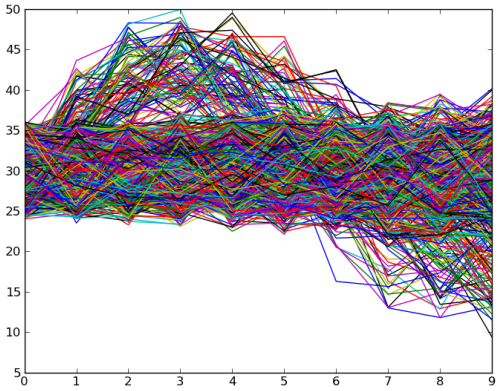
SVD ile Kumeleme

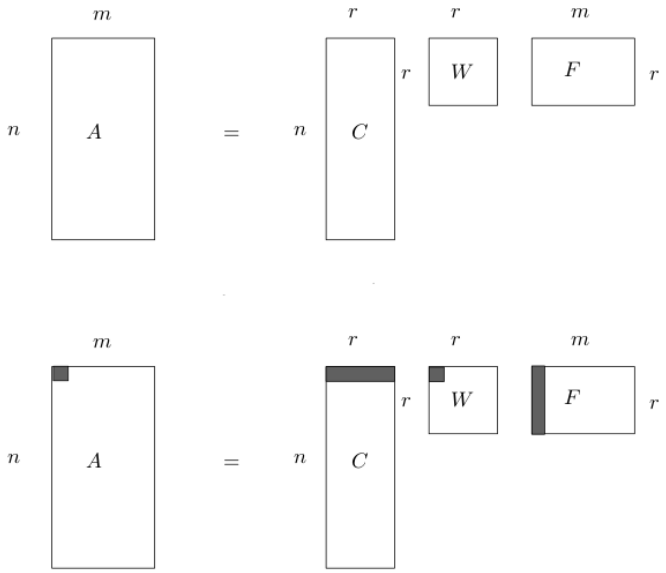
Tekil Deger Ayristirma (Singular Value Decomposition -SVD-) ile bir veri madenciligi ornegi gorecegiz. Ornek olarak [1] adresinde tarif edilen / paylasilan zaman serisini kullandik. Once veriyi grafikledik,



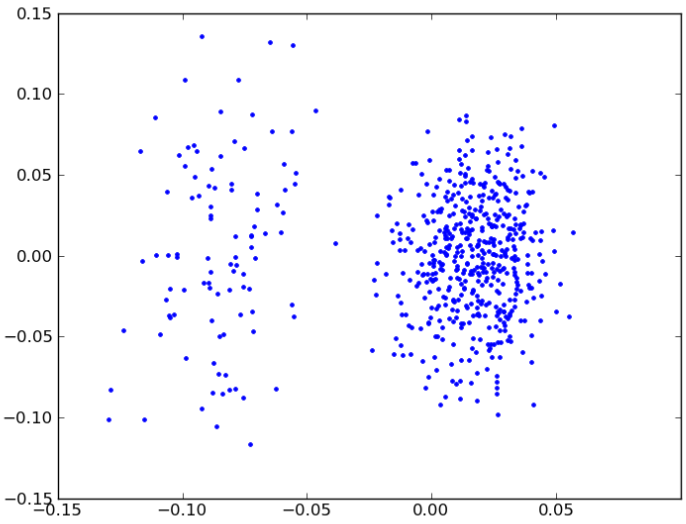
Verinin tamami kullanilmadi, serinin ilk 10 noktasini aldik, ve grafige bakinca iki tane ana seri oldugunu goruyoruz.

```
import numpy as np
from pylab import *

data = np.genfromtxt("synthetic_control.data", dtype=float)

print data.shape

for t in data[:,0:10]:
    plot(t); hold(True)

show()
```

Peki bu serileri nasil otomatik olarak kumeleyerek bulurduk / birbirinden ayirtederdik? *Lineer Cebir Ders 29*'da SVD'nin matematigini isledik. SVD bir matris A uzerinde ayristirma yapar, ve A herhangi boyutta, turde bir matris olabilir.

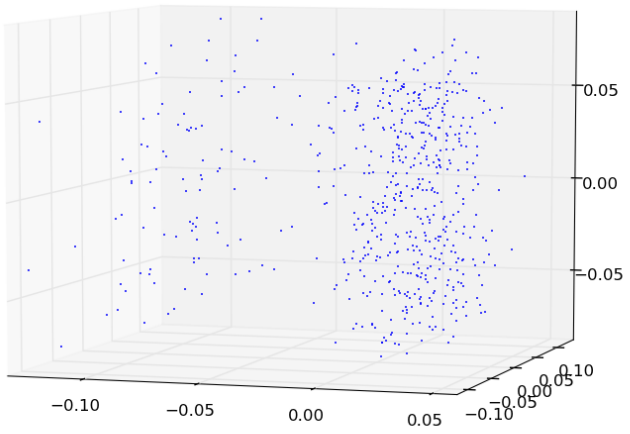


Ayristirmanin $A = CWF$ sonucunu verir, burada C, ana matris ile ayni miktarda satira sahiptir, F ayni miktarda kolona sahiptir. Ayristirma sonrasi A'nin kertesi (rank) ortaya cikar, eger tum A kolonlari birbirinden bagimsiz ise, o zaman $r = m$ olacaktir, ama kolonlarin bazilari mesela ayni olcumu degisik katlarda tekrarliyor ise, o zaman matriste tekillik vardir, ve bu durumda $r < m$ olur, ve ortadaki W matrisi $r \times r$ oldugu icin beklenenden daha ufak boyutlarda olabilir.



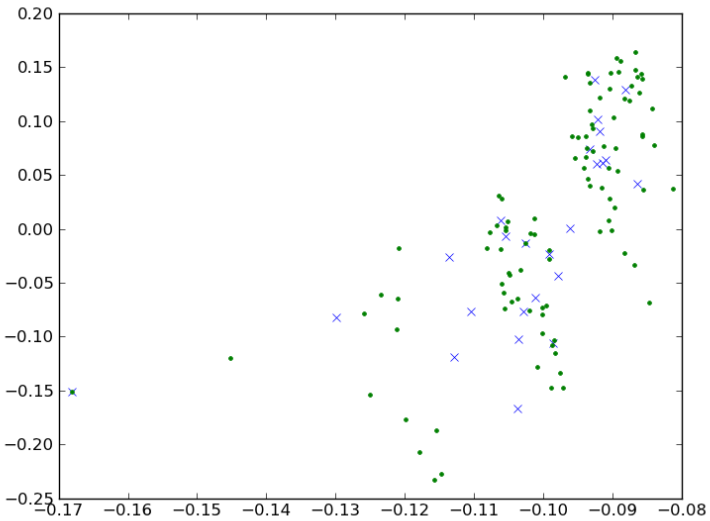

```
import scipy.linalg as lin
import numpy as np
from pylab import *

data = np.genfromtxt("synthetic_control.data", dtype=float)

# before norm, and take only 10 data points
data = data[:,0:10]
```

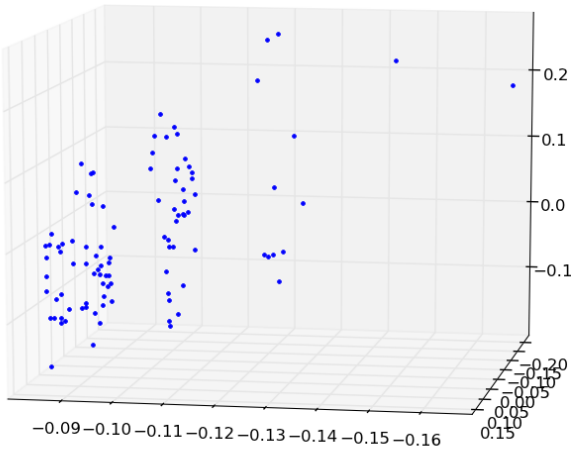
```
print data.shape

# show the mean, and std of the first time series
print data[0,:]
print np.mean(data[0,:], axis=0)
print np.std(data[0,:], axis=0)

# normalize
data -= np.mean(data, axis=0)
data /= np.std(data, axis=0)

# after norm
print data[0,:]

u,s,v = lin.svd(data, full_matrices=False)
print 'svd'
print u.shape
print s
print v.shape

plot(u[:,0], u[:,1], '.')

print u[:,0] < -0.025

print u[:,0].shape

show()
```




```
from mpl_toolkits.mplot3d import Axes3D
import scipy.linalg as lin
import numpy as np
from pylab import *

data = np.genfromtxt("synthetic_control.data", dtype=float)
```

```
data = data[:,0:10]

print data.shape

data -= np.mean(data, axis=0)
data /= np.std(data, axis=0)

u,s,v = lin.svd(data)
print 'svd'
print u.shape
print s
print v.shape

fig = plt.figure()
ax = Axes3D(fig)
ax.plot(u[:,0], u[:,1], u[:,2],',', zs=0, zdir='z', label='zs=0, zdir=z')

show()
```






```
import numpy as np
import scipy.linalg as lin
import Levenshtein as leven
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
import itertools
import time

words = np.array(
    ['the', 'be', 'to', 'of', 'and', 'a', 'in', 'that', 'have',
     'I', 'it', 'for', 'not', 'on', 'with', 'he', 'as', 'you',
     'do', 'at', 'this', 'but', 'his', 'by', 'from', 'they', 'we',
     'say', 'her', 'she', 'or', 'an', 'will', 'my', 'one', 'all',
     'would', 'there', 'their', 'what', 'so', 'up', 'out', 'if',
     'about', 'who', 'get', 'which', 'go', 'me', 'when', 'make',
     'can', 'like', 'time', 'no', 'just', 'him', 'know', 'take',
     'people', 'into', 'year', 'your', 'good', 'some', 'could',
     'them', 'see', 'other', 'than', 'then', 'now', 'look',
     'only', 'come', 'its', 'over', 'think', 'also', 'back',
     'after', 'use', 'two', 'how', 'our', 'work', 'first', 'well',
     'way', 'even', 'new', 'want', 'because', 'any', 'these',
     'give', 'day', 'most', 'us'])

print "calculating distances..."

(dim,) = words.shape

f = lambda (x,y): leven.distance(x,y)
res=np.fromiter(itertools.imap(f, itertools.product(words, words)),
                dtype=float)
A = np.reshape(res,(dim,dim))

print "svd..."

u,s,v = lin.svd(A, full_matrices=False)
```

```
print u.shape
print s.shape
print s
print v.shape

k=KMeans(init='k-means++', k=25, n_init=10)
k.fit(u[:,0:10])
centroids = k.cluster_centers_
labels = k.labels_
print labels

for i in range(np.max(labels)):
    print words[labels==i]

plt.plot(centroids[:,0],centroids[:,1],'x')
plt.hold(True)
plt.plot(u[:,0], u[:,1], '.')
plt.show()

from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure()
ax = Axes3D(fig)
ax.plot(u[:,0], u[:,1], u[:,2],'.', zs=0,
        zdir='z', label='zs=0, zdir=z')
plt.show()
```

[1] http://kdd.ics.uci.edu/databases/synthetic_control/synthetic_control.data.html