**Large-Scale Data Mining: Models and Algorithms** **ECE 2019 Spring 2018**
Prof Vwani Roychowdhury TA: Tianyi Wang
UCLA, Department of ECE Behnam Shahbazi

# Project 2: Clustering

Due Monday Apr. 30, 2018 by 11:59 pm

## Team Members

Jennifer MacDonald, UID: 604501712
Nguyen Nguyen, UID: 004870721
Sam Yang, UID: 604034791

## Introduction

In this project, we explore unsupervised machine learning algorithms. Unsupervised learning is a machine learning task that infers hidden structure from "unlabeled" data. It is useful when we do not know much about the features of the training dataset, or we want to see additional hidden features that a training dataset does not include. There are multiple approaches to unsupervised learning such as clustering, neural networks, etc. However we will only focus on clustering algorithms, particularly k-means clustering.

Clustering algorithms are unsupervised learning that can also be used to find groups of data points that have similar representations in the same feature space. Clustering is different from classification (supervised learning) in that prior labeling or grouping of data points is not available. Clustering algorithms can also be used to group data together when groups aren't already predetermined in the training dataset. To name a few, some clustering algorithms include K-means, kNN (K-nearest neighbour), DBSCAN, Birch, Agglomerative clustering, etc. In this project, we will focus on K-means algorithm.

The goals for this project is to find good representations of the data, demonstrate that the clustering algorithm is both efficient and yields accurate results, and try a variety of preprocessing methods that could increase the efficiency of the clustering algorithm.

## K-means Algorithm

K-means algorithm clustering is a simple and popular clustering algorithm that clusters data by trying to separate samples into clusters. Each data point can only be in one cluster and the sum of the squares of the distances between each data point and the "centroid" of the cluster it belongs

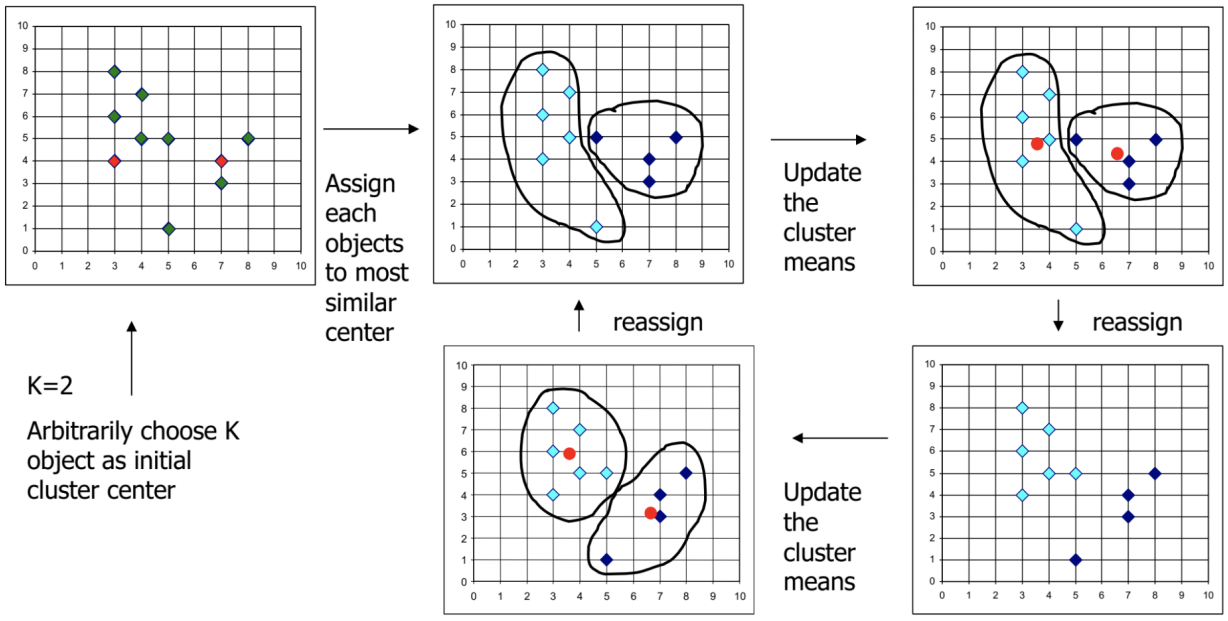
to is minimized. For example, given a number of clusters *k*, each data point is sorted into one of *k* clusters, where $\mu_k$ is the center of cluster *k*. Then the goal is to find $r_{nk}$'s and k's such that J is minimized as seen below.

$$r_{nk} = \begin{cases} 1, & \text{if } \mathbf{x}_n \text{ is assigned to cluster } k \\ 0, & \text{otherwise} \end{cases}, \quad n = 1, \ldots, N \quad k = 1, \ldots, K$$

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2.$$

The K-means clustering algorithm tests each datapoint, and finds the cluster whose center is closest, and then recalculates the new cluster centers with the additional datapoint. The pseudo-code of k-means algorithm and the diagram can be seen below.

1. Arbitrarily choose k objects as the initial cluster centers
2. Until no change, do
    a. (Re)assign each object to the cluster to which the object is the most similar, based on the mean value of the objects in the cluster
    b. Update the cluster means by calculating the mean value of the objects for each cluster




K-means is relatively efficient in large dataset such that big O(t*k*n) where n is number of objects, k is number of clusters, and t is number of iterations. It often terminates at a local optimum. However, k-means does not support categorical data, and it implicitly assumes that the clusters are isotropically shapes (round shapes). Therefore, it may fail to cluster properly in abstract shapes. Even when the shapes are round, k-Means algorithm may fail when the clusters

have unequal variances. It also requires the user to specify number of clusters, and unsuitable to discover any non-convex clusters. K-means is also unable to handle noisy data and outliers.

## Dataset

We used the same dataset as project 1, the "20 Newsgroups" dataset. It contains about 20,000 newsgroup posts split evenly among 20 topics, and individually is split into training and testing sets.

Table 1: Two well-separated classes

| | | | | |
|---|---|---|---|---|
| Class 1 | comp.graphics | comp.os.ms-windows.misc | comp.sys.ibm.pc.hardware | comp.sys.mac.hardware |
| Class 2 | rec.autos | rec.motorcycles | rec.sport.baseball | rec.sport.hockey |

However, in order to use the clustering algorithm, we are going to ignore the labels and instead group the data points together based on spatial proximity. The class labels can be then used as the ground truth to evaluate how well the clustering algorithm performed.

We load the data in a similar manner to project 1 to get started:

```
categories = ['comp.sys.ibm.pc.hardware', 'comp.graphics',
              'comp.sys.mac.hardware', 'comp.os.ms-windows.misc',
              'rec.autos', 'rec.motorcycles',
              'rec.sport.baseball', 'rec.sport.hockey']

dataset = fetch_20newsgroups(subset='all', categories=categories,
                             shuffle=True, random_state=42)
```

## Problem Statement

*Question 1: Report the dimensions of the TF-IDF matrix you get.*

Similar to project 1 and using the same 8 categories in the dataset, we shuffled the data and set *random_state=42*. We also removed headers and footers since we did not find much useful information in them. We then used CountVectorizer (as discussed previously in project 1) to vectorize terms into "Bag of Words" representation. Again, we set *min_df=3* (minimum degrees of freedom) and *stop_words='english'* to remove any english stop words. However, we did not do any stemming since it was not required in this project.We then used TfidfTransformer to transform into TF-IDF matrix using sklearn library. The dimensions we extracted came out to be:

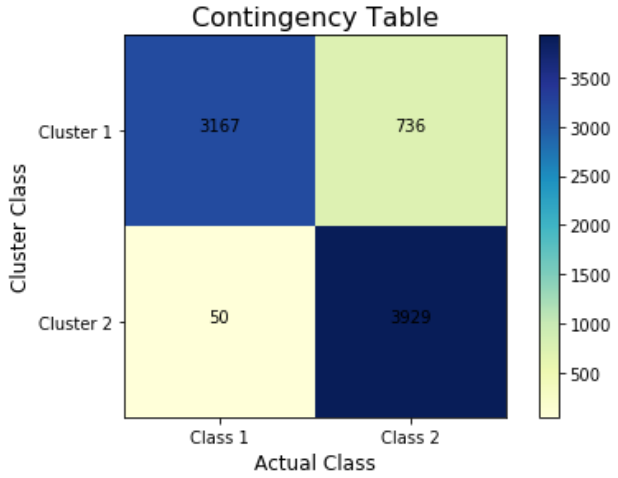
X_tfidf: (7882, 23522)

*Question 2: Report the contingency table of your clustering result.*

In our clustering process, we wanted to cluster only 2 classes using K-means clustering algorithm (as previously discussed above) by setting *k=2* and using the TF-IDF matrix we transformed above. We also set *random_state=0* and *max_iter* to at least 1000 and *n_init* to at least 30. *Max_iter* parameter is the maximum number of iterations of k-Means algorithm in a single run. Therefore, larger max_iter provides better accuracy but at a cost of computational power. *N_init* is the number of time the k-means algorithm will be running with different centroid seeds. The final results will be the best output consecutive runs. Higher *n_init* values also provide better accuracy but at a computational cost.

In project 1, we used confusion matrix to provide a visualization of our classification results. Since we do not have a test dataset, to visualize clustering results, we used contingency table as seen in Figure 1 below. The difference between two tables is that confusion matrix is used to evaluate which data points are correctly classified and which ones are misclassed by comparing "actual classes" against "predicted classes". In contingency table, each element $A_{ij}$ is the number of data points that are members of class $c_i$ and elements of cluster $k_j$. Class $c_i$ can be viewed as "actual classes" and cluster $k_j$ can be viewed as "predicted classes" or "cluster classes".

Figure 1: Contingency Table for k-Means with TF-IDF




*Question 3: Report the 5 measures above for the K-means clustering results you get.*

In addition to the contingency table above, we had to find a way to quantify the clustering results. There are various measurements that we can perform on the data points with respect to the ground truth such as *homogeneity score*, *completeness score, V-measure, adjusted Rand score*, and *adjusted mutual info score;* all scores return a value between 0.0 and 1.0.

- A clustering result satisfies **homogeneity** if all of its clusters contain only data points which are members of a single class.
- A clustering result satisfies **completeness** if all the data points that are members of a given class are elements of the same cluster.

$$h = 1 - \frac{H(C|K)}{H(C)}$$

$$c = 1 - \frac{H(K|C)}{H(K)}$$

where $H(C|K)$ is the **conditional entropy of the classes given the cluster assignments** and is given by:

$$H(C|K) = -\sum_{c=1}^{|C|}\sum_{k=1}^{|K|}\frac{n_{c,k}}{n}\cdot\log\left(\frac{n_{c,k}}{n_k}\right)$$

and $H(C)$ is the **entropy of the classes** and is given by:

$$H(C) = -\sum_{c=1}^{|C|}\frac{n_c}{n}\cdot\log\left(\frac{n_c}{n}\right)$$

with $n$ the total number of samples, $n_c$ and $n_k$ the number of samples respectively belonging to class $c$ and cluster $k$, and finally $n_{c,k}$ the number of samples from class $c$ assigned to cluster $k$.

- **V-measure** is the harmonic mean between homogeneity, and completeness.

```
v = 2 * (homogeneity * completeness) / (homogeneity + completeness)
```

- **Adjusted Rand Index** is the score of Rand Index which computes a similarity measurement between two clusterings by considering all pairs of samples and conuning pairs that are assigned in the same or different clusters.

```
ARI = (RI - Expected_RI) / (max(RI) - Expected_RI)
```

- **Adjusted Mutual Rand Information** is an adjustment of the Mutual Information (MI) score to account for chance. It accounts for MI which is generally higher for two clusterings with a large number of clusters, regardless of whether there is actually more information shared.

```
AMI(U, V) = [MI(U, V) - E(MI(U, V))] / [max(H(U), H(V)) - E(MI(U, V))]
```

Table 1 below is the summary of measurements we have found for the same dataset as contingency table above. The scores range from 0.5805 to 0.6408. A homogeneity score of 0.5805 is saying that our cluster is only half "pure" since it contains half of data points from each class. A completeness score of 0.5951 is telling that our clustering result contains equal amount of data points that are assigned to two different clusters. Since our homogeneity score and completeness score are around 0.5, it's expected that our V-measure score is 0.5877. Therefore, our adjusted rand index, similar to accuracy measure that computes similarity between clustering labels and ground truth labels, is only 0.6408. Our last adjusted mutual information score is only at 0.5805.

Table 1: Measures for K-means Clustering

| Measure | Score |
|---|---|

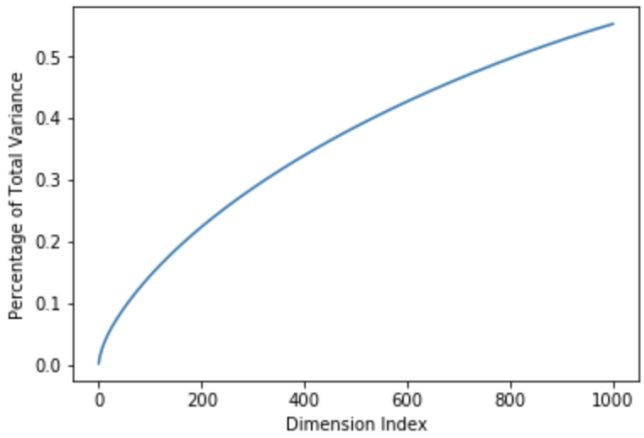
| | |
|---|---|
| Homogeneity Score | 0.5805 |
| Completeness Score | 0.5951 |
| V-Measure Score | 0.5877 |
| Adjusted Rand Index | 0.6408 |
| Adjusted Mutual Information | 0.5805 |

*Question 4: Report the plot of the percent of variance the top r principal components can retain v.s. r, for r = 1 to 1000.*

In our previous analysis, we ran our k-Means algorithm through TF-IDF matrix. However, because TF-IDF is a high dimensional sparse matrix, it does not yield good clustering result. In a high-dimensional space, Euclidean distance is not a good metric since distances between data points are almost the same. As we discussed above, k-Means algorithm does not work well with non-rounded shapes, or unequal variances. Therefore, for the following problems below, we used dimension reduction methods such as Latent Semantic Index (LSI) and Non-Negative Matrix Factorization (NMF) to reduce our TF-IDF matrix.

First off, we observed the percent of variance from r = 1 to 1000 in LSI method and found that as r increases, the percentage of total variance increases monotonically which can be seen in Figure 2 below.

Figure 2: Plot of Percent of Variance Top R Principal Components Retains



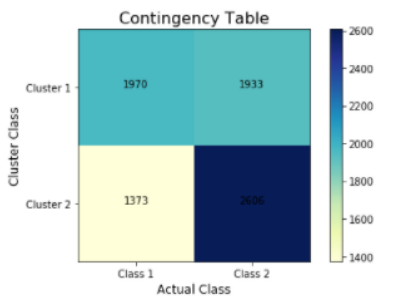
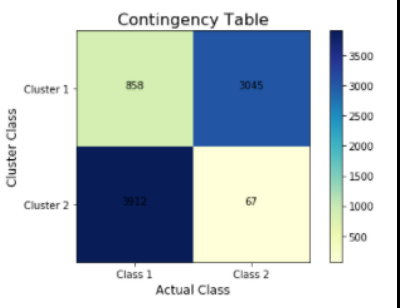
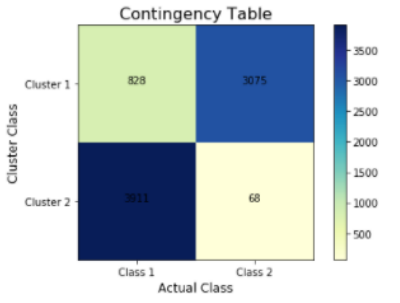

*Question 5: Let r be the dimension that we want to reduce the data to (i.e. n components). Try r = 1,2,3,5,10,20,50,100,300, and plot the 5 measure scores v.s. r for both SVD and NMF. Report the best r choice for SVD and NMF respectively.*
*Note: what is "best" after all? What if some measures contradict with each other? Here you are faced with this challenge that you need to decide which measure you value the most, and design your own standard of "best".* ***Please explain your standard and justify it****.*

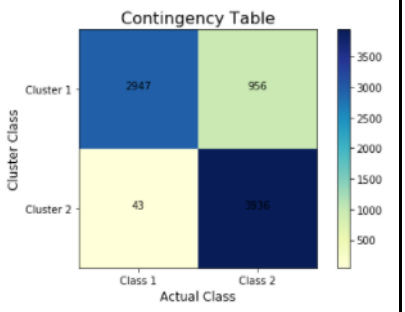
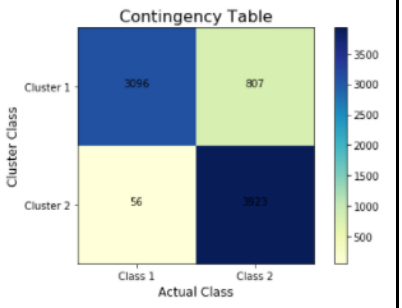
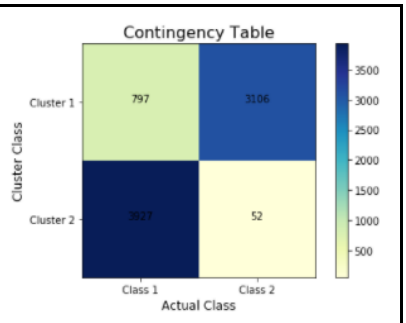
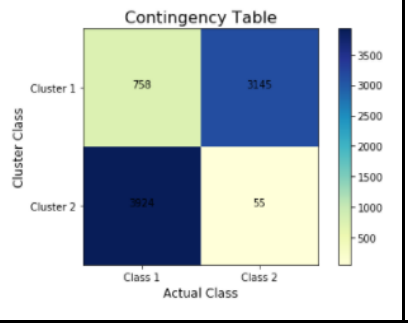
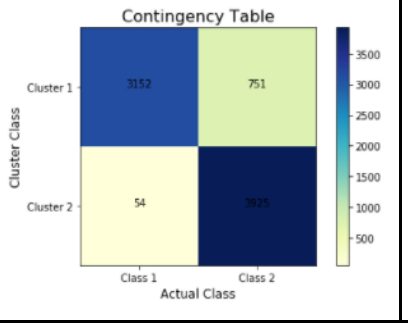
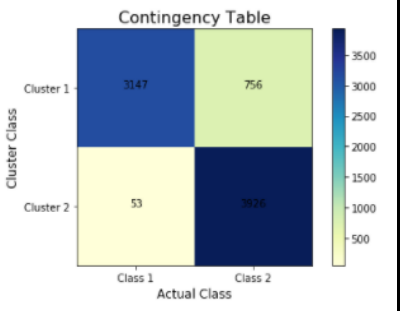
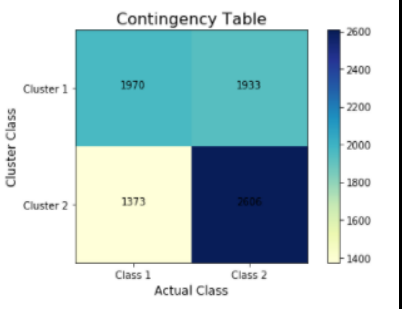
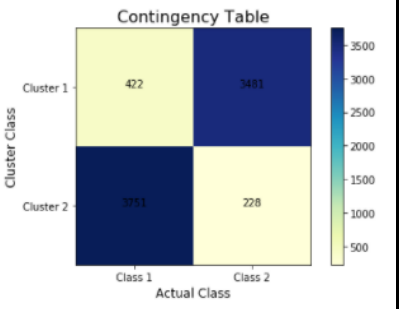
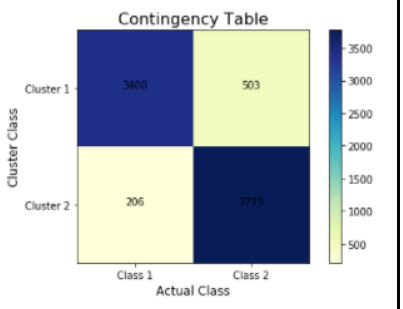
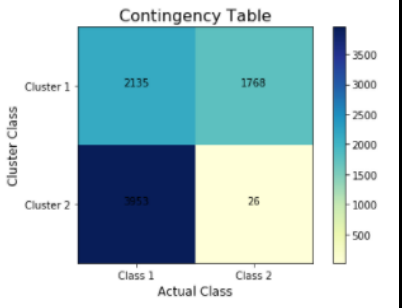
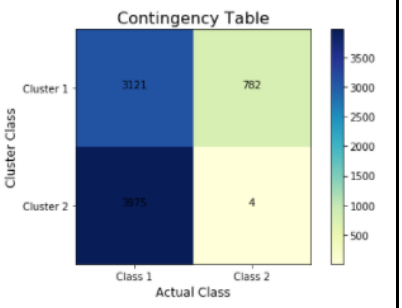
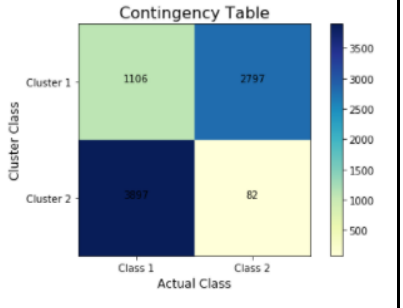
Since we found that percentage of total variance increases monotonically as number of r components increases, we wanted to find if the measurements would also increase monotonically also. We tried a range of r principal components: 1, 2, 3, 5, 10, 20, 50, 100, 300 and plotted 5 measurement scores for both SVD (note that SVD is  LSI) and NMF as seen in Table 2 and 3 below. By observing the 5 measurement scores in SVD, it is relatively consistent across all measurements. The first r principal component of 1 has the lowest values while r principal component of 100 has the highest. However, there is a slight drop in r principal component of 5. On the other hand, the 5 measurement scores in NMF are not consistent at all. The lowest measurement values have r principal component is at 100, while the highest values have r principal component of 2.

The best r choice for SVD and NMF are 100 and 2 respectively. We looked at all 5 measurements and they consistently showed highest values when r=100 in SVD, and r=2 in NMF.

All five measurements for both SVD and NMF consistently had the highest score for the same r-value, so we did not need to pick a "best" measurement to use for these cases. However, if some measurements contradicted each other it seems that V-measure score would be a the first score to rely on as the "best" since it's a harmonic mean between homogeneity, and completeness. The second score used to evaluate if the measurements contradicted each other would be the adjusted rand index since it is similar to accuracy score in classification.

Table 2: SVD and NMF Contingency Tables for Varying r Principal Components

| SVD | | |
|---|---|---|
| R = 1 | R = 2 | R = 3 |
|  |  |  |
| R = 5 | R = 10 | R = 20 |

| | | |
|---|---|---|
|  |  |  |
| R = 50 | R = 100 | R = 300 |
|  |  |  |
| NMF | | |
| R = 1 | R = 2 | R = 3 |
|  |  |  |
| R = 5 | R = 10 | R = 20 |
|  |  |  |
| R = 50 | R = 100 | R = 300 |

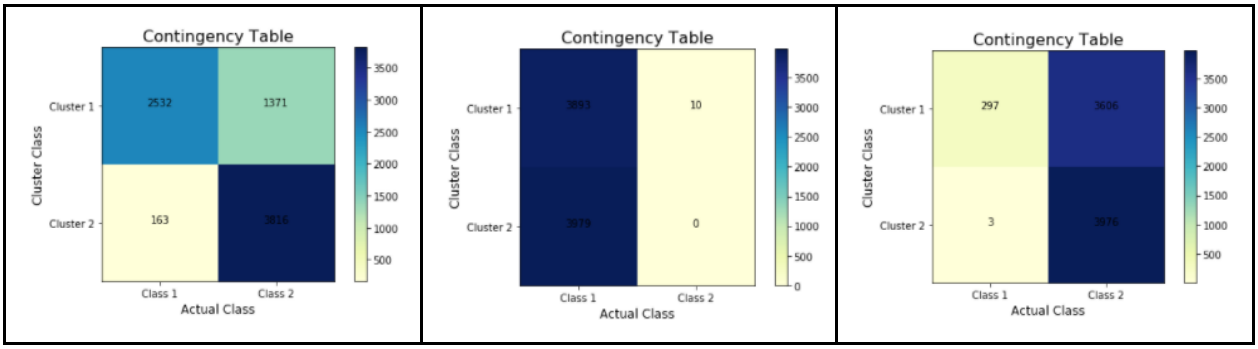
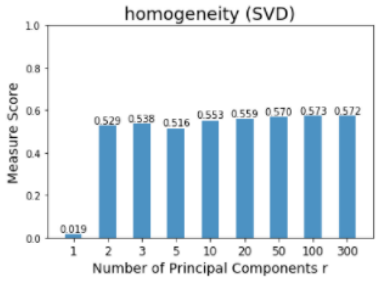
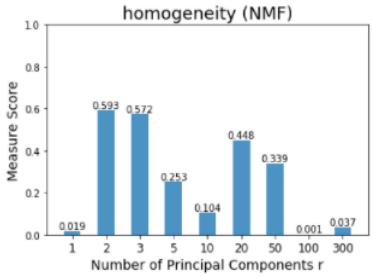
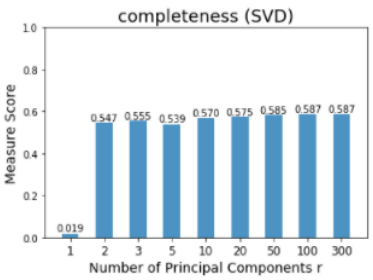
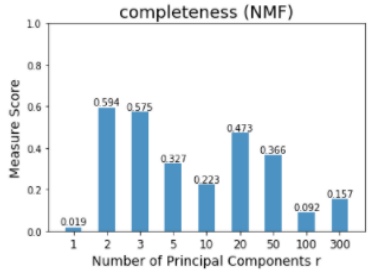
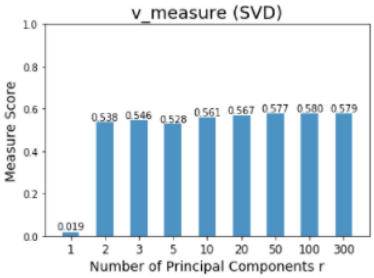
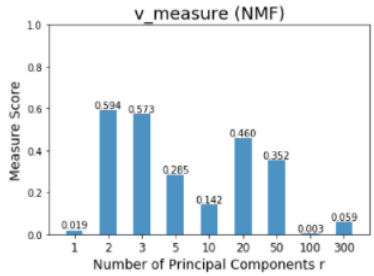
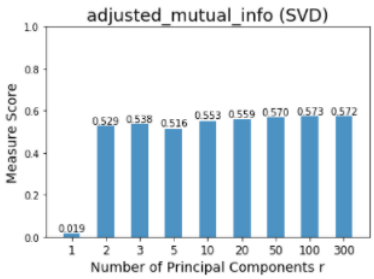
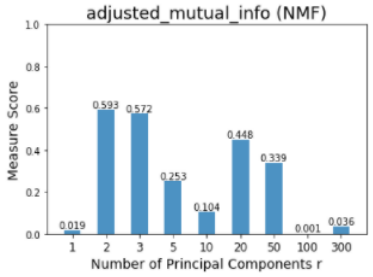
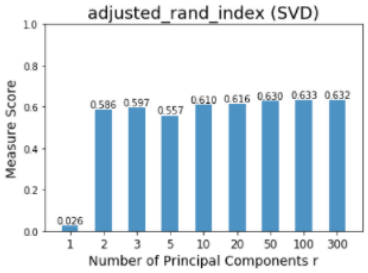
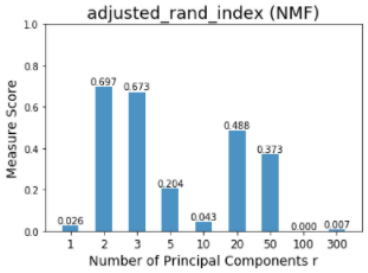



Table 3: SVD and NMF Measurements for Varying r Principal Components

| Measure Score | SVD | NMF |
|---|---|---|
| Homogeneity Score |  |  |
| Completeness Score |  |  |
| V-Measure Score |  |  |

| | | |
|---|---|---|
| Adjusted Rand Index |  |  |
| Adjusted Mutual Information |  |  |

*Question 6: How do you explain the non-monotonic behavior of the measures as r increases?*

We observed that the 5 measurement scores are non-monotonic as r increases. This contradicts our initial assumption because when r is too low, the information is limited thus cannot guarantee the correctness of the clustering which caused all measurement scores low. As r increases, there are more information to help clustering more accurately which contributes to increase in measurement scores. However, when r is too big values, it then becomes a high dimensional matrix. As discussed earlier, Euclidean distance is not a good metric for high dimensional matrix because the distances between data points tend to be the same.

Compared to SVD, the results for NMF look much more volatile given changes in the number of principal components. While we do expect a certain degree of non-monotonicity, the drastic changes we see in the results do not seem justified by the reasons we mentioned earlier. These changes are more likely due to the fact that NMF is a random, non-unique process that can converge prematurely at local optima. It seems feasible that the drastic decreases in the adjusted Rand index can be attributed to the fact that the dimensionality reduction converged before an optimal solution was found. More dimensions or features that are introduced, the noisier the data gets, which leads to inaccuracy, but the more features there are, the more information there is to work with, which can enhance the accuracy of the data. These two can fluctuate and cause increases and decreases in the measurement, which accounts for the non-monotonic behavior.

*Question 7: Visualize the clustering results for:*

- *SVD with its best r*
- *NMF with its best r*

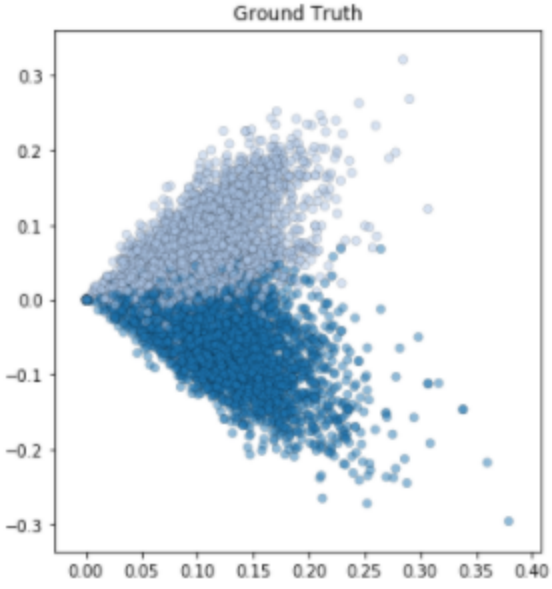
In order to visualize the data, we can plot the points on a 2D graph and color the data points according to the cluster it belongs to. We plotted the two clusters according to the clustering algorithm we ran with the best r value, and plotted the data points using the actual labels to compare.

The best r-value (see above for how it is calculated) for SVD was 100, and the clustering result we obtained was very similar to the ground truth, or the actual classifications given for the dataset.

The best r-value for NMF was 2, and the clustering result we obtained was very similar to the ground truth, albeit less so than for SVD. This could be attributed to a less sharply-defined line in the ground truth clustering.

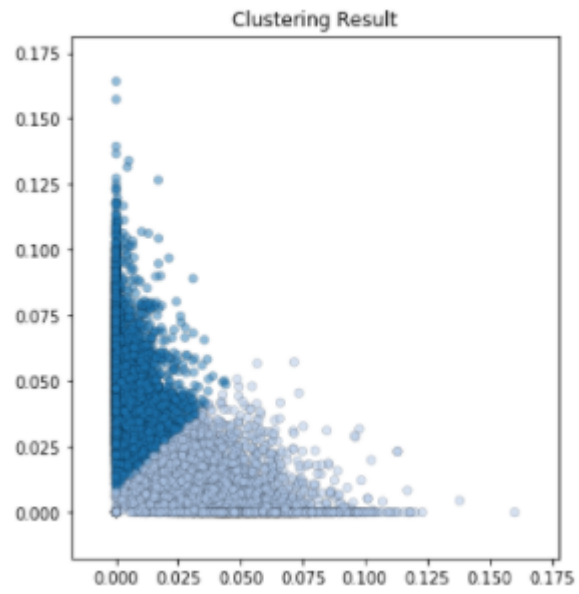
Table 4: Clustering Visualization for SVD and NMF with best r parameter.

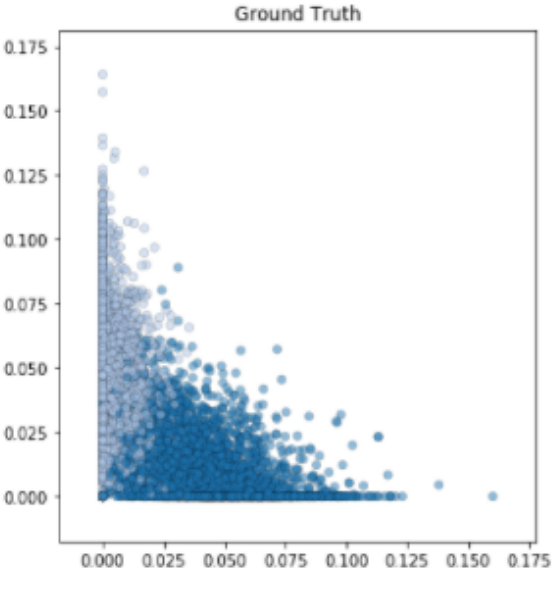
| SVD, r = 100 | |
|---|---|
| Clustering Result | Ground Truth |


NMF, r = 2

| Clustering Result | Ground Truth |
|---|---|




*Question 8: Visualize the transformed data as in part (a).*

As in question 7 we visualize the transformed data of and use color the data based on its corresponding cluster and compare with its ground truth to see how well the clustering algorithm performed. The transformations we performed were logarithmic transformation, where

$$\mathbf{f}(\mathbf{x}) = \mathbf{sign}(\mathbf{x}) \cdot \big(\log(|\mathbf{x}| + c) - \log c\big), \quad \big(\mathbf{sign}(\mathbf{x})\big)_i \equiv \begin{cases} 1 & x_i > 0 \\ 0 & x_i = 0 \\ -1 & x_i < 0 \end{cases}$$

and c was set to 0.01. A scaling feature, in which each data column was given unit variance by preprocessing with standard deviation, was also applied to test how these processing and transformation affected the measurements for both SVD and NMF. Additionally, the logarithmic transformation of the scaled unit variance data and the scaled unit variance of the logarithmic transformation were also applied. In the transformations for SVD, the logarithmic transformation performed much better across all measures than the other transformations. For the transformations for NMF, the logarithmic function also performed the best, but the scores were much closer than the other measures using SVD.

Table 5: Measures for K-means Clustering with Different Logarithmic Transformations

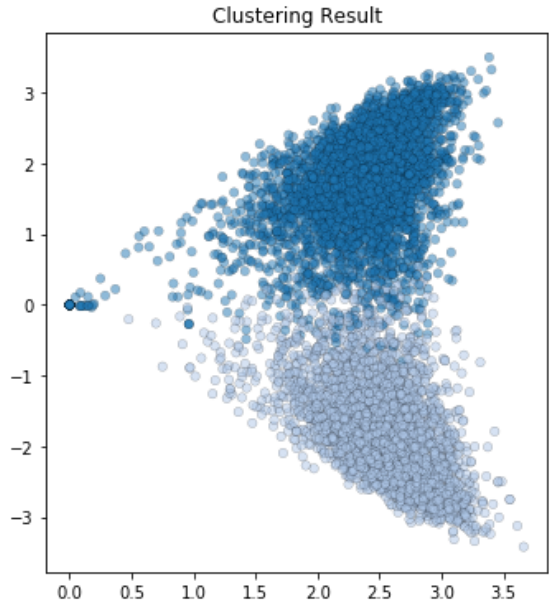
| SVD | | | | | |
|---|---|---|---|---|---|
| | Homogeneity Score | Completeness Score | V-Measure Score | Adjusted Rand Index | Adjusted Mutual Information |
| Logarithmic Transformation | 0.6049 | 0.6093 | 0.6071 | 0.6986 | 0.6048 |
| Unit Variance Scaled | 0.0457 | 0.0488 | 0.0472 | 0.0584 | 0.0456 |
| Logarithmic Transformation of Unit Variance Scaled | 0.1792 | 0.1914 | 0.1851 | 0.2097 | 0.1791 |
| Unit Variance Scaling of Logarithmic Transformation | 0.0077 | 0.0078 | 0.0078 | 0.0106 | 0.0076 |
| NMF | | | | | |
| | Homogeneity Score | Completeness Score | V-Measure Score | Adjusted Rand Index | Adjusted Mutual Information |

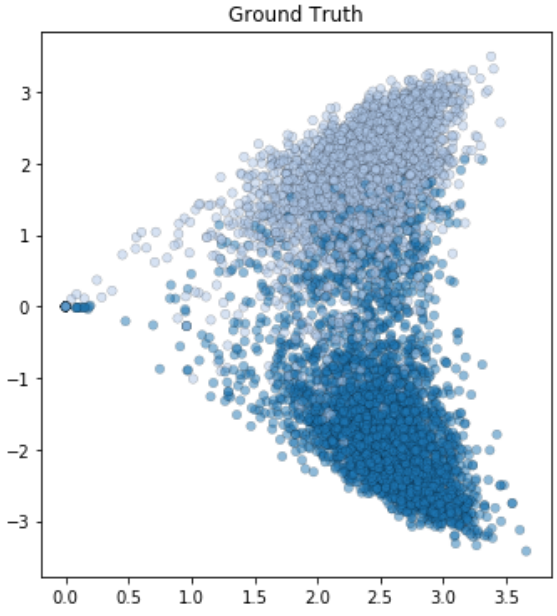
| | | | | | |
|---|---|---|---|---|---|
| Logarithmic Transformation | 0.6040 | 0.6072 | 0.6056 | 0.7020 | 0.6040 |
| Unit Variance Scaled | 0.5865 | 0.5930 | 0.5897 | 0.6746 | 0.5865 |
| Logarithmic Transformation of Unit Variance Scaled | 0.4637 | 0.4840 | 0.4737 | 0.5036 | 0.4637 |
| Unit Variance Scaling of Logarithmic Transformation | 0.5978 | 0.6014 | 0.5996 | 0.6948 | 0.5977 |

The visual representation of the clustering data for the transformations of SVD and NMF. The logarithmic transformations clearly performed the best for both SVD and NMF, but the NMF responded best to the data transformations overall.

Table 6: Clustering Visualization for Transformed Data

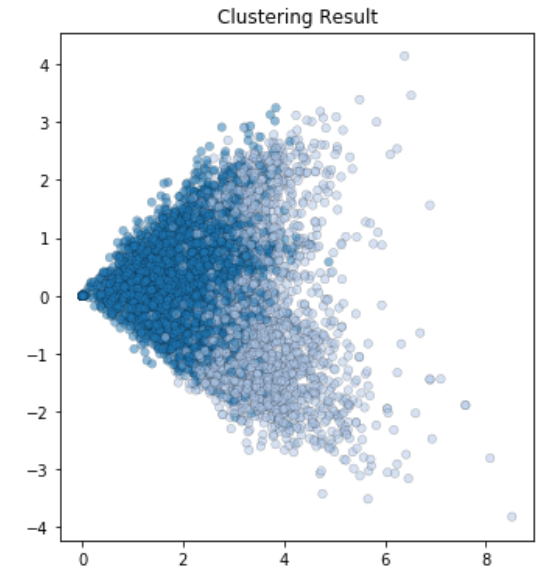
| SVD Transformations | |
|---|---|
| Logarithmic Transformation | |
| Clustering Result | Ground Truth |

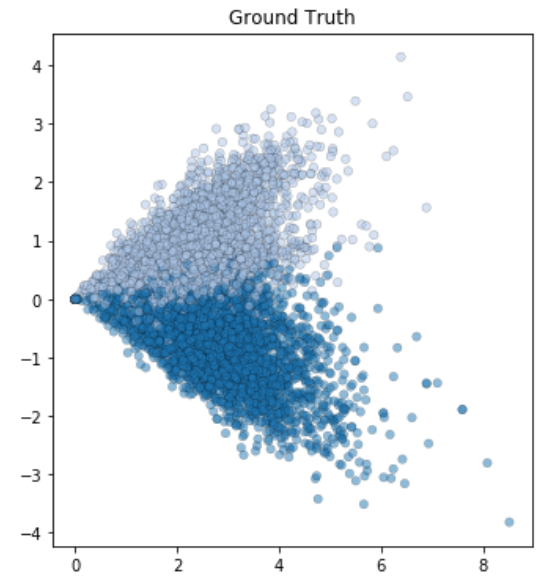



Unit Variance Scaled

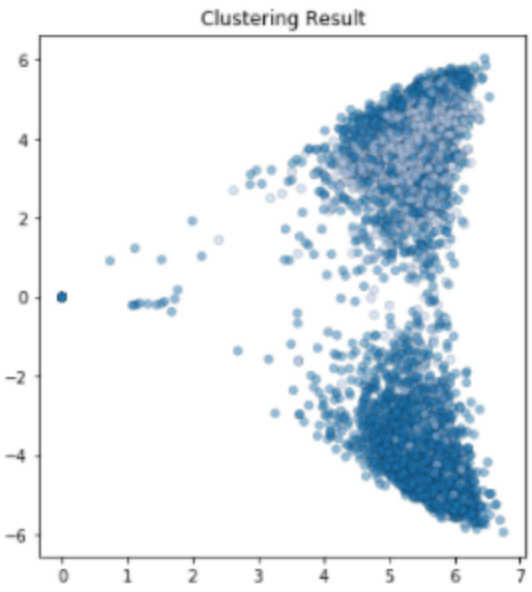
| Clustering Result | Ground Truth |
|---|---|

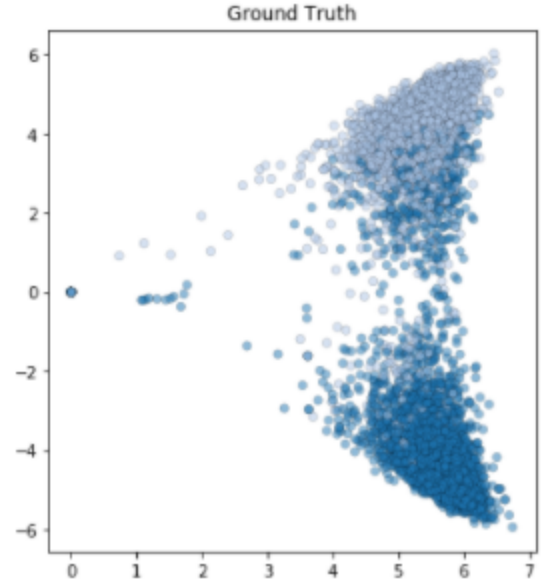



Logarithmic Transformation of Unit Variance Scaled

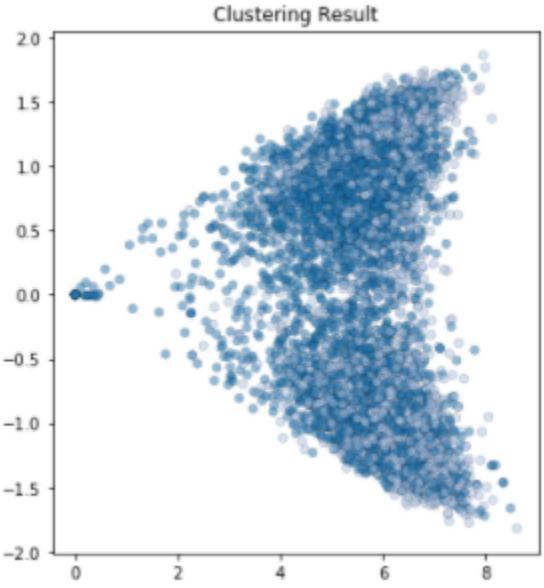
| Clustering Result | Ground Truth |
|---|---|

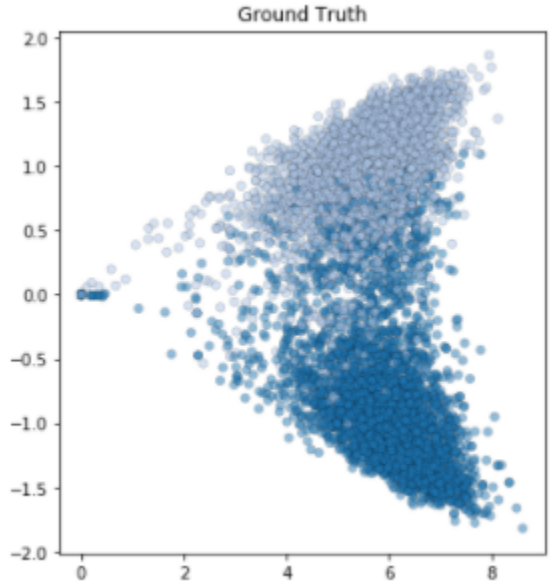



Unit Variance Scaling of Logarithmic Transformation

| Clustering Result | Ground Truth |
|---|---|

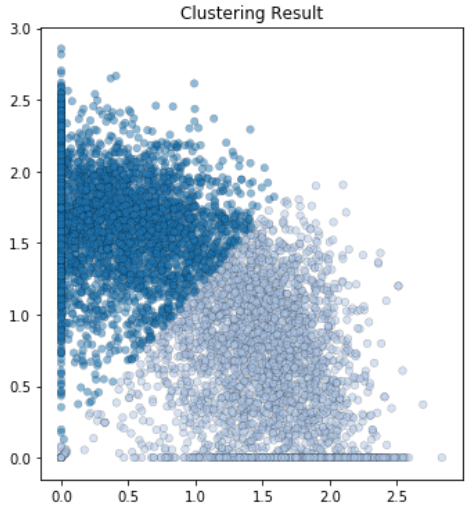
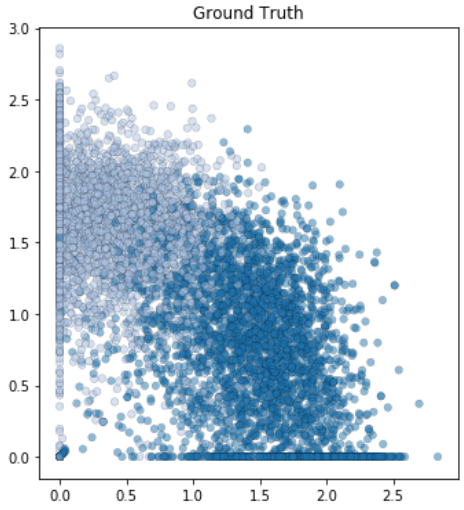
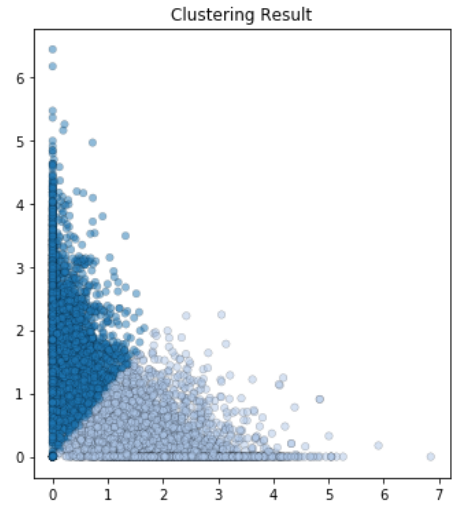
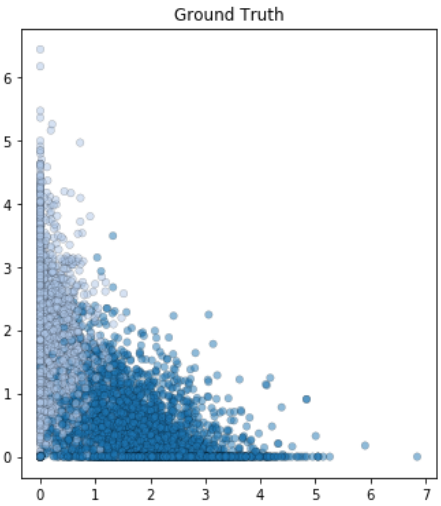


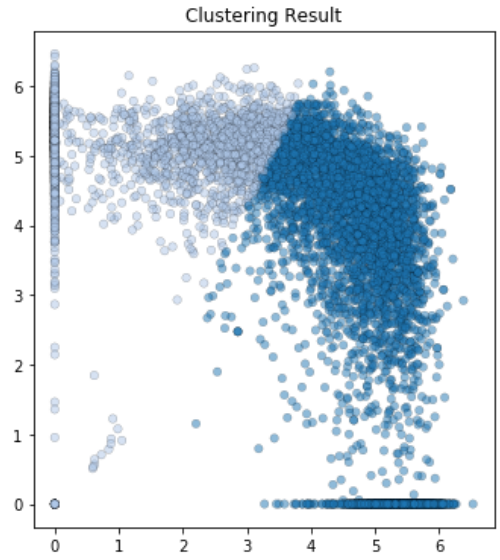
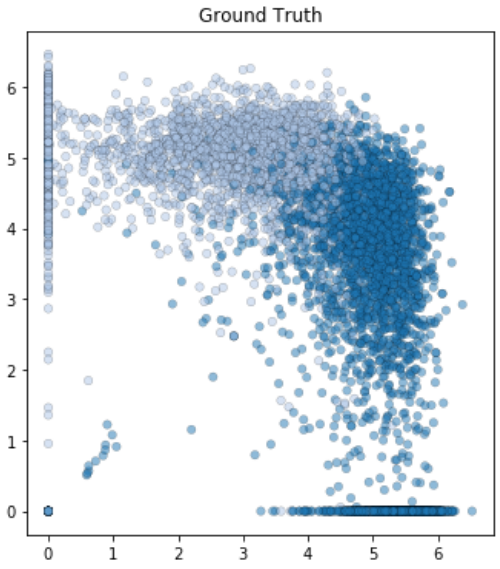
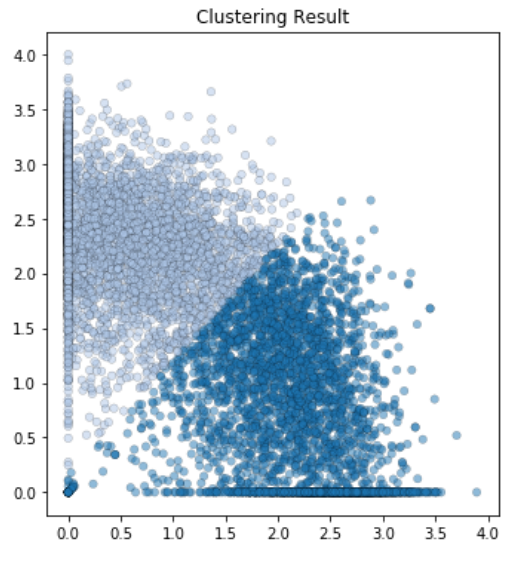
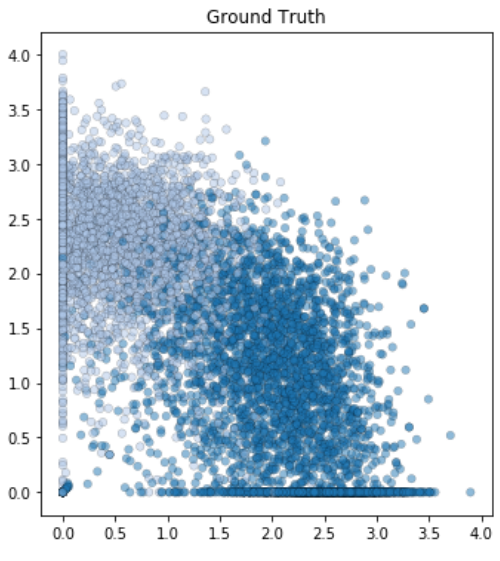




NMF Transformations

Logarithmic Transformation

| Clustering Result | Ground Truth |
|---|---|

<table>
<tr><td> 
</td><td> 
</td></tr>
<tr><td colspan="2">Unit Variance Scaled</td></tr>
<tr><td>Clustering Result</td><td>Ground Truth</td></tr>
<tr><td> 
</td><td> 
</td></tr>
<tr><td colspan="2">Logarithmic Transformation of Unit Variance Scaled</td></tr>
<tr><td>Clustering Result</td><td>Ground Truth</td></tr>
</table>

<table>
<tr><td> 
</td><td> 
</td></tr>
<tr><td colspan="2">Unit Variance Scaling of Logarithmic Transformation</td></tr>
<tr><td>Clustering Result</td><td>Ground Truth</td></tr>
<tr><td> 
</td><td> 
</td></tr>
</table>

*Question 9: Can you justify why the "logarithm transformation" may improve the clustering results?*

The "logarithm transformation" might improve clustering results since it provides a sublinear smoothing to measurements related to term frequency. If a term appears 20 times in document A and 10 times in document B, it is not necessarily true that document A should weigh the term twice as much as document B. A potential justification for this is Heap's law, an empirical law

stating that the probability of encountering an unseen word in a document decreases as more words are processed. In other words, we should expect to words repeat themselves as the word count increases. Due to this "expected" increase in word frequency, it does not seem obvious that linearly correlating term frequency to a word vector weight is the best choice.

The logarithm transformation still weighs frequent terms more heavily than infrequent terms, but it does so in a way where the increases at the larger frequencies provide diminishing returns. It is not obvious that this will improve the clustering results after applying the dimensionality reductions, although the results do seem to indicate an increase in performance when the logarithm transformation is added. In fact, in comparing the 2-D projections of SVD with and without the nonlinear transformation, the transformed results qualitatively show more of a separation between the two classes. We attribute this improvement to diminishing the weight increases of term-frequencies at higher magnitudes.

We also tested the effects of scaling each column of the data matrix to have unit variance, and found that this made the clustering worse. This is likely due to the fact that scaling every feature to have the same variance will distort the relative importance of that feature. For instance, if one feature captures a "topic score" for the "computers" category, it likely has a high variance since roughly half of the documents have a high score, and the other half have a low score. This is good, since the scores will look drastically different for different topics. Alternatively, if another feature captures a topic that is common to both classes, the variance should be quite low and should not contribute much to the clustering results.

If we now scale both of these features to have unit variance, the noise in the data for the common category now becomes amplified, and the importance of the data in the "computers" category is diminished. We see these effects in the clustering results below (Question 10), as the unit variance scaling lowers the clustering scores.

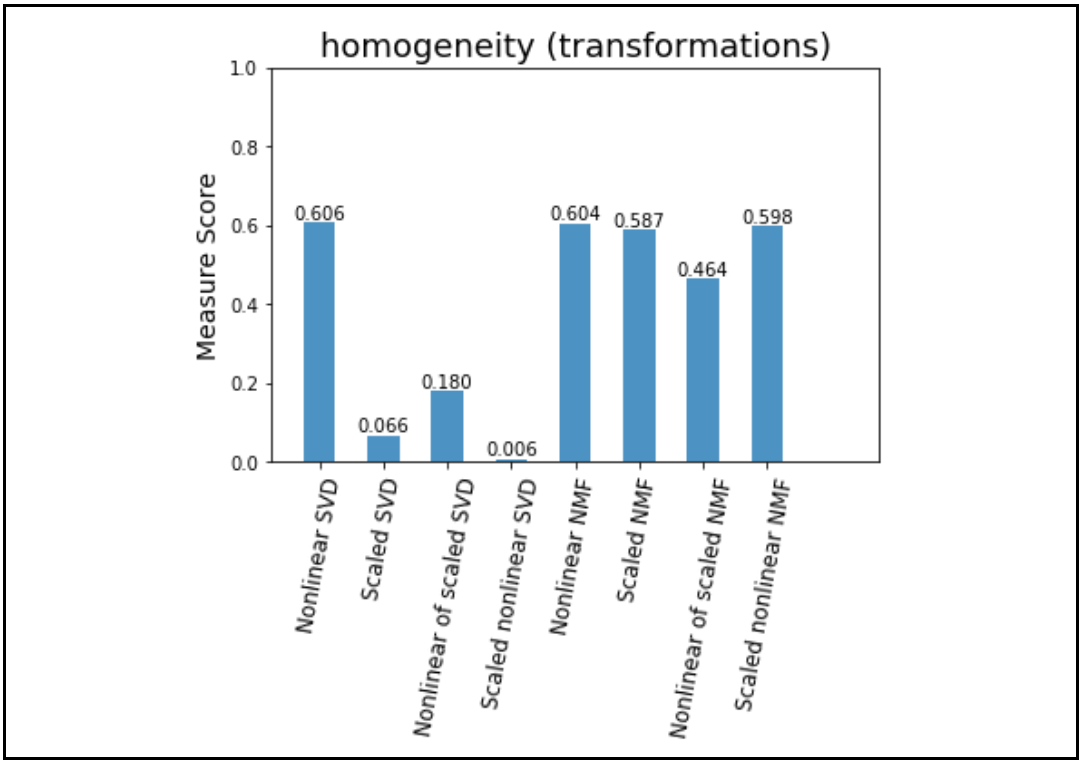
*Question 10: Report the new clustering measures (except for the contingency matrix) for the clustering results of the transformed data.*

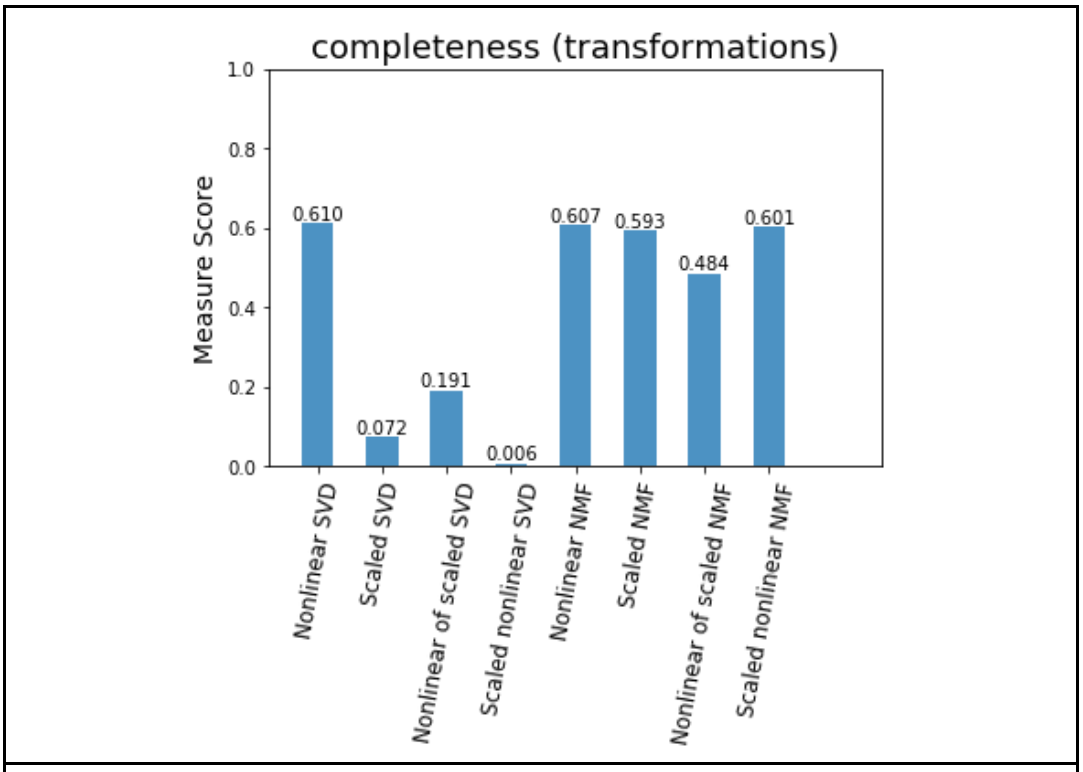
The results comparing the measurements of the different clustering transformations can be seen below. Across all measures, it is seen that the logarithmic (non-linear) SVD transformations greatly outperformed the other SVD transformations, and that the logarithmic transformation for NMF, while still performing better than the other NMF transformations, was not as drastic a difference. In fact, the scaled unit variance for NMF and the unit variance scaling of logarithmic transformation performed almost as well for NMF across all measures. The logarithmic transformation for SVD tended to perform slightly better than the logarithmic transformation for NMF for all measures.

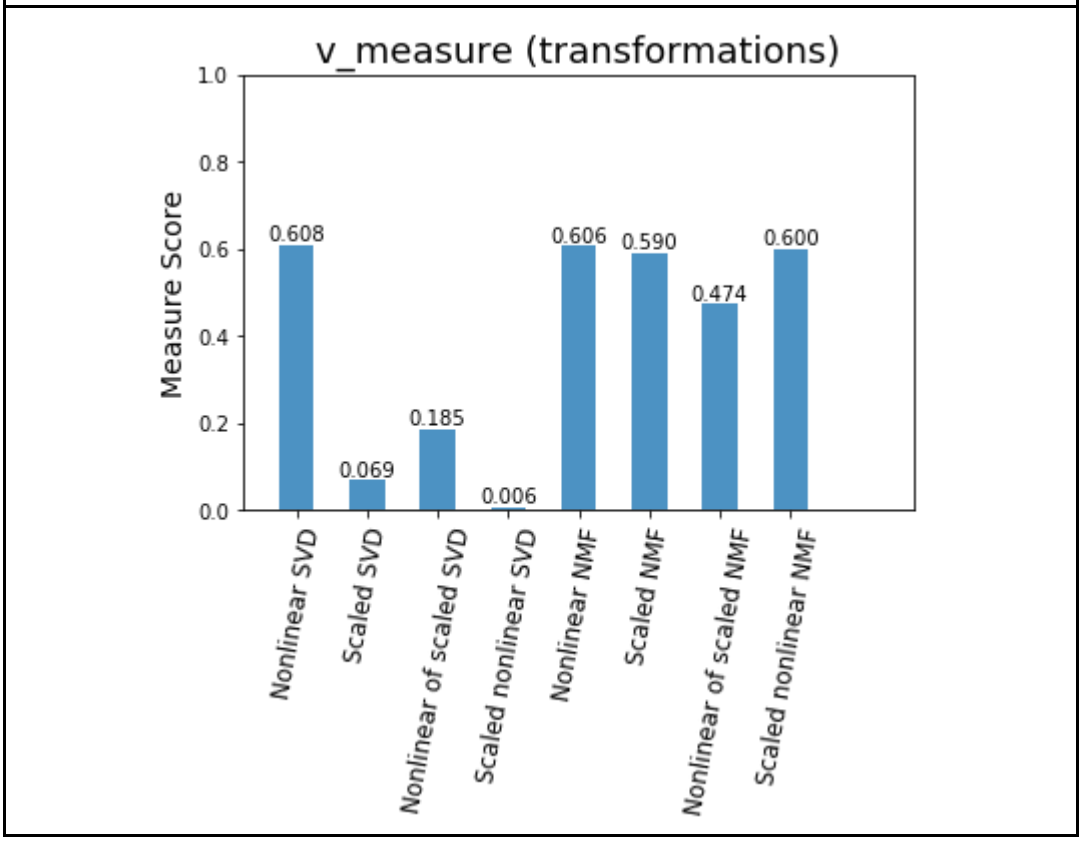
With respect to the non-transformed data, the logarithmic transformation for SVD was the only transformation that performed better than the non-transformed data, and it did so across all measures. For NMF, the logarithmic transformation performed better again for the non-transformed data, as did the unit variance scaling of logarithmic transformation.

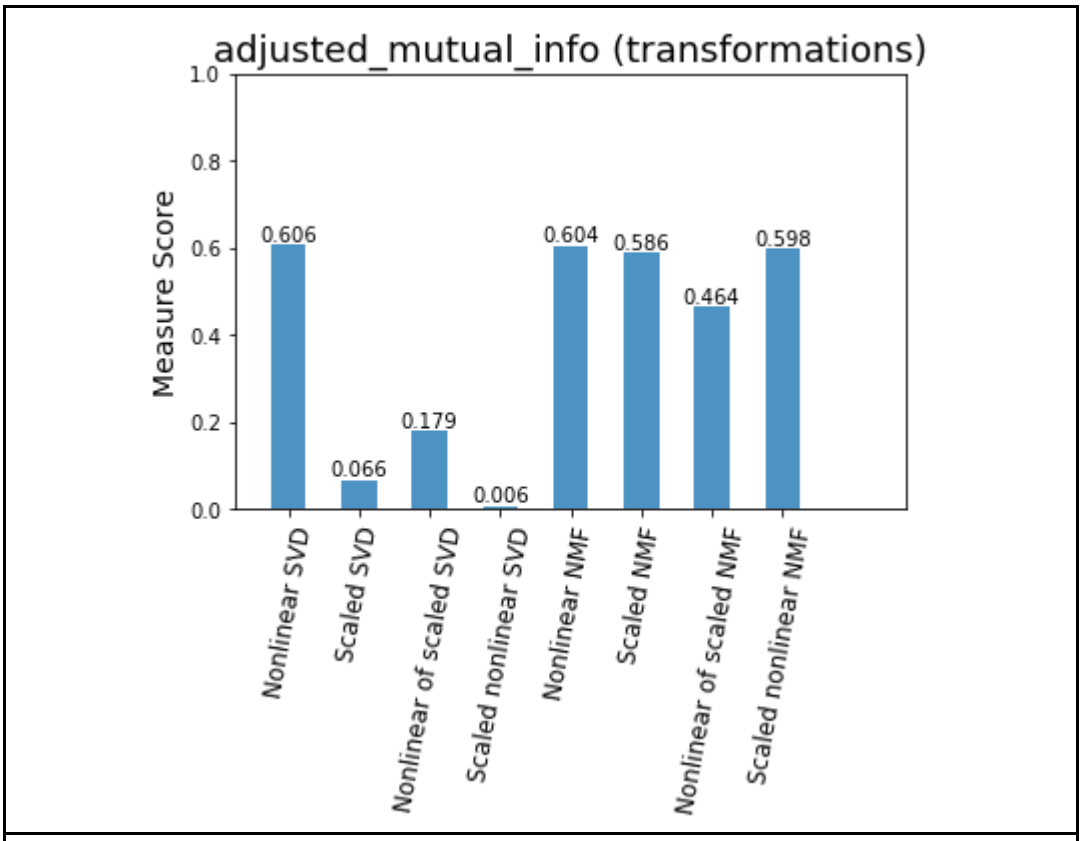
Note: The label “nonlinear” refers to the logarithmic transformation from question 9, and the label “scaled” refers to performing a unit-variance scaling on each data column.

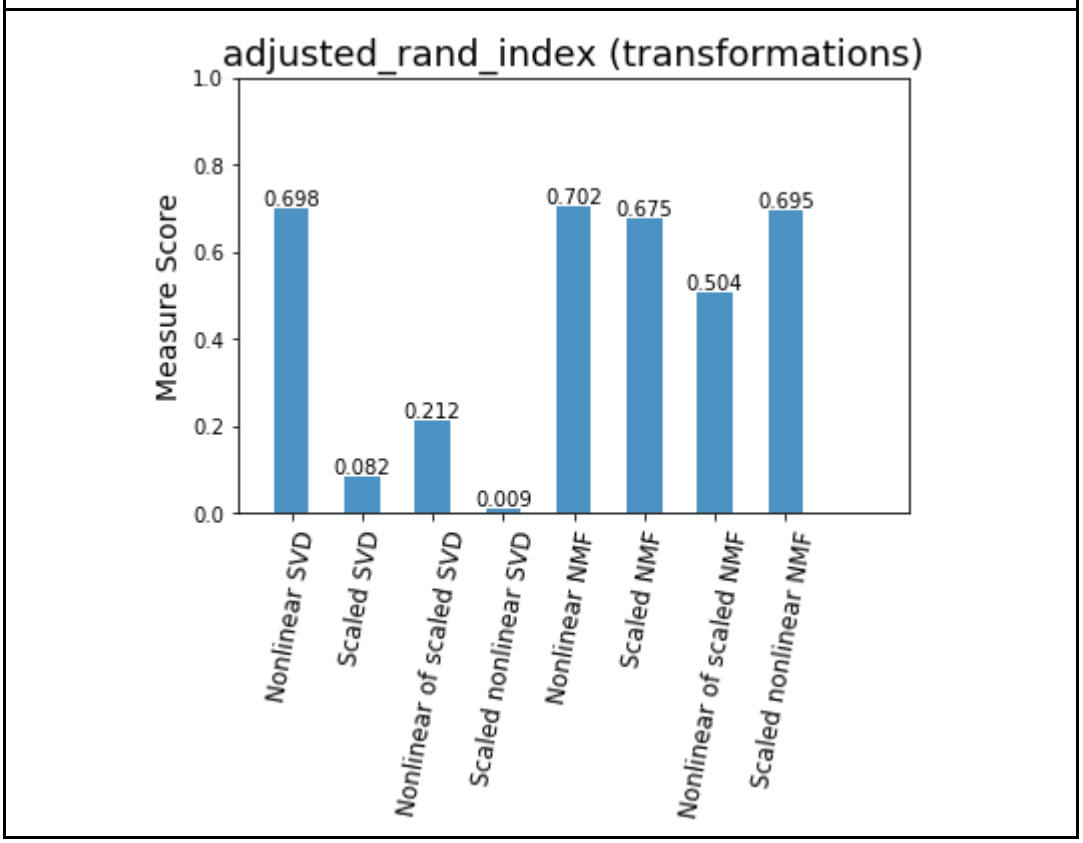
Table 7: SVD and NMF Measurements for Transformed r Principal Components




completeness (transformations)
Measure Score
1.0
0.8
0.6
0.4
0.2
0.0
0.610
0.072
0.191
0.006
0.607
0.593
0.484
0.601
Nonlinear SVD
Scaled SVD
Nonlinear of scaled SVD
Scaled nonlinear SVD
Nonlinear NMF
Scaled NMF
Nonlinear of scaled NMF
Scaled nonlinear NMF


v_measure (transformations)
Measure Score
1.0
0.8
0.6
0.4
0.2
0.0
0.608
0.069
0.185
0.006
0.606
0.590
0.474
0.600
Nonlinear SVD
Scaled SVD
Nonlinear of scaled SVD
Scaled nonlinear SVD
Nonlinear NMF
Scaled NMF
Nonlinear of scaled NMF
Scaled nonlinear NMF


adjusted_mutual_info (transformations)
Measure Score
1.0
0.8
0.6
0.4
0.2
0.0
0.606
0.066
0.179
0.006
0.604
0.586
0.464
0.598
Nonlinear SVD
Scaled SVD
Nonlinear of scaled SVD
Scaled nonlinear SVD
Nonlinear NMF
Scaled NMF
Nonlinear of scaled NMF
Scaled nonlinear NMF


adjusted_rand_index (transformations)
Measure Score
1.0
0.8
0.6
0.4
0.2
0.0
0.698
0.082
0.212
0.009
0.702
0.675
0.504
0.695
Nonlinear SVD
Scaled SVD
Nonlinear of scaled SVD
Scaled nonlinear SVD
Nonlinear NMF
Scaled NMF
Nonlinear of scaled NMF
Scaled nonlinear NMF

*Question 11: Repeat the following for 20 categories using the same parameters as in 2-class case:*

- *Transform corpus to TF-IDF matrix;*
- *Directly perform K-means and report the 5 measures and the contingency matrix;*

In this part, we wanted to explore how "pure" or clustering is using all 20 original categories. We included all documents and terms in the data matrix. Therefore, we repeated the following 20 categories clustering using the same parameters as in 2-class case; that is *random_state=0*, max_iter=1000*, n_iter=30, and n_clusters=20.* Since this does take an long time to run, we decided to put that through Pickle (a persistent object that can be loaded from a file). We found that the homogeneity score to be 0.5008, completeness score to be 0.3902, V-Measure score to be 0.4387, Adjusted Rand Index score to be 0.3902, and Adjusted Mutual Information to be 0.3883.

Figure 3: Contingency Table for 20 Categories

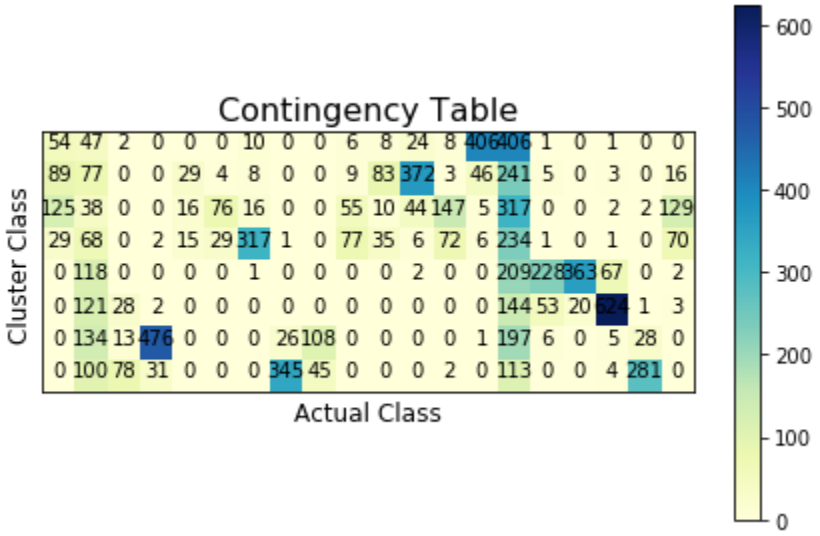


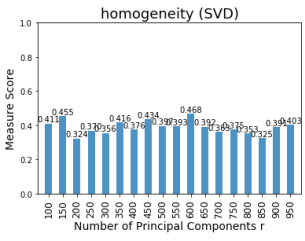
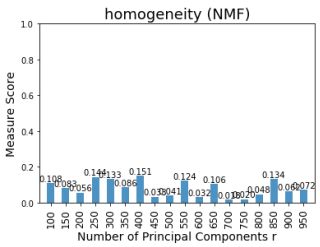
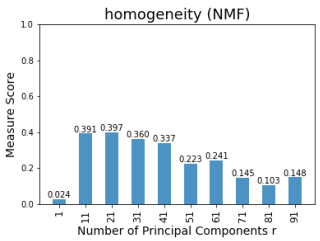
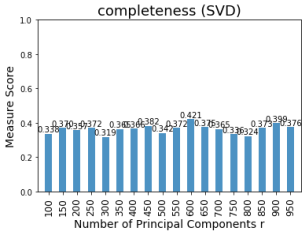
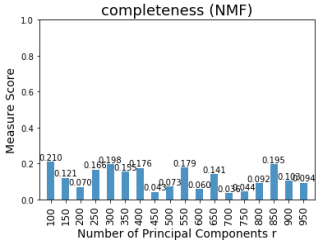
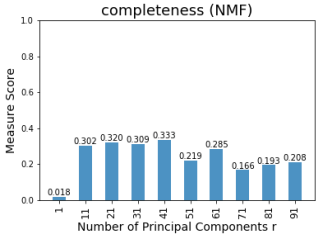
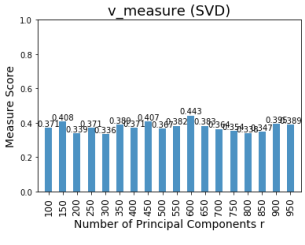
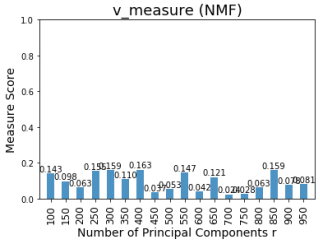
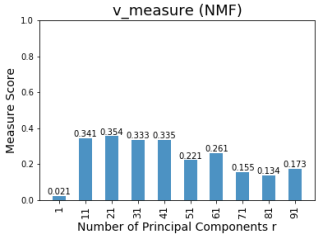

Table 8: Measurement Scores for 20 Categories

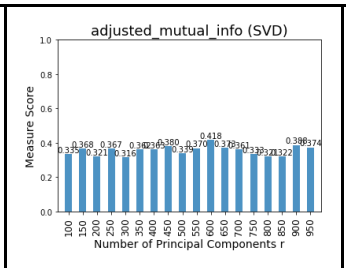
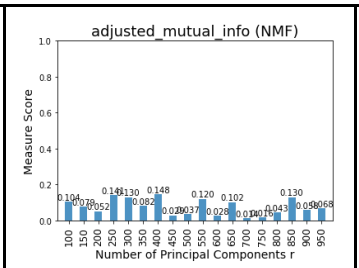
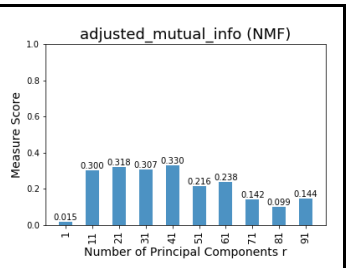
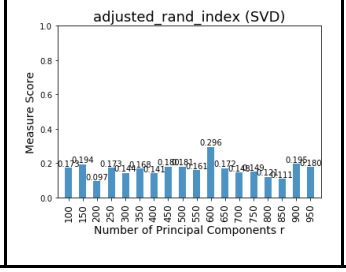
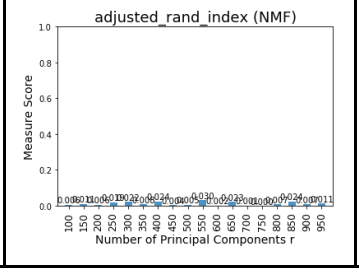
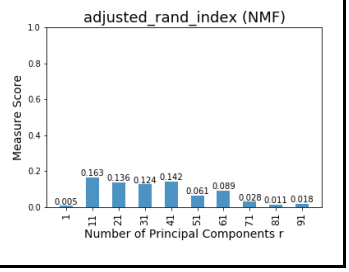
| **Measure** | **Score** |
|---|---|
| Homogeneity Score | 0.5008 |
| Completeness Score | 0.3902 |
| V-Measure Score | 0.4387 |
| Adjusted Rand Index | 0.2195 |
| Adjusted Mutual Information | 0.3883 |

*Question 12: Try different dimensions for both truncated SVD and NMF dimensionality reduction techniques and the different transformations of the obtained feature vectors as outlined in above parts.*
*You don't need to report everything you tried, which will be tediously long.* ***You are asked, however, to report your best combination, and quantitatively report how much better it is compared to other combinations. You should also include typical combinations showing what choices are desirable (or undesirable)****.*

In this problem, we explored different dimensions for both truncated SVD and NMF. However because it took excessively long time to run for each iteration, we decided to lower max_iter and n_init to 5 in order to save some time. We iterated starting at 100 dimensions to 1000 with increment of 50 for both SVD and NMF as seen in Table 9 below. According to the 5 measurement scores, it seems that NMF does not perform well at all when principal component is greater than 100. Therefore, we decided to try a different range for NMF; the range is from 1 to 100 with increment of 10.

Table 9: SVD and NMF Measurements for Varying r Principal Components with 20 categories

| Measure Score | SVD (100-1000) | NMF (100-1000) | NMF (1-100) |
|---|---|---|---|
| Homogeneity Score |   |   |   |
| Completeness Score |   |   |   |
| V-Measure Score |   |   |   |

| | | | |
|---|---|---|---|
| Adjusted Rand Index |  |  |  |
| Adjusted Mutual Information |  |  |  |

Interestingly, we found that NMF does not perform well for higher principal components, even for higher number of clusters. We found that the measurement scores have highest values at 600 principal components for SVD, and at 21 principal components for NMF. Overall, as expected, we found that SVD performs much better than NMF. Again, it is likely due to the fact that NMF is a random, non-unique process that can converge prematurely at local optima. Therefore, more dimensions are introduced create noisier data which explains why NMF performs poorly at higher principal components. Table 10 below is Contingency Table for optimized SVD and NMF respectively.

Table 10: Contingency Table optimized SVD and NMF using 20 categories

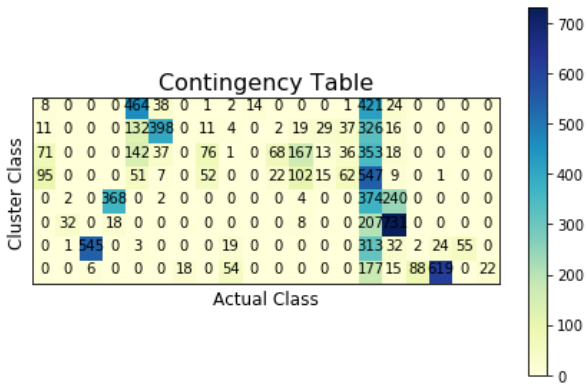
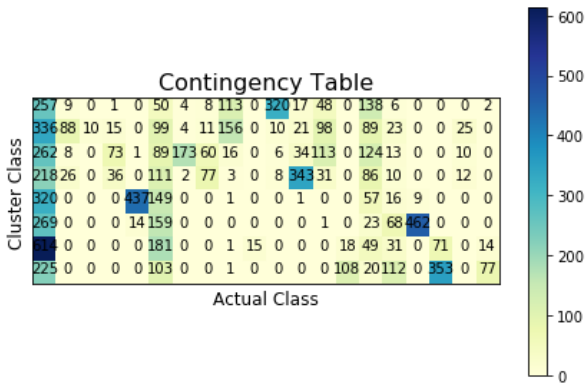
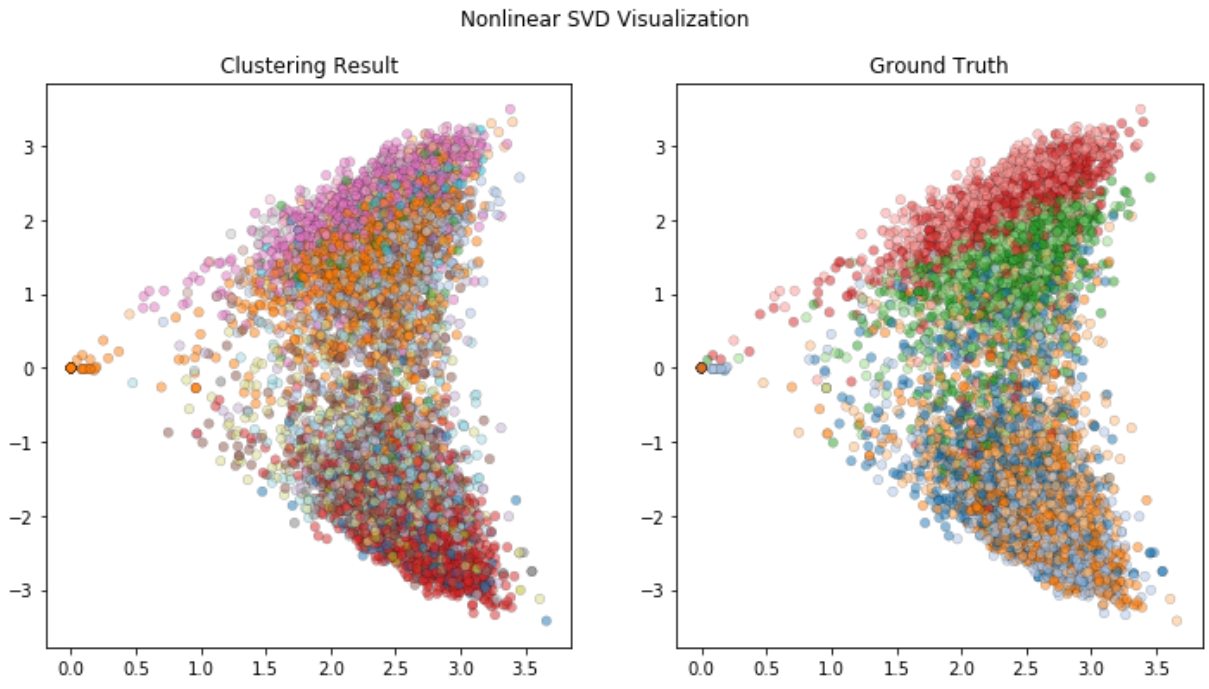
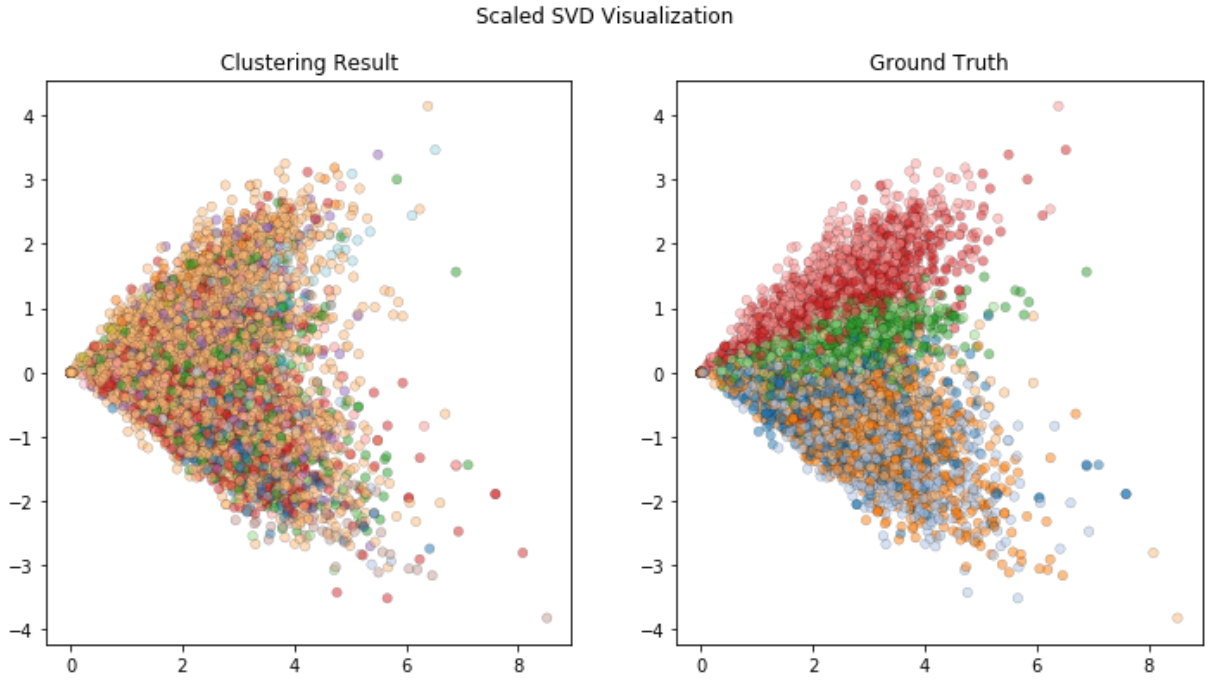
| SVD (r=600) | NMF (r=21) |
|---|---|
|  |  |

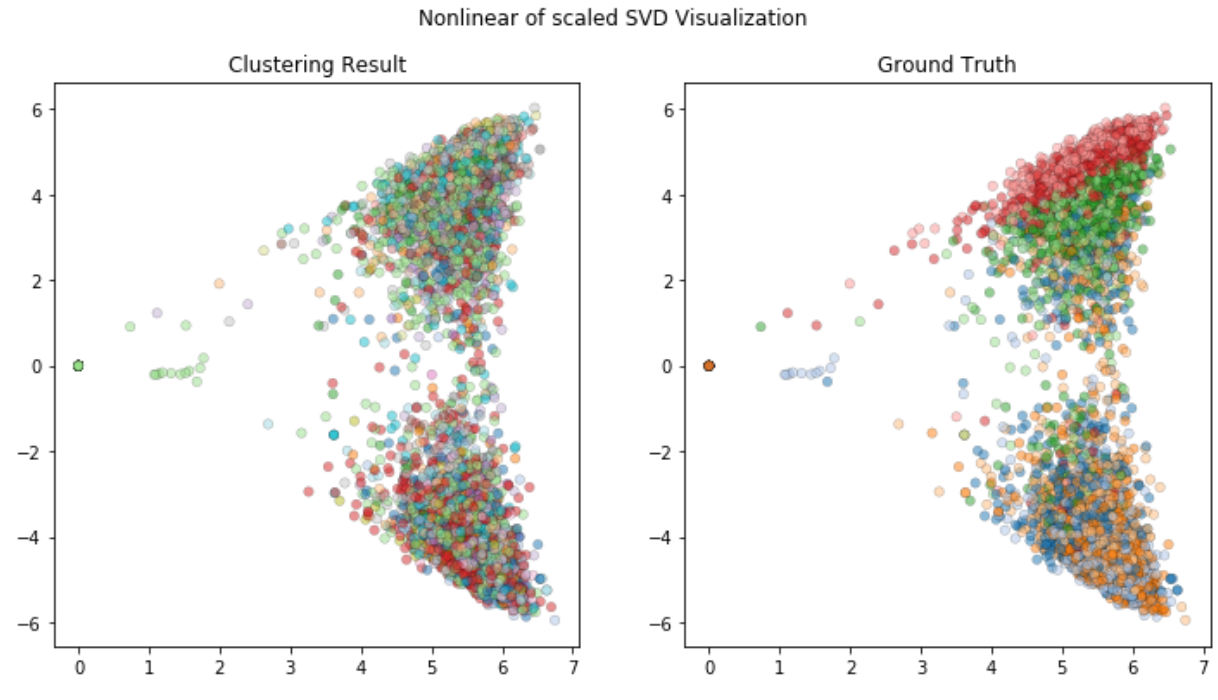
Table 11: Measures for K-means Clustering with Different Transformations with 20 categories

| SVD | | | | | |
|---|---|---|---|---|---|
| | Homogeneity Score | Completeness Score | V-Measure Score | Adjusted Rand Index | Adjusted Mutual Information |
| Logarithmic Transformation | 0.4170 | 0.3487 | 0.3798 | 0.2528 | 0.3464 |
| Unit Variance Scaled | 0.0913 | 0.1161 | 0.1022 | 0.0121 | 0.0873 |
| Logarithmic Transformation of Unit Variance Scaled | 0.1478 | 0.1210 | 0.1331 | 0.0452 | 0.1180 |
| Unit Variance Scaling of Logarithmic Transformation | 0.1230 | 0.1184 | 0.1207 | 0.0315 | 0.1148 |
| NMF | | | | | |
| | Homogeneity Score | Completeness Score | V-Measure Score | Adjusted Rand Index | Adjusted Mutual Information |
| Logarithmic Transformation | 0.4667 | 0.3367 | 0.3912 | 0.2481 | 0.3348 |
| Unit Variance Scaled | 0.4141 | 0.3217 | 0.3621 | 0.1511 | 0.3195 |
| Logarithmic Transformation of Unit Variance Scaled | 0.4256 | 0.2987 | 0.3510 | 0.1864 | 0.2967 |
| Unit Variance Scaling of Logarithmic Transformation | 0.5258 | 0.3771 | 0.4392 | 0.2890 | 0.3753 |

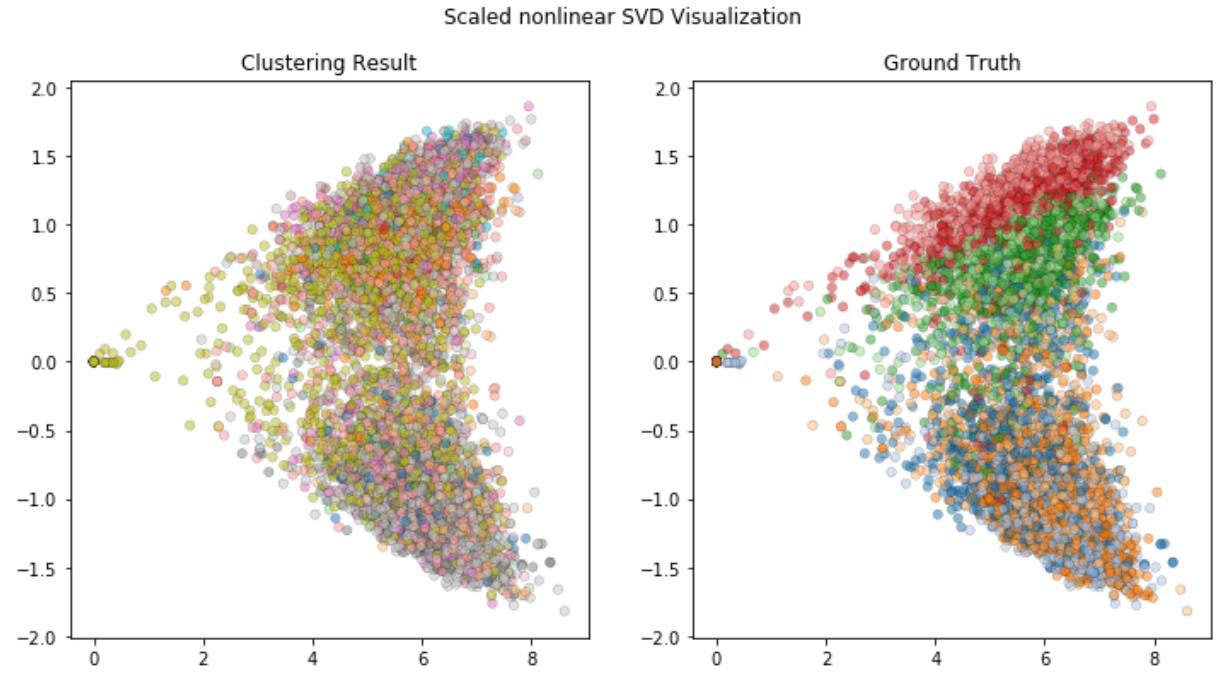
Table 12: Clustering Visualization for Transformed Data

| SVD Transformations |
| --- |
| Logarithmic Transformation |
|  |
| Unit Variance Scaled |
|  |

Logarithmic Transformation of Unit Variance Scaled

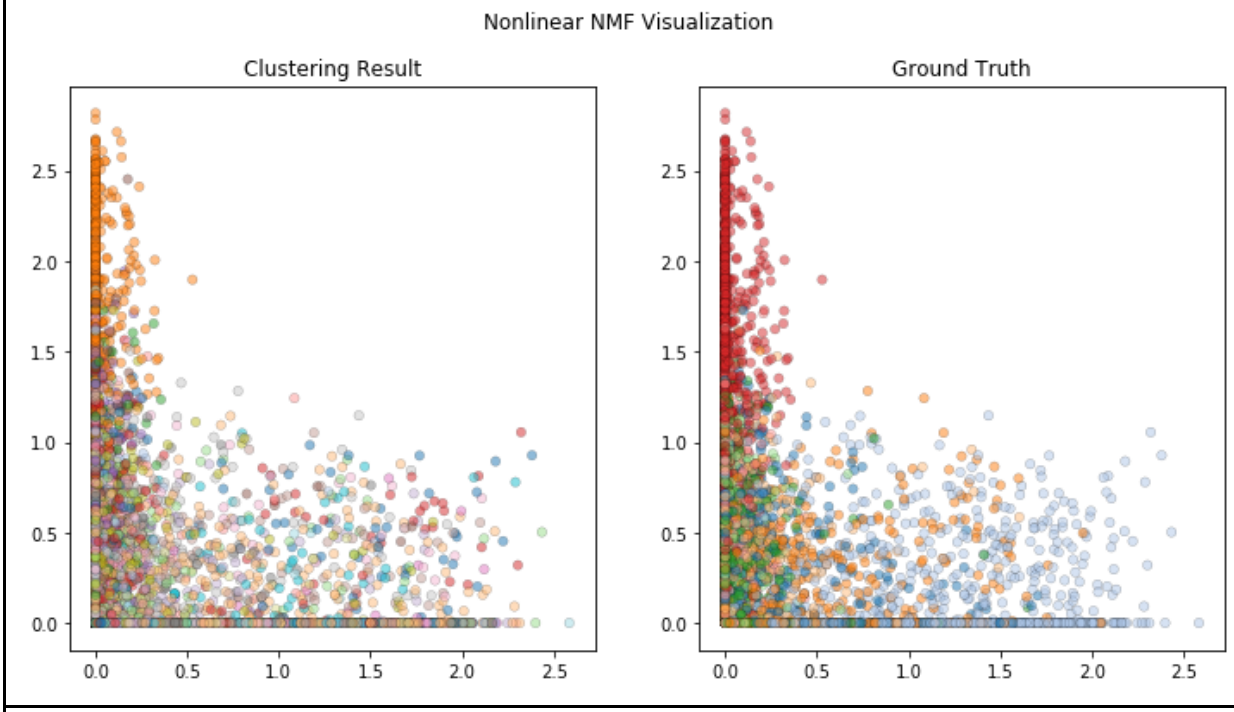



Unit Variance Scaling of Logarithmic Transformation

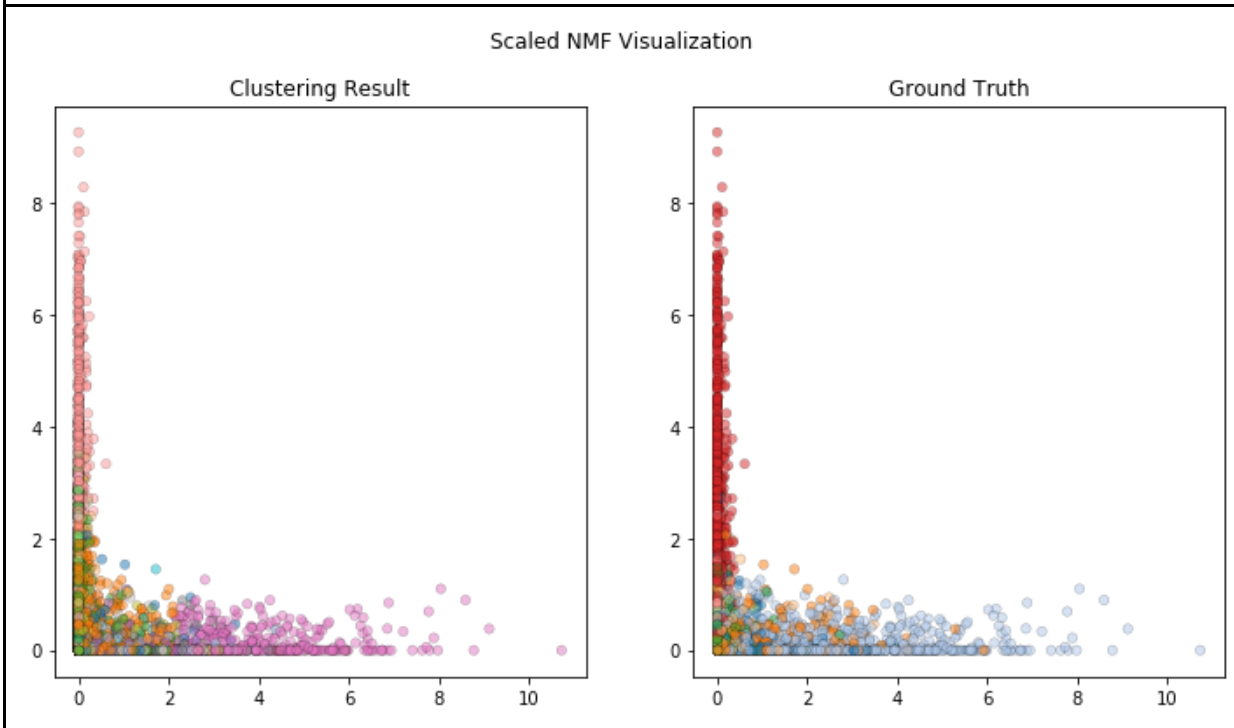



NMF Transformations

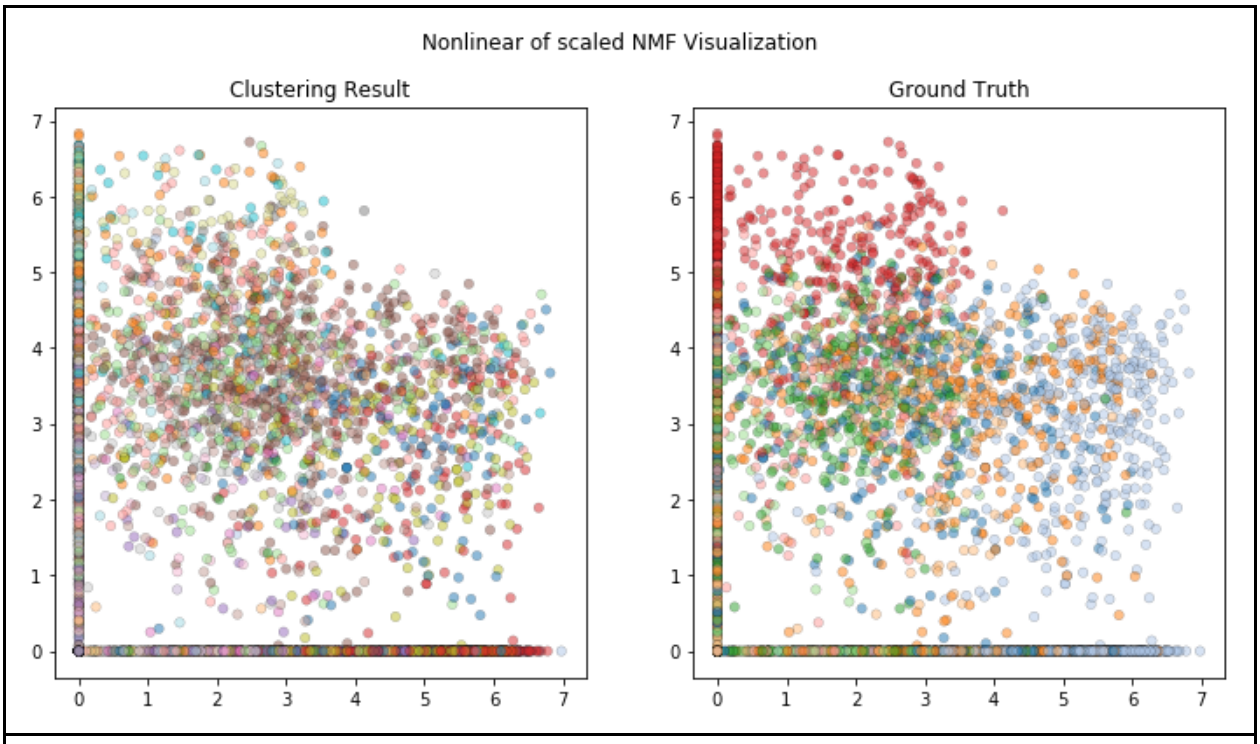
Logarithmic Transformation



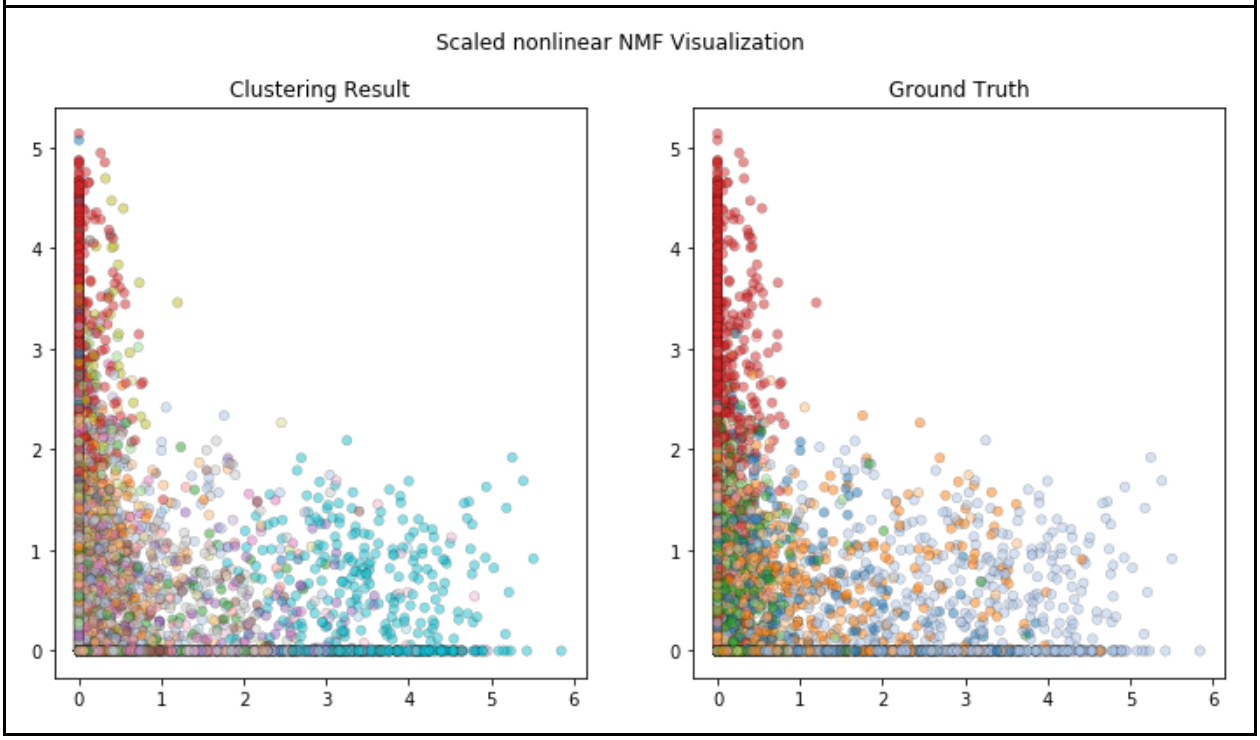

Unit Variance Scaled




Logarithmic Transformation of Unit Variance Scaled




Unit Variance Scaling of Logarithmic Transformation



For further transformation, we performed  different transformations similar to problem 8 above against best SVD and NMF which is r=600 and r=21 respectively. As seen in table 13, he unit variance scaling of the logarithmic transformation improved the results of NMF when using the best r-value of 21. However, none of the transformations of SVD using the best r-value of 600 improved the measurements, and in all cases except for the logarithmic transformation, actually made all five scores markedly worse.

The delta, or differences between a few combinations and the "best" combination can be seen in table 14. The difference from the best SVD combination (r=600, no transformation) and the other two compared  (r=600, logarithmic; r=100, no transformation) was modest, although the logarithmic transformation when r=600 for the adjusted mutual index measurement was slightly better than the best SVD combination. The difference from the best NMF combination (r=21, unit variance scaling of logarithmic transformation) was noticeably larger than the two compared (r=1, no transformation; r=21, no transformation), especially compared to the the r=1 value, with the homogeneity score difference being the highest at -0.502!

From the many different combinations that we've run, it seems that for SVD, a high r-value is desirable, and depending on the number of iterations used (unfortunately, a low number of iterations had to be used for question 12 due to the sheer volume of calculations) a logarithmic transformation could help improve that score a bit further. For NVD, an extremely low r-value seems to be desirable (although not too low, as seen when the r-value is set to 1), along with a unit variance scaling of logarithmic transformation or a logarithmic transformation.

Table 13: Comparisons of SVD and NMF with Different R-values and Transformations

| SVD | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | Measures | | | | |
| | R-Value | Transformation | Homog | Comp | V-Meas | Adj. Rand | Adj. Mut |
| Combo | 100 | None | 0.411 | 0.338 | 0.371 | 0.335 | 0.173 |
| | 600 | None | 0.468 | 0.421 | 0.443 | 0.418 | 0.296 |
| | 600 | Logarithmic | 0.417 | 0.349 | 0.380 | 0.253 | 0.346 |
| NMF | | | | | | | |

| | | | Measures | | | | |
|---|---|---|---|---|---|---|---|
| | R-Value | Transformation | Homog | Comp | V-Meas | Adj. Rand | Adj. Mut |
| Combo | 1 | None | 0.024 | 0.018 | 0.021 | 0.015 | 0.005 |
| | 21 | None | 0.397 | 0.320 | 0.354 | 0.318 | 0.136 |
| | 21 | Scaling of Logarithmic | 0.526 | 0.377 | 0.439 | 0.289 | 0.375 |

Table 14: Differences Comparisons of SVD and NMF with Different R-values and Transformations Compared to Best Combination

| SVD | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | Measures | | | | |
| | R-Value | Transformation | Homog | Comp | V-Meas | Adj. Rand | Adj. Mut |
| Combo | 100 | None | -0.057 | -0.083 | -0.108 | -0.083 | -0.123 |
| | 600 | None | 0 | 0 | 0 | 0 | 0 |
| | 600 | Logarithmic | -0.051 | -0.072 | -0.0632 | -0.165 | +0.050 |
| NMF | | | | | | | |
| | | | Measures | | | | |
| | R-Value | Transformation | Homog | Comp | V-Meas | Adj. Rand | Adj. Mut |
| Combo | 1 | None | -0.502 | -0.359 | -0.418 | -0.274 | -0.370 |
| | 21 | None | -0.129 | -0.057 | -0.085 | +0.029 | -0.239 |
| | 21 | Scaling of Logarithmic | 0 | 0 | 0 | 0 | 0 |

**Conclusion**

This project provided a meaningful introduction to k-Means clustering against reduced dimensions such as Latent Semantic Index (LSI) and Non-Negative Matrix Factorization (NMF). We were able to find good representations of the data, performed and provided analysis of k-Means clustering with 5 main measurements scores: *homogeneity score*, *completeness score, V-measure, adjusted Rand score*, and *adjusted mutual info score.* We also explored a wide variety of different parameters to find the most optimized for k-Means algorithm. In conclusion, we found that SVD (LSI) performs much better than NMF.