Project 2

**Introduction:**

Clustering algorithms are unsupervised methods for finding groups of data points that

have similar representations in a proper space. K-means clustering is a simple and popular clustering algorithm. In this project, we need to:

1. To find proper representations of the data, s.t. the clustering is efficient and gives

out reasonable results.

2. To perform K-means clustering on the dataset, and evaluate the performance of the

clustering.

3. To try different preprocess methods which may increase the performance of the

clustering.

In order to define the clustering task, we pretend as if the class labels are not available and aim to find groupings of the documents. We then use class labels as the ground truth to evaluate the performance of the clustering task.  
 To get started with a simple clustering task, we take all the documents in the following classes: class 1(com) class 2(rec).

1. **Building the TF-IDF matrix**

We transform the documents into TF-IDF vectors using min\_df=3 and exclude the stopwords.

1. **2-class Clustering**

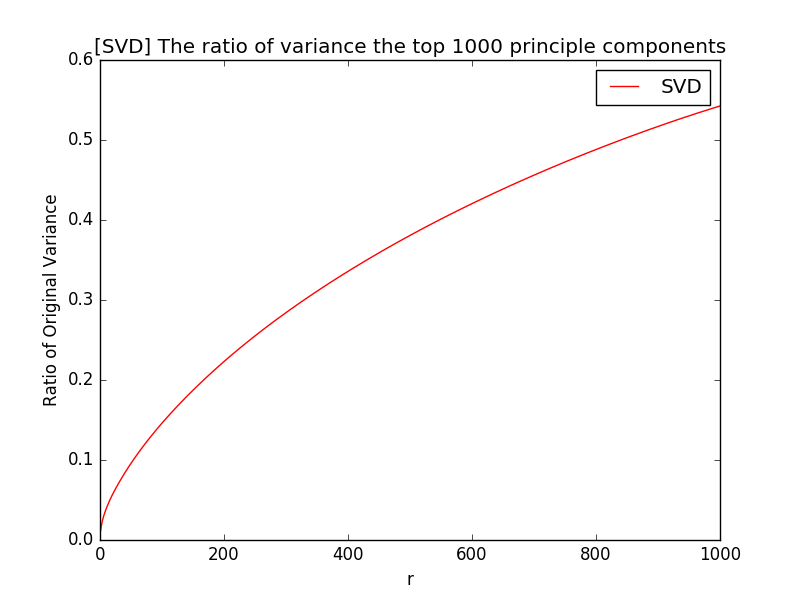
In this part, we apply K-means clustering to classify TF-IDF data into 2 classes. And then we examine the result with homogeneity score, completeness score, V-measure, adjusted Rand score and adjusted mutual info score.

1. **Preprocess the data**

For the high dimensional sparse TF-IDF vectors, they cannot yield a good result. Also, when the clusters are not round-shaped, K-means may fail to identify the clusters properly. Thus, we use the package in sklearn – Demonstration of k-means assumptions.

To reduce the dimension, we use NMF and LSI method to dimensionality reduction. Through SVD we calculate the variance remained after dimensionality reduction and sweep over parameters for each method, and choose one that yields better results in clustering purity metrics.

Firstly, we plot the ratio of variance of the original data retained after dimensionality reduction.



We can see that with the increase of the dimension, matrix will contain more information of the original TF-IDF matrix.

Then, we try r=1,2,3,5,10,20,50,100,300 to find the best one for LSI and NMF result

LSI:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **R** | **1** | **2** | **3** | **5** | **10** |
| Homogeneity | 0.0792 | 0.4105 | 0.4080 | 0.3892 | 0.4027 |
| Completeness | 0.0816 | 0.4474 | 0.4454 | 0.4395 | 0.4452 |
| V-measure | 0.0804 | 0.4282 | 0.4259 | 0.4128 | 0.4229 |
| Adjusted Rand | 0.1042 | 0.4219 | 0.4186 | 0.3705 | 0.4010 |
| Adjusted Mutual | 0.0791 | 0.4104 | 0.4080 | 0.3891 | 0.4026 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **R** | **20** | **50** | **100** | **300** |
| Homogeneity | 0.4099 | 0.4032 | 0.4051 | 0.3974 |
| Completeness | 0.4529 | 0.4477 | 0.4492 | 0.4430 |
| V-measure | 0.4303 | 0.4242 | 0.4260 | 0.4190 |
| Adjusted Rand | 0.4065 | 0.3969 | 0.3994 | 0.3892 |
| Adjusted Mutual | 0.4099 | 0.4031 | 0.4051 | 0.3974 |

matrix

NMF:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **R** | **1** | **2** | **3** | **5** | **10** |
| Homogeneity | 0.0792 | 0.4418 | 0.0405 | 0.2721 | 0.4145 |
| Completeness | 0.0816 | 0.44524 | 0.1382 | 0.3213 | 0.4375 |
| V-measure | 0.0804 | 0.4470 | 0.0627 | 0.2947 | 0.4257 |
| Adjusted Rand | 0.1042 | 0.5116 | 0.0107 | 0.2600 | 0.4600 |
| Adjusted Mutual | 0.0791 | 0.4418 | 0.0404 | 0.2721 | 0.4144 |

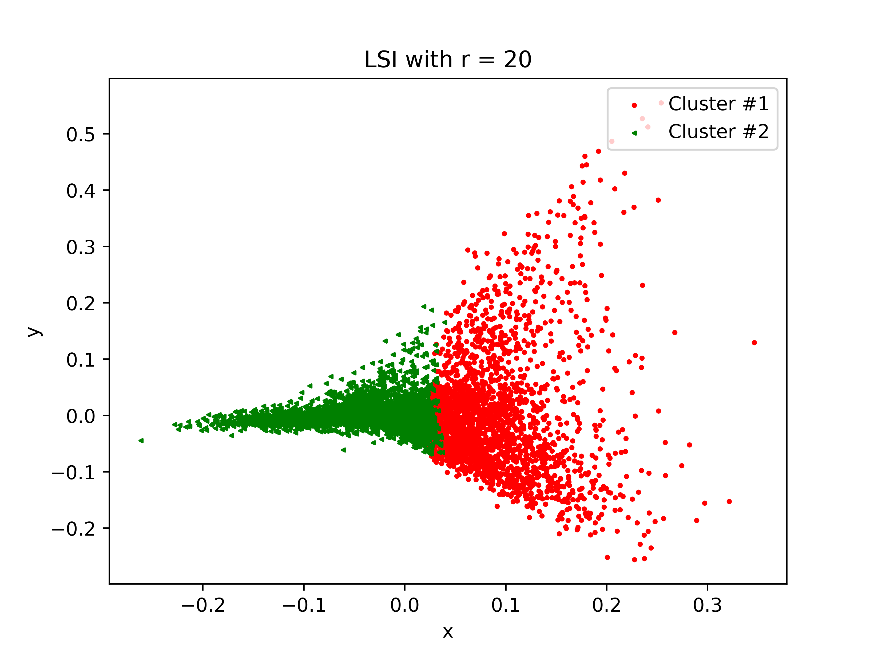
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **R** | **20** | **50** | **100** | **300** |
| Homogeneity | 0.3822 | 0.3499 | 0.0688 | 0.0000 |
| Completeness | 0.4238 | 0.4074 | 0.1777 | 0.0000 |
| V-measure | 0.4019 | 0.3765 | 0.0992 | 0.0000 |
| Adjusted Rand | 0.3842 | 0.3204 | 0.0185 | 0.0000 |
| Adjusted Mutual | 0.3821 | 0.3499 | 0.0687 | 0.0001 |

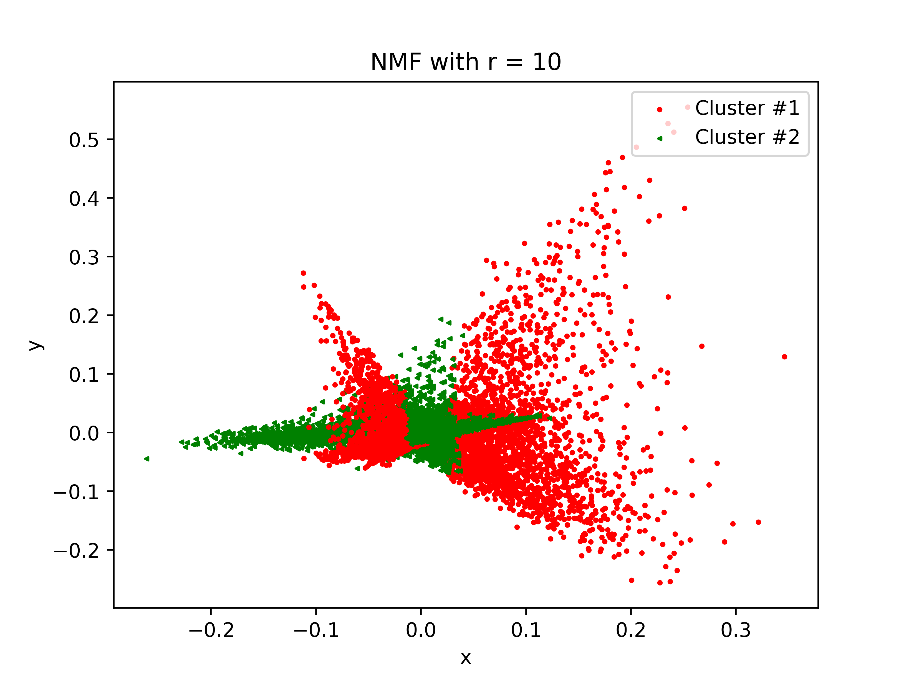
From the result we could see that r=20 and r=10 are the best result for LSI and NMF. With the increase of r (more dimension), the important values are included in the new matrix, but when it goes through the threshold, the more dimension will have little effect on result. That is because the dimension we added are not so important in TF-IDF matrix, thus the result shows non-monotonic behavior of the measures as r increases.

1. **Normalization & Non-linear Transform**

First, we visualize the performance of the case with the best clustering result. And then, based on the best r we got, we used 3 methods to see whether they increase the clustering performance. Firstly, we use normalization and then non-linear transformation and the combination of both.

We plot the best clustering result in previous part by projecting final data vectors onto 2-dimensional plane and color- coding the classes.





Through the plot, we can see that 2 clusters are overlapping. Thus, we need to try some method to scatter them.

1. LSI with normalization

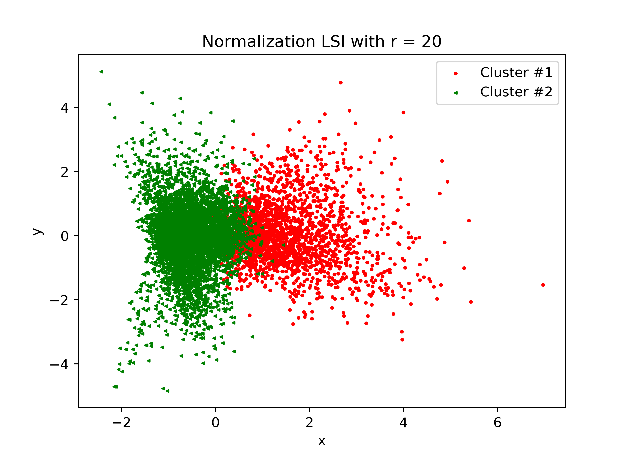
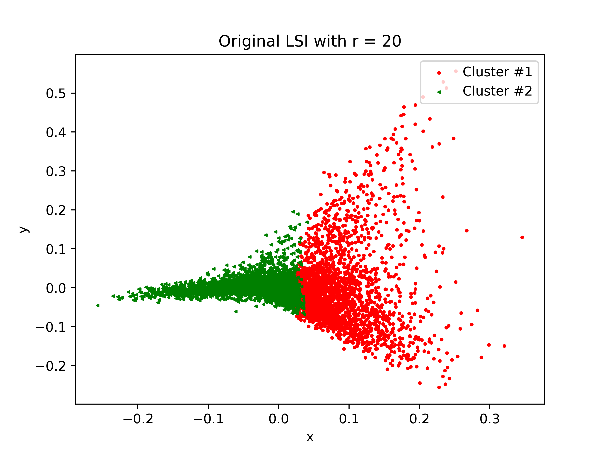
Table 1 LSI with r = 20

|  |  |  |
| --- | --- | --- |
|  | **Origin** | **Normalization** |
| Homogeneity | 0.4025 | 0.3483 |
| Completeness | 0.4480 | 0.4069 |
| V-measure | 0.4240 | 0.3754 |
| Adjusted Rand | 0.3937 | 0.3167 |
| Adjusted Mutual | 0.4024 | 0.3483 |

Contingency matrix:

Table 2 Contingency matrix

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Origin** |  | **Normalization** |  |
|  | cluster\_0 | cluster\_1 | cluster\_0 | cluster\_1 |
| class\_0 | 2457 | 1446 | 2195 | 1708 |
| class\_1 | 22 | 3957 | 16 | 3964 |



Although, normalizing on LSI nearly has no impact on purity, it scatters 2 clusters a lot.

1. NMF with logarithm and normalizing:

Table 3 NMF with logarithm and normalizing

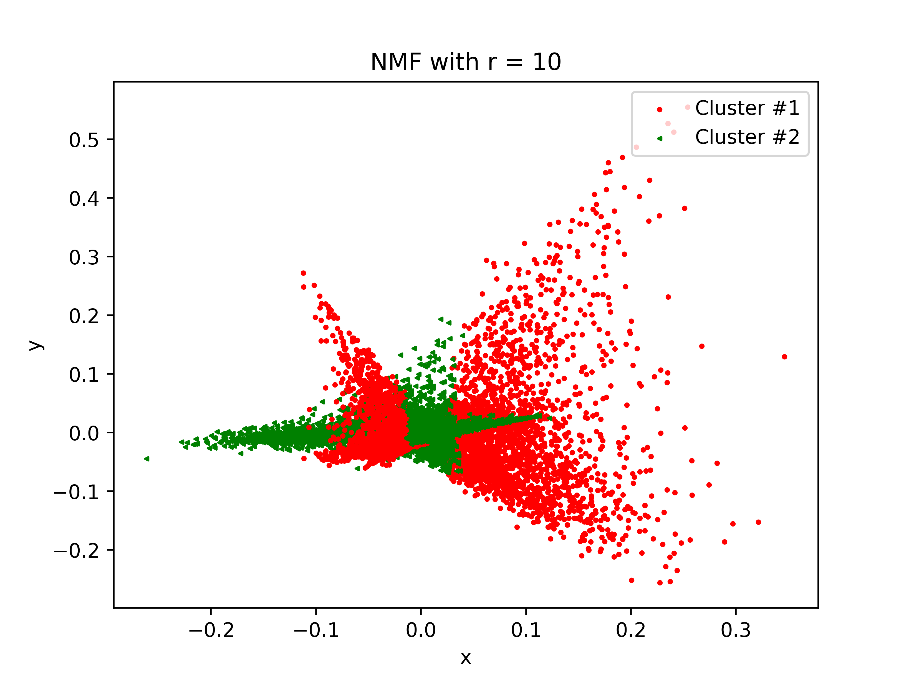
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Origin** | **Normalization** | **Logarithm** | **L+N** | **N+L** |
| Homogeneity | 0.4145 | 0.4740 | 0.1871 | 0.2043 | 0.0818 |
| Completeness | 0.4375 | 0.4943 | 0.1899 | 0.2054 | 0.0833 |
| V-measure | 0.4257 | 0.4840 | 0.1885 | 0.2048 | 0.0825 |
| Adjusted Rand | 0.4600 | 0.5259 | 0.2397 | 0.2658 | 0.1072 |
| Adjusted Mutual | 0.4144 | 0.4740 | 0.1870 | 0.2042 | 0.0817 |

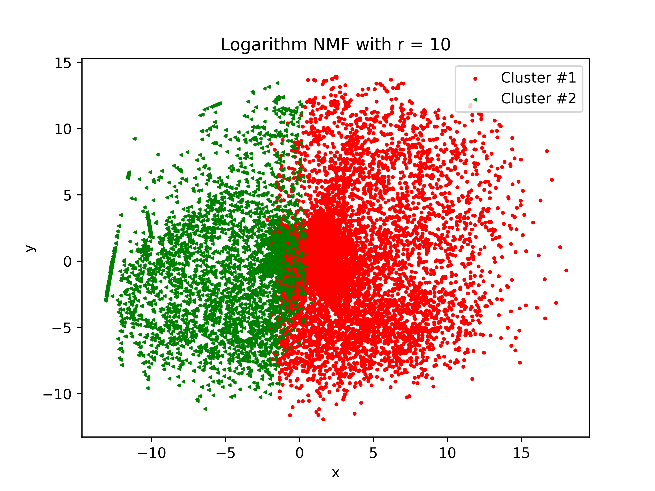
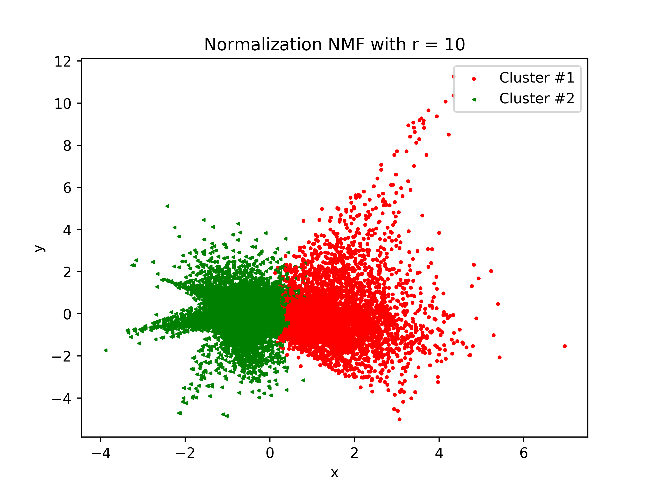
Contingency matrix:

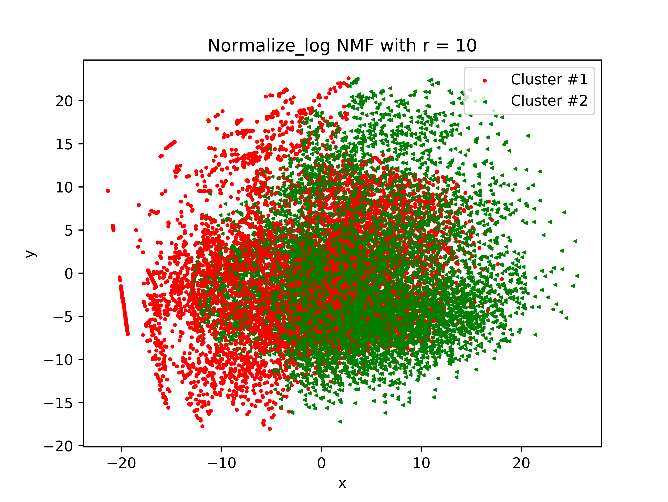
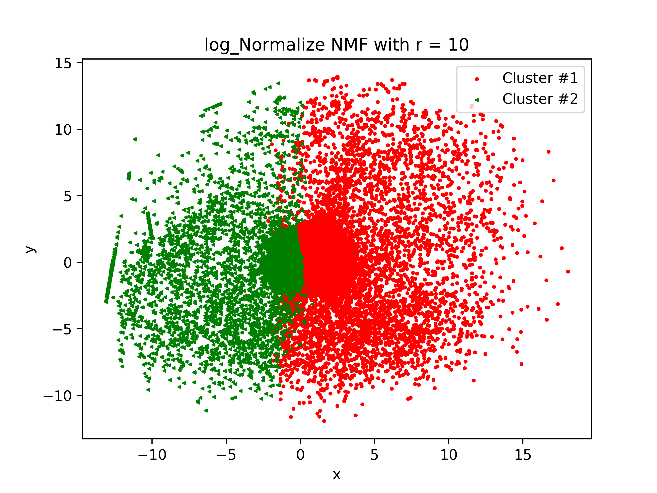
Table 4 Contingency matrix

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Origin** | | **Normalization** | |
|  | cluster\_0 | cluster\_1 | cluster\_0 | cluster\_1 |
| class\_0 | 2785 | 1145 | 2912 | 991 |
| class\_1 | 123 | 3856 | 92 | 3887 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Logarithm** | | **L+N** | | **N+L** | |
|  | cluster\_0 | cluster\_1 | cluster\_0 | cluster\_1 | cluster\_0 | cluster\_1 |
| class\_0 | 3203 | 700 | 3140 | 763 | 1001 | 2902 |
| class\_1 | 1331 | 2668 | 1146 | 2833 | 2330 | 1649 |







From the result above, with normalization on NMF, the purity increased a lot. With logarithm, the 2 clusters could scatter a lot.

With the logarithm transformation, it could reduce the range of data and make the variance of 2 clusters more balanced, which is better for k-means algorithm.

1. **Multi- class Clustering**

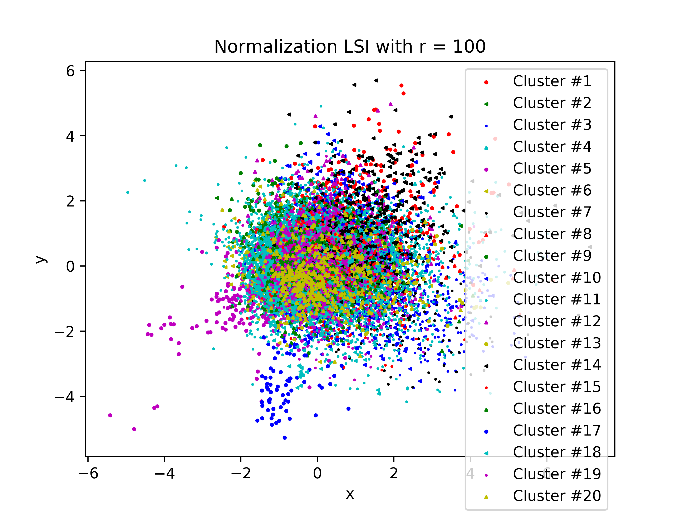
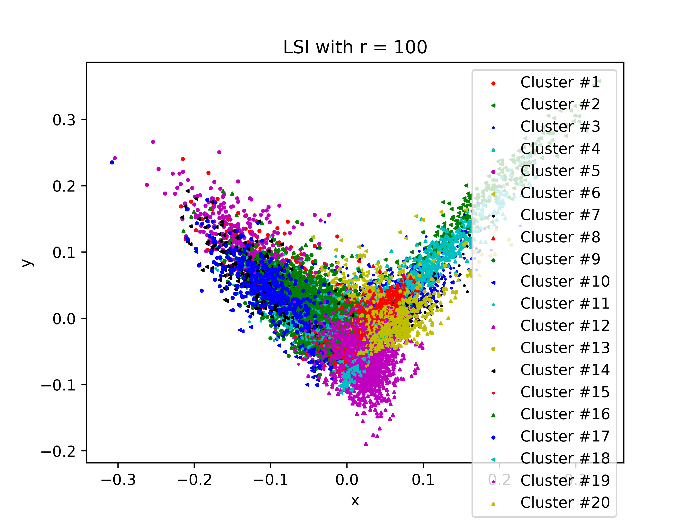
In this part, we include all the documents and the corresponding terms in the data matrix and find proper representation through dimensionality reduction of the TF-IDF representation. We try different dimensionality reduction techniques and transformations.

Firstly, we exam the best r when k-cluster is 20. We found that r=20,10 is not the best parameter when we need to do the high-level clustering. We found the best for LSI and NMF is the same parameter—r =100. (we record the result in task 5 a.txt which will be uploaded)

Secondly, we plot the origin clustering result and new clustering with different transformations and output the contingency matrix.

**R=100**

1. LSI with normalization

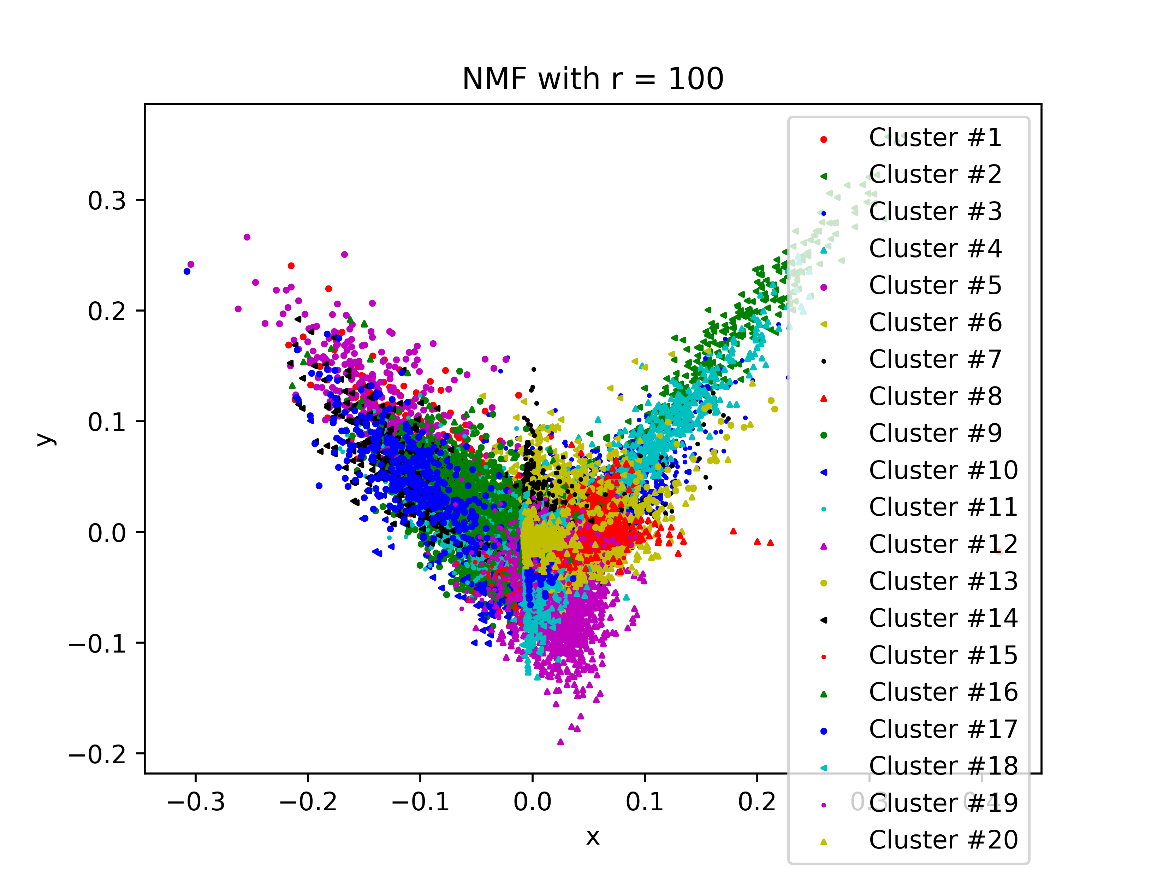
Table 5 LSI with normalization

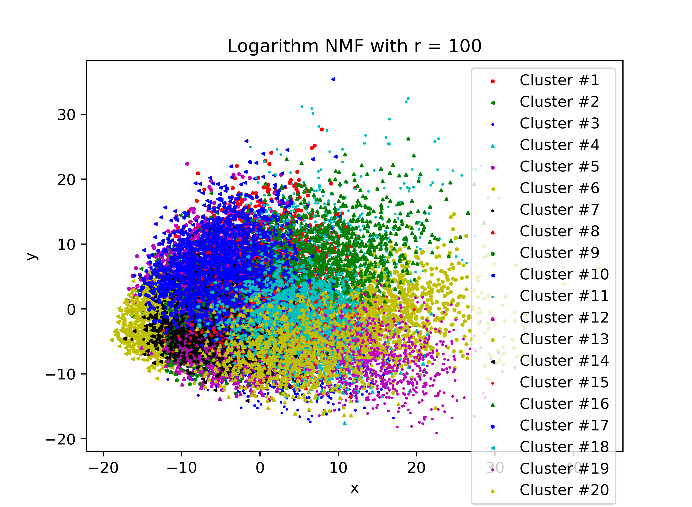
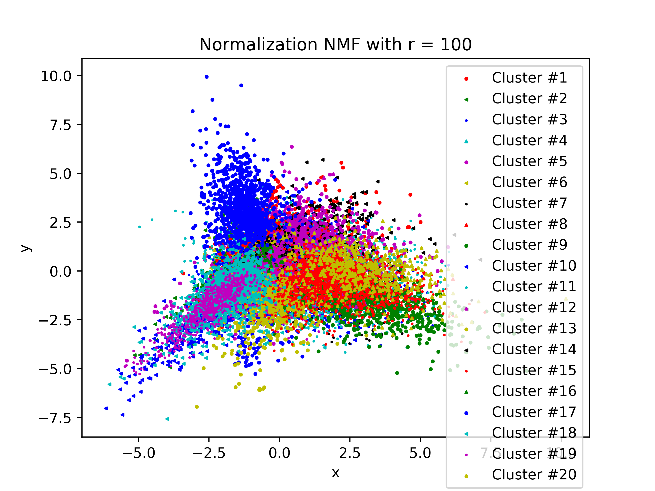
|  |  |  |
| --- | --- | --- |
|  | **Origin** | **Normalization** |
| Homogeneity | 0.2871 | 0.243 |
| Completeness | 0.3915 | 0.3979 |
| V-measure | 0.3313 | 0.3018 |
| Adjusted Rand | 0.0595 | 0.0276 |
| Adjusted Mutual | 0.2848 | 0.2406 |

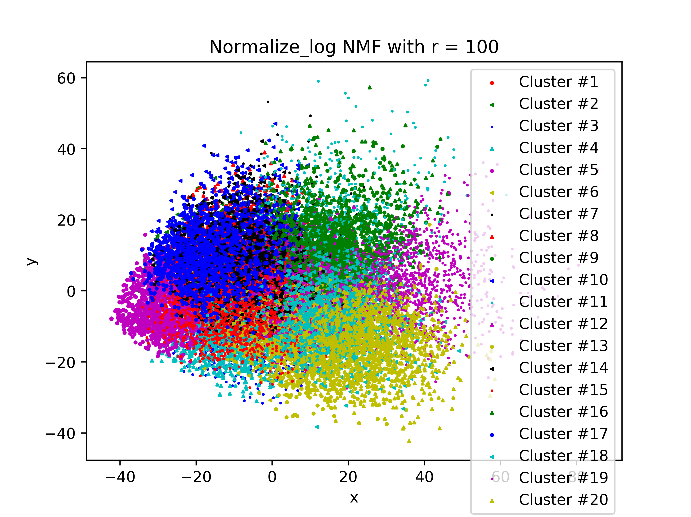
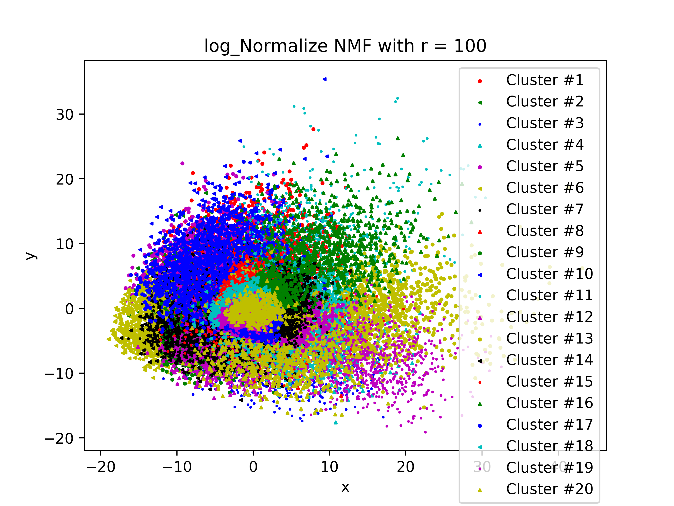
Contingency matrix:

Although, normalizing on LSI nearly has no impact on purity, it scatters 20 clusters a lot.

1. NMF with logarithm and normalizing







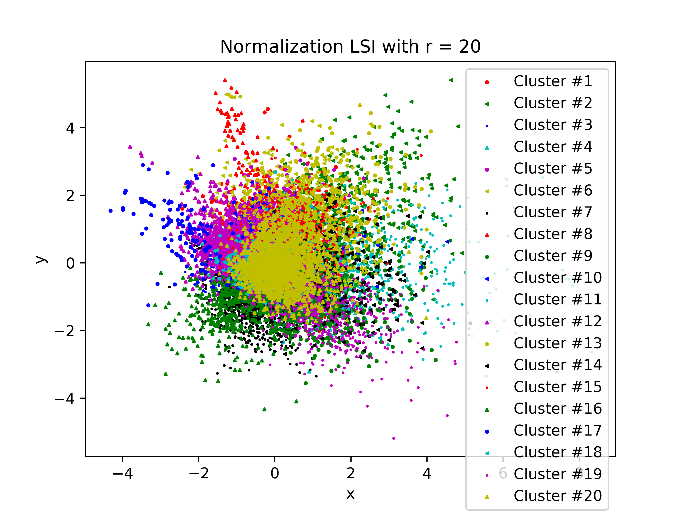
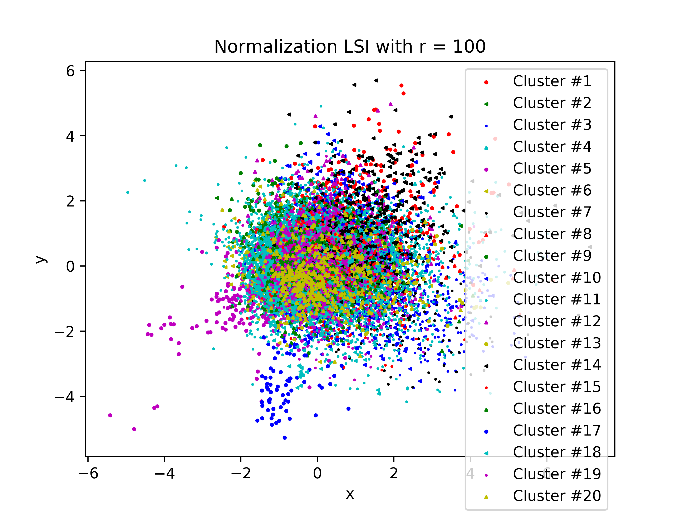
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Origin** | **Normalization** | **Logarithm** | **L+N** | **N+L** |
| Homogeneity | 0.0792 | 0.2892 | 0.2146 | 0.2703 | 0.1672 |
| Completeness | 0.1362 | 0.4236 | 0.2169 | 0.2751 | 0.169 |
| V-measure | 0.1001 | 0.3437 | 0.2157 | 0.2727 | 0.1681 |
| Adjusted Rand | 0.007 | 0.0599 | 0.0978 | 0.1346 | 0.0718 |
| Adjusted Mutual | 0.0761 | 0.2868 | 0.2121 | 0.2679 | 0.1645 |

Contingency matrix:

From the result above, with normalization and logarithm on NMF, the purity increased a lot. With logarithm, the 20 clusters could scatter a lot.

**R=20,10**

1. LSI with normalization

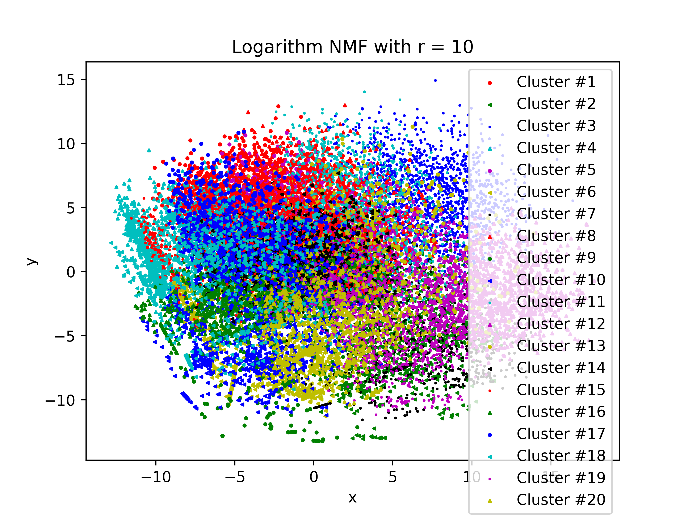
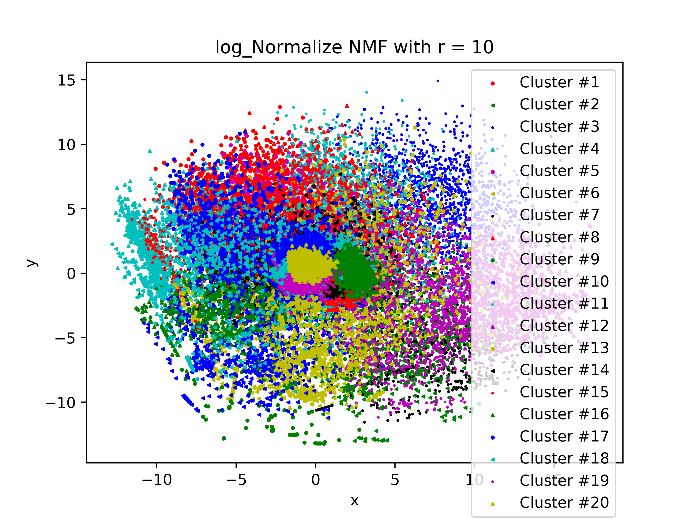


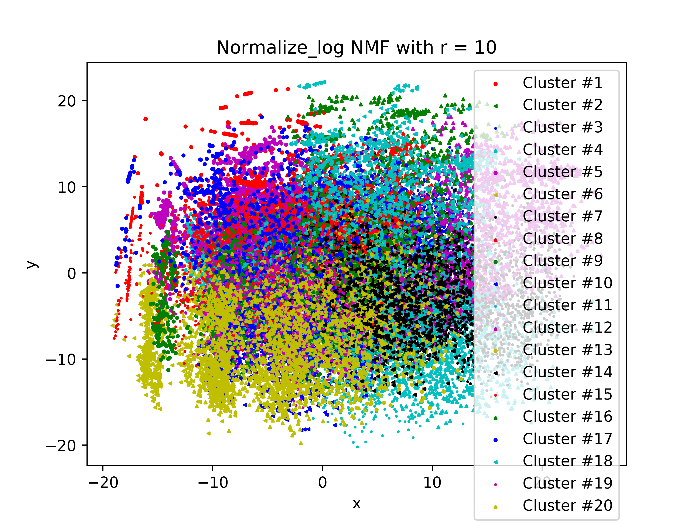
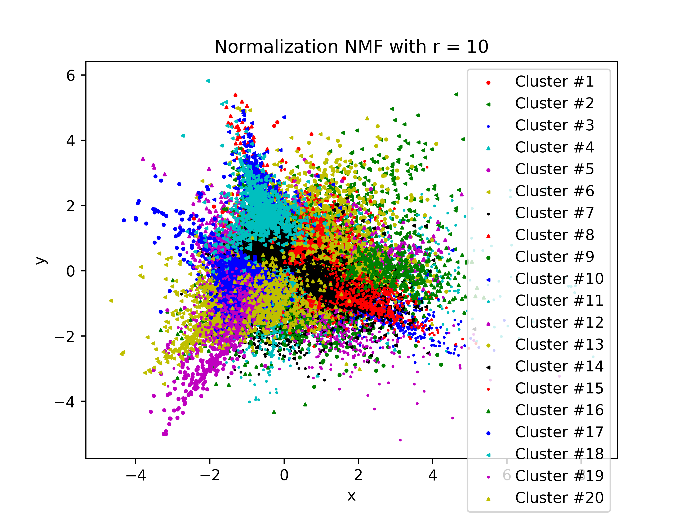
|  |  |  |
| --- | --- | --- |
|  | **Norm 20** | **Norm 100** |
| Homogeneity | 0.2534 | 0.243 |
| Completeness | 0.3082 | 0.3979 |
| V-measure | 0.2781 | 0.3018 |
| Adjusted Rand | 0.0629 | 0.0276 |
| Adjusted Mutual | 0.251 | 0.2406 |

Contingency matrix:

From the result, we could see that when r=100 it’s result is better than r=20 and in the plot parameter 100 scatters 20 clusters a lot.

1. NMF with logarithm and normalizing



Table 6 r=100

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Origin** | **Normalization** | **Logarithm** | **L+N** | **N+L** |
| Homogeneity | 0.0792 | 0.2892 | 0.2146 | 0.2703 | 0.1672 |
| Completeness | 0.1362 | 0.4236 | 0.2169 | 0.2751 | 0.169 |
| V-measure | 0.1001 | 0.3437 | 0.2157 | 0.2727 | 0.1681 |
| Adjusted Rand | 0.007 | 0.0599 | 0.0978 | 0.1346 | 0.0718 |
| Adjusted Mutual | 0.0761 | 0.2868 | 0.2121 | 0.2679 | 0.1645 |

Table 7 r=10

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Origin** | **Normalization** | **Logarithm** | **L+N** | **N+L** |
| Homogeneity | 0.0792 | 0.2643 | 0.1804 | 0.1966 | 0.1362 |
| Completeness | 0.1362 | 0.3097 | 0.1817 | 0.1977 | 0.1369 |
| V-measure | 0.1001 | 0.2852 | 0.1810 | 0.1971 | 0.1365 |
| Adjusted Rand | 0.007 | 0.0815 | 0.0703 | 0.0791 | 0.0458 |
| Adjusted Mutual | 0.0761 | 0.2619 | 0.1778 | 0.1940 | 0.1334 |

Contingency matrix:

From the result above, we can see the parameter 100 is better than 10 for all the transformations and the purity increased a lot within each parameter when applying transformations. That may because when the dimension of cluster increases, it need more row/column’s information to make a better clustering. Thus, r=100 is better than r=10,20.