Matrix Decompositions in Data Analysis Assignment Report: Non-negative Matrix Factorization for newsgroups data set.

Ta Quoc Viet (299954)

Table of Contents

1 Introduction

Negative Matrix Factorization (NMF) is a set of matrix factorization algorithms those address the problem where negative values in the component matrices do not seem appropriate [1]. Occurrences of words in documents are one of those scenarios.

In this assignment, we have to implement, experiment and compare some of the NMF algorithms with the newsgroup real-world data set.

1.1 The newsgroup data set

The *20 Newsgroups* data set is a collection of approximately 20,000 newsgroup documents, partitioned (nearly) evenly across 20 different newsgroups. The data is organized into 20 different newsgroups, each corresponding to a different topic [2].

comp.**graphics** comp.os.ms-windows.**misc** comp.sys.ibm.pc.**hardware** comp.sys.mac.**hardware** comp.windows.**x**	rec.**autos** rec.**motorcycles** rec.sport.**baseball** rec.sport.**hockey**	sci.**crypt** sci.**electronics** sci.**med** sci.**space**
misc.**forsale**	talk.politics.**misc** talk.politics.**guns** talk.politics.**mideast**	talk.religion.**misc** alt.**atheism** soc.religion.**christian**

Table 1: Newsgroup data organization

The form of the data is a matrix $A(2000, 5136)$ document-term matrix. One cell a_{ij} represent the frequency of term frequency of word j in document i. We can notice that this is a very sparse matrix, where most of the cells are zeros.

1.2 Technologies and Equipment

Programing language: Python 3

Libraries: SciPy, NumPy, sklearn, matplotlib

Testing machine: MacBook Pro Late 2013 2.4 GHz Intel Core i5, 8GB RAM

2 Task 1: ALS vs. multiplicative NMF

In the first task, we have to implement several versions of the NMF algorithm:

- NMF based on Alternating Least Squares (ALS)
- NMF multiplicative updates by Lee and Seung (LNS)
- NMF via Oblique Projected Landweber (OPL) gradient descent updates

2.1 NMF based on Alternating Least Squares

This algorithm is the simplest one to implement. I followed the pseudo code provided in the course's slide to implement the optimization function for the matrices W and H. It worked great out of the box.

1. $\boldsymbol{W} \leftarrow \mathrm{random}(n, k)$
2. **repeat**
   2.1. $\boldsymbol{H} \leftarrow [\boldsymbol{W}^{+}\boldsymbol{A}]_{+}$
   2.2. $\boldsymbol{W} \leftarrow [\boldsymbol{A}\boldsymbol{H}^{+}]_{+}$
3. **until** convergence

2.2 NMF multiplicative updates by Lee and Seung

For this algorithm, I started by following the algorithm described in the slide:

1. $\boldsymbol{W} \leftarrow \mathrm{random}(n, k)$
2. $\boldsymbol{H} \leftarrow \mathrm{random}(k, m)$
3. **repeat**
   3.1. $h_{ij} \leftarrow h_{ij} \frac{(\boldsymbol{W}^T\boldsymbol{A})_{ij}}{(\boldsymbol{W}^T\boldsymbol{W}\boldsymbol{H})_{ij} + \varepsilon}$
   3.2. $w_{ij} \leftarrow w_{ij} \frac{(\boldsymbol{A}\boldsymbol{H}^T)_{ij}}{(\boldsymbol{W}\boldsymbol{H}\boldsymbol{H}^T)_{ij} + \varepsilon}$
4. **until** convergence

The algorithm also worked as expected and did converge. After that, I tried to optimize this algorithm and found that we can normalize the matrix $\boldsymbol{W}$ to sum to 1 after each iteration [3]. The result is a little bit better.

2.3 NMF via Oblique Projected Landweber (OPL) gradient descent updates

OPL provides the mechanism to select the step size in the $\boldsymbol{H} \leftarrow \boldsymbol{H} - \varepsilon_{\boldsymbol{H}} \frac{\partial f}{\partial \boldsymbol{H}}$ updates. We can set the learning rates to $\frac{1}{rowSums(\boldsymbol{W}^T\boldsymbol{W})}$. Here is the algorithm for OPL:

```
1. W ← random(n, k)
2. H ← random(k, m)
3. repeat
   3.1. H ← H − ε_H ∂f/∂H
   3.2. W ← W − ε_W ∂f/∂W
4. until convergence
```

And here is how to update $\boldsymbol{H}$ in the loop:

```
1. η ← diag(1 / rowSums(WᵀW))
2. repeat
   2.1. G ← WᵀWH – WᵀA
   2.2. H ← [H – ηG]₊
3. until a stopping criterion is met
```

During implementation, I encountered some problem inferring the update for $\boldsymbol{W}$ from the $\boldsymbol{H}$ updating rule. After some debugging and study, I finally derived the update step for $\boldsymbol{W}$:

1. $\boldsymbol{\eta} \leftarrow \mathrm{diag}(1 / \mathrm{rowSums}(\boldsymbol{H}\boldsymbol{H}^T))$
2. **repeat**
   - 2.1. $\boldsymbol{G} \leftarrow \boldsymbol{W}\boldsymbol{H}\boldsymbol{H}^T - \boldsymbol{A}\boldsymbol{H}^T$
   - 2.2. $\boldsymbol{W} \leftarrow [\boldsymbol{W} - \boldsymbol{G}\boldsymbol{\eta}]_+$
3. **until** a stopping criterion is met

With this the update step works correctly.

Note that the number of iterations here from the slide stated that it's a small number. To find the ideal one, I firstly tried 20 iterations, however, the program took significantly longer time in comparisons to others update methods to finish. Hence, after some more trials and errors, I settled with the number of iterations equal 5. From this the algorithm works fine.

2.4 Implementation

I define the convergence condition is when the current repetition reconstruction error does not differ by a threshold value of 1% to the average of the last 5 reconstruction errors. This can be achieved by utilizing the fixed size queue (FIFO) data structure. When we queue one element, if the queue is full, the oldest value will be dequeued. By this we will always have the last 5 recent reconstruction errors tracked.

I modified the template NMF to make it possible to detect convergence and stop on it by a while loop which check if the convergence has happened or the max number of iterations has been reached. As a requirement in the assignment, the maximum number of iterations is 300.

2.5 Experiment method

Since the initialization of $\boldsymbol{W}$ and $\boldsymbol{H}$ can affect the result, so to compare the performance of each version of the NMF algorithm fairly, we have to run all the test on the same $\boldsymbol{W}$ and $\boldsymbol{H}$. Also, the number of repetitions for each version should be large enough to achieve statistical stability; 300 should be a safe number for this.

During the NMF algorithm is running, there're several types of information tracked:

- Best results (best $\boldsymbol{W}$ and $\boldsymbol{H}$) after all repetitions
- Best convergence after all repetitions
- The number of iterations needed for the algorithm to converge for each repetition
- Reconstruction errors at convergence point of each repetition
- Convergence time of each repetition

2.6 Results & Discussions

In total there were there runs were performed. The results look consistently similar over the runs. Here is the console output of the last run for the first task. The average reconstruction errors, average number of iterations needed to convergence, and average convergence time over 300 repetitions are printed out:

- Finished NMF with 300 repetitions of NMF **ALS** optimization function
    - Average reconstruction errors: **76833.35863997308**
    - Average number of iterations needed to convergence: **7.87(3)**
    - Average convergence time (ms): **1533.3933333333334**
- Finished NMF with 300 repetitions of NMF **Lee and Seung** optimization function
    - Average reconstruction errors: **77821.63773435664**
    - Average number of iterations needed to convergence: **12.8**
    - Average convergence time (ms): **3000.77(6)**
- Finished NMF with 300 repetitions of NMF **OPL** optimization function
    - Average reconstruction errors: **76939.62832184952**
    - Average number of iterations needed to convergence: **8.02(6)**
    - Average convergence time (ms): **4232.05**

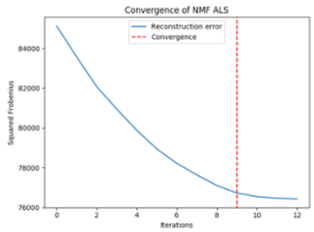
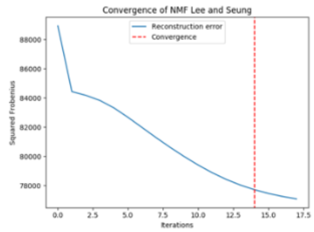
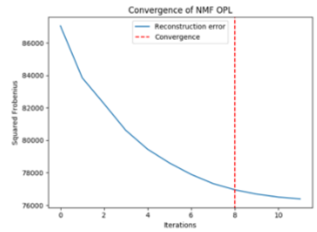
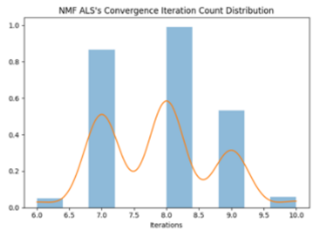
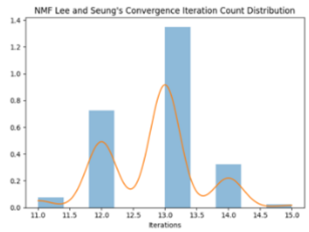
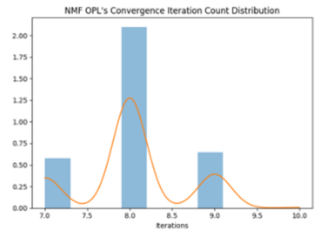
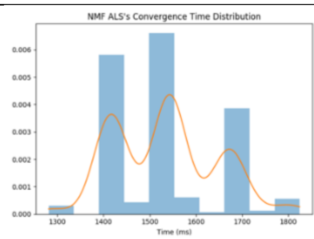
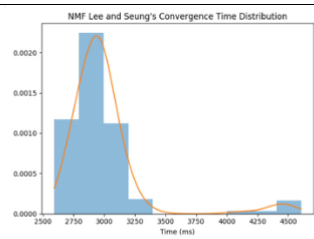
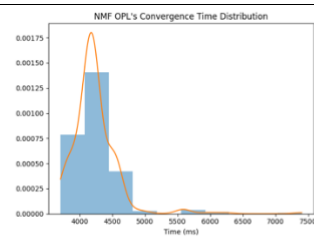
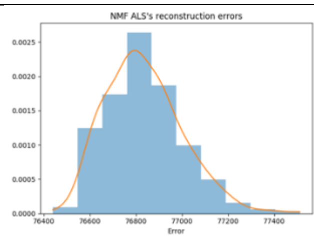
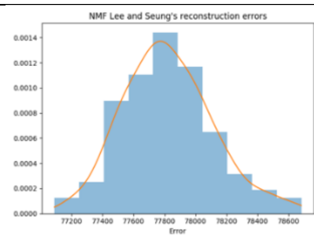
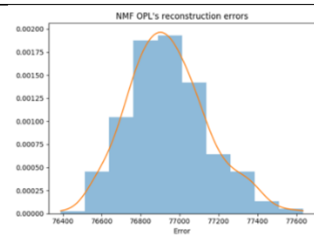
Test	NMF's ALS	NMF's Lee and Seung	NMF's OPL
Best convergence			
Convergence iteration count (number of iterations for convergence)			
Convergence time			
Reconstruction errors			

Table 2: Experiment result for task 1 from 300 repetitions

ALS and OPL both took approximately the same number of iterations to reach convergence, around 8 iterations, though ALS still ahead with a small margin. ALS was the quickest algorithm with average time of 1.5s to converge. Lee and Seung's takes more iterations to finish but actually it's still quicker than OPL with average time of 3s and 4.2s respectively. Finally, with the reconstruction errors, OPL again was just a bit behind ALS with the error of 76940 and 76833 accordingly. Lee and Seung's was quicker than OPL but yielded a higher reconstruction error of 77821. As we can see the plotted data histograms formed normal distribution bell shapes, which make sense with respect to the Central Limit Theorem.

From the above results, we can see that the **ALS version is significantly** better than the other versions (in term of speed and reconstruction error), nonetheless all the algorithms perform relatively well in this task. The reason why ALS performed better may come to the fact that our data is extremely sparse in this case, and ALS is good at factorizing sparse data [4].

3 Task 2: Analyzing the data

In this task, we have to use NMF to perform topic discovering on the newsgroup data set. Two versions of NMF (one must be Generalized Kullback-Leibler Divergence) will be put to test to see which is better.

3.1 Generalized Kullback-Leibler (GKL) Divergence method

Kullback-Leibler Divergence measures the expected number of extra bits required to code samples from P when using a code optimized for Q:

$$D_{KL}(P \parallel Q) = \sum_i P(i) ln \frac{P(i)}{Q(i)}$$

The value of $D_{KL} \in [0, 1]$, where 0 means the two probability distributions are perfectly correlated, 1 means no mutual information is found.

However, the standard KL-divergence can only apply when P and Q are probability distributions. The Generalized KL-divergence lifted this requirement, and in NMF, $P = \boldsymbol{A}$ and $Q = \boldsymbol{WH}$, thus, we have:

$$D_{GKL}(\boldsymbol{A} \parallel \boldsymbol{WH}) = \sum_{i,j} \boldsymbol{A}_{ij} ln \frac{\boldsymbol{A}_{ij}}{(\boldsymbol{WH})_{ij}} - \boldsymbol{A}_{ij} + (\boldsymbol{WH})_{ij}$$

Update rules for multiplicative GKL NMF are:

$$\boldsymbol{H}_{kj} \leftarrow \boldsymbol{H}_{kj} \frac{\sum_{i=1}^{n} \boldsymbol{W}_{ik}(\boldsymbol{A}_{ij}/(\boldsymbol{WH})_{ij})}{\sum_{i=1}^{n} \boldsymbol{W}_{ik}}$$

$$\boldsymbol{W}_{ik} \leftarrow \boldsymbol{W}_{ik} \frac{\sum_{j=1}^{m} (\boldsymbol{A}_{ij}/(\boldsymbol{WH})_{ij})\boldsymbol{H}_{kj}}{\sum_{j=1}^{m} \boldsymbol{H}_{kj}}$$

The columns of $\boldsymbol{W}$ are normalized to sum to $\mathbf{1}$ after every iteration.

3.2 Implementation

I decided to implement the GKL NMF instead of using any library. For this, I have to implement the error function, update function, and modify the current NMF implementation to be able to take an error function as a parameter.

By following the slide, I was successfully implemented the GKL NMF (`def nmf_gkl_vanila(A, w, h)`). However, my update function implementation wasn't efficient enough for a feasible running time. So, I decided to find some better approach to implement the update function for GKL. Luckily, I have found one cleaver way to write the updates:

Algorithm KL-NMF

initialize $\mathbf{W}, \mathbf{H}$

repeat

$$\mathbf{H} \leftarrow \mathbf{H} .* \frac{\mathbf{W}^T \frac{\mathbf{V}}{\mathbf{W}\mathbf{H}}}{\mathbf{W}^T \mathbf{1}}$$

$$\mathbf{W} \leftarrow \mathbf{W} .* \frac{\frac{\mathbf{V}}{\mathbf{W}\mathbf{H}} \mathbf{H}^T}{\mathbf{1}\, \mathbf{H}^T}$$

until convergence **return** $\mathbf{W}, \mathbf{H}$

Figure 1: New GKL NMF algorithm [5]

This new approach gives the same results but much faster.

3.3 Experiment method

I decided to use ALS NMF to compare with GKL since it performed best in the previous task. I performed 5 runs for each $k \in \{5,14,20,32,40\}$ for each algorithm.

The row 2 of $\boldsymbol{W}$ will be selected for analyzing. The top 10 terms with the highest value in the right factor matrix $\boldsymbol{H}$ will be selected.

3.4 Results & Discussions

k	ALS NMF	GKL NMF
5	**code** 8.135044252333036e-07 **color** 6.771078359739697e-07 **user** 6.73487060076795e-07 **fax** 6.607786073010279e-07 **disk** 6.491193590974238e-07 **screen** 6.439583546774883e-07 **error** 6.323133080938553e-07 **full** 6.11090687877732e-07 **manual** 5.96294727756101e-07 **displai** 5.890340945124344e-07	**valu** 0.0006669705649682361 **e** 0.0006439565195747647 **practic** 0.0005818823548332084 **pai** 0.0005347396172687092 **natur** 0.0005208130872905069 **appli** 0.000518450100187515 **monei** 0.0005161832148307061 **space** 0.0005127392975135956 **decid** 0.00046775271603448213 **kei** 0.00046504042092450066
	• Possible topic: computer hardware and graphics. • Confident: High • Related terms: fax, disk, screen, error, full, manual, display, color.	• Possible topic: budget for space research • Confident: Medium • Related terms: space, pay, money, apply, decide.

14	**code** 4.466459181013235e-07 **copi** 4.290488811905655e-07 **contact** 3.999745792796934e-07 **th** 3.972193851663223e-07 **specif** 3.969125644735367e-07 **correct** 3.924376453737878e-07 **request** 3.900095702057493e-07 **appear** 3.7768003079756726e-07 **special** 3.752351344004882e-07 **instead** 3.724498759342003e-07	**research** 0.000635281919749141 **space** 0.0005607514095223592 **night** 0.00040641187631740896 **develop** 0.00036018698542125574 **nasa** 0.0003301639286260597 **technolog** 0.00032995076573953046 **hit** 0.00031702368143423206 **launch** 0.000284320769860608 **shuttl** 0.00028036881183231234 **project** 0.0002717341824719111
	• Possible topic: cannot infer.	• Possible topic: space research project • Confident: Very high • Related terms: all terms
20	**studi** 2.898787193184064e-07 **medic** 2.8820604183705304e-07 **develop** 2.8442118798250747e-07 **agenc** 2.7531818715328045e-07 **organ** 2.734233692201469e-07 **project** 2.6956408505729837e-07 **associ** 2.66149069629556e-07 **research** 2.6238420072797504e-07 **april** 2.610676977541929e-07 **present** 2.604561165727819e-07	**argument** 0.0005880199217004587 **evid** 0.0004912116548532228 **religion** 0.000444504584480999 **truth** 0.0003594346766982993 **natur** 0.0003489336071202540З **jame** 0.0003127498010222540З **explain** 0.0002770635705792626 **argu** 0.0002760088775111112 **belief** 0.00027398650734858387 **faith** 0.00025926131444889606
	• Possible topic: Medical science research. • Confident: Medium • Related terms: medic, study, develop, organ, project, research.	• Possible topic: Atheism • Confident: High • Related terms: all terms except jame (?)

32	**mb** 8.05305990304906e-07 **memori** 7.781425372707154e-07 **mac** 6.82133661071673e-07 **disk** 6.294417180437741e-07 **ram** 5.917562102106015e-07 **instal** 5.722567273230132e-07 **driver** 5.348454622971537e-07 **scsi** 5.05071986739496e-07 **board** 4.992579380609567e-07 **pc** 4.986229006291679e-07	**hit** 0.00028705203599667785 **late** 0.0002543235864353643 **fail** 0.00018592071882135675 **th** 0.00017415100415993115 **decis** 0.00016976961478037134 **team** 0.00016947315399955387 **final** 0.00016346065815530606 **save** 0.00015747428665193432 **score** 0.00015325249434477385 **win** 0.00014868762012644853
	• Possible topic: Computer hardware. • Confident: Medium • Related terms: medic, study, develop, organ, project, research.	• Possible topic: sport • Confident: Medium • Related terms: team, score, win, save, final, hit
40	**creat** 2.71969479776224e-07 **valu** 2.478372592389254e-07 **happi** 2.1836481963905265e-07 **absolut** 2.166689530956787e-07 **truth** 2.044340155364166e-07 **although** 2.0417823506373594e-07 **voic** 2.0311779885926976e-07 **natur** 2.0228858880286746e-07 **realiti** 1.9902487605886527e-07 **peac** 1.9898622381739523e-07	**steve** 0.00031988730568636905 **andrew** 0.0002383709790060761 **canada** 0.00013672817398617735 **studi** 0.00013449590129556272 **commerci** 0.00013326454562050528 **theyr** 0.0001324308115082889 **air** 0.00012659728987629812 **white** 0.00012183135264351663 **kevin** 0.00012012522431033951 **sharewar** 0.00011814736575319293
	• Possible topic: Politics. • Confident: Medium • Related terms: peace, create, value, happy, truth.	• Possible topic: politics. • Confident: Low Related terms: Canada, sharewar, white.

Table 3: Discovered topic for ALS and GKL

We have the following chart:

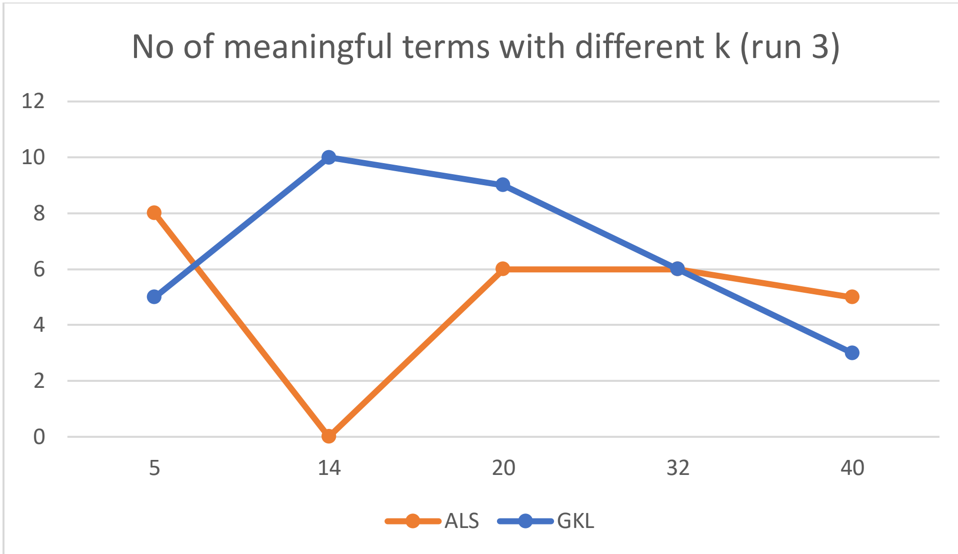


Figure 2: Number of meaningful terms with different k (run 3)

From other 2 runs performed, here're the charts for each run:

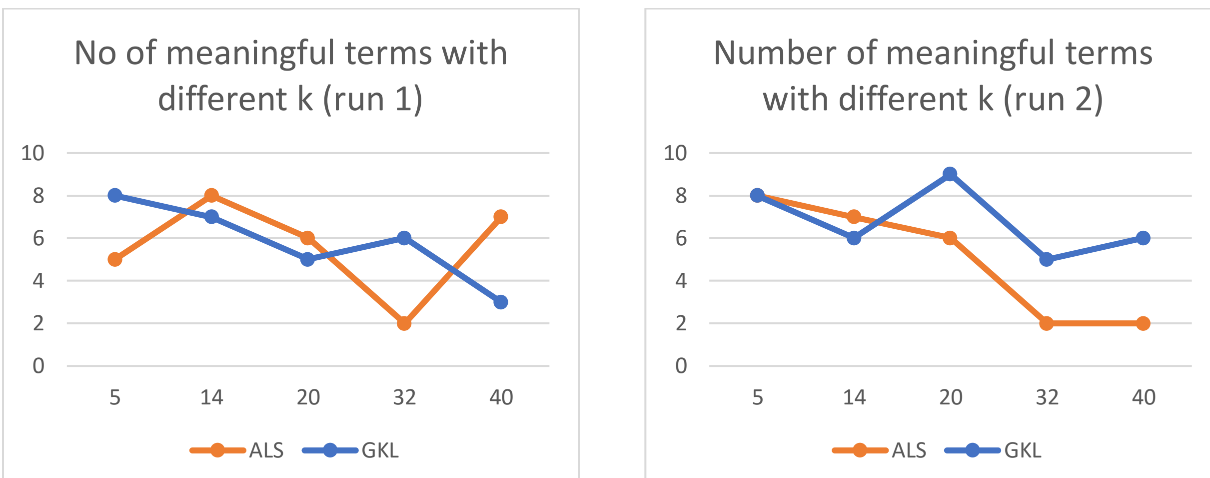


Figure 3: Number of meaningful terms with different k (run 1 & 2)

The general trend is that the result goes worse when the k is high like 32 or 40. Rank 20 seems to be the best rank overall. This make sense since the original data has 20 clusters.

We can see that generally GKL is better that ALS in topic discovery.

4 Task 3: Clustering and pLSA

In this task, we have to utilize pLSA to reduce dimension and compare the result with Karhunen-Lóeve transformation. We use Normalized Mutual Information (NMI) to measure the quality of our clustering. This takes values from $[0, 1]$, value 0 means perfect match, and 1 in reverse.

4.1 Experiment method

Three scenarios will be put to the test:

- k-means: on the normalized data, with 20 clusters.
- k-means on the first k principal components, $k = 20$.
- k-means on the $\boldsymbol{W}$ matrix of the NMF (using K-L Divergence), $k \in \{5,14,20,32,40\}$

4.2 Results and Discussions

Here's the raw output of the test on the 3 runs:

Run	Output

1	`NMI for k-mean = 0.9255533028402121`
	`NMI for KL = 0.8704720040282852`
	`NMI for NMF of rank 5 = 0.8816685869036163`
	`NMI for NMF of rank 14 = 0.8471480385791297`
	`NMI for NMF of rank 20 = 0.8415870723745043`
	`NMI for NMF of rank 32 = 0.8260407766635935`
	`NMI for NMF of rank 40 = 0.8213070688864785`

2	`NMI for k-mean = 0.8832249230407034`
	`NMI for KL = 0.8990083659934769`
	`NMI for NMF of rank 5 = 0.8828396805623329`
	`NMI for NMF of rank 14 = 0.8495514360815736`
	`NMI for NMF of rank 20 = 0.8200568114986974`
	`NMI for NMF of rank 32 = 0.8227199233594451`
	`NMI for NMF of rank 40 = 0.8107404679505668`

3	`NMI for k-mean = 0.8841741863880289`
	`NMI for KL = 0.9071604862445866`
	`NMI for NMF of rank 5 = 0.8817180405938404`
	`NMI for NMF of rank 14 = 0.8237431111008482`
	`NMI for NMF of rank 20 = 0.8546663964670154`
	`NMI for NMF of rank 32 = 0.8292075247628473`
	`NMI for NMF of rank 40 = 0.808934819743111`

Table 4: Output of the test on the 3 runs

We can see a consistent pattern here is that the GKL NMF with rank 40 topped the test for the whole 3 runs. The PCA clustering is the worst of for having worst points 2 times out of 3.

We can also see that when we increase the rank for GKL NMF, the points are generally getting better. A more detailed investigating about why GKL is a good choice for $\text{word} \times \text{document}$ data is described here [6].

5 Bibliography

[1] D. Skillicorn, "Non-Negative Matrix Factorization (NNMF)," in *Understanding Complex datasets: data mining with matrix decompositions*, Chapman & Hall/CRC, 2007, p. 173.

[2] K. Lang, "Jason Rennie," 1995. [Online]. Available: http://qwone.com/~jason/20Newsgroups/. [Accessed 10 February 2019].

[3] V. PaulPauca, J. Piper, Robert J. Plemmons, "Nonnegative matrix factorization for spectral data analysis," *ScienceDirect*, vol. 416, no. 1, pp. 29-47, 2005.

[4] C. R. Aberger, "Recommender : An Analysis of Collaborative Filtering Techniques," *Semantic Scholar*, 2014.

[5] Nicholas Bryan, Dennis Sun, "Nicholas J. Bryan," 09 April 2013. [Online]. Available: https://ccrma.stanford.edu/~njb/teaching/sstutorial/part2.pdf. [Accessed 20 February 2019].

[6] Fernando Pereira, Naftali Tishby, Lillian Lee, "Distributional clustering of English words," *ACM*, pp. 183-190, 1993.

6 Appendix

Full source code and test results can be found here: https://github.com/envil/nmf-newsgroups