspired from
# https://gist.github.com/bbarrilleaux/9841297

%matplotlib inline

from sklearn import datasets, cluster
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D


# load data
iris = datasets.load_iris()
X_iris = iris.data
```

```
y_iris = iris.target
# do the clustering
centers = kmeans(X_iris, k=3)
labels = [find_closest_centroid(p, centers) for p in X_iris]
#plot the clusters in color
fig = plt.figure(1, figsize=(8, 8))
plt.clf()
ax = Axes3D(fig, rect=[0, 0, 1, 1], elev=8, azim=200)
plt.cla()
ax.scatter(X_iris[:, 3], X_iris[:, 0], X_iris[:, 2], c=y_iris)

# moon
# np.random.seed(0)
# X, y = datasets.make_moons(2000, noise=0.2)

# blob
# np.random.seed(0)
# X, y = datasets.make_blobs(n_samples=2000, centers=3, n_features=20, rand
om_state=0)

# centers = kmeans(X, k=3)X_iris
# labels = [find_closest_centroid(p, centers) for p in X]

# fig = plt.figure(1, figsize=(8, 8))
# plt.clf()
# plt.scatter(X[:,0], X[:,1], s=40, c=labels, cmap=plt.cm.Spectral)

ax.w_xaxis.set_ticklabels([])
ax.w_yaxis.set_ticklabels([])
ax.w_zaxis.set_ticklabels([])
ax.set_xlabel('Petal width')
ax.set_ylabel('Sepal length')
ax.set_zlabel('Petal length')
plt.show()
```

```
/opt/conda/lib/python3.5/site-packages/sklearn/utils/fixes.py:64: Deprecati
onWarning: inspect.getargspec() is deprecated, use inspect.signature() inst
ead
  if 'order' in inspect.getargspec(np.copy)[0]:
/opt/conda/lib/python3.5/site-packages/matplotlib/collections.py:590: Futur
eWarning: elementwise comparison failed; returning scalar instead, but in t
he future will perform elementwise comparison
  if self._edgecolors == str('face'):
```





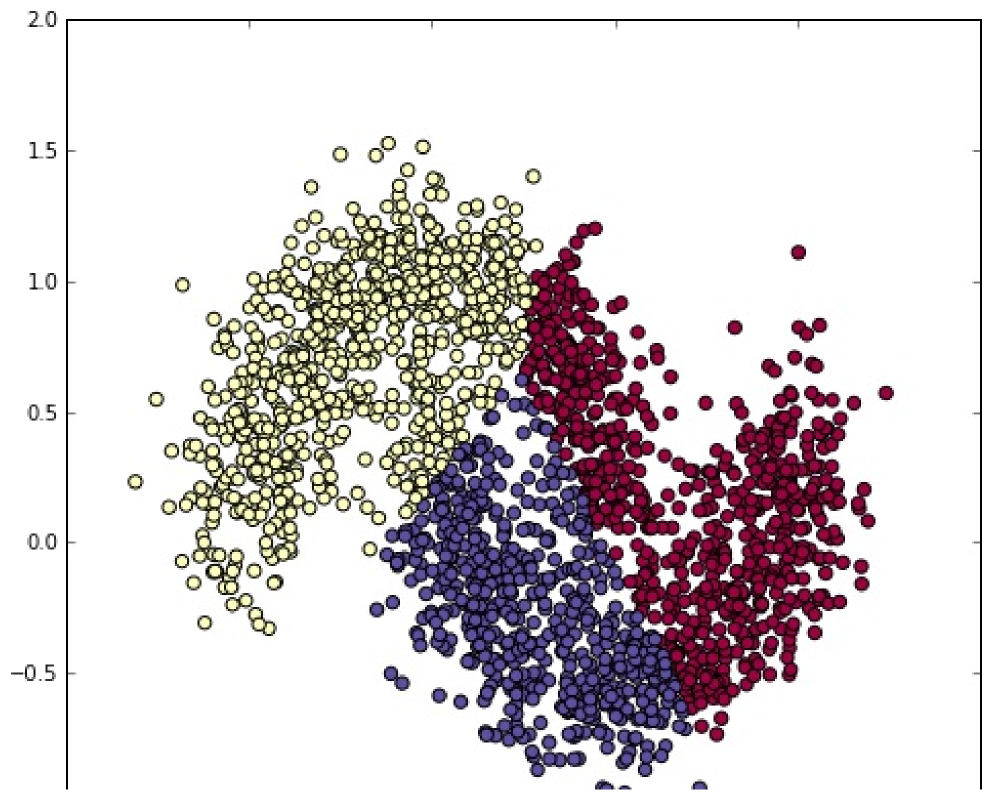
In [8]:

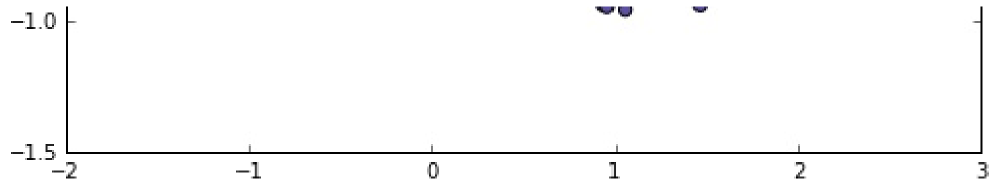
```
# moon
np.random.seed(0)
X, y = datasets.make_moons(2000, noise=0.2)
centers = kmeans(X, k=3)
labels = [find_closest_centroid(p, centers) for p in X]
fig = plt.figure(1, figsize=(8, 8))
plt.clf()
plt.scatter(X[:,0], X[:,1], s=40, c=labels, cmap=plt.cm.Spectral)
```

Out[8]:

```
<matplotlib.collections.PathCollection at 0x7faf6c6ed710>
```

```
/opt/conda/lib/python3.5/site-packages/matplotlib/collections.py:590: Futur
eWarning: elementwise comparison failed; returning scalar instead, but in t
he future will perform elementwise comparison
  if self._edgecolors == str('face'):
```

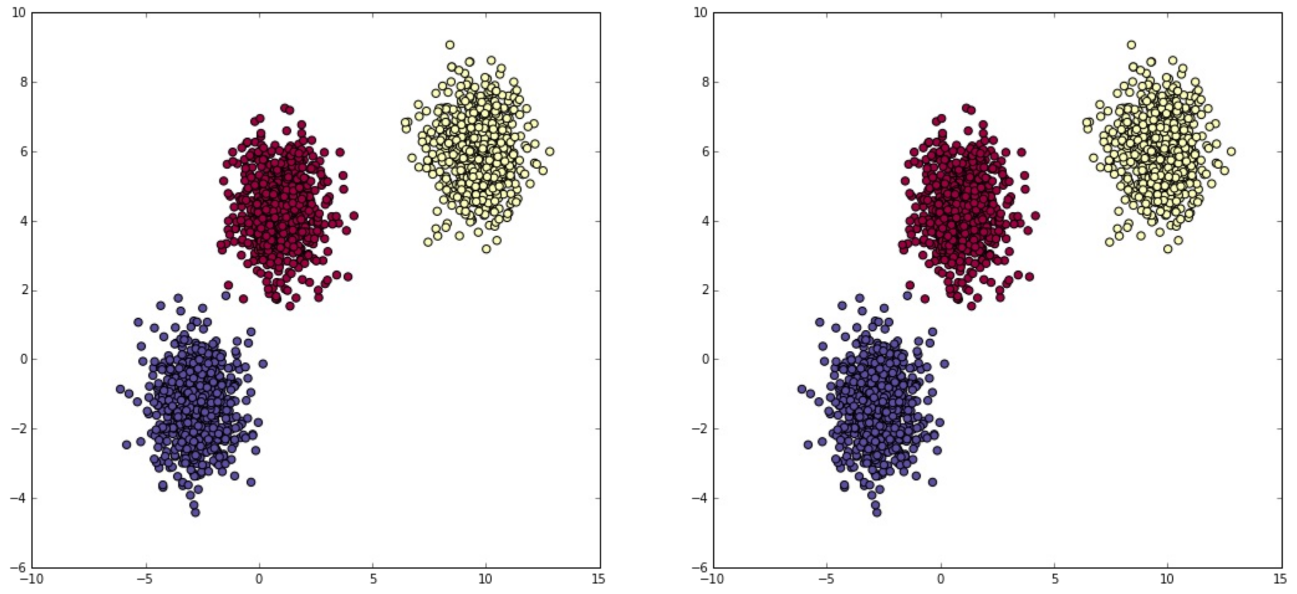




In [9]:

```
# blob
plt.figure(figsize=(18,8))
plt.subplot(1,2,1)
np.random.seed(0)
X, y = datasets.make_blobs(n_samples=2000, centers=3, n_features=20,
random_state=0)
centers = kmeans(X, k=3)
labels = [find_closest_centroid(p, centers) for p in X]
fig = plt.figure(1, figsize=(8, 8))
plt.scatter(X[:,0], X[:,1], s=40, c=labels, cmap=plt.cm.Spectral)


plt.subplot(1,2,2)
np.random.seed(3)
X, y = datasets.make_blobs(n_samples=2000, centers=3, n_features=20,
random_state=0)
centers = kmeans(X, k=3)
labels = [find_closest_centroid(p, centers) for p in X]
fig = plt.figure(1, figsize=(8, 8))
plt.scatter(X[:,0], X[:,1], s=40, c=labels, cmap=plt.cm.Spectral)

plt.show()
```

```
/opt/conda/lib/python3.5/site-packages/matplotlib/collections.py:590: Futur
eWarning: elementwise comparison failed; returning scalar instead, but in t
he future will perform elementwise comparison
  if self._edgecolors == str('face'):
```



Comment

We can see that in the first plot the clustering works quite well indeed it is divided in three groups where the elements inside are actually close to each other. This is not always true because in the green one. This is due to the randomisation of the centroids and the outcome of our algorithm it's strongly related to the set of initial centroids and data itself. In the second one the elements are really close to each other but still the algorithm is able to evaluate 3 well separated groups. For the last dataset the outcome it's strongly correleted to the initial centroids. Here, if we try to plot data we can see 3 different groups but if we try to cluster this set, the outcome of most of iterations is really bad.

That's enough about K-means for now. In the next section, we will apply MMLIB's K-means on Spark to deal with a large data in the real usecase.

2. Usecase: Network Intrusion

Some attacks attempt to flood a computer with network traffic. In some other cases, attacks attempt to exploit flaws in networking software in order to gain unauthorized access to a computer. Detecting an exploit in an incredibly large haystack of network requests is not easy.

Some exploit behaviors follow known patterns such as scanning every port in a short of time, sending a burst of request to a port... However, the biggest threat may be the one that has never been detected and classified yet. Part of detecting potential network intrusions is detecting anomalies. These are connections that aren't known to be attacks, but, do not resemble connections that have been observed in the past.

In this notebook, K-means is used to detect anomalous network connections based on statistics about each of them.

2.1. Data

The data comes from KDD Cup 1999. The dataset is about 708MB and contains about 4.9M connections. For each connection, the data set contains information like the number of bytes sent, login attempts, TCP errors, and so on. Each connection is one line of CSV-formatted data, containing 38 features: back, buffer_overflow, ftp_write, guess_passwd, imap, ipsweep, land, loadmodule, multihop, neptune, nmap, normal, perl, phf, pod, portsweep, rootkit, satan, smurf, spy, teardrop, warezclient, warezmaster. For more details about each feature, please follow this link.

Many features take on the value 0 or 1, indicating the presence or absence of a behavior such as `su_attempted` in the 15th column. Some features are counts, like `num_file_creations` in the 17th columns. Some others are the number of sent and received bytes.

2.2. Clustering without using categorical features

First, we need to import some packages that are used in this notebook.

In [2]:

```
import os
import sys
import re
from pyspark import SparkContext
from pyspark import SparkContext
from pyspark.sql import SQLContext
from pyspark.sql.types import *
from pyspark.sql import Row
from pyspark.sql.functions import *
%matplotlib inline
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import pyspark.sql.functions as func
import matplotlib.patches as mpatches
from pyspark.mllib.clustering import KMeans, KMeansModel

input_path = "/datasets/k-means/kddcup.data"
raw_data = sc.textFile(input_path, 12)
```

2.2.1. Loading data

There are two types of features: numerical features and categorical features. Currently, to get familiar with the data and the problem, we only use numerical features. In our data, we also have pre-defined groups for each connection, which we can use later as our "ground truth" for verifying our results.

Note 1: we don't use the labels in the training phase!!!

Note 2: in general, since clustering is un-supervised, you don't have access to ground truth. For this reason, several metrics to judge the quality of clustering have been devised. For a short overview of such metrics, follow this link. Note that computing such metrics, that is trying to assess the quality of your clustering results, is as computationally intensive as computing the clustering itself!

Question 2

Write function `parseLine` to construct a tuple of `(label, vector)` for each connection, extract the data that contains only the data points (without label), then print the number of connections.

Where,

- `label` is the pre-defined label of each connection
- `vector` is a numpy array that contains values of all features, but the label and the categorial features at index `1,2,3` of each connection. Each `vector` is a data point.

In [3]:

```
def parseLine(line):
    cols = line.split(',')
    # label is the last column
    label = cols[-1]

    # vector is every column, except the label
    # vector is every column, except the label
    vector = cols[:len(cols)-1]

    # delete values of columns that have index 1->3 (categorical features)
    del(vector[1:4])

    # convert each value from string to float
    vector = np.array(vector)
    vector=vector.astype(np.float)

    return (label, vector)

labelsAndData = raw_data.map(lambda x:parseLine(x))

# we only need the data, not the label
data = labelsAndData.map(lambda x:x[1]).cache()

# number of connections
n = data.count()
print(n)
```

```
4898431
```

In [12]:

```
data.take(1)
```

Out[12]:

```
[array([  0.00000000e+00,   2.15000000e+02,   4.50760000e+04,
          0.00000000e+00,   0.00000000e+00,   0.00000000e+00,
          0.00000000e+00,   0.00000000e+00,   1.00000000e+00,
          0.00000000e+00,   0.00000000e+00,   0.00000000e+00,
          0.00000000e+00,   0.00000000e+00,   0.00000000e+00,
          0.00000000e+00,   0.00000000e+00,   0.00000000e+00,
          0.00000000e+00,   1.00000000e+00,   1.00000000e+00,
          0.00000000e+00,   0.00000000e+00,   0.00000000e+00,
          0.00000000e+00,   1.00000000e+00,   0.00000000e+00,
          0.00000000e+00,   0.00000000e+00,   0.00000000e+00,
          0.00000000e+00,   0.00000000e+00,   0.00000000e+00,
          0.00000000e+00,   0.00000000e+00,   0.00000000e+00,
          0.00000000e+00,   0.00000000e+00])]
```

Question 3

Using K-means algorithm of MLLIB, cluster the connections into two groups then plot the result. Why two groups? In this case, we are just warming up, we're testing things around, so "two groups" has no particular meaning.

You can use the following parameters:

- `maxIterations=10`
- `runs=10`
- `initializationMode="random"`

Discuss the result from your figure.

In [56]:

```
model_clusters = KMeans.train(data, 2, 10,initializationMode="random")
```

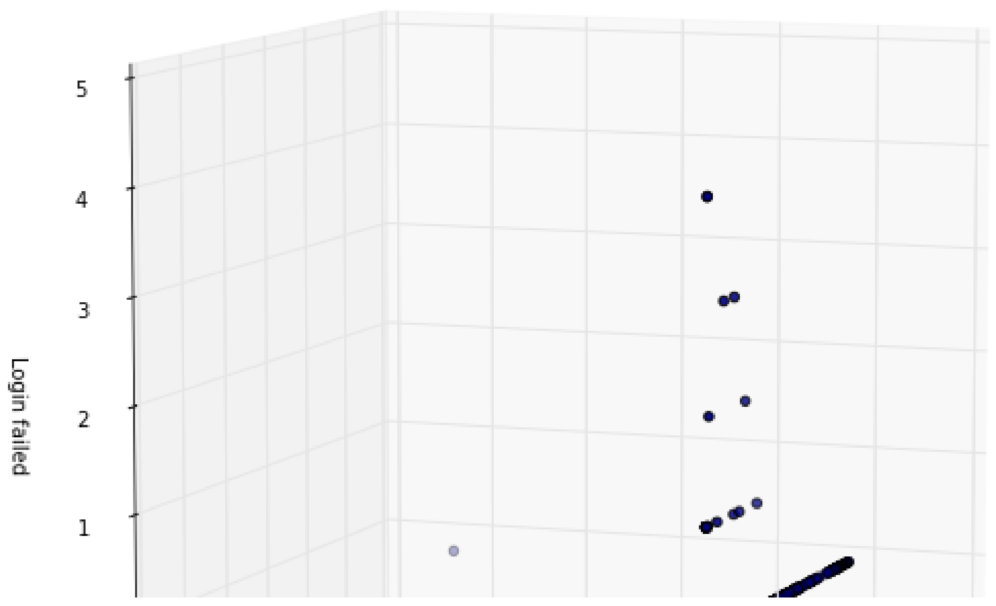
In [50]:

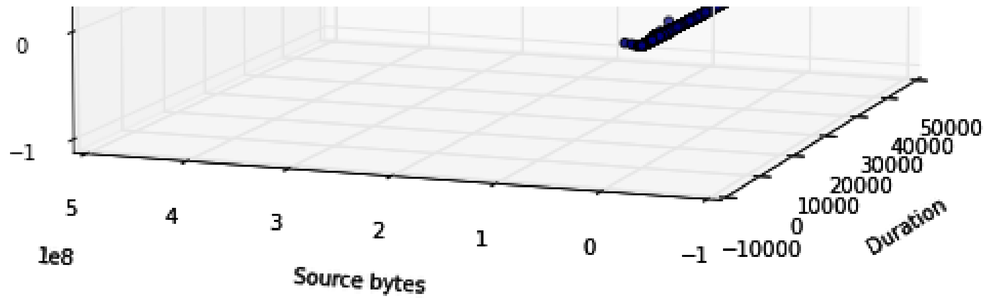
```
import random
# do the clustering
centers = model_clusters.centers
SAMPLE_SIZE = 0.3
RANDOM_SEED = 42

data2 = np.array(data.sample(False, SAMPLE_SIZE, random.seed(RANDOM_SEED)).
collect())
labels = data.map(lambda p: model_clusters.predict(p)).sample(False,
SAMPLE_SIZE,random.seed(RANDOM_SEED)).collect()

num = len(data2[:, 10])
```

In [51]:

```
#plot the clusters in color
fig = plt.figure(1, figsize=(8, 8))
plt.clf()
ax = Axes3D(fig, rect=[0, 0, 1, 1], elev=8, azim=200)
plt.cla()
ax.scatter(data2[:, 0], data2[:, 1], data2[:, 7], c=labels)

ax.set_xlabel('Duration')
ax.set_ylabel('Source bytes')
ax.set_zlabel('Login failed')

plt.show()
```

```
/opt/conda/lib/python3.5/site-packages/matplotlib/collections.py:590: Futur
eWarning: elementwise comparison failed; returning scalar instead, but in t
he future will perform elementwise comparison
  if self._edgecolors == str('face'):
```





Comment

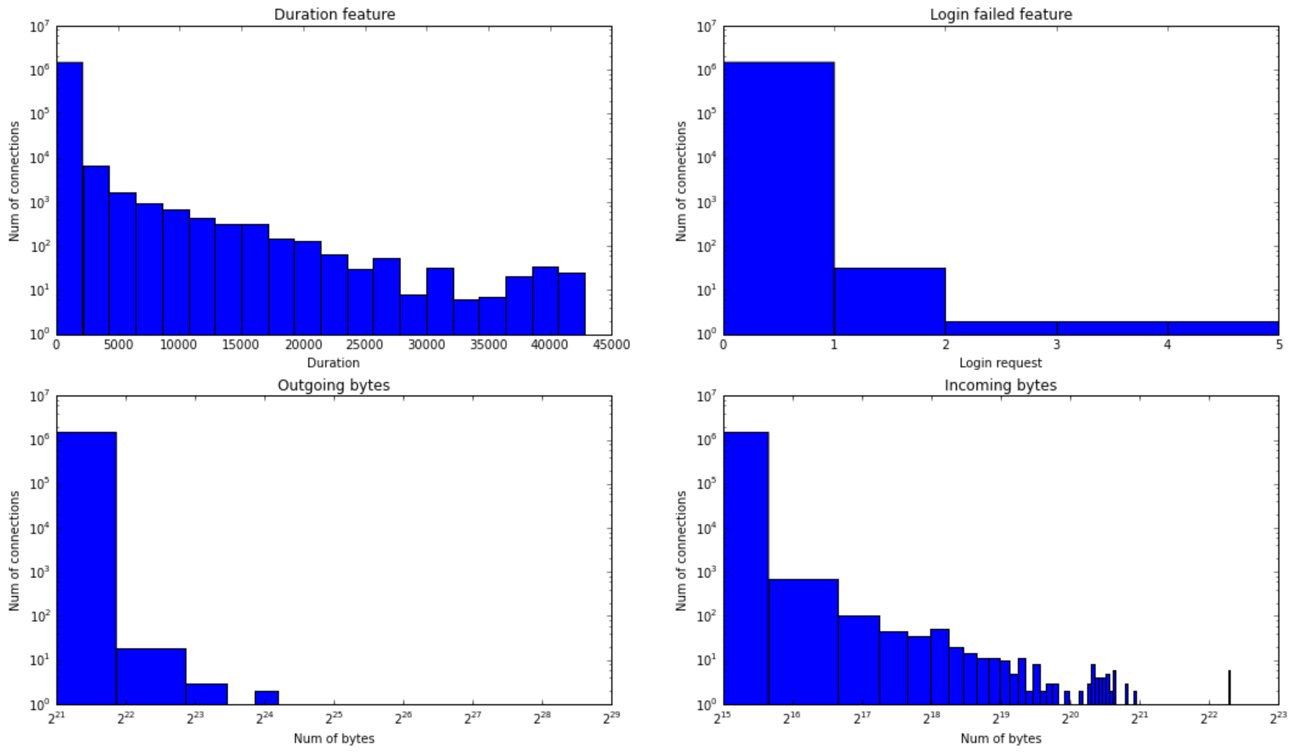
This is the 3D representation of the clustering we decided to put on the axis some feature that we think are relevant for identify an network intrusion.

In [52]:

```
plt.figure(figsize=(18,10))

plt.subplot(2, 2, 1)
plt.hist(data2[:, 0], 20)
plt.yscale('log', nonposy='clip')
plt.title("Duration feature")
plt.xlabel("Duration")
plt.ylabel("Num of connections")

plt.subplot(2, 2, 2)
plt.hist(data2[:, 7], range(6))
plt.yscale('log', nonposy='clip')
plt.title("Login failed feature")
plt.xlabel("Login request")
plt.ylabel("Num of connections")

fig = plt.subplot(2, 2, 3)
plt.hist(data2[:, 1], 100)
plt.yscale('log', nonposy='clip')
fig.set_xscale('log', basex=2)
plt.title("Outgoing bytes")
plt.xlabel("Num of bytes")
plt.ylabel("Num of connections")

fig = plt.subplot(2, 2, 4)
plt.hist(data2[:, 2], 100)
plt.yscale('log', nonposy='clip')
fig.set_xscale('log', basex=2)
plt.title("Incoming bytes")
plt.xlabel("Num of bytes")
plt.ylabel("Num of connections")

num_root = len(list(filter(lambda x: x==1, data2[:, 10])))
print("Percentage of login with root privileges: ", num_root/num*100)
num_hot = len(list(filter(lambda x: x==1, data2[:, 17])))
print("Percentage of hot login: ", num_hot/num*100)
```

```
num_guest = len(list(filter(lambda x: x==1, data2[:, 18])))
print("Percentage of guest login: ", num_guest/num*100)
num_out_data = len(list(filter(lambda x: x>0, data2[:, 1])))
print("Percentage of connection width outgoing byte greater than 0: ", num_
out_data/num*100)

plt.show()
```

```
Percentage of login with root privileges:  0.006467411394761942
Percentage of hot login:  6.807801468170464e-05
Percentage of guest login:  0.08577829849894786
Percentage of connection width outgoing byte greater than 0:
76.49313807651016
```



Comment

We didn't plot the firsts features because even if we think that they are very useful for the intrusion detection they are boolean feature: - login with root privileges: the value is really low as expected, most of connection should be indeed with user privileges; - hot login: user in the hot list; - guest login: logins of not registered user, so possible attackers; Then we plotted the duration of the connections and most of them lasts less then one hour but we can see that a non negligible number of connections last 11 hours. Form login failure plot we could say that there was no attempt of brute force attack on some user password because the maximum value of failure is 5. For the incoming bytes most of connection are in the order of hundreds of kB instead the outgoing connection are in the order of MB.

2.2.3. Evaluating model

Question 4

One of the simplest method to evaluate our result is calculate the Within Set Sum of Squared Errors (WSSSE), or simply, 'Sum of Squared Errors'. An error of a data point is defined as it's distance to the closest cluster center.

In [57]:

```
from operator import add

# Evaluate clustering by computing Within Set Sum of Squared Errors
def error(clusters, point):
    closest_center = clusters.centers[clusters.predict(point)]
    return np.sum([x**2 for x in (point - closest_center)])

WSSSE = data.map(lambda x : error(model_clusters,x)).reduce(add)
print("Within Set Sum of Squared Error = " + str(WSSSE))
```

```
Within Set Sum of Squared Error = 6.37937321457e+18
```

Question 5

This is a good opportunity to use the given labels to get an intuitive sense of what went into these two clusters, by counting the labels within each cluster. Complete the following code that uses the model to assign each data point to a cluster, and counts occurrences of cluster and label pairs. What do you think about the result?

In [18]:

```
clusterLabelCount = data.map(lambda x :
(model_clusters.predict(x),1)).reduceByKey(lambda freq1, freq2: freq1 +
freq2).collect()
labelpairs=labelsAndData.map(lambda x : (x[0],1)).reduceByKey(lambda freq1,
freq2: freq1 + freq2).collect()
```

In [19]:

```
for item in clusterLabelCount:
    print(item)
for item in labelpairs:
    print(item)
```

```
(0, 2925804)
(1, 1972627)
('guess_passwd.', 53)
('loadmodule.', 9)
('spy.', 2)
('phf.', 4)
('back.', 2203)
('pod.', 264)
```

```
('nmap.', 2316)
('smurf.', 2807886)
('satan.', 15892)
('land.', 21)
('warezmaster.', 20)
('ipsweep.', 12481)
('teardrop.', 979)
('multihop.', 7)
('neptune.', 1072017)
('buffer_overflow.', 30)
('perl.', 3)
('rootkit.', 10)
('imap.', 12)
('warezclient.', 1020)
('ftp_write.', 8)
('normal.', 972781)
('portsweep.', 10413)
```

In [58]:

```
cluster0=labelsAndData.map(lambda x : (x[0],model_clusters.predict(x[1]))).f
ilter(lambda x:x[1]==0).cache()
cluster1=labelsAndData.map(lambda x : (x[0],model_clusters.predict(x[1]))).f
ilter(lambda x:x[1]==1).cache()
```

In [59]:

```
cluster0=cluster0.map(lambda x : (x[0],1)).reduceByKey(lambda freq1, freq2:
freq1 + freq2).collect()
cluster1=cluster1.map(lambda x : (x[0],1)).reduceByKey(lambda freq1, freq2:
freq1 + freq2).collect()
```

In [60]:

```
for item in cluster1:
    print(item)
print("\n")
for item in cluster0:
    print(item)
```

```
('guess_passwd.', 53)
('loadmodule.', 9)
('phf.', 4)
('spy.', 2)
('nmap.', 2316)
('satan.', 15885)
('land.', 21)
('warezmaster.', 20)
('ipsweep.', 12481)
('teardrop.', 979)
('neptune.', 1072016)
('multihop.', 6)
('buffer_overflow.', 30)
('perl.', 3)
('rootkit.', 10)
('imap.', 11)
('warezclient.', 897)
('ftp_write.', 6)
('normal.', 857475)
('portsweep.', 10403)
```

```
('back.', 2203)
('smurf.', 2807886)
('pod.', 264)
('satan.', 7)
('multihop.', 1)
('neptune.', 1)
('imap.', 1)
('warezclient.', 123)
('ftp_write.', 2)
('normal.', 115306)
('portsweep.', 10)
```

The result seems to be good because the two clusters have enough elements even if there is not an equivalent concentration between them. We could not expect a very good result trying to cluster the datasets in 2 groups because we have a lot of connections and features and 2 groups are not enough to describe them.

We can see by comparing the label pairs in the two clusters that there are more labels in the first cluster. Most of the data in cluster1 is labelled 'smurf.' (2 807 886 out of 2 925 804)

2.2.4. Choosing K

How many clusters are appropriate for a dataset? In particular, for our own dataset, it's clear that there are 23 distinct behavior patterns in the data, so it seems that k could be at least 23, or likely, even more. In other cases, we even don't have any information about the number of patterns at all (remember, generally your data is not labelled!). Our task now is finding a good value of $k$. For doing that, we have to build and evaluate models with different values of $k$. A clustering could be considered good if each data point were near to its closest centroid. One of the ways to evaluate a model is calculating the Mean of Squared Errors of all data points.

Question 6

Complete the function below to calculate the MSE of each model that is corresponding to each value of $k$. Plot the results. From the obtained result, what is the best value for $k$? Why?

In [61]:

```
# k: the number of clusters
def clusteringScore(data, k):
    clusters = KMeans.train(data, k, 10,initializationMode="random")
    # calculate mean square error
    return data.map(lambda x : error(clusters,x)).reduce(add)
```

In [69]:

```
scores = [clusteringScore(data, k) for k in [50,60,70]]
```

In [73]:

```
for item in scores:
    print(item)
```

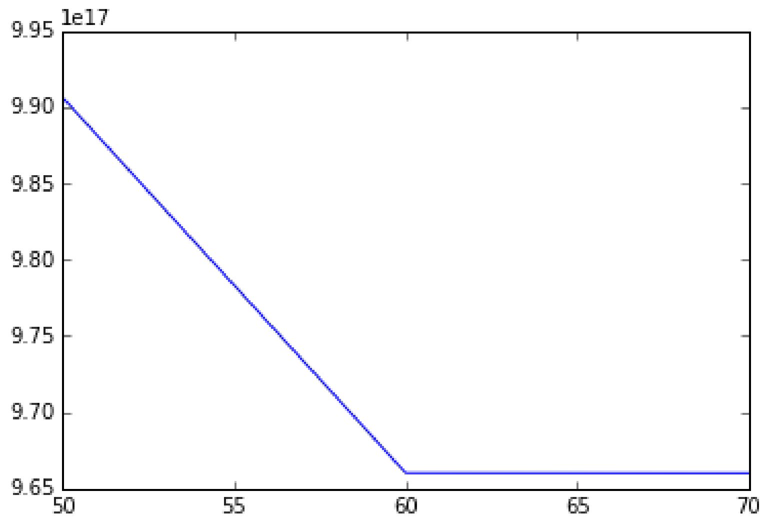
```
9.90622369892e+17
9.66015654168e+17
9.66004619428e+17
```

In [67]:

```
# plot results
plt.plot([50,60,70],scores)
```

Out[67]:

```
[<matplotlib.lines.Line2D at 0x7faf5bfb8c50>]
```



As expected the error decreses as we increse the number of cluster. But the higher is the number of clusters K, the higher is the cost of running Kmeans (There is a trade-off to be made between the accuracy and the cost). In the case of K=70 the curve is slightly higher compared with k=50 that's because the algorithm in that iteration converged in an unfortunate situation probably due to the initial centroids.

2.2.5 Normalizing features

K-means clustering treats equally all dimensions/directions of the space and therefore tends to produce more or less spherical (rather than elongated) clusters. In this situation, leaving variances uneven is equivalent to putting more weight on variables with smaller variance, so clusters will tend to be separated along variables with greater variance.

In our notebook, since Euclidean distance is used, the clusters will be influenced strongly by the magnitudes of the variables, especially by outliers. Normalizing will remove this bias.

Each feature can be normalized by converting it to a standard score. This means subtracting the mean of the feature’s values from each value, and dividing by the standard deviation

$normalize_i=\frac{feature_i - \mu_i}{\sigma_i}$

Where.

- $normalize_i$ is the normalized value of feature $i$
- $\mu_i$ is the mean of feature $i$
- $\sigma_i$ is the standard deviation of feature $i$

Question 7

Complete the code below to normalize the data. Print the first 5 lines of the new data.

HINT If $\sigma_i = 0$ then $normalize_i=feature_i - \mu_i$

In [9]:

```
from operator import add
```

In [10]:

```
def normalizeData(data):
    # number of connections
    n = data.count()

    # calculate the sum of each feature
    sums = data.reduce(add)
    print(sums)

    # calculate means
    means =[x*(1/n) for x in sums]

    # calculate the sum square of each feature
    sumSquares =data.map(lambda x: (x-means)**2).reduce(add)
    print(sumSquares)
    variance=[(x/n) for x in sumSquares]
    # calculate standard deviation of each feature
    stdevs = [np.sqrt(x) for x in variance]
    print(stdevs)

    def normalize(point):
        for i in range(0,len(stdevs)):
            if stdevs[i]==0:
                stdevs[i]=1
        return (point-means)/stdevs

    return data.map(normalize)

normalizedData = normalizeData(data).cache()
print(normalizedData.take(5))
```

```
[   2.36802060e+08    8.98676524e+09    5.35703589e+09    2.80000000e+01
    3.17800000e+03    3.90000000e+01    6.09250000e+04    1.57000000e+02
```

```
   7.03067000e+05   3.96200000e+04   3.34000000e+02   1.80000000e+02
   6.33610000e+04   5.82300000e+03   3.64000000e+02   5.00200000e+03
   0.00000000e+00   2.00000000e+00   4.09100000e+03   1.64084428e+09
   1.44634545e+09   8.71775140e+05   8.72101730e+05   2.82468470e+05
   2.82786920e+05   3.86919313e+06   1.03746840e+05   1.38433600e+05
   1.14124176e+09   9.26852923e+08   3.69201228e+06   1.50436230e+05
   2.96380553e+06   3.16639800e+04   8.72367200e+05   8.71361620e+05
   2.83755350e+05   2.82440660e+05]
[  2.56288804e+12   4.34144161e+18   2.03794728e+18   2.79998399e+01
   8.99593818e+03   2.54999689e+02   1.07736324e+06   2.60994968e+02
   6.02156483e+05   7.28516355e+07   3.33977226e+02   3.19993386e+02
   7.59669934e+07   7.55440779e+04   3.73972951e+02   6.17689224e+03
   0.00000000e+00   1.99999918e+00   4.08758334e+03   2.20135894e+11
   2.96415821e+11   7.14332989e+05   7.15749707e+05   2.64228272e+05
   2.65156244e+05   7.42363185e+05   3.35136000e+04   9.67782387e+04
   2.00770991e+10   5.49482085e+10   8.28196854e+05   5.77114862e+04
   1.13324779e+06   8.33893726e+03   7.14193132e+05   7.15462699e+05
   2.61255735e+05   2.61334681e+05]
[723.32973742373622, 941430.97839223535, 645012.26790443365,
0.0023908331945205314, 0.042854332380513317, 0.0072150829486976121, 0.46897
811672005524, 0.0072994068308639103, 0.35061152330824669,
3.8564806417008048, 0.0082571453448041215, 0.0080824309543176768, 3.9380748
990133552, 0.12418573580623804, 0.0087375887179610111,
0.035510477731763503, 0.0, 0.00063897874457037737, 0.028887157723358654, 21
1.99076084188857, 245.99268506864763, 0.38187555255765865,
0.38225404703536964, 0.23225289950485706, 0.23266037928851638,
0.3892957976533023, 0.082714575134118951, 0.14055955047916366,
64.020930815773255, 105.91275654036558, 0.41118597274525992,
0.10854320270282473, 0.48098766909351814, 0.041259775034444515, 0.381838167
69577189, 0.38217739927266686, 0.23094279563844014, 0.23097768597343582]
[array([ -6.68331854e-02,  -1.72038228e-03,   6.81884351e-02,
        -2.39084686e-03,  -1.51391734e-02,  -1.10348462e-03,
        -2.65207600e-02,  -4.39091558e-03,   2.44279187e+00,
        -2.09732783e-03,  -8.25770840e-03,  -4.54646139e-03,
        -3.28458917e-03,  -9.57233922e-03,  -8.50457842e-03,
        -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
        -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
        -4.66042614e-01,  -4.65755574e-01,  -2.48285775e-01,
        -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
        -2.01059296e-01,  -3.63913926e+00,  -1.78651044e+00,
        -1.83302273e+00,  -2.82939000e-01,  -1.25793664e+00,
        -1.56668488e-01,  -4.66404784e-01,  -4.65453641e-01,
        -2.50831829e-01,  -2.49631966e-01]), array([ -6.68331854e-02,  -1.7
7667956e-03,   5.32451452e-03,
        -2.39084686e-03,  -1.51391734e-02,  -1.10348462e-03,
        -2.65207600e-02,  -4.39091558e-03,   2.44279187e+00,
        -2.09732783e-03,  -8.25770840e-03,  -4.54646139e-03,
        -3.28458917e-03,  -9.57233922e-03,  -8.50457842e-03,
        -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
        -2.89113034e-02,  -1.57069789e+00,  -1.19217808e+00,
        -4.66042614e-01,  -4.65755574e-01,  -2.48285775e-01,
        -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
        -2.01059296e-01,  -3.62351937e+00,  -1.77706870e+00,
         5.98966843e-01,  -2.82939000e-01,   8.21118739e-01,
        -1.56668488e-01,  -4.66404784e-01,  -4.65453641e-01,
        -2.50831829e-01,  -2.49631966e-01]), array([ -6.68331854e-02,  -1.6
9807581e-03,   2.08332760e-04,
        -2.39084686e-03,  -1.51391734e-02,  -1.10348462e-03,
        -2.65207600e-02,  -4.39091558e-03,   2.44279187e+00,
        -2.09732783e-03,  -8.25770840e-03,  -4.54646139e-03,
        -3.28458917e-03,  -9.57233922e-03,  -8.50457842e-03,
        -3.28458917e-03,  -9.57233922e-03,  -8.50457842e-03,
        -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
        -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
        -4.66042614e-01,  -4.65755574e-01,  -2.48285775e-01,
        -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
        -2.01059296e-01,  -3.60789948e+00,  -1.76762697e+00,
         5.98966843e-01,  -2.82939000e-01,  -2.18408950e-01,
        -1.56668488e-01,  -4.66404784e-01,  -4.65453641e-01,
        -2.50831829e-01,  -2.49631966e-01]), array([ -6.68331854e-02,  -1.7
0126245e-03,   1.45482068e-03,
        -2.39084686e-03,  -1.51391734e-02,  -1.10348462e-03,
        -2.65207600e-02,  -4.39091558e-03,   2.44279187e+00,
        -2.09732783e-03,  -8.25770840e-03,  -4.54646139e-03,
        -3.28458917e-03,  -9.57233922e-03,  -8.50457842e-03,
        -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
        -2.89113034e-02,  -1.57069789e+00,  -1.19217808e+00,
        -4.66042614e-01,  -4.65755574e-01,  -2.48285775e-01,
        -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
        -2.01059296e-01,  -3.59227958e+00,  -1.75818524e+00,
         5.98966843e-01,  -2.82939000e-01,  -5.71848364e-01,
        -1.56668488e-01,  -4.66404784e-01,  -4.65453641e-01,
        -2.50831829e-01,  -2.49631966e-01]), array([ -6.68331854e-02,  -1.6
9488918e-03,  -9.42032957e-04,
        -2.39084686e-03,  -1.51391734e-02,  -1.10348462e-03,
        -2.65207600e-02,  -4.39091558e-03,   2.44279187e+00,
        -2.09732783e-03,  -8.25770840e-03,  -4.54646139e-03,
        -3.28458917e-03,  -9.57233922e-03,  -8.50457842e-03,
        -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
        -2.89113034e-02,  -1.56598070e+00,  -1.18811292e+00,
        -4.66042614e-01,  -4.65755574e-01,  -2.48285775e-01,
        -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
        -2.01059296e-01,  -3.57665969e+00,  -1.74874350e+00,
         5.98966843e-01,  -2.82939000e-01,  -7.38172794e-01,
        -1.56668488e-01,  -4.66404784e-01,  -4.65453641e-01,
        -2.50831829e-01,  -2.49631966e-01])]
```

Question 8

Using the new data, build different models with different values of $k \in [60,70,80,90,100,110]$. Evaluate the results by plotting them and choose the best value of k.

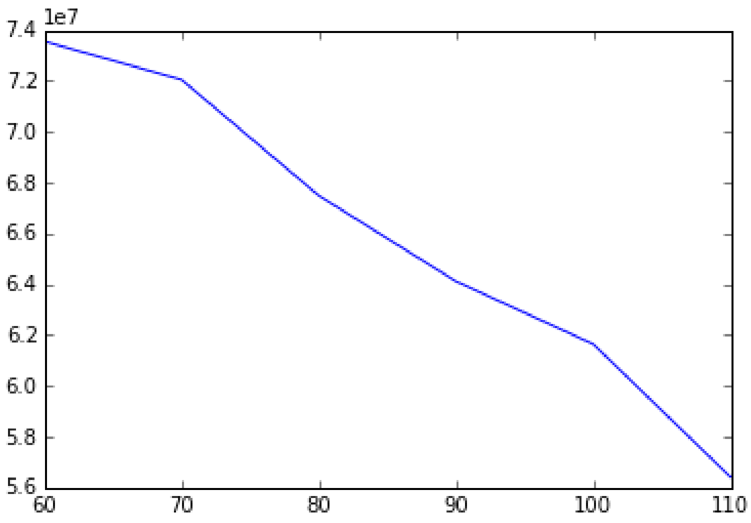
In [47]:

```python
scores = [clusteringScore(normalizedData, k) for k in [60,70,80,90,100,110]]
for score in scores:
    print(score)
plt.plot([60,70,80,90,100,110],scores)
```

```
73552436.9967
72045959.0331
67485919.529
64110959.7966
61642981.7933
56414424.306
```

Out[47]:

```
[<matplotlib.lines.Line2D at 0x7f6fdd177cc0>]
```



The best value of k is 110. As in the previous case the trend of the curve is the same but we can see that the range of values is completly different (1e07 instead of 1e17) due to the normalization of the data. The curve is always decreasing because we normalized the data and removed outliers.

Question 9

Plot the clustering result to see the difference between before and after normalizing features. Discuss about the difference and explain why and if normalization was useful.

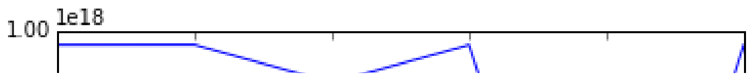
In [52]:

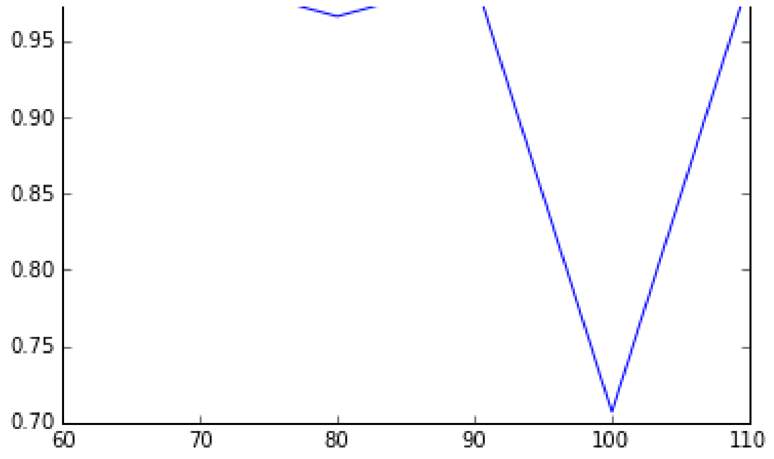
```
scores = [clusteringScore(data, k) for k in [60,70,80,90,100,110]]
for score in scores:
    print(score)
plt.plot([60,70,80,90,100,110],scores)
```

```
9.90484381914e+17
9.90561404052e+17
9.66011504232e+17
9.90525607156e+17
7.06824145889e+17
9.90453230446e+17
```

Out[52]:

```
[<matplotlib.lines.Line2D at 0x7f6fdd06c438>]
```





in the previous question, we plot the scoring curve for normalized data. When using the normalized Data, k=110 is the best value of k, but when using the data without normalization k=100 is the best choice. Normalization was useful, because we see that when using the not normalized data the curve is not monotonous because of outliers, but in the normalized data we get a decreasing curve.

2.3. Clustering using categorical features

2.3.1 Loading data

In the previous section, we ignored the categorical features of our data: this is not a good idea, since these categorical features can be important in providing useful information for clustering. The problem is that K-means (or at least, the one we have developed and the one we use from MLLib) only work with data points in a metric space. Informally, this means that operations such as addition, subtraction and computing the mean of data points are trivial and well defined. For a more formal definition of what a metric space is, follow this link.

What we will do next is to transform each categorical feature into one or more numerical features. This approach is very widespread: imagine for example you wanted to use K-means to cluster text data. Then, the idea is to transform text data in d-dimensional vectors, and a nice way to do it is to use word2vec. If you're interested, follow this link to a nice blog post on the problem.

There are two approaches:

- **Approach 1**: mapping **one** categorical feature to **one** numerical feature. The values in each categorical feature are encoded into unique numbers of the new numerical feature. For example, ['VERY HOT','HOT', 'COOL', 'COLD', 'VERY COLD'] will be encoded into [0,1,2,3,4,5]. However, by using this method, we implicit assume that the value of 'VERY HOT' is smaller than 'HOT'... This is not generally true.
- **Approach 2**: mapping **one** categorical feature to **multiple** numerical features. Basically, a single variable with n observations and d distinct values, to d binary variables with n observations each. Each observation indicating the presence (1) or absence (0) of the d^{th} binary variable. For example, ['house', 'car', 'tooth', 'car'] becomes

```
[
[1,0,0,0],
[0,1,0,0],
[0,0,1,0],
[0,0,0,1],
]
```

We call the second approach "one-hot encoding". By using this approach, we keep the same role for all values of categorical features.

Question 10

Calculate the number of distinct categorical features value (at index `1,2,3`). Then construct a new input data using one-hot encoding for these categorical features (don't throw away numerical features!).

In [11]:

```
vColumn1 = raw_data.map(lambda x: x[1]).distinct().collect()
numValuesColumn1 = len(vColumn1)
vColumn1 = dict(zip(vColumn1, range(0, numValuesColumn1)))


vColumn2 = raw_data.map(lambda x: x[2]).distinct().collect()
numValuesColumn2 = len(vColumn2)
vColumn2 = dict(zip(vColumn2, range(0, numValuesColumn2)))



vColumn3 = raw_data.map(lambda x: x[3]).distinct().collect()
numValuesColumn3 = len(vColumn3)
vColumn3 = dict(zip(vColumn3, range(0, numValuesColumn3)))





def parseLineWithHotEncoding(line):
    cols = line.split(',')
    # label is the last column
    label = cols[-1]

    vector = cols[0:-1]
    featureOfCol1 = [0]*numValuesColumn1
    featureOfCol2 = [0]*numValuesColumn2
    featureOfCol3 = [0]*numValuesColumn3
    featureOfCol1[vColumn1[vector[1]]] = 1
    featureOfCol2[vColumn2[vector[2]]] = 1
    featureOfCol3[vColumn3[vector[3]]] = 1

    vector = ([vector[0]] + featureOfCol1 + featureOfCol2 +
        featureOfCol3 + vector[4:])

    # convert each value from string to float
    vector = np.array(list(map(lambda x: float(x), vector)))

    return (label, vector)

labelsAndData = raw_data.map(parseLine)

# we only need the data, not the label
data = labelsAndData.values().cache()
```

In [12]:

```
normalizedData = normalizeData(data).cache()
```

```
[  2.36802060e+08   8.98676524e+09   5.35703589e+09   2.80000000e+01
   3.17800000e+03   3.90000000e+01   6.09250000e+04   1.57000000e+02
   7.03067000e+05   3.96200000e+04   3.34000000e+02   1.80000000e+02
   6.33610000e+04   5.82300000e+03   3.64000000e+02   5.00200000e+03
   0.00000000e+00   2.00000000e+00   4.09100000e+03   1.64084428e+09
   1.44634545e+09   8.71775140e+05   8.72101730e+05   2.82468470e+05
   2.82786920e+05   3.86919313e+06   1.03746840e+05   1.38433600e+05
   1.14124176e+09   9.26852923e+08   3.69201228e+06   1.50436230e+05
   2.96380553e+06   3.16639800e+04   8.72367200e+05   8.71361620e+05
   2.83755350e+05   2.82440660e+05]
[  2.56288804e+12   4.34144161e+18   2.03794728e+18   2.79998399e+01
   8.99593818e+03   2.54999689e+02   1.07736324e+06   2.60994968e+02
   6.02156483e+05   7.28516355e+07   3.33977226e+02   3.19993386e+02
   7.59669934e+07   7.55440779e+04   3.73972951e+02   6.17689224e+03
   0.00000000e+00   1.99999918e+00   4.08758334e+03   2.20135894e+11
   2.96415821e+11   7.14332989e+05   7.15749707e+05   2.64228272e+05
   2.65156244e+05   7.42363185e+05   3.35136000e+04   9.67782387e+04
   2.00770991e+10   5.49482085e+10   8.28196854e+05   5.77114862e+04
   1.13324779e+06   8.33893726e+03   7.14193132e+05   7.15462699e+05
   2.61255735e+05   2.61334681e+05]
[723.32973742373622, 941430.97839223535, 645012.26790443365,
0.0023908331945205314, 0.042854332380513317, 0.0072150829486976121, 0.46897
811672005524, 0.0072994068308639103, 0.35061152330824669,
3.8564806417008048, 0.0082571453448041215, 0.0080824309543176768, 3.9380748
990133552, 0.12418573580623804, 0.0087375887179610111,
0.035510477731763503, 0.0, 0.00063897874457037737, 0.028887157723358654, 21
1.99076084188857, 245.99268506864763, 0.38187555255765865,
0.38225404703536964, 0.23225289950485706, 0.23266037928851638,
0.3892957976533023, 0.082714575134118951, 0.14055955047916366,
64.020930815773255, 105.91275654036558, 0.41118597274525992,
0.10854320270282473, 0.48098766909351814, 0.041259775034444515, 0.381838167
69577189, 0.38217739927266686, 0.23094279563844014, 0.23097768597343582]
```

2.3.2. Building models

Question 11

Using the new data, cluster the connections with different values of $k \in [80,90,100,110,120,130,140,150,160]$. Evaluate the results and choose the best value of k as previous questions.

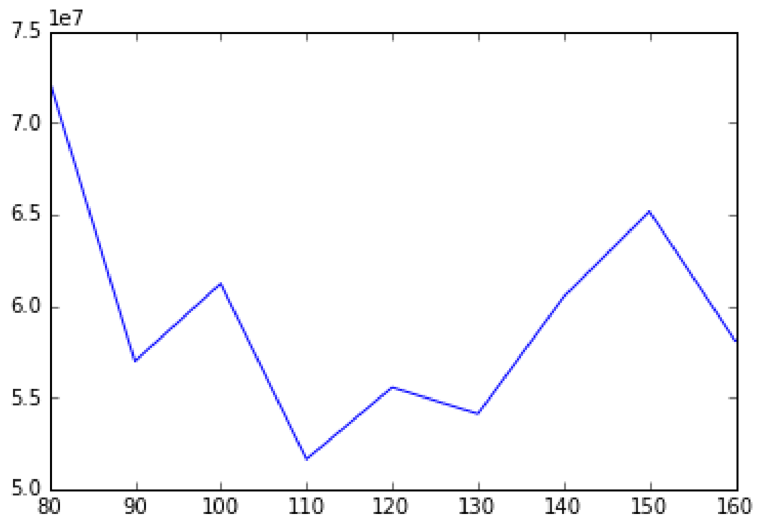
In [51]:

```
scores = [clusteringScore(normalizedData, k) for k in [80,90,100,110,120,130
,140,150,160]]
for score in scores:
    print(score)
plt.plot([80,90,100,110,120,130,140,150,160],scores)
```

```
72339647.1236
56976578.0083
61214839.542
51622478.7582
55552811.1386
54099463.571
60489459.9097
65157899.1846
58087740.3312
```

Out[51]:

```
[<matplotlib.lines.Line2D at 0x7f6fdd08ebe0>]
```



The best value of k is 110 because it gets us the smaller error.

2.4. Anomaly detection

When we have a new connection data (e.g., one that we never saw before), we simply find the closest cluster for it, and use this information as a proxy to indicate whether the data point is anomalous or not. A simple approach to decide when there is an anomaly or not, amounts to measuring the new data point’s distance to its nearest centroid. If this distance exceeds some thresholds, it is anomalous.

Question 12

Build your model with the best value of k in your opinion. Then, detect the anomalous connections in our data. Plot and discuss your result.

HINT The threshold has strong impact on the result. Be careful when choosing it! A simple way to choose the threshold's value is picking up a distance of a data point from among known data. For example, the 100th-farthest data point distance can be an option.

In [13]:

```
clusters = KMeans.train(normalizedData, 110, 10,initializationMode="random"
)
```

In [15]:

```
def distanceToCluster(clusters, point):
    closest_center = clusters.centers[clusters.predict(point)]
    return euclidean_distance(point,  closest_center)
```

In [30]:

```
listDist=normalizedData.map(lambda x :distanceToCluster(clusters,
x)).collect()
```

In [31]:

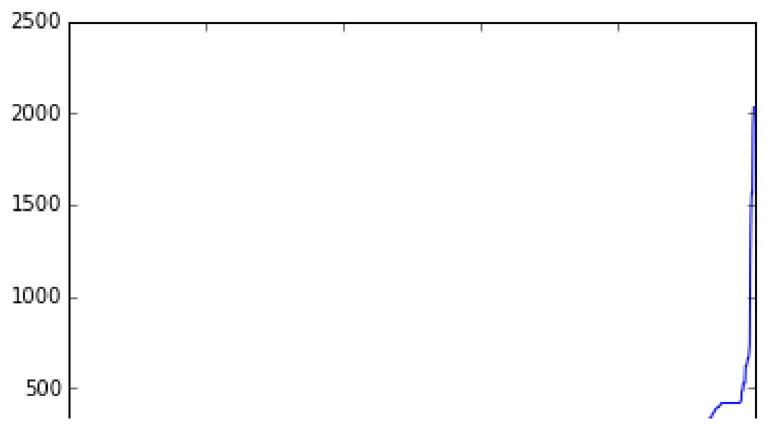
```
listDist=sorted(listDist)
```

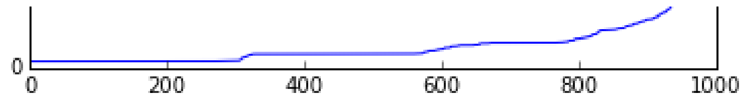
In [32]:

```
plt.plot(listDist[-1000:-1])
```

Out[32]:

```
[<matplotlib.lines.Line2D at 0x7f4d074695c0>]
```





Comment

We plotted here the 1000th-farthest data points. We can see that after 900, the distance increase brutally, so we decided to choose the threshold equal to the distance of the 100th-farthest data point.

In [33]:

```
threshold=listDist[-100]
```

In [34]:

```
threshold
```

Out[34]:

```
264.19547573380254
```

In [35]:

```
def Anomalous(clusters, point):
    closest_center = clusters.centers[clusters.predict(point)]
    if euclidean_distance(point,  closest_center)>=threshold:
        return 1 #1 anomalous
    else :return 0 #0 not anomalous
```

In [36]:

```
listData=normalizedData.map(lambda x :(x,Anomalous(clusters, x))).filter(lam
bda x:x[1]==1).collect()
len(listData)
```

Out[36]:

```
100
```

In [37]:

```
for item in listData:
    print(item)
```

```
(array([  1.89065330e+01,   9.45728261e-03,   2.86667062e-02,
         -2.39084686e-03,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,   2.44279187e+00,
          2.57206450e-01,  -8.25770840e-03,  -4.54646139e-03,
         -3.28458917e-03,   2.97931248e+02,  -8.50457842e-03,
          2.81319464e+01,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
         -4.66042614e-01,  -4.65755574e-01,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,  -3.32674143e+00,  -1.72041830e+00,
         -1.22502534e+00,   1.09899922e+00,  -1.15398387e+00,
          6.87196896e+00,  -4.66404784e-01,  -4.65453641e-01,
         -2.50831829e-01,  -2.49631966e-01]), 1)
 (array([ -3.50357923e-02,  -1.83828790e-03,  -1.26760816e-03,
         -2.39084686e-03,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,   6.84982775e+02,  -4.09367663e-01,
         -2.09732783e-03,  -8.25770840e-03,  -4.54646139e-03,
         -3.28458917e-03,  -9.57233922e-03,  -8.50457842e-03,
         -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
         -4.66042614e-01,  -4.65755574e-01,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,  -3.62351937e+00,  -1.76762697e+00,
          5.98966843e-01,  -2.82939000e-01,   8.21118739e-01,
          2.40800124e+01,  -4.66404784e-01,  -4.65453641e-01,
         -2.50831829e-01,  -2.49631966e-01]), 1)
 (array([ -4.88607458e-02,  -1.68745369e-03,   1.68126681e-02,
         -2.39084686e-03,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,   5.47985342e+02,   2.44279187e+00,
         -2.09732783e-03,  -8.25770840e-03,  -4.54646139e-03,
         -3.28458917e-03,   1.60953369e+01,  -8.50457842e-03,
         -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
         -4.66042614e-01,  -4.65755574e-01,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,  -2.24896891e+00,  -1.76762697e+00,
         -1.78438294e+00,   8.55778577e-02,  -1.23714608e+00,
         -1.56668488e-01,  -4.66404784e-01,  -4.65453641e-01,
         -2.50831829e-01,  -2.49631966e-01]), 1)
 (array([  2.09235937e+01,  -5.94436754e-04,   3.63264993e-01,
         -2.39084686e-03,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,   2.44279187e+00,
          1.94735040e+02,   1.21098972e+02,   2.47445758e+02,
          2.13552837e+02,  -9.57233922e-03,  -8.50457842e-03,
          1.68935459e+02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
         -4.66042614e-01,  -4.65755574e-01,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,  -1.23367595e+00,  -1.66376790e+00,
         -1.63846357e+00,   4.04713408e+00,  -1.23714608e+00,
         -1.56668488e-01,  -4.40215678e-01,  -2.56126767e-01,
          1.69770265e+00,   9.67218375e-02]), 1)
 (array([  1.52830127e+01,  -1.16484501e-03,   1.70434552e-01,
         -2.39084686e-03,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,   2.44279187e+00,
          1.17203210e+02,   1.21098972e+02,   2.47445758e+02,
          1.28231961e+02,  -9.57233922e-03,  -8.50457842e-03,
          1.12614054e+02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
         -4.66042614e-01,  -4.65755574e-01,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,  -3.13930273e+00,  -1.64488444e+00,
         -1.44390440e+00,   5.46223930e-01,  -1.19556498e+00,
          2.99410002e+00,  -4.66404784e-01,  -4.65453641e-01,
         -2.50831829e-01,  -2.49631966e-01]), 1)
 (array([ -6.68331854e-02,  -1.94875803e-03,  -1.69550700e-03,
          4.18261837e+02,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,  -4.09367663e-01,
         -2.09732783e-03,  -8.25770840e-03,  -4.54646139e-03,
         -3.28458917e-03,  -9.57233922e-03,  -8.50457842e-03,
         -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
          2.15261153e+00,   2.15030568e+00,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,  -3.62351937e+00,  -1.71097657e+00,
          5.98966843e-01,  -2.82939000e-01,   8.21118739e-01,
          9.05327024e+00,   2.15250575e+00,  -1.51463331e-01,
         -2.50831829e-01,  -2.49631966e-01]), 1)
 (array([  1.00115579e+01,  -1.05225045e-03,   1.22825217e-02,
         -2.39084686e-03,  -1.51391734e-02,   6.92991622e+02,
         -2.65207600e-02,  -4.39091558e-03,   2.44279187e+00,
          1.29442156e+00,   1.21098972e+02,  -4.54646139e-03,
          5.58320134e+00,  -9.57233922e-03,  -8.50457842e-03,
         -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
         -4.66042614e-01,  -4.65755574e-01,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,  -1.59293346e+00,  -1.40884110e+00,
         -1.80870284e+00,  -9.86805713e-02,  -1.23714608e+00,
          1.05516556e+00,  -4.66404784e-01,  -4.65453641e-01,
         -2.50831829e-01,  -2.49631966e-01]), 1)
 (array([  2.08392615e+01,  -7.22964499e-04,   2.72237267e-01,
         -2.39084686e-03,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,   2.44279187e+00,
          2.29222442e+02,   1.21098972e+02,   2.47445758e+02,
          2.47579614e+02,  -9.57233922e-03,  -8.50457842e-03,
          1.40774757e+02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
         -4.66042614e-01,  -4.65755574e-01,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,  -3.40484088e+00,  -1.74874350e+00,
         -1.34662482e+00,   9.14740788e-01,  -1.11240276e+00,
          1.19616719e+01,  -4.66404784e-01,  -4.65453641e-01,
         -2.50831829e-01,  -2.49631966e-01]), 1)
 (array([  1.18421192e-01,  -1.84253675e-03,   5.94583686e-02,
         -2.39084686e-03,  -1.51391734e-02,   2.77195987e+02,
         -2.65207600e-02,  -4.39091558e-03,   2.44279187e+00,
          2.57206450e-01,  -8.25770840e-03,  -4.54646139e-03,
          2.50646589e-01,  -9.57233922e-03,  -8.50457842e-03,
          2.81319464e+01,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
         -4.66042614e-01,  -4.65755574e-01,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,  -3.60789948e+00,  -1.77706870e+00,
         -6.17027945e-01,   8.92998245e+00,  -2.18408950e-01,
         -1.56668488e-01,  -4.66404784e-01,  -4.65453641e-01,
         -2.50831829e-01,  -2.49631966e-01]), 1)
 (array([  1.84489271e+01,  -1.04481497e-03,   2.02069610e-01,
         -2.39084686e-03,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,   2.44279187e+00,
          1.40799854e+02,   1.21098972e+02,   2.47445758e+02,
          1.54894734e+02,  -9.57233922e-03,  -8.50457842e-03,
          1.68935459e+02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
         -4.66042614e-01,  -4.65755574e-01,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,  -2.29582858e+00,  -1.68265137e+00,
         -1.51686409e+00,   3.61965501e-01,  -1.23714608e+00,
         -1.56668488e-01,  -4.40215678e-01,  -2.29960908e-01,
         -2.50831829e-01,  -2.49631966e-01]), 1)
 (array([  1.07235430e+01,  -7.03309683e-06,   2.28876852e-01,
         -2.39084686e-03,  -1.51391734e-02,  -1.10348462e-03,
          2.10577489e+00,   2.73990476e+02,   2.44279187e+00,
          4.92467445e+00,  -8.25770840e-03,   1.23720606e+02,
          2.53602720e+00,   6.44100645e+01,  -8.50457842e-03,
         -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
         -4.66042614e-01,  -4.65755574e-01,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,  -2.84252479e+00,  -1.72041830e+00,
         -1.49254419e+00,   4.54094715e-01,  -1.21635553e+00,
         -1.56668488e-01,  -4.66404784e-01,  -4.65453641e-01,
         -2.50831829e-01,  -2.49631966e-01]), 1)
 (array([  2.08461740e+01,  -6.43298541e-04,   2.85333142e-01,
         -2.39084686e-03,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,   2.44279187e+00,
          1.98883900e+02,   1.21098972e+02,   2.47445758e+02,
          2.17615735e+02,  -9.57233922e-03,  -8.50457842e-03,
          1.68935459e+02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
         -4.66042614e-01,  -4.65755574e-01,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,  -2.70194576e+00,  -1.70153484e+00,
         -1.46822430e+00,   3.61965501e-01,  -1.21635553e+00,
         -1.56668488e-01,  -4.66404784e-01,  -4.65453641e-01,
         -2.50831829e-01,  -2.49631966e-01]), 1)
 (array([  5.20614149e-02,  -1.77774177e-03,   9.36062604e-05,
         -2.39084686e-03,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,   4.10987909e+02,   2.44279187e+00,
         -2.09732783e-03,  -8.25770840e-03,  -4.54646139e-03,
         -3.28458917e-03,  -9.57233922e-03,  -8.50457842e-03,
         -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
         -4.66042614e-01,  -4.65755574e-01,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,  -2.48326728e+00,  -1.68265137e+00,
         -1.46822430e+00,   1.77707072e-01,  -1.23714608e+00,
         -1.56668488e-01,  -4.66404784e-01,  -4.65453641e-01,
         -2.50831829e-01,  -2.49631966e-01]), 1)
 (array([ -3.08883063e-02,  -1.83616347e-03,  -1.31721962e-03,
         -2.39084686e-03,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,   2.73990476e+02,  -4.09367663e-01,
         -2.09732783e-03,  -8.25770840e-03,  -4.54646139e-03,
         -3.28458917e-03,  -9.57233922e-03,  -8.50457842e-03,
         -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
         -4.66042614e-01,  -4.65755574e-01,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,  -2.23334901e+00,  -1.66376790e+00,
         -1.49254419e+00,   2.69836287e-01,  -1.23714608e+00,
         -1.56668488e-01,  -4.66404784e-01,  -4.65453641e-01,
         -2.50831829e-01,  -2.49631966e-01]), 1)
 (array([  3.40876013e+00,   1.08748094e-04,   1.29698893e-01,
         -2.39084686e-03,  -1.51391734e-02,   2.77195987e+02,
          2.10577489e+00,  -4.39091558e-03,   2.44279187e+00,
          1.91863822e+01,   1.21098972e+02,   2.47445758e+02,
          1.95494162e+01,   7.24625191e+01,  -8.50457842e-03,
         -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
         -4.66042614e-01,  -4.65755574e-01,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,   3.43933101e-01,  -1.76762697e+00,
         -1.80870284e+00,  -1.90809786e-01,  -1.25793664e+00,
         -1.56668488e-01,  -4.66404784e-01,  -4.65453641e-01,
         -2.50831829e-01,  -2.49631966e-01]), 1)
 (array([ -6.40681947e-02,   7.36510505e+02,  -1.69550700e-03,
         -2.39084686e-03,  -1.51391734e-02,  -1.10348462e-03,
          2.10577489e+00,  -4.39091558e-03,  -4.09367663e-01,
         -2.09732783e-03,  -8.25770840e-03,  -4.54646139e-03,
         -3.28458917e-03,  -9.57233922e-03,  -8.50457842e-03,
         -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.31125262e+00,  -1.18811292e+00,
          1.60269416e+00,   1.28700546e+00,   6.55900999e-01,
          1.17024609e+00,  -1.90057067e+00,   4.45895289e+00,
         -2.01059296e-01,   3.43933101e-01,  -1.75818524e+00,
         -1.80870284e+00,   5.46223930e-01,  -8.00544455e-01,
         -1.56668488e-01,   4.99911271e-03,   1.28765892e+00,
         -3.43279980e-02,   1.17907747e+00]), 1)
 (array([  9.09086827e-04,  -1.75012424e-03,   1.03213187e-02,
         -2.39084686e-03,  -1.51391734e-02,  -1.10348462e-03,
          2.10577489e+00,  -4.39091558e-03,   2.44279187e+00,
          2.59094045e+00,  -8.25770840e-03,  -4.54646139e-03,
          2.53602720e+00,  -9.57233922e-03,  -8.50457842e-03,
         -2.87561127e-02,   0.00000000e+00,   1.56499665e+03,
         -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
         -4.66042614e-01,  -4.65755574e-01,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,   3.43933101e-01,  -1.61655924e+00,
         -1.66278346e+00,   5.06055544e+00,  -1.25793664e+00,
         -1.56668488e-01,   1.05256333e+00,  -2.06340350e-02,
         -2.50831829e-01,   1.01333867e-02]), 1)
 (array([  1.30863650e-01,  -1.79792381e-03,  -1.19319097e-03,
         -2.39084686e-03,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,   5.47985342e+02,  -4.09367663e-01,
         -2.09732783e-03,  -8.25770840e-03,  -4.54646139e-03,
         -3.28458917e-03,  -9.57233922e-03,  -8.50457842e-03,
         -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57069789e+00,  -1.19624324e+00,
          8.43284460e-01,  -4.65755574e-01,  -2.48285775e-01,
         -2.48130352e-01,  -7.44637309e-01,   1.18337112e+01,
         -2.01059296e-01,   3.43933101e-01,  -1.50325844e+00,
         -1.54118398e+00,   3.21797115e+00,  -1.25793664e+00,
         -1.56668488e-01,   5.28781220e-01,  -2.03795049e-01,
         -2.50831829e-01,  -1.19749290e-01]), 1)
 (array([  1.90281926e+01,   9.72602244e-03,   1.28167139e-01,
         -2.39084686e-03,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,   2.44279187e+00,
         -2.09732783e-03,  -8.25770840e-03,  -4.54646139e-03,
         -3.28458917e-03,   3.22088612e+02,  -8.50457842e-03,
          2.81319464e+01,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
         -4.66042614e-01,  -4.65755574e-01,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,   3.43933101e-01,  -1.41828284e+00,
         -1.46822430e+00,   2.69836287e-01,  -1.25793664e+00,
         -1.56668488e-01,   9.47806905e-01,   2.14858697e-01,
         -7.76287642e-02,   3.13192964e-01]), 1)
 (array([  8.64968682e-01,  -1.68957811e-03,   2.59494578e-04,
         -2.39084686e-03,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,   2.73990476e+02,   2.44279187e+00,
         -2.09732783e-03,  -8.25770840e-03,  -4.54646139e-03,
         -3.28458917e-03,  -9.57233922e-03,  -8.50457842e-03,
         -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
         -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
         -4.66042614e-01,  -4.65755574e-01,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,   3.43933101e-01,  -1.38995764e+00,
         -1.44390440e+00,   1.77707072e-01,  -1.25793664e+00,
         -1.56668488e-01,   7.38294062e-01,   3.16976833e-02,
         -3.43279980e-02,   2.69898739e-01]), 1)
 (array([  3.24977316e+00,   4.28474136e-04,   7.12011849e-01,
         -2.39084686e-03,  -1.51391734e-02,   4.15794532e+02,
         -2.65207600e-02,   2.73990476e+02,   2.44279187e+00,
          5.44328201e+00,  -8.25770840e-03,  -4.54646139e-03,
          3.04388955e+00,   8.04288226e+00,  -8.50457842e-03,
         -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
         -4.66042614e-01,  -4.65755574e-01,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,  -3.07682316e+00,  -1.77706870e+00,
         -1.76006305e+00,   7.30482359e-01,  -1.19556498e+00,
         -1.56668488e-01,  -4.66404784e-01,  -4.65453641e-01,
         -2.50831829e-01,  -2.49631966e-01]), 1)
 (array([ -6.68331854e-02,  -1.94875803e-03,  -1.69550700e-03,
          4.18261837e+02,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,  -4.09367663e-01,
         -2.09732783e-03,  -8.25770840e-03,  -4.54646139e-03,
         -3.28458917e-03,  -9.57233922e-03,  -8.50457842e-03,
         -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19217808e+00,
          2.15261153e+00,   2.15030568e+00,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
          6.91336300e+00,  -3.62351937e+00,  -1.77706870e+00,
          5.98966843e-01,  -2.82939000e-01,   8.21118739e-01,
         -1.56668488e-01,   2.15250575e+00,   2.15113227e+00,
         -2.50831829e-01,  -2.49631966e-01]), 1)
 (array([  1.62092846e+01,  -1.00976194e-03,   2.24757550e-01,
         -2.39084686e-03,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,   2.44279187e+00,
          1.76843079e+02,   1.21098972e+02,   2.47445758e+02,
          1.90191168e+02,  -9.57233922e-03,  -8.50457842e-03,
          8.44533515e+01,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
         -4.66042614e-01,  -4.65755574e-01,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,  -3.29550164e+00,  -1.74874350e+00,
         -1.39526461e+00,   1.37538686e+00,  -1.15398387e+00,
         -1.56668488e-01,  -2.30702835e-01,   1.88692838e-01,
         -2.50831829e-01,  -2.49631966e-01]), 1)
 (array([  1.82802626e+01,   7.60159692e-03,   4.69500794e-02,
         -2.39084686e-03,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,   2.44279187e+00,
         -2.09732783e-03,  -8.25770840e-03,  -4.54646139e-03,
         -3.28458917e-03,   2.73773884e+02,  -8.50457842e-03,
         -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
         -4.66042614e-01,  -4.65755574e-01,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,  -3.23302208e+00,  -1.73930177e+00,
         -1.37094471e+00,   1.09899922e+00,  -1.17477442e+00,
         -1.56668488e-01,  -2.56891941e-01,   5.78635425e-02,
         -2.50831829e-01,  -2.49631966e-01]), 1)
 (array([  2.12430876e-01,  -1.46120237e-03,   1.06034839e-02,
 (array([  2.12430876e-01,  -1.46120237e-03,   1.06034839e-02,
         -2.39084686e-03,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,   2.73990476e+02,   2.44279187e+00,
          7.75814006e-01,  -8.25770840e-03,  -4.54646139e-03,
         -3.28458917e-03,   1.60953369e+01,  -8.50457842e-03,
         -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
         -4.66042614e-01,  -4.65755574e-01,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,  -3.10806294e+00,  -1.72986004e+00,
         -1.39526461e+00,   8.22611573e-01,  -1.19556498e+00,
         -1.56668488e-01,  -3.09270151e-01,  -2.06340350e-02,
         -2.50831829e-01,  -2.49631966e-01]), 1)
 (array([  2.28466448e+01,  -7.95194966e-04,   2.73787625e-01,
         -2.39084686e-03,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,   2.44279187e+00,
          1.85659408e+02,   1.21098972e+02,   2.47445758e+02,
          2.00348415e+02,  -9.57233922e-03,  -8.50457842e-03,
          8.44533515e+01,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
         -4.66042614e-01,  -4.65755574e-01,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,  -9.21278119e-01,  -1.60711750e+00,
         -1.56550388e+00,  -6.55135681e-03,  -1.23714608e+00,
         -1.56668488e-01,  -4.14026573e-01,  -1.77629190e-01,
          5.22735344e-02,  -2.49631966e-01]), 1)
 (array([  1.06627132e+01,  -1.64708960e-03,   9.80411099e-04,
         -2.39084686e-03,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,   4.10987909e+02,   2.44279187e+00,
         -2.09732783e-03,  -8.25770840e-03,  -4.54646139e-03,
         -3.28458917e-03,  -9.57233922e-03,  -8.50457842e-03,
         -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
         -4.66042614e-01,  -4.65755574e-01,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,  -3.27722238e-01,  -1.58823404e+00,
         -1.58982378e+00,  -6.55135681e-03,  -1.25793664e+00,
         -1.56668488e-01,  -4.40215678e-01,  -2.03795049e-01,
          9.55743006e-02,  -2.49631966e-01]), 1)
 (array([ -6.68331854e-02,  -1.94875803e-03,  -1.69550700e-03,
          4.18261837e+02,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,  -4.09367663e-01,
         -2.09732783e-03,  -8.25770840e-03,  -4.54646139e-03,
         -3.28458917e-03,  -9.57233922e-03,  -8.50457842e-03,
         -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
          2.15261153e+00,   2.15030568e+00,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,  -3.62351937e+00,  -1.72986004e+00,
          5.98966843e-01,  -2.82939000e-01,   8.21118739e-01,
          7.84143620e+00,   2.15250575e+00,  -2.06340350e-02,
         -2.50831829e-01,  -2.49631966e-01]), 1)
 (array([ -6.68331854e-02,  -1.94875803e-03,  -1.69550700e-03,
          4.18261837e+02,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,  -4.09367663e-01,
         -2.09732783e-03,  -8.25770840e-03,  -4.54646139e-03,
         -3.28458917e-03,  -9.57233922e-03,  -8.50457842e-03,
         -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19217808e+00,
          2.15261153e+00,   2.15030568e+00,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
          6.91336300e+00,  -3.45170056e+00,  -1.70153484e+00,
         -1.63846357e+00,   1.28325765e+00,  -1.09161221e+00,
          7.84143620e+00,   1.05256333e+00,  -1.77629190e-01,
         -2.50831829e-01,  -2.49631966e-01]), 1)
 (array([ -4.60957551e-02,  -1.80323488e-03,  -1.48465829e-03,
         -2.39084686e-03,  -1.51391734e-02,   2.77195987e+02,
         -2.65207600e-02,  -4.39091558e-03,  -4.09367663e-01,
         -2.09732783e-03,  -8.25770840e-03,  -4.54646139e-03,
         -3.28458917e-03,  -9.57233922e-03,  -8.50457842e-03,
         -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
         -4.66042614e-01,  -4.65755574e-01,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,  -2.32706836e+00,  -1.77706870e+00,
         -1.80870284e+00,   2.69836287e-01,  -1.23714608e+00,
         -1.56668488e-01,  -2.56891941e-01,  -4.65453641e-01,
         -2.50831829e-01,  -2.49631966e-01]), 1)
 (array([ -4.05657737e-02,  -1.78092841e-03,  -1.34512606e-03,
         -2.39084686e-03,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,   2.73990476e+02,  -4.09367663e-01,
         -2.09732783e-03,  -8.25770840e-03,  -4.54646139e-03,
         -3.28458917e-03,  -9.57233922e-03,  -8.50457842e-03,
         -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
         -4.66042614e-01,  -4.65755574e-01,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,   3.43933101e-01,  -1.77706870e+00,
         -1.83302273e+00,  -1.90809786e-01,  -1.25793664e+00,
         -1.56668488e-01,  -4.66404784e-01,  -4.65453641e-01,
         -3.43279980e-02,  -2.49631966e-01]), 1)
 (array([ -6.68331854e-02,  -1.94875803e-03,  -1.69550700e-03,
          4.18261837e+02,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,  -4.09367663e-01,
         -2.09732783e-03,  -8.25770840e-03,  -4.54646139e-03,
         -3.28458917e-03,  -9.57233922e-03,  -8.50457842e-03,
         -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19217808e+00,
          2.15261153e+00,   2.15030568e+00,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
          6.91336300e+00,  -3.62351937e+00,  -1.77706870e+00,
          5.98966843e-01,  -2.82939000e-01,   8.21118739e-01,
         -1.56668488e-01,   2.15250575e+00,   2.15113227e+00,
         -2.50831829e-01,  -2.49631966e-01]), 1)
 (array([ -6.68331854e-02,  -1.94875803e-03,  -1.69550700e-03,
          4.18261837e+02,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,  -4.09367663e-01,
         -2.09732783e-03,  -8.25770840e-03,  -4.54646139e-03,
         -3.28458917e-03,  -9.57233922e-03,  -8.50457842e-03,
         -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19217808e+00,
          2.15261153e+00,   2.15030568e+00,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
          6.91336300e+00,  -3.62351937e+00,  -1.76762697e+00,
          5.98966843e-01,  -2.82939000e-01,   8.21118739e-01,
          2.40800124e+01,   2.15250575e+00,   2.15113227e+00,
         -2.50831829e-01,  -2.49631966e-01]), 1)
 (array([ -6.68331854e-02,  -1.94875803e-03,  -1.69550700e-03,
          4.18261837e+02,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,  -4.09367663e-01,
         -2.09732783e-03,  -8.25770840e-03,  -4.54646139e-03,
         -3.28458917e-03,  -9.57233922e-03,  -8.50457842e-03,
         -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19217808e+00,
          2.15261153e+00,   2.15030568e+00,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
          6.91336300e+00,  -3.62351937e+00,  -1.75818524e+00,
          5.98966843e-01,  -2.82939000e-01,   8.21118739e-01,
          2.40800124e+01,   2.15250575e+00,   2.15113227e+00,
         -2.50831829e-01,  -2.49631966e-01]), 1)
 (array([ -6.68331854e-02,  -1.94875803e-03,  -1.69550700e-03,
          4.18261837e+02,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,  -4.09367663e-01,
         -2.09732783e-03,  -8.25770840e-03,  -4.54646139e-03,
         -3.28458917e-03,  -9.57233922e-03,  -8.50457842e-03,
         -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19217808e+00,
          2.15261153e+00,   2.15030568e+00,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
          6.91336300e+00,  -3.62351937e+00,  -1.74874350e+00,
          5.98966843e-01,  -2.82939000e-01,   8.21118739e-01,
          2.40800124e+01,   2.15250575e+00,   2.15113227e+00,
         -2.50831829e-01,  -2.49631966e-01]), 1)
 (array([ -6.68331854e-02,  -1.94875803e-03,  -1.69550700e-03,
          4.18261837e+02,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,  -4.09367663e-01,
         -2.09732783e-03,  -8.25770840e-03,  -4.54646139e-03,
         -3.28458917e-03,  -9.57233922e-03,  -8.50457842e-03,
         -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19217808e+00,
          2.15261153e+00,   2.15030568e+00,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
          6.91336300e+00,  -3.62351937e+00,  -1.73930177e+00,
          5.98966843e-01,  -2.82939000e-01,   8.21118739e-01,
          2.40800124e+01,   2.15250575e+00,   2.15113227e+00,
         -2.50831829e-01,  -2.49631966e-01]), 1)
 (array([ -6.68331854e-02,  -1.94875803e-03,  -1.69550700e-03,
          4.18261837e+02,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,  -4.09367663e-01,
         -2.09732783e-03,  -8.25770840e-03,  -4.54646139e-03,
         -3.28458917e-03,  -9.57233922e-03,  -8.50457842e-03,
         -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19217808e+00,
          2.15261153e+00,   2.15030568e+00,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
          6.91336300e+00,  -3.62351937e+00,  -1.72986004e+00,
          5.98966843e-01,  -2.82939000e-01,   8.21118739e-01,
          7.84143620e+00,   2.15250575e+00,  -2.06340350e-02,
         -2.50831829e-01,  -2.49631966e-01]), 1)
 (array([  1.98134500e+01,   8.14863649e-03,   2.26823239e-02,
         -2.39084686e-03,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,   2.44279187e+00,
          2.57206450e-01,  -8.25770840e-03,  -4.54646139e-03,
         -3.28458917e-03,   2.73773884e+02,  -8.50457842e-03,
          2.81319464e+01,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
         -4.66042614e-01,  -4.65755574e-01,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,  -2.87376457e+00,  -1.69209310e+00,
         -1.34662482e+00,   4.54094715e-01,  -1.21635553e+00,
         -1.56668488e-01,  -4.14026573e-01,  -4.65453641e-01,
         -7.76287642e-02,   6.16252542e-01]), 1)
 (array([ -4.19482691e-02,  -1.78623947e-03,  -1.30636711e-03,
         -2.39084686e-03,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,   4.10987909e+02,  -4.09367663e-01,
         -2.09732783e-03,  -8.25770840e-03,  -4.54646139e-03,
         -3.28458917e-03,  -9.57233922e-03,  -8.50457842e-03,
         -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
         -4.66042614e-01,  -4.65755574e-01,   4.05736553e+00,
          4.04998007e+00,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,  -1.46797433e+00,  -1.54102537e+00,
         -1.37094471e+00,  -6.55135681e-03,  -1.23714608e+00,
         -1.56668488e-01,  -4.40215678e-01,  -3.60790204e-01,
         -3.43279980e-02,   9.19312120e-01]), 1)
 (array([  2.04729003e+01,  -6.17805435e-04,   3.34679490e-01,
         -2.39084686e-03,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,   2.44279187e+00,
          1.97328078e+02,   1.21098972e+02,   2.47445758e+02,
          2.15584286e+02,  -9.57233922e-03,  -8.50457842e-03,
          2.53417567e+02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
         -4.66042614e-01,  -4.65755574e-01,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,  -1.28053563e+00,  -1.50325844e+00,
         -1.34662482e+00,  -6.55135681e-03,  -1.23714608e+00,
         -1.56668488e-01,  -4.40215678e-01,  -3.86956063e-01,
         -3.43279980e-02,   7.46135219e-01]), 1)
 (array([ -6.68331854e-02,  -1.94875803e-03,  -1.69550700e-03,
          4.18261837e+02,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,  -4.09367663e-01,
         -2.09732783e-03,  -8.25770840e-03,  -4.54646139e-03,
         -3.28458917e-03,  -9.57233922e-03,  -8.50457842e-03,
         -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
          2.15261153e+00,   2.15030568e+00,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,  -3.62351937e+00,  -1.71097657e+00,
          5.98966843e-01,  -2.82939000e-01,   8.21118739e-01,
          5.90250173e+00,   2.15250575e+00,  -1.51463331e-01,
         -2.50831829e-01,  -2.49631966e-01]), 1)
 (array([ -6.68331854e-02,  -1.94875803e-03,  -1.69550700e-03,
          4.18261837e+02,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,  -4.09367663e-01,
         -2.09732783e-03,  -8.25770840e-03,  -4.54646139e-03,
         -3.28458917e-03,  -9.57233922e-03,  -8.50457842e-03,
         -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19217808e+00,
          2.15261153e+00,   2.15030568e+00,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
          6.91336300e+00,  -3.62351937e+00,  -1.72986004e+00,
          5.98966843e-01,  -2.82939000e-01,   8.21118739e-01,
          7.84143620e+00,   2.15250575e+00,  -2.06340350e-02,
         -2.50831829e-01,  -2.49631966e-01]), 1)
 (array([ -6.68331854e-02,  -1.94875803e-03,  -1.69550700e-03,
          4.18261837e+02,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,  -4.09367663e-01,
         -2.09732783e-03,  -8.25770840e-03,  -4.54646139e-03,
         -3.28458917e-03,  -9.57233922e-03,  -8.50457842e-03,
         -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19217808e+00,
          2.15261153e+00,   2.15030568e+00,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
          6.91336300e+00,  -3.62351937e+00,  -1.76762697e+00,
          5.98966843e-01,  -2.82939000e-01,   8.21118739e-01,
          2.40800124e+01,   2.15250575e+00,   8.42839317e-01,
         -2.50831829e-01,  -2.49631966e-01]), 1)
 (array([ -6.68331854e-02,  -1.94875803e-03,  -1.69550700e-03,
          4.18261837e+02,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,  -4.09367663e-01,
         -2.09732783e-03,  -8.25770840e-03,  -4.54646139e-03,
         -3.28458917e-03,  -9.57233922e-03,  -8.50457842e-03,
         -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19217808e+00,
          2.15261153e+00,   2.15030568e+00,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
          6.91336300e+00,  -3.62351937e+00,  -1.75818524e+00,
          5.98966843e-01,  -2.82939000e-01,   8.21118739e-01,
          2.40800124e+01,   2.15250575e+00,   1.28765892e+00,
         -2.50831829e-01,  -2.49631966e-01]), 1)
 (array([ -6.68331854e-02,  -1.94875803e-03,  -1.69550700e-03,
          4.18261837e+02,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,  -4.09367663e-01,
         -2.09732783e-03,  -8.25770840e-03,  -4.54646139e-03,
         -3.28458917e-03,  -9.57233922e-03,  -8.50457842e-03,
         -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19217808e+00,
          2.15261153e+00,   2.15030568e+00,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
          6.91336300e+00,  -3.62351937e+00,  -1.74874350e+00,
          5.98966843e-01,  -2.82939000e-01,   8.21118739e-01,
          2.40800124e+01,   2.15250575e+00,   1.49698580e+00,
         -2.50831829e-01,  -2.49631966e-01]), 1)
 (array([ -6.68331854e-02,  -1.94875803e-03,  -1.69550700e-03,
          4.18261837e+02,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,  -4.09367663e-01,
         -2.09732783e-03,  -8.25770840e-03,  -4.54646139e-03,
         -3.28458917e-03,  -9.57233922e-03,  -8.50457842e-03,
         -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19217808e+00,
          2.15261153e+00,   2.15030568e+00,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
          6.91336300e+00,  -3.62351937e+00,  -1.73930177e+00,
          5.98966843e-01,  -2.82939000e-01,   8.21118739e-01,
          2.40800124e+01,   2.15250575e+00,   1.62781509e+00,
         -2.50831829e-01,  -2.49631966e-01]), 1)
 (array([ -6.68331854e-02,  -1.94875803e-03,  -1.69550700e-03,
          4.18261837e+02,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,  -4.09367663e-01,
         -2.09732783e-03,  -8.25770840e-03,  -4.54646139e-03,
         -3.28458917e-03,  -9.57233922e-03,  -8.50457842e-03,
         -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19217808e+00,
          2.15261153e+00,   2.15030568e+00,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
          6.91336300e+00,  -3.62351937e+00,  -1.72986004e+00,
          5.98966843e-01,  -2.82939000e-01,   8.21118739e-01,
          2.40800124e+01,   2.15250575e+00,   1.70631267e+00,
         -2.50831829e-01,  -2.49631966e-01]), 1)
 (array([  2.22259044e+01,  -1.01188637e-03,   1.39492505e-01,
         -2.39084686e-03,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,   2.44279187e+00,
          1.61284853e+02,   1.21098972e+02,   2.47445758e+02,
          1.73685641e+02,  -9.57233922e-03,  -8.50457842e-03,
          2.81319464e+01,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
         -4.66042614e-01,  -4.65755574e-01,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,  -3.38922099e+00,  -1.73930177e+00,
         -1.07910597e+00,   1.46751607e+00,  -1.13319332e+00,
         -1.56668488e-01,  -4.66404784e-01,  -4.65453641e-01,
         -2.50831829e-01,  -2.49631966e-01]), 1)
 (array([  1.54530596e+01,  -8.56803307e-04,   2.83736273e-01,
         -2.39084686e-03,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,   2.44279187e+00,
          1.39244031e+02,   1.21098972e+02,   2.47445758e+02,
          1.55148665e+02,  -9.57233922e-03,  -8.50457842e-03,
          1.40774757e+02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
         -4.66042614e-01,  -4.65755574e-01,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,  -3.12368284e+00,  -1.69209310e+00,
         -1.10342586e+00,   8.22611573e-01,  -1.19556498e+00,
         -1.56668488e-01,  -4.66404784e-01,  -4.65453641e-01,
         -2.50831829e-01,  -2.49631966e-01]), 1)
 (array([  1.95037710e+01,   8.00736219e-03,   2.24249645e-02,
         -2.39084686e-03,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,   2.44279187e+00,
          2.57206450e-01,  -8.25770840e-03,  -4.54646139e-03,
         -3.28458917e-03,   2.73773884e+02,  -8.50457842e-03,
         -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
         -4.66042614e-01,  -4.65755574e-01,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,  -2.09276999e+00,  -1.68265137e+00,
         -1.56550388e+00,   1.77707072e-01,  -1.23714608e+00,
         -1.56668488e-01,  -4.66404784e-01,  -4.65453641e-01,
         -1.64230297e-01,   5.29664092e-01]), 1)
 (array([  2.12401852e+01,  -7.20840073e-04,   2.74294593e-01,
         -2.39084686e-03,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,   2.44279187e+00,
          2.09774659e+02,   1.21098972e+02,   2.47445758e+02,
          2.25741533e+02,  -9.57233922e-03,  -8.50457842e-03,
          1.97096162e+02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
         -4.66042614e-01,  -4.65755574e-01,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,   3.43933101e-01,  -1.71097657e+00,
         -1.76006305e+00,   3.61965501e-01,  -1.25793664e+00,
         -1.56668488e-01,   1.62872364e+00,  -1.51463331e-01,
         -2.50831829e-01,  -2.49631966e-01]), 1)
 (array([  1.86452414e+01,   9.31707053e-03,   2.58962163e-02,
         -2.39084686e-03,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,   2.44279187e+00,
          1.03511778e+00,  -8.25770840e-03,  -4.54646139e-03,
          5.04577768e-01,   3.22088612e+02,  -8.50457842e-03,
         -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
         -4.66042614e-01,  -4.65755574e-01,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,   3.43933101e-01,  -1.67320964e+00,
         -1.71142326e+00,   2.69836287e-01,  -1.25793664e+00,
         -1.56668488e-01,   1.52396722e+00,  -2.56126767e-01,
         -2.50831829e-01,  -2.49631966e-01]), 1)
 (array([  1.47341621e+01,  -8.76985349e-04,   2.56470125e-01,
         -2.39084686e-03,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,   2.44279187e+00,
          1.44689411e+02,   1.21098972e+02,   2.47445758e+02,
          1.59719427e+02,  -9.57233922e-03,  -8.50457842e-03,
          1.40774757e+02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
         -4.66042614e-01,  -4.65755574e-01,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,   3.43933101e-01,  -1.29554031e+00,
         -1.34662482e+00,  -9.86805713e-02,  -1.25793664e+00,
         -1.56668488e-01,   1.18350885e+00,   1.54931751e+00,
         -2.50831829e-01,  -2.49631966e-01]), 1)
 (array([ -6.68331854e-02,  -1.83935011e-03,  -1.24280243e-03,
         -2.39084686e-03,  -1.51391734e-02,  -1.10348462e-03,
          2.10577489e+00,   5.47985342e+02,  -4.09367663e-01,
         -2.09732783e-03,  -8.25770840e-03,  -4.54646139e-03,
         -3.28458917e-03,   1.60953369e+01,  -8.50457842e-03,
         -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
          3.45885478e+01,  -1.57541507e+00,  -1.19624324e+00,
         -4.66042614e-01,  -4.65755574e-01,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,  -1.81161194e+00,  -1.76762697e+00,
         -1.78438294e+00,   7.30482359e-01,  -1.23714608e+00,
         -1.56668488e-01,  -4.66404784e-01,  -4.65453641e-01,
         -1.20929530e-01,  -2.49631966e-01]), 1)
 (array([  1.51807081e+01,   7.76942654e-03,   3.31875505e-02,
         -2.39084686e-03,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,   2.44279187e+00,
         -2.09732783e-03,  -8.25770840e-03,  -4.54646139e-03,
         -3.28458917e-03,   2.89878793e+02,  -8.50457842e-03,
          2.81319464e+01,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
         -4.66042614e-01,  -4.65755574e-01,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,  -3.56103980e+00,  -1.77706870e+00,
         -1.34662482e+00,   3.40222958e+00,  -8.42125563e-01,
         -1.56668488e-01,  -4.66404784e-01,  -4.65453641e-01,
         -2.50831829e-01,  -2.49631966e-01]), 1)
 (array([  7.29910205e+00,  -2.44968756e-04,   1.42557657e+00,
         -2.39084686e-03,  -1.51391734e-02,  -1.10348462e-03,
          4.23807055e+00,  -4.39091558e-03,   2.44279187e+00,
          1.93933086e+03,  -8.25770840e-03,  -4.54646139e-03,
          1.89635476e+03,   1.60953369e+01,  -8.50457842e-03,
         -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
         -4.66042614e-01,  -4.65755574e-01,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,  -3.32674143e+00,  -1.71097657e+00,
         -8.60226903e-01,   6.38353144e-01,  -1.15398387e+00,
         -1.56668488e-01,  -4.66404784e-01,  -4.65453641e-01,
         -2.50831829e-01,  -2.49631966e-01]), 1)
 (array([  1.83770373e+01,   7.60690798e-03,   2.38528443e-02,
         -2.39084686e-03,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,   2.44279187e+00,
         -2.09732783e-03,  -8.25770840e-03,  -4.54646139e-03,
         -3.28458917e-03,   2.81826339e+02,  -8.50457842e-03,
         -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
         -4.66042614e-01,  -4.65755574e-01,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,  -2.62384630e+00,  -1.59767577e+00,
         -1.07910597e+00,   1.77707072e-01,  -1.21635553e+00,
         -1.56668488e-01,  -4.66404784e-01,  -4.65453641e-01,
         -2.50831829e-01,  -2.49631966e-01]), 1)
 (array([  2.00028518e+01,  -1.10536109e-03,   2.34129465e-01,
         -2.39084686e-03,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,   2.44279187e+00,
          1.34835867e+02,   1.21098972e+02,   2.47445758e+02,
          1.45245349e+02,  -9.57233922e-03,  -8.50457842e-03,
          2.25256864e+02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
         -4.66042614e-01,  -4.65755574e-01,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,  -3.51418013e+00,  -1.75818524e+00,
         -9.08866695e-01,   2.02029136e+00,  -1.00844999e+00,
         -1.56668488e-01,  -4.66404784e-01,  -4.65453641e-01,
         -2.50831829e-01,  -2.49631966e-01]), 1)
 (array([  2.11917978e+01,  -9.13100583e-04,   2.55527508e-01,
         -2.39084686e-03,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,   2.44279187e+00,
          1.75287257e+02,   1.21098972e+02,   2.47445758e+02,
          1.91460824e+02,  -9.57233922e-03,  -8.50457842e-03,
          1.68935459e+02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
         -4.66042614e-01,  -4.65755574e-01,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,  -1.60855335e+00,  -1.55046710e+00,
         -1.37094471e+00,  -6.55135681e-03,  -1.23714608e+00,
         -1.56668488e-01,  -4.40215678e-01,  -3.60790204e-01,
         -1.64230297e-01,   9.67218375e-02]), 1)
 (array([  5.50518740e+01,  -1.94875803e-03,   6.20592795e+02,
         -2.39084686e-03,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,  -4.09367663e-01,
         -2.09732783e-03,  -8.25770840e-03,  -4.54646139e-03,
         -3.28458917e-03,  -9.57233922e-03,  -8.50457842e-03,
         -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
         -4.66042614e-01,  -4.65755574e-01,   4.05736553e+00,
          4.04998007e+00,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,   3.43933101e-01,  -1.77706870e+00,
         -1.83302273e+00,   4.32352172e+00,   8.21118739e-01,
         -1.56668488e-01,  -4.66404784e-01,  -4.65453641e-01,
          4.07924479e+00,   4.07979057e+00]), 1)
 (array([  5.51362062e+01,  -1.94875803e-03,   6.20592795e+02,
         -2.39084686e-03,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,  -4.09367663e-01,
         -2.09732783e-03,  -8.25770840e-03,  -4.54646139e-03,
         -3.28458917e-03,  -9.57233922e-03,  -8.50457842e-03,
         -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57069789e+00,  -1.19217808e+00,
         -4.66042614e-01,  -4.65755574e-01,   4.05736553e+00,
          4.04998007e+00,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,   3.43933101e-01,  -1.76762697e+00,
         -1.80870284e+00,   4.32352172e+00,   8.21118739e-01,
         -1.56668488e-01,  -4.66404784e-01,  -4.65453641e-01,
          4.07924479e+00,   4.07979057e+00]), 1)
 (array([  1.51302332e+01,  -1.94875803e-03,   2.03087038e+03,
 (array([  1.51392332e+01,  -1.94875805e-03,   2.03087038e+03,
         -2.39084686e-03,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,  -4.09367663e-01,
         -2.09732783e-03,  -8.25770840e-03,  -4.54646139e-03,
         -3.28458917e-03,  -9.57233922e-03,  -8.50457842e-03,
         -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
         -4.66042614e-01,  -4.65755574e-01,   4.05736553e+00,
          4.04998007e+00,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,   3.43933101e-01,  -1.77706870e+00,
         -1.83302273e+00,   5.70545994e+00,   8.21118739e-01,
         -1.56668488e-01,  -4.66404784e-01,  -4.65453641e-01,
          4.07924479e+00,   4.07979057e+00]), 1)
 (array([  6.49448976e+00,  -4.31483307e-05,   5.96986533e+00,
         -2.39084686e-03,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,   4.10987909e+02,   2.44279187e+00,
          4.66537068e+00,   1.21098972e+02,   2.47445758e+02,
          2.28209602e+00,  -9.57233922e-03,  -8.50457842e-03,
         -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
         -4.66042614e-01,  -4.65755574e-01,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,   3.43933101e-01,  -1.76762697e+00,
         -1.80870284e+00,  -9.86805713e-02,  -1.25793664e+00,
         -1.56668488e-01,  -4.66404784e-01,  -4.65453641e-01,
          3.08332717e+00,  -2.49631966e-01]), 1)
 (array([ -6.68331854e-02,  -1.71082237e-03,   4.30261487e+00,
         -2.39084686e-03,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,   2.44279187e+00,
          4.50927173e+02,   1.21098972e+02,   2.47445758e+02,
          4.42598759e+02,  -9.57233922e-03,  -8.50457842e-03,
         -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
          2.15261153e+00,   2.15030568e+00,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,  -3.52980002e+00,  -1.73930177e+00,
         -1.06310134e-01,   3.67861722e+00,  -9.66868885e-01,
         -1.56668488e-01,  -9.97573087e-02,   5.78635425e-02,
         -2.50831829e-01,  -2.49631966e-01]), 1)
 (array([ -2.25933342e-02,  -1.78517726e-03,  -1.30636711e-03,
         -2.39084686e-03,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,   4.10987909e+02,  -4.09367663e-01,
         -2.09732783e-03,  -8.25770840e-03,  -4.54646139e-03,
         -3.28458917e-03,  -9.57233922e-03,  -8.50457842e-03,
         -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
         -4.66042614e-01,  -4.65755574e-01,   4.05736553e+00,
          4.04998007e+00,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,  -2.74880544e+00,  -1.63544270e+00,
         -1.15206565e+00,   7.30482359e-01,  -1.21635553e+00,
         -1.56668488e-01,  -4.14026573e-01,  -3.08458486e-01,
         -3.43279980e-02,   5.72958317e-01]), 1)
 (array([  2.05917949e+01,  -5.07335308e-04,   3.51068326e-01,
         -2.39084686e-03,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,   2.44279187e+00,
          1.96031559e+02,   1.21098972e+02,   2.47445758e+02,
          2.17615735e+02,  -9.57233922e-03,  -8.50457842e-03,
          1.97096162e+02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
         -4.66042614e-01,  -4.65755574e-01,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01
         2.48130352e-01,   5.39733093e-01,   2.56056520e-01,
         -2.01059296e-01,  -3.60789948e+00,  -1.76762697e+00,
         -6.17027945e-01,   8.92998245e+00,  -2.18408950e-01,
          2.40800124e+01,  -4.66404784e-01,  -4.65453641e-01,
         -2.50831829e-01,  -2.49631966e-01]), 1)
 (array([  1.79249613e+01,  -1.17121828e-03,   1.87948948e-01,
         -2.39084686e-03,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,   2.44279187e+00,
          9.67182118e+01,   1.21098972e+02,   2.47445758e+02,
          1.06901742e+02,  -9.57233922e-03,  -8.50457842e-03,
          1.40774757e+02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
         -4.66042614e-01,  -4.65755574e-01,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,  -2.95186403e+00,  -1.70153484e+00,
         -1.34662482e+00,   7.30482359e-01,  -1.21635553e+00,
         -1.56668488e-01,  -4.66404784e-01,  -4.65453641e-01,
         -2.50831829e-01,  -2.49631966e-01]), 1)
 (array([  1.84821070e+01,  -9.01416243e-04,   2.22193258e-01,
         -2.39084686e-03,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,   2.44279187e+00,
          2.04588583e+02,   1.21098972e+02,   2.47445758e+02,
          2.20155047e+02,  -9.57233922e-03,  -8.50457842e-03,
          5.62926489e+01,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
         -4.66042614e-01,  -4.65755574e-01,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,  -1.40283539e-01,  -1.43716630e+00,
         -1.41958451e+00,  -6.55135681e-03,  -1.25793664e+00,
         -1.56668488e-01,  -4.66404784e-01,  -3.86956063e-01,
         -2.07531063e-01,  -3.31608388e-02]), 1)
 (array([  5.41464769e-01,  -1.77667956e-03,   9.25467233e-03,
         -2.39084686e-03,  -1.51391734e-02,   2.77195987e+02,
         -2.65207600e-02,  -4.39091558e-03,   2.44279187e+00,
         -2.09732783e-03,  -8.25770840e-03,  -4.54646139e-03,
         -3.28458917e-03,  -9.57233922e-03,  -8.50457842e-03,
         -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
         -4.66042614e-01,  -4.65755574e-01,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,  -3.60789948e+00,  -1.69209310e+00,
          5.98966843e-01,  -2.82939000e-01,  -2.18408950e-01,
          1.68090081e+01,  -4.66404784e-01,  -4.65453641e-01,
         -2.50831829e-01,  -2.49631966e-01]), 1)
 (array([  1.23604175e+01,  -1.45482909e-03,   4.94980318e-01,
         -2.39084686e-03,  -1.51391734e-02,   1.38597442e+02,
         -2.65207600e-02,   2.73990476e+02,   2.44279187e+00,
          2.70451743e+02,   1.21098972e+02,   2.47445758e+02,
          2.65354797e+02,  -9.57233922e-03,  -8.50457842e-03,
         -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
         -4.66042614e-01,  -4.65755574e-01,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,  -3.62351937e+00,  -1.68265137e+00,
          5.98966843e-01,  -2.82939000e-01,   8.21118739e-01,
          1.75361085e+01,  -4.66404784e-01,  -4.65453641e-01,
         -2.50831829e-01,  -2.49631966e-01]), 1)
 (array([  2.02517010e+01,  -4.41478117e-04,   4.95267134e-01,
         -2.39084686e-03,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,   2.44279187e+00,
          2.29222442e+02,   1.21098972e+02,   2.47445758e+02,
          2.52150376e+02,  -9.57233922e-03,  -8.50457842e-03,
          2.25256864e+02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
         -4.66042614e-01,  -4.65755574e-01,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,  -2.98310381e+00,  -1.68265137e+00,
         -1.20070544e+00,   6.38353144e-01,  -1.21635553e+00,
         -1.56668488e-01,  -4.14026573e-01,  -2.29960908e-01,
         -2.50831829e-01,  -2.49631966e-01]), 1)
 (array([ -6.68331854e-02,  -1.94875803e-03,  -1.69550700e-03,
          4.18261837e+02,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,  -4.09367663e-01,
         -2.09732783e-03,  -8.25770840e-03,  -4.54646139e-03,
         -3.28458917e-03,  -9.57233922e-03,  -8.50457842e-03,
         -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.21219170e+00,  -1.19217808e+00,
          2.15261153e+00,   2.15030568e+00,  -2.48285775e-01,
         -2.48130352e-01,  -2.00332030e+00,   7.11124897e-01,
          6.91336300e+00,  -3.62351937e+00,  -1.72986004e+00,
          5.98966843e-01,  -2.82939000e-01,   8.21118739e-01,
          1.19616719e+01,   2.15250575e+00,  -2.06340350e-02,
         -2.50831829e-01,  -2.49631966e-01]), 1)
 (array([ -6.68331854e-02,  -1.94875803e-03,  -1.69550700e-03,
          4.18261837e+02,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,  -4.09367663e-01,
         -2.09732783e-03,  -8.25770840e-03,  -4.54646139e-03,
         -3.28458917e-03,  -9.57233922e-03,  -8.50457842e-03,
         -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
          2.15261153e+00,   2.15030568e+00,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,  -3.62351937e+00,  -1.72986004e+00,
          5.98966843e-01,  -2.82939000e-01,   8.21118739e-01,
          1.19616719e+01,   2.15250575e+00,  -2.06340350e-02,
         -2.50831829e-01,  -2.49631966e-01]), 1)
 (array([ -6.68331854e-02,  -1.94875803e-03,  -1.69550700e-03,
          4.18261837e+02,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,  -4.09367663e-01,
         -2.09732783e-03,  -8.25770840e-03,  -4.54646139e-03,
         -3.28458917e-03,  -9.57233922e-03,  -8.50457842e-03,
         -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19217808e+00,
          2.15261153e+00,   2.15030568e+00,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
          6.91336300e+00,  -3.62351937e+00,  -1.68265137e+00,
          5.98966843e-01,  -2.82939000e-01,   8.21118739e-01,
          6.38723535e+00,   2.15250575e+00,  -2.29960908e-01,
         -2.50831829e-01,  -2.49631966e-01]), 1)
 (array([  3.58571953e+00,   1.83803047e-03,   1.34956157e-01,
         -2.39084686e-03,  -1.51391734e-02,   1.94037853e+03,
          8.50266186e+00,  -4.39091558e-03,   2.44279187e+00,
          1.65933445e+01,   1.21098972e+02,   2.47445758e+02,
          4.95132952e+01,   1.60953369e+01,  -8.50457842e-03,
          5.62926489e+01,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
         -4.66042614e-01,  -4.65755574e-01,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,  -3.45170056e+00,  -1.77706870e+00,
         -1.63846357e+00,   2.75732508e+00,  -1.09161221e+00,
         -1.56668488e-01,  -2.56891941e-01,  -4.65453641e-01,
         -2.50831829e-01,  -2.49631966e-01]), 1)
 (array([ -6.68331854e-02,  -1.94875803e-03,  -1.69550700e-03,
          4.18261837e+02,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,  -4.09367663e-01,
         -2.09732783e-03,  -8.25770840e-03,  -4.54646139e-03,
         -3.28458917e-03,  -9.57233922e-03,  -8.50457842e-03,
         -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19217808e+00,
          2.15261153e+00,   2.15030568e+00,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
          6.91336300e+00,  -3.62351937e+00,  -1.77706870e+00,
          5.98966843e-01,  -2.82939000e-01,   8.21118739e-01,
         -1.56668488e-01,   2.15250575e+00,   2.15113227e+00,
         -2.50831829e-01,  -2.49631966e-01]), 1)
 (array([ -6.68331854e-02,  -1.94875803e-03,  -1.69550700e-03,
          4.18261837e+02,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,  -4.09367663e-01,
         -2.09732783e-03,  -8.25770840e-03,  -4.54646139e-03,
         -3.28458917e-03,  -9.57233922e-03,  -8.50457842e-03,
         -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19217808e+00,
          2.15261153e+00,   2.15030568e+00,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
          6.91336300e+00,  -3.62351937e+00,  -1.76762697e+00,
          5.98966843e-01,  -2.82939000e-01,   8.21118739e-01,
          2.40800124e+01,   2.15250575e+00,   2.15113227e+00,
         -2.50831829e-01,  -2.49631966e-01]), 1)
 (array([ -6.68331854e-02,  -1.94875803e-03,  -1.69550700e-03,
          4.18261837e+02,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,  -4.09367663e-01,
         -2.09732783e-03,  -8.25770840e-03,  -4.54646139e-03,
         -3.28458917e-03,  -9.57233922e-03,  -8.50457842e-03,
         -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19217808e+00,
          2.15261153e+00,   2.15030568e+00,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
          6.91336300e+00,  -3.62351937e+00,  -1.75818524e+00,
          5.98966843e-01,  -2.82939000e-01,   8.21118739e-01,
          2.40800124e+01,   2.15250575e+00,   2.15113227e+00,
         -2.50831829e-01,  -2.49631966e-01]), 1)
 (array([ -6.68331854e-02,  -1.94875803e-03,  -1.69550700e-03,
          4.18261837e+02,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,  -4.09367663e-01,
         -2.09732783e-03,  -8.25770840e-03,  -4.54646139e-03,
         -3.28458917e-03,  -9.57233922e-03,  -8.50457842e-03,
         -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19217808e+00,
          2.15261153e+00,   2.15030568e+00,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
          6.91336300e+00,  -3.62351937e+00,  -1.74874350e+00,
          5.98966843e-01,  -2.82939000e-01,   8.21118739e-01,
          2.40800124e+01,   2.15250575e+00,   2.15113227e+00,
         -2.50831829e-01,  -2.49631966e-01]), 1)
 (array([ -6.68331854e-02,  -1.94875803e-03,  -1.69550700e-03,
          4.18261837e+02,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,  -4.09367663e-01,
         -2.09732783e-03,  -8.25770840e-03,  -4.54646139e-03,
         -3.28458917e-03,  -9.57233922e-03,  -8.50457842e-03,
         -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19217808e+00,
          2.15261153e+00,   2.15030568e+00,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
          6.91336300e+00,  -3.62351937e+00,  -1.73930177e+00,
          5.98966843e-01,  -2.82939000e-01,   8.21118739e-01,
          2.40800124e+01,   2.15250575e+00,   2.15113227e+00,
         -2.50831829e-01,  -2.49631966e-01]), 1)
 (array([  1.18571892e+01,  -1.02888177e-03,   1.98260380e-01,
         -2.39084686e-03,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,   2.44279187e+00,
          1.39503335e+02,   1.21098972e+02,   2.47445758e+02,
          1.53625078e+02,  -9.57233922e-03,  -8.50457842e-03,
          1.40774757e+02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
         -4.66042614e-01,  -4.65755574e-01,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,   3.43933101e-01,  -1.68265137e+00,
         -1.73574315e+00,   2.69836287e-01,  -1.25793664e+00,
         -1.56668488e-01,   1.81204738e+00,  -4.65453641e-01,
         -2.07531063e-01,   5.29664092e-01]), 1)
 (array([  1.84973144e+01,   8.97822466e-03,   2.41892720e-02,
         -2.39084686e-03,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,   2.44279187e+00,
          2.57206450e-01,  -8.25770840e-03,  -4.54646139e-03,
         -3.28458917e-03,   3.05983703e+02,  -8.50457842e-03,
         -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
         -4.66042614e-01,  -4.65755574e-01,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,   3.43933101e-01,  -1.67320964e+00,
         -1.71142326e+00,   2.69836287e-01,  -1.25793664e+00,
         -1.56668488e-01,   1.81204738e+00,  -4.65453641e-01,
         -2.07531063e-01,   4.86369866e-01]), 1)
 (array([  5.28260565e+01,   6.60236218e+02,  -1.69550700e-03,
         -2.39084686e-03,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,  -4.09367663e-01,
         -2.09732783e-03,  -8.25770840e-03,  -4.54646139e-03,
         -3.28458917e-03,  -9.57233922e-03,  -8.50457842e-03,
         -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
         -4.66042614e-01,  -4.65755574e-01,   4.05736553e+00,
          4.04998007e+00,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,  -2.48326728e+00,  -1.77706870e+00,
         -1.80870284e+00,   4.54094715e-01,  -1.19556498e+00,
         -1.56668488e-01,  -4.66404784e-01,  -4.65453641e-01,
         -1.20929530e-01,   4.07979057e+00]), 1)
 (array([  4.98011567e+01,   1.46581330e+03,  -1.69550700e-03,
         -2.39084686e-03,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,  -4.09367663e-01,
         -2.09732783e-03,  -8.25770840e-03,  -4.54646139e-03,
         -3.28458917e-03,  -9.57233922e-03,  -8.50457842e-03,
         -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
         -4.66042614e-01,  -4.65755574e-01,   4.05736553e+00,
          4.04998007e+00,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,  -2.43640760e+00,  -1.77706870e+00,
         -1.80870284e+00,   8.22611573e-01,  -1.13319332e+00,
         -1.56668488e-01,  -4.66404784e-01,  -4.65453641e-01,
          8.97276823e-03,   4.07979057e+00]), 1)
 (array([  4.92633660e+01,   4.05454318e+02,  -1.69550700e-03,
         -2.39084686e-03,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,  -4.09367663e-01,
         -2.09732783e-03,  -8.25770840e-03,  -4.54646139e-03,
         -3.28458917e-03,  -9.57233922e-03,  -8.50457842e-03,
         -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
         -4.66042614e-01,  -4.65755574e-01,   4.05736553e+00,
          4.04998007e+00,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,  -2.42078771e+00,  -1.76762697e+00,
         -1.76006305e+00,   8.22611573e-01,  -1.09161221e+00,
         -1.56668488e-01,  -4.66404784e-01,  -4.65453641e-01,
          9.55743006e-02,   1.91507930e+00]), 1)
 (array([  4.77370911e+01,   1.24015216e+03,  -1.69550700e-03,
         -2.39084686e-03,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,  -4.09367663e-01,
         -2.09732783e-03,  -8.25770840e-03,  -4.54646139e-03,
         -3.28458917e-03,  -9.57233922e-03,  -8.50457842e-03,
         -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
         -4.66042614e-01,  -4.65755574e-01,   4.05736553e+00,
          4.04998007e+00,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,  -2.38954793e+00,  -1.77706870e+00,
         -1.80870284e+00,   8.22611573e-01,  -1.05003110e+00,
         -1.56668488e-01,  -4.66404784e-01,  -4.65453641e-01,
          1.82175833e-01,   4.07979057e+00]), 1)
 (array([ -6.68331854e-02,  -1.64496518e-03,   2.86564656e-03,
         -2.39084686e-03,  -1.51391734e-02,  -1.10348462e-03,
          2.10577489e+00,  -4.39091558e-03,   2.44279187e+00,
          1.55372534e+00,  -8.25770840e-03,  -4.54646139e-03,
          2.28209602e+00,  -9.57233922e-03,  -8.50457842e-03,
         -2.87561127e-02,   0.00000000e+00,   1.56499665e+03,
         -2.89113034e-02,  -1.57069789e+00,  -1.19624324e+00,
          8.43284460e-01,   2.15030568e+00,  -2.48285775e-01,
         -2.48130352e-01,  -7.44637309e-01,   1.18337112e+01,
         -2.01059296e-01,   3.43933101e-01,  -1.77706870e+00,
         -1.83302273e+00,  -9.86805713e-02,  -1.25793664e+00,
         -1.56668488e-01,  -4.66404784e-01,   2.15113227e+00,
         -2.50831829e-01,  -2.49631966e-01]), 1)
 (array([  1.70000719e+01,   6.62329896e-03,   2.08017395e-02,
         -2.39084686e-03,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,   2.44279187e+00,
          2.57206450e-01,  -8.25770840e-03,  -4.54646139e-03,
         -3.28458917e-03,   2.81826339e+02,  -8.50457842e-03,
          1.40774757e+02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
         -4.66042614e-01,  -4.65755574e-01,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,  -3.60789948e+00,  -1.74874350e+00,
          5.98966843e-01,  -2.82939000e-01,  -2.18408950e-01,
          1.19616719e+01,  -4.66404784e-01,  -4.65453641e-01,
         -2.50831829e-01,  -2.49631966e-01]), 1)
 (array([  2.08959438e+01,   9.01433989e-03,   2.20869864e-02,
         -2.39084686e-03,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,   2.44279187e+00,
         -2.09732783e-03,  -8.25770840e-03,  -4.54646139e-03,
         -3.28458917e-03,   2.89878793e+02,  -8.50457842e-03,
          2.81319464e+01,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
         -4.66042614e-01,  -4.65755574e-01,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,   3.43933101e-01,  -1.15391431e+00,
         -1.20070544e+00,  -9.86805713e-02,  -1.25793664e+00,
         -1.56668488e-01,   1.91680380e+00,   1.88947368e+00,
         -2.50831829e-01,  -2.49631966e-01]), 1)
 (array([  1.90682850e+01,   8.60432577e-03,   3.64479536e-02,
         -2.39084686e-03,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,   2.44279187e+00,
          5.16510228e-01,  -8.25770840e-03,  -4.54646139e-03,
         -3.28458917e-03,   3.46245976e+02,  -8.50457842e-03,
          1.40774757e+02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
         -4.66042614e-01,  -4.65755574e-01,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,   3.43933101e-01,  -1.20112297e+00,
         -1.24934524e+00,  -9.86805713e-02,  -1.25793664e+00,
         -1.56668488e-01,   1.49777812e+00,   1.65398095e+00,
         -2.50831829e-01,  -2.49631966e-01]), 1)
 (array([  1.52028280e+01,   1.46460527e-06,   1.99387490e-01,
         -2.39084686e-03,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,   2.73990476e+02,   2.44279187e+00,
          1.94456860e+01,  -8.25770840e-03,  -4.54646139e-03,
          1.87876226e+01,   1.60953369e+01,  -8.50457842e-03,
         -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
         -4.66042614e-01,  -4.65755574e-01,   4.05736553e+00,
          4.04998007e+00,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,   3.43933101e-01,  -1.28609857e+00,
         -1.32230492e+00,  -9.86805713e-02,  -1.25793664e+00,
         -1.56668488e-01,   8.69239589e-01,   1.15682963e+00,
         -2.50831829e-01,  -1.63043515e-01]), 1)
 (array([  1.71811788e+01,   6.65728977e-03,   3.05938013e-02,
         -2.39084686e-03,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,   2.44279187e+00,
          2.57206450e-01,  -8.25770840e-03,  -4.54646139e-03,
         -3.28458917e-03,   2.81826339e+02,  -8.50457842e-03,
          1.40774757e+02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
         -4.66042614e-01,  -4.65755574e-01,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,  -3.60789948e+00,  -1.76762697e+00,
          5.98966843e-01,  -2.82939000e-01,  -2.18408950e-01,
         -1.56668488e-01,  -4.66404784e-01,  -4.65453641e-01,
         -2.50831829e-01,  -2.49631966e-01]), 1)
 (array([  2.02060787e+01,   7.31373726e-03,   2.21676050e-02,
         -2.39084686e-03,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,   2.44279187e+00,
          2.57206450e-01,  -8.25770840e-03,  -4.54646139e-03,
         -3.28458917e-03,   3.30141066e+02,  -8.50457842e-03,
         -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
         -4.66042614e-01,  -4.65755574e-01,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,  -3.12368284e+00,  -1.72041830e+00,
         -1.32230492e+00,   5.46223930e-01,  -1.19556498e+00,
         -1.56668488e-01,  -4.66404784e-01,  -4.65453641e-01,
         -2.50831829e-01,  -2.49631966e-01]), 1)
 (array([ -6.68331854e-02,  -1.94875803e-03,  -1.69550700e-03,
          4.18261837e+02,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,  -4.09367663e-01,
         -2.09732783e-03,  -8.25770840e-03,  -4.54646139e-03,
         -3.28458917e-03,  -9.57233922e-03,  -8.50457842e-03,
         -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
          2.15261153e+00,   2.15030568e+00,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,  -3.62351937e+00,  -1.77706870e+00,
          5.98966843e-01,  -2.82939000e-01,   8.21118739e-01,
         -1.56668488e-01,   2.15250575e+00,   2.15113227e+00,
         -2.50831829e-01,  -2.49631966e-01]), 1)
 (array([  1.78890164e+01,  -1.07243250e-03,   1.53523246e-01,
         -2.39084686e-03,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,   2.44279187e+00,
          1.47282449e+02,   1.21098972e+02,   2.47445758e+02,
          1.58957633e+02,  -9.57233922e-03,  -8.50457842e-03,
          1.12614054e+02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
         -4.66042614e-01,  -4.65755574e-01,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,  -3.15492262e+00,  -1.71097657e+00,
         -1.20070544e+00,   9.14740788e-01,  -1.19556498e+00,
         -1.56668488e-01,  -4.66404784e-01,  -4.65453641e-01,
         -2.50831829e-01,  -2.49631966e-01]), 1)
 (array([ -6.68331854e-02,  -1.94875803e-03,  -1.69550700e-03,
          4.18261837e+02,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,  -4.09367663e-01,
         -2.09732783e-03,  -8.25770840e-03,  -4.54646139e-03,
         -3.28458917e-03,  -9.57233922e-03,  -8.50457842e-03,
         -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19217808e+00,
          2.15261153e+00,   2.15030568e+00,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
          6.91336300e+00,  -3.40484088e+00,  -1.77706870e+00,
         -1.66278346e+00,   1.55964529e+00,  -1.11240276e+00,
         -1.56668488e-01,  -2.83081046e-01,   2.15113227e+00,
          5.22735344e-02,  -2.49631966e-01]), 1)
 (array([ -6.68331854e-02,  -1.94875803e-03,  -1.69550700e-03,
          4.18261837e+02,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,  -4.09367663e-01,
         -2.09732783e-03,  -8.25770840e-03,  -4.54646139e-03,
         -3.28458917e-03,  -9.57233922e-03,  -8.50457842e-03,
         -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.56598070e+00,  -1.19217808e+00,
          3.98113255e-01,   2.15030568e+00,   2.63650060e+00,
         -2.48130352e-01,  -1.18132325e+00,   1.18337112e+01,
          6.91336300e+00,  -3.38922099e+00,  -1.76762697e+00,
         -1.54118398e+00,   1.46751607e+00,  -1.00844999e+00,
         -1.56668488e-01,  -1.52135519e-01,   2.15113227e+00,
          8.97276823e-03,  -2.49631966e-01]), 1)
 (array([  2.23116191e+01,  -7.60141945e-04,   2.71688441e-01,
         -2.39084686e-03,  -1.51391734e-02,  -1.10348462e-03,
         -2.65207600e-02,  -4.39091558e-03,   2.44279187e+00,
          1.79176813e+02,   1.21098972e+02,   2.47445758e+02,
          1.94507998e+02,  -9.57233922e-03,  -8.50457842e-03,
          2.25256864e+02,   0.00000000e+00,  -6.38979005e-04,
         -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
         -4.66042614e-01,  -4.65755574e-01,  -2.48285775e-01,
         -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
         -2.01059296e-01,  -3.60789948e+00,  -1.76762697e+00,
          5.98966843e-01,  -2.82939000e-01,  -2.18408950e-01,
         -1.56668488e-01,  -4.66404784e-01,  -4.65453641e-01,
         -2.50831829e-01,  -2.49631966e-01]), 1)
```

```
(array([  1.38340444e+00,  -1.14147633e-03,   1.03792409e-01,
        -2.39084686e-03,  -1.51391734e-02,  -1.10348462e-03,
        -2.65207600e-02,   2.73990476e+02,   2.44279187e+00,
        -2.09732783e-03,  -8.25770840e-03,  -4.54646139e-03,
        -3.28458917e-03,  -9.57233922e-03,  -8.50457842e-03,
        -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
        -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
        -4.66042614e-01,  -4.65755574e-01,  -2.48285775e-01,
        -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
        -2.01059296e-01,   2.65833643e-01,  -7.29036307e-01,
        -7.38627424e-01,  -1.90809786e-01,  -1.25793664e+00,
        -1.56668488e-01,   1.62872364e+00,   1.86330782e+00,
        -2.50831829e-01,  -2.49631966e-01]), 1)
(array([  2.10341934e+01,   8.60220134e-03,   2.12063830e-02,
        -2.39084686e-03,  -1.51391734e-02,  -1.10348462e-03,
        -2.65207600e-02,  -4.39091558e-03,   2.44279187e+00,
        -2.09732783e-03,  -8.25770840e-03,  -4.54646139e-03,
        -3.28458917e-03,   3.22088612e+02,  -8.50457842e-03,
        -2.87561127e-02,   0.00000000e+00,  -6.38979005e-04,
        -2.89113034e-02,  -1.57541507e+00,  -1.19624324e+00,
        -4.66042614e-01,  -4.65755574e-01,  -2.48285775e-01,
        -2.48130352e-01,   5.39733093e-01,  -2.56056520e-01,
        -2.01059296e-01,   3.43933101e-01,  -7.19594574e-01,
        -7.62947320e-01,  -1.90809786e-01,  -1.25793664e+00,
        -1.56668488e-01,   1.57634543e+00,   1.83714196e+00,
        -2.50831829e-01,  -2.49631966e-01]), 1)
```

Question 13

Try other methods to find the best value for $k$ such as `silhouette`, `entropy`... In particular, with this data, you can take advantage of predefined labels to calculate the quality of model using entropy... However, we suggest you to try with `silhouette`. It's more general and can work with any dataset (with and without predefined labels).

Here are some additional information about the metrics we suggest to use:

- Silhouette
- Hack approach to Silhouette
- Entropy [Lookup for entropy]

Note you are free to play with any relevant evaluation metric you think appropriate for your work!

In [14]:

```python
# Silhouette

def Silhouette(pair):
    point_cluster = clusters.predict(pair[0])

    groups = pair[1].groupBy(lambda x: clusters.predict(x))

    aAndC = groups.map(lambda x: x[1].map(lambda y: euclidean_distance(pair[
0], y)).reduceByKey(np.mean))
    a = aAndC.filter(lambda x: x[0]==point_cluster)[0][1]
    c = aAndC.filter(lambda x: x[0]!=point_cluster)[:][1]
    b = c.min()
    return 1-a/b if a<b else a/b-1
```

In [ ]:

```
#s = data.map(lambda point: Silhouette(point)).collect()
s = data.cartesian(data).groupBy(lambda x: tuple(x[0])).map(lambda pair: Sil
houette(pair))
ss = s.collect()

plt.hist(s, 100)
plt.title("Silhouette distridution")
plt.xlabel("S index")
plt.ylabel("Num of items")
plt.show()
```

Comment

We tried to implement Silhouette algorithm because compared with the previous one gives a better eximation of how good data is splitted amoung different groups. But it takes a time that is the square of the previous one because the distance it's evaluated between one point and all the others and this must be done for all the points.

Question 14

Implement K-means on Spark so that It can work with large datasets in parallel. Test your algorithm with our dataset in this notebook. Compare our algorithm with the algorithm from MLLIB.
Let's clarify the meaning of this question: what we want is for students to design the K-means algorithm for the parallel programming model exposed by Spark. You are strongly invited to use the Python API (pyspark). So, at the end of the day, you will operate on RDDs, and implement a `map/reduce` algorithm that performs the two phases of the standard K-means algorithm, i.e. the assignment step and the update step.