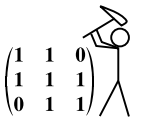


You must hand in a machine-typed report in PDF format and a script file (a `.R` *file). The report must explain your approach to the problem, the results you obtained, and your interpretation of the results. Naturally, the report must also answer to any direct question presented in the problems. You can, and in many cases should, add plots and other illustrations to your answers. The script file must show every step you have taken to solve these tasks, that is to say, if we run the script file we must get the same results you reported and see the same figures you presented. You can discuss these problems with other students, and you are encouraged to discuss with the tutor, but everybody must hand in their own answers and own code. Return your answers by email to* `smetzler@mpi-inf.mpg.de`*. Remember to write your name and matriculation number to every answer sheet!*

Task 1: ALS vs. multiplicative NMF

Download the data and utility files from `http://resources.mpi-inf.mpg.de/departments/d5/teaching/ss17/dmm/assignments/assignment_2.zip`. That package contains file `assignment2.R`. You can fill your answers to that file and return it as a part of your solution.

Your first task is to implement three versions of the NMF algorithm:

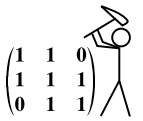
1. NMF based on alternating least squares
2. Lee and Seung's multiplicative NMF algorithm
3. NMF via gradient descent using Oblique Projected Landweber (OPL) updates

You can truncate negative values to zero. Your implementations should be reasonably efficient.

Load the `news` data. It is a sample of 2000 news articles of the 20-newsgroups dataset (`http://qwone.com/~jason/20Newsgroups/`). Terms have been stemmed and very frequent and infrequent words have been removed. The data is given in form of an 2000×5136 document-term matrix; entry (d, w) denotes the term frequency (tf) of word w in document d.

Run the three NMF algorithms on the `news` data. Compare the reconstruction errors and convergence rates. Notice that any two runs of the algorithm might result to very different outcomes, depending on the initial $\boldsymbol{W}$ and $\boldsymbol{H}$. Also, the default 300 iterations might not be enough (or it might be too much) for the methods to converge. Play around with the number of re-starts and iterations. Based on your experiments, which one of the three methods you consider better for this data and why?

Hint: The `news` data is reasonably large. We advise you to start early enough with solving the assignment as the computations need some time to run.



Task 2: Analysing the data

In this task we try to analyse the `news` data. Before proceeding further, normalize the data such that the sum of all entries in the data equals 1. Then use one of the methods you implemented in the first task to find $k = 20$ NMF of the data and study the top-10 terms of the right factor matrix $\boldsymbol{H}$. Can you infer some "topics" based on these terms? Recall that the terms are stemmed. The topics can be very broad (e.g. "terms associated with sports") and they might not be the ones of the newsgroups. Also, some factors might not correspond to any sensible topic. Argue why (or why not) you think the factors correspond to the topics you claim they do.

Repeat the analysis with $k = 5, 14, 32, 40$. How do the results change with increased k? Can you name the single best rank for this data?

Repeat the analysis, but this time using the generalized K–L divergence optimizing version of NMF (provided in `utils.R`). Do the results change? Are they better or worse? Is a different k better with K–L divergence than with Euclidean distance?

Task 3: Clustering and pLSA

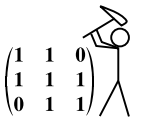
In this task, we study the use of pLSA as a dimensionality reduction tool, and compare it to Karhunen–Lóeve transformation. We use the (normalized) `news` data. The documents of the data came from 20 newsgroups. Your task is to cluster the documents in such a way that the clusters correspond to the newsgroups. To evaluate the quality of the clustering, we use normalized mutual information (NMI).[1] NMI takes values from $[0, 1]$ and obtains value 0 for perfect match. Notice that NMI does not care about cluster labels or the ordering of the clusters. You can use the provided function `nmi.news` to evaluate a clustering (see provided example in `assignment2.R`).

To compute the pLSA, first compute the K–L divergence optimizing NMF (of the normalized data), and then use the provided script to normalize $\boldsymbol{W}$ and $\boldsymbol{H}$ to obtain decomposition of type $\boldsymbol{W}'\boldsymbol{\Sigma}\boldsymbol{H}'$. Use also another script to compute normalization of type $\boldsymbol{\Sigma}\boldsymbol{W}'\boldsymbol{H}'$, where $\boldsymbol{\Sigma}$ is n-by-n diagonal.

Cluster the normalized newsgroup data into 20 clusters using each of the methods below and compute the NMI. Try different ranks for the matrix factorizations. Which clustering(s) perform well, which do not? Why?

1. k-means,
2. k-means on the first k principal components (Karhunen–Lóeve transform, similar to Task 3 of Assignment 1),
3. k-means on the $\boldsymbol{W}$ matrix of the NMF (using K–L divergence),
4. k-means on the $\boldsymbol{W}'$ matrix of factorization $\boldsymbol{W}'\boldsymbol{\Sigma}\boldsymbol{H}'$ obtained from the NMF, and
5. k-means on the $\boldsymbol{W}'$ matrix of factorization $\boldsymbol{\Sigma}\boldsymbol{W}'\boldsymbol{H}'$ obtained from the NMF.

[1] Strictly speaking, we are using the normalized metric variant $D(X, Y)$, `http://en.wikipedia.org/wiki/Mutual_information#Metric`



Task 4: Competition (optional)

This task is fully optional; not doing it will not affect your grade in any way, and you can achieve an excellent grading without doing it. Doing it, however, will allow you to learn much more about NMF and will give you an alternative way of obtaining an excellent grade from this assignment.

Your task is to implement (and design) an algorithm for NMF that is fast and reasonably accurate. Implement your algorithm in the separate file `competitionNMF.R`. Your algorithm must be called `nmf` and it must expect exactly two arguments: a nonnegative matrix $\boldsymbol{A}$, and a positive integer k (the size of the decomposition). The algorithm must return a list with two elements, $\boldsymbol{W}$ and $\boldsymbol{H}$, containing the left and right nonnegative factor matrices, respectively. You do not have to validate the correctness of the input (e.g. that $\boldsymbol{A}$ is nonnegative), but your algorithm must work with any n-by-m nonnegative $\boldsymbol{A}$ with $\min(n, m) > 1$ and $\max(n, m) < 10000$ and with any k that is between 1 and $\min(n, m)$. You must implement your algorithm using only basic R operations: you cannot call external code or programs, and you cannot load any library except `Matrix`. You can use any of the algorithms from Task 1 as the baselines, or use any other NMF algorithm (in the latter case, provide a short justification as to why your code works).

Eligibility: You are eligible for the competition if you solve the mandatory parts of this assignment in at least passing level and if your code admits the above guidelines. Your code must also compute the decomposition within a reasonable time frame (to be decided by the judges) to be considered for the evaluation.

Evaluation: We will run your code with a matrix that is another sample from the 20 Newsgroups data. The algorithms are executed on a dedicated system with R version 3.4. The algorithms that finish in reasonable time are ranked based on the squared Frobenius error of their solution. All algorithms that have error that is within 5% of the best submission qualify for the next evaluation step. The winner is the qualifying algorithm that had the fastest wall-clock running time when run on the evaluation data.

Price: The winning submission will receive an excellent grade from this assignment. The Jury withholds the right to award the excellent grade for multiple submissions if they are found to perform essentially similar, or to none, if no eligible submissions are made.