Multi-platform Cancer Data Analysis with Machine Learning Methods

Xiong Yuan Peng
Zhejiang University
yuanpengxiong@foxmail.com
3140200128

Ge Xian Long
Zhejiang University
18868104399@163.com
3120102146

Hu ChunWang
Zhejiang University
hcw@zju.edu.cn
3120102551

Abstract

In our paper, a method of dealing with large number of cancer data is proposed. This method is suitable for high dimension raw data. Multi-layer PCA is used to lower data dimension. And prior to k-means algorithm, Neural network is used to learn features automatically from raw data. Our experimental result indicates the ability of clustering proper number of cancer subtypes.

1. Introduction

Given a great amount of cancer data, gathered from multi-platform, it's very hard to handle such big data for its redundant properties. And, especially with such a high dimension, it's easy to be trapped into over fitting problem while sample set is small. With deep learning method, however, we can train a neural network and learn underlying features, and well describe the data sets, automatically from raw data. But everything doesn't goes as well as we thought. It's well known that bio-data always born with high dimensions, even over 10,000 sometimes, but our computer resources restrict some reasonable algorithms to work well. So it's very important to reduce the dimensions of our data while preserve most of their information. With our analysis system, we tested the GBM cancer data from Synapse (ID: syn1710282) and tried to cluster it into several subtype groups with K-means algorithm.

2. System Overview

Since some of data in the set of multi-platform cancer genomic data we used has near 20,000 dimensions of features, a huge computing cost will be demanded if we directly run neural network. At the same time, in order to make use of these multi-platform data, we need to joint all of these data, which make our situation even worse. So we introduce multi-layer PCA here and use auto encoder to learn lower dimension feature from raw data. Our system is separated into 3 parts shown as follow:

2.1 Multi-layer PCA

For traditional PCA method we need to normalize our data, then construct a co-variance matrix and compute its eigenvectors, finally project raw data to top-k eigenvectors. But our data has 20,000 dimensions of features such that co-variance matrix become a 20,000*20,000 matrix, that's truly a huge matrix, and it's very hard to run SVD(Singular Value Decomposition) on that matrix with restricted computing resources.

Since PCA will find a lower-dimensional subspace onto which to project our data. We can view our data as several subsets who has lower dimensions, and run PCA on each subset. Denote $A=[\mathrm{A}_1.....\mathrm{A}_n]$ as our raw data, $B=[\mathrm{B}_1.....\mathrm{B}_n]$ as A processed by PCA, C as B

processed by PCA. Then we can view our multi-layer PCA as follow.

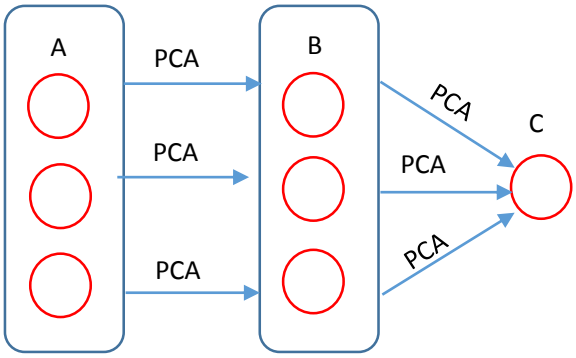


Figure 1. Multi-layer PCA

More mathematically we need to resolve that why this multi-layer PCA works.

In traditional PCA, we need to minimize function J as follow:

$$J=(X^T U)^T (X^T U) \qquad \text{s.t.} \quad u_{i,j} > 0$$

With Lagrange Multiplier Method, we rewrite J as follow:

$$J = (X^T U)^T (X^T U) - \lambda U^T U$$

$$\nabla_U J = \nabla_U tr(J)$$

$$= \nabla_U tr(UU^T XX^T - \lambda UU^T)$$
$$= \nabla_U tr(UU^T XX^T) - \nabla_U tr(\lambda UU^T$$
$$= XX^T U - \lambda U$$

In order to get minimum value, we set our gradient as:

$$\nabla_U J = 0 \rightarrow XX^T U = \lambda U$$

In order to resolve multi-layer PCA, here we denote X as [a b], then

$$X^T U = \begin{bmatrix} a^T \\ b^T \end{bmatrix} U = \begin{bmatrix} a^T U \\ b^T U \end{bmatrix}$$

$$(X^T U)^T (X^T U) = \begin{bmatrix} a^T U \\ b^T U \end{bmatrix}^T \begin{bmatrix} a^T U \\ b^T U \end{bmatrix}$$

$$= (a^T U)^T (a^T U) + (b^T U)^T (b^T U)$$

Which means that running PCA separately might also be reasonable instead of running with the whole matrix. We need to know that here we just make U the same while in real situation, U is different for different sub matrix x_i .

2.2 Auto Encoder

Neural network is always a good method to learn sophisticated functions from labeled data, and it truly works well in most of cases. But if we just label our data with itself, we will get a new method to learn features automatically from raw data. That process is what we call auto encoder.

Usually we learn an abstract function: y = f(x) to classify our data, or fit data.

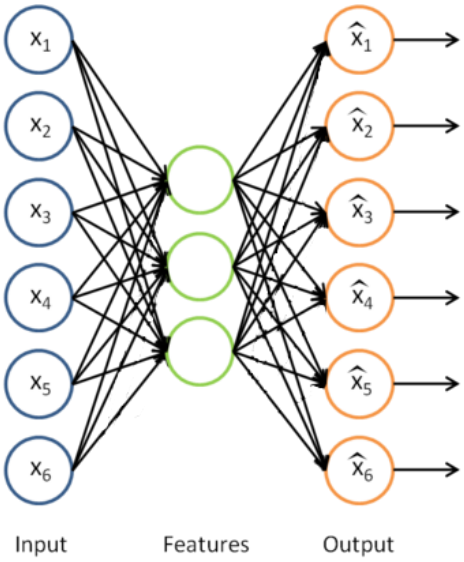


Figure 2 Auto encoder[3]

And if we trained a neural network as a function: x = f(x), it seems that this network is nonsense, but when the hidden layer has less features than its input, it's interesting for us to realize that we can reconstruct our data from a lower-dimension features.

Here we train our neural network with BP (Back Propagation) algorithm.

Cost function is[3]:

$$J(W,b)=[\frac{1}{m}\sum_{i=1}^{m}(\frac{1}{2}|h_{w,b}(x^{(i)}-y^{(i)})|^2)]+\lambda\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}(W_{ji}^{(l)})^2$$

$$\delta_i^{(l)} = ((W^{(l)})^T \delta^{(l+1)}) f^1(z_i^{(l)})$$

Gradients for hidden layer are

$$\frac{\partial}{\partial W_{ij}^{(l)}} J = a_j^{(l)} \delta_i^{(l+1)}$$

$$\frac{\partial}{\partial b_i^{(l)}} J = \delta_i^{(l+1)}$$

And to choose better parameters for learning rate and λ, we separated our samples into 3 parts: training set, cross validation set and test set. Cross validation set is used to choose parameters that has lowest error in training set.

2.3 K-means clustering

K-means Clustering is a popular unsupervised learning method to separate data set into K clusters. Here we use this method to find sub type of cancer.

3. Performance

3.1 Multi-layer PCA

Instead of deciding k arbitrary, we choose k dynamically while running multi-layer algorithm. While run SVD, we can not only get eigenvectors matrix, but also eigenvalue matrix. And we decide k by computing η, K is the lowest one that makes η>0.98(0.98 means that we preserve 98 percent of valid information of raw data).

$$\eta = \frac{\sum_{i=1}^{k}\sum_{j=1}^{k} S_{ij}}{\sum_{i=1}^{n}\sum_{j=1}^{n} S_{ij}}$$

Let's define a function P(X) as PCA operation on X, and our multi-layer PCA algorithm is:

1. $x_i = X[\mathrm{i}:\min(\mathrm{i}+5000,\mathrm{size}(\mathrm{X}))]$ while index is belong to Z={1,2,3…flooer(size(X)/5000)}
2. For each subset i($i \le l$), $z_i = \mathrm{P}(\mathrm{x}_i)$
3. $Z = [\mathrm{z}_1 \ldots . \mathrm{z}_l]$, if size(Z)>5000, X=Z, go to step 1;
4. $X_{\mathrm{new}} = \mathrm{P}(\mathrm{Z})$

After running this multi-PCA algorithm, we reduce raw data to lower dimension successfully, and the results are shown in the following chart.

Data Type	CNV	Methyl atio	miRNA	mRNA
Formal	106	16383	533	17813
First layer	74	738	133	710
Second layer	74	189	133	180

3.2 Auto Encoder

After running multi-layer PCA, we pre-process all of multi-platform data, then we group them together as the input of our auto encoder. [1]

3.2.1 Different Platform Data Layer

Here we use cancer type GBM as an example. We just run auto encoder identically for CNV (Copy Number Variations), DNA methylation, miRNA, mRNA. For each group of data, we didn't choose the size of hidden layer number arbitrarily, we choose the one that makes the lowest cost errors, and here are some notations:

1. Choose k = [floor(n/2), floor(n/3), floor(n/4)]
2. For each k, train auto encoder 100 times, and use the best W which has lowest cost error as the current best W
3. Choose sigmoid function:

   $f(\mathrm{x}) = \frac{1}{1+\exp(-\theta^T \mathrm{x})}$ for hidden layer's activation function

4. Data pre-process: $\hat{x} = \frac{x - \min(\mathrm{x})}{\max(\mathrm{x}) - \min(\mathrm{x})}$

3.2.2 Learning Curve

Here we used GBM_CNV as example, trained data with MATLAB built-in function, and plotted its learning curve as follow.

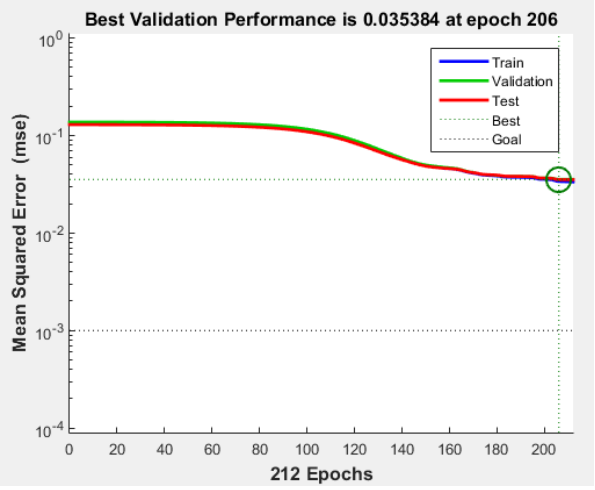


Figure 4. Learning Curve at thefirst iteration

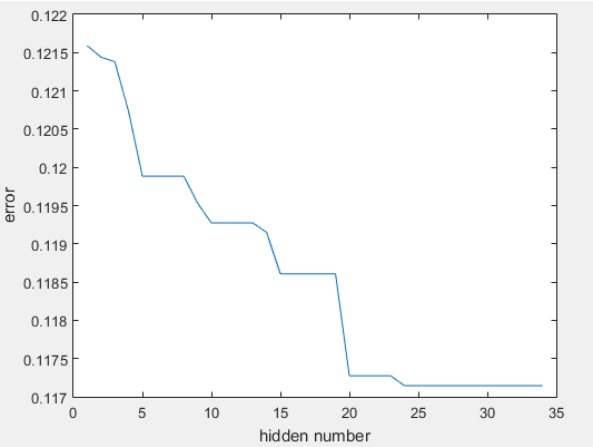
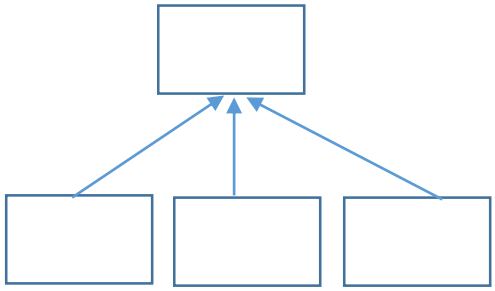


Figure 5 Error curve with different hidden layer number

Now we know exactly that our neural network works well. For every iteration, error function decreased, and the more hidden layer number, the less error cost, but when it comes to 35, error goes down slowly. The following chart show the final hidden layer number of different data subset.

3.2.3 Top Layer of auto encoder

After learning data from different groups, now we turned our eyes on making all of the features getting together, and learn new features from the whole data. This procedure can be frustrated as follow:



Here we chosen 100 arbitrarily as a reasonable feature number, and trained the final auto encoder again, and as a result the mean error is 0.2669. This part passes the activation values of hidden layer to the next part of the system.

3.2.3 The experimental verification

we run multi-layer PCA and auto-encoder on KIRC and GBM data set, and we use this two group of data to train a classifier, and here is our training result. (repeated 10 times)

(1) Neural network

	Neural network size	**Train data**	**Error number**	**Test data**	**Error number**
Non-randomized data	100/10/2	300	5(2%)	153	20(13%)
Randomized data	100/10/2	300	2(0.6%)	153	12(7.8%)

(2) SVM: 0 error on both test data and training data.

3.3 K-means algorithm

SPSS is a very convenient tool on statistical analysis, developed by IBM, and we use it to find our k-mean center, and check it with our own MATLAB k-means program. Finally we visualize all of the data on 2 dimension axis with PCA method.

Data Type	CNV	methylation	miRNA	mRNA
Input layer	74	189	133	180
Hidden layer	37	47	66	45
Mean Error	0.1101	0.1233	0.1170	0.1208

3.3.1 SPSS Analysis

Cluster distance	**1**	**2**	**3**	**4**
1		1.290	1.316	1.172
2	1.290		1.302	1.335
3	1.316	1.302		1.380
4	1.172	1.335	1.380	
Element number	63	69	31	47

Table above shows the distance between different clusters, and the element number of different clusters.

3.3.2 Variants Visualization

With the clustering center compute by SPSS, we checked again on our MATLAB, and iterated for about 100 times. Figure 6 below shows how different subtype data scattered on 2 dimension axis. It seems that all of the data overlapped. That problem may come from the PCA: high dimension data will overlap in low dimension.

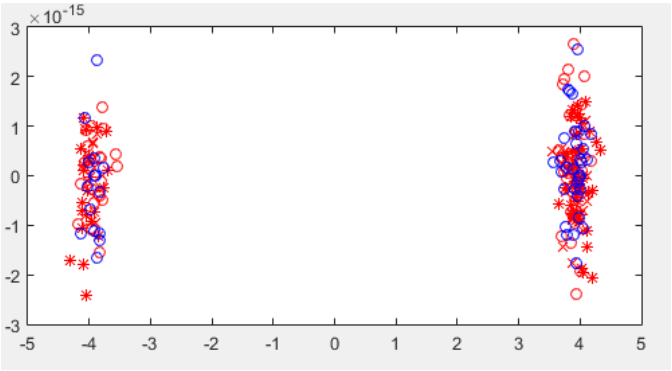


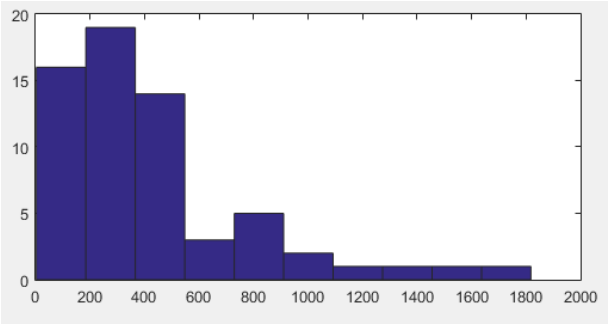
Figure 6 sample clustered into 2 groups in 2 dimensions

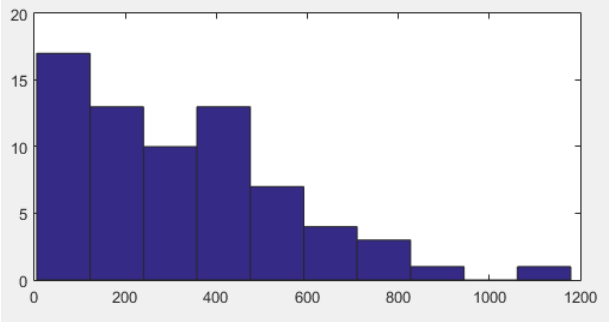
Group ID	1	2	3	4
Mean value	421.428	323.826	438.677	382.44
Standard deviation	379.35	249.36	679.29	267.31
Max value	1818	1179	3880	1245
Min value	3	5	4	6

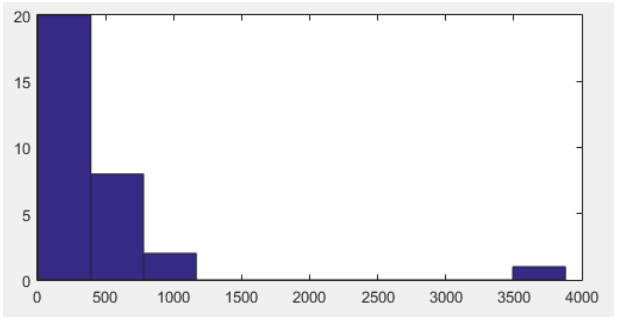
axis

3.3.3 Statistical Analysis

With data divided into 4 groups, we added survival time in, and plotted histogram of each group. It seems apparently that these 4 group of data have different distribution which strongly reveals that we have clustered data successfully.







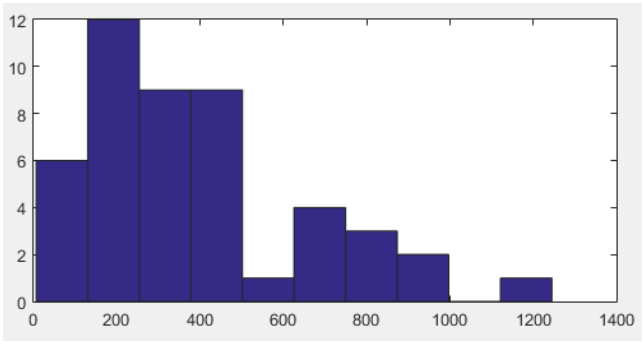
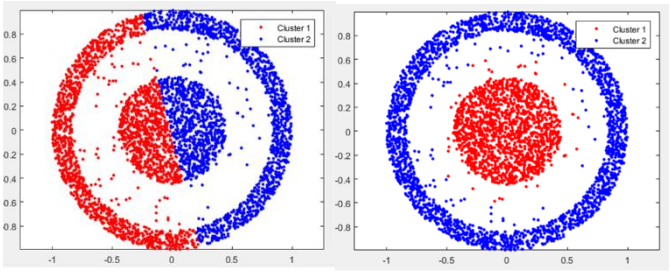


Figure 7 histogram of different group

By analyzing the table above, we can draw the conclusion statistically that there are 4 subtype of cancer in GBM cancer.

3.3.4 Spectral Clustering

K-means algorithm is sort of unstable with different initial centroids. And for some special structure of data, it can't works well. So we turned to spectral clustering.



k-means spectral clustering

Spectral clustering is a clustering method based on dimension reduction. when it's hard to cluster on the original features space, we need to find a method to project it to a lower(may be higher space just like some kernel method) dimension space, and make it easier to be clustered.

Though at the last step of spectral clustering, we applied the k-means method, but It becomes much more stable(center doesn't change much as we experimented).

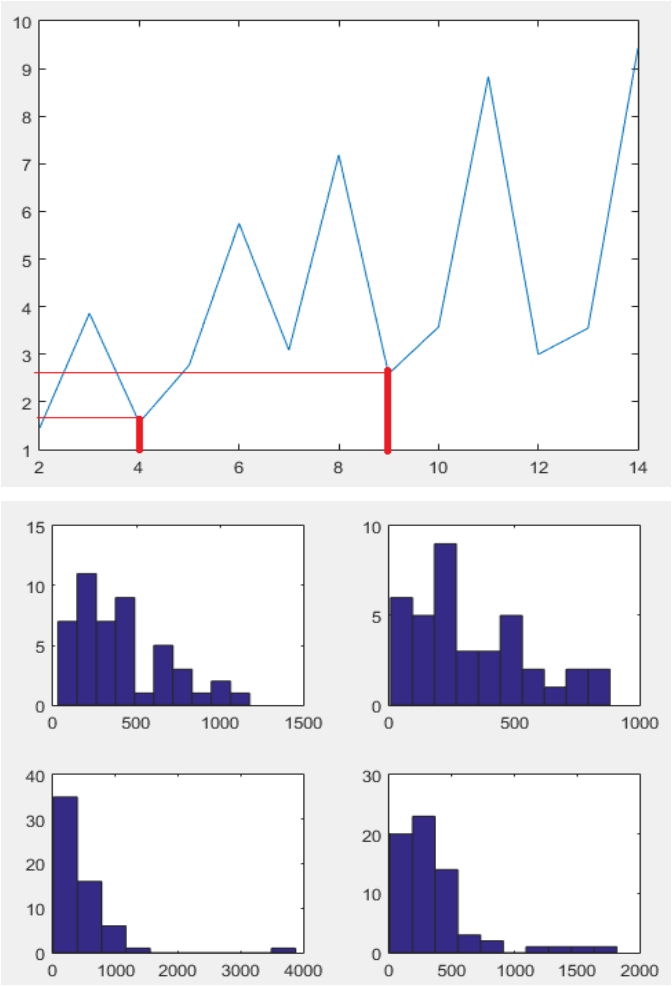
We choose k as follow:

$$sim\_inner = \sum_{i=1}^{K}\sum_{j=1}^{C^i} | x^{(j)} - \mu^i |$$

$$sim\_inter = \frac{1}{K^2}\sum_{i=1}^{K}\sum_{j=1}^{K} center^{(i)} * (center^{(j)})^T$$

$$J = sim\_inner - sim\_inter$$

The result as follows:



4. Discussion

Our analysis system provides a valid method to analyze data from different platforms with high dimensions. There are always conflicts between precision and computing resource restriction. In most of cases (except top layer of auto encoder), we choose parameter, such as hidden layer number, PCA dimension, and so on, dynamically. This makes the whole system more reliable.

However, when it goes to the last step: K-means algorithm, how to decide the number of clusters becomes a big problem. Here we draw K = 8 as a reasonable example to analyze this kind of problem.

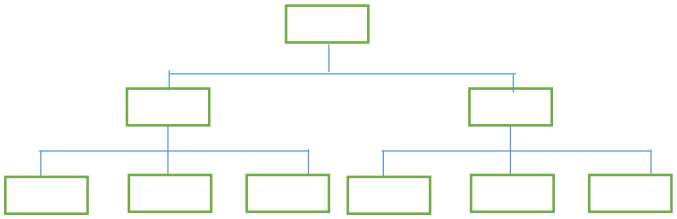


Figure 8. Decision Tree

It seems that all of the 8 group data has different distributions (shown in figure 8). To explain this kind of problem, we think we should regard our clustering procedures as a decision tree. Father node can be divided into several child nodes. That shows and explains why our group can be clustered into 4 or 8 groups

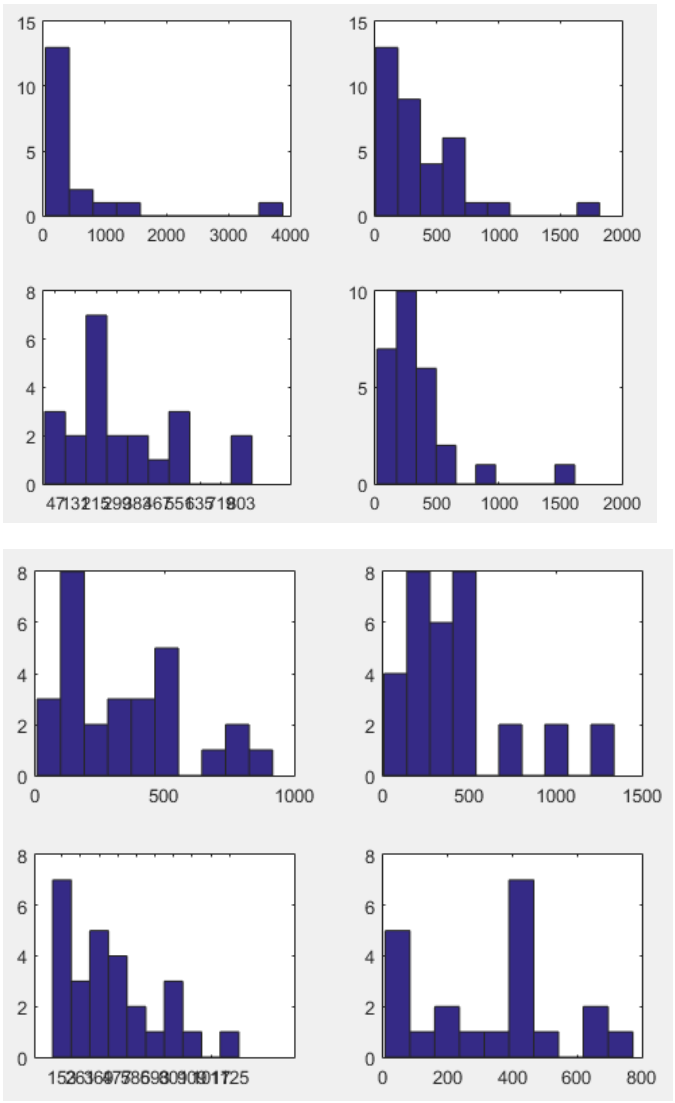


Figure 9. K-means clustering with 8 group

5. Reference

[1] Muxuan Liang, Zhizhong Li, Ting Chen and Jianyang Zeng," Integrative Data Analysis of Multi-platform Cancer Data with a Multimodal Deep Learning Approach"

[2] 2k-Gamer.neural network tool [EB/OL].

http://blog.csdn.net/evan123mg/article/details/39178485.
[3] Andrew Ng. Deep learning. [EB/OL] http://ufldl.stanford.edu/tutorial/selftaughtlearning/SelfTaughtLearnin/
[4] Kanungo, Tapas, et al. "An efficient k-means clustering algorithm: Analysis and implementation." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 2002:881--892.
[5] L Ma，Y Jiang，F Wang Multi-PCA Models for Process Monitoring and Fault Diagnosis. *Adchm Hong Kong*
[6] Lange, S., and M. Riedmiller. "Deep auto-encoder neural networks in reinforcement learning." *Neural Networks (IJCNN), The 2010 International Joint Conference on* IEEE, 2010:1 - 8.
[7] Ng, Andrew Y., M. I. Jordan, and Y. Weiss. "On Spectral Clustering: Analysis and an algorithm." *Advances in Neural Information Processing Systems* 2001:849--856.