UNIVERSITY OF EASTERN FINLAND

You must hand in a machine-typed report in PDF format and a script file (a `.R` file for R, a `.m` file for Matlab, or a `.py` file for Python). 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 anybody runs the script file they 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.

Only reports in PDF format are accepted (no doc[x], eps, ps, jpg, txt, rtf, otf, tiff, png, or other files types). The source code must be self-contained file made with Python (suffix `.py`), Matlab (suffix `.m`), or R (suffix `.R`). The source code files must be encoded in ASCII or UTF-8, provided that the software supports it. You can include libraries that can be obtained from standard sources (Mathworks toolboxes licensed by UEF for Matlab, CRAN for R and installable with PyPi in Python; numpy, scipy, and scikit-learn are also allowed libraries with Python). All other code must be included. If you have copies some code from the internet (e.g. preprocessing script from Stackexchange), you must write a comment in your source code where the code snippet is from, including the URL and the name of the author of the code. You can naturally only copy small snippets.

Return your report and source code to Moodle. Remember to write your name and student ID number to every page of the report and to the top of your source code file.

Task 1: ALS vs. multiplicative NMF

Download the data (`news.csv`) from the course Moodle page. There you can also find three template files, called `assignment.{m, py, R}`. You can use these as templates for your answers, and you can also find some helper functions and boilerplate code in there.

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

a) NMF based on alternating least squares

b) Lee and Seung's multiplicative NMF algorithm

c) NMF via gradient descent using Oblique Projected Landweber (OPL) updates

You can truncate negative values to zero. Your implementations should be reasonably efficient. As a convergence criterion, you can stop after 300 iterations. For initial solutions, you can use either matrices where elements are sampled uniformly at random form the unit interval, or use scaling based on the input matrix. But you have to use the same initial solution generation for all methods.

The data is a sample of 2000 news articles of the 20-newsgroups dataset.[1] 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 for $k = 20$. 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.

Analyse the convergence speed of the algorithms. Use either the number of iterations the algorithms take to reach error below some reasonable threshold (e.g. error that is less than 95 % of the best error you have got), or the wall-clock time it takes for them to reach that level. Is one of the methods clearly better than the other? Can you show statistically significantly faster (in iterations or in wall-clock time) convergence times? Give at least one plot of convergence rates (iteration or time vs. reconstruction error) for each method.

Compare also the best reconstruction errors. Does any of the methods give statistically significantly lower reconstruction errors over different re-starts? In all these tests, use an appropriate statistical test.

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. It is advisable to start early enough with solving the assignment as the computations need some time to run. It's also a good idea to start with a smaller sample to make sure your code actually works. Make sure you use the same computer if you do any wall-clock tests.

[1] `http://qwone.com/~jason/20Newsgroups/`

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 $\boldsymbol{H}$. Recall that the columns of the data correspond to the terms that are the column names in the csv file and the top-10 terms correspond to the columns in a row of $\boldsymbol{H}$ that have the ten highest values.

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 (for this, you can either use an existing implementation or do your own). 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. For pLSA, we use the normalized `news` data from the previous task; for Karhunen–Lóeve, you have to first normalize the data to z-scores. The documents of the data came from 20 newsgroups and we will use $k = 20$ factors in NMF/pLSA and in Karhunen–Lóeve. Our aim 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).[2] This 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.

The template files have functions to compute the NMI. You will also need the file `news_ground_truth.txt` from Moodle. Please remember that you must not use this file to guide your clustering, only to evaluate the results.

To compute the pLSA, first compute the K–L divergence optimizing NMF (of the normalized data), and then normalize the columns of $\boldsymbol{W}$ to sum to one. To compute the Karhunen–Lóeve-transform (or PCA), normalize the data to z-scores, compute the SVD of the data (using existing implementations) and then do the transformation e.g. using the equation from slide set 4, slide 19.

Cluster the normalized newsgroup data into 20 clusters using each of the methods below and compute the NMI. You can use existing implementations for k-means, but do re-start it multiple times and take the best solution. Try different ranks for the matrix factorizations. Which clustering(s) perform well, which do not? Why?

a) k-means

b) k-means on the first k principal components

c) k-means on the $\boldsymbol{W}$ matrix of the NMF (using K–L divergence)

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