**CSCI 1952Q: Algorithmic Aspects of Machine Learning (Spring 2024)**

# Coding Assignment 2

Due at 1:00pm ET, Friday, Mar 8

**Getting Started.**

- You can use any programming language for this coding assignment.
- You cannot use packages or functions that directly solve the problem.

**Overview.** In this assignment, you will use Nonnegative Matrix Factorization (NMF) for topic modeling. You will be given a word-word correlation matrix $M$ obtained from a real-world dataset, where $M_{i,j}$ is proportional to the empirical probability that word $i$ and word $j$ co-occurring in a document. Your goal is to find nonnegative matrices $A$ and $W$ such that $M \approx AW$.

You cannot use packages like `stats/nnmf` in Matlab or `sklearn.decomposition.NMF` in Python.

**Input.** You will be given an input file `word_word_correlation`, which specifies an $m \times m$ matrix and the desired factorization rank $r$. The first line of this file has two integers: $m$ and $r$. This is followed by an $m \times m$ matrix described in $m$ lines, each containing $m$ nonnegative real numbers.

You must use the specified factorization rank $r$.

**Output.** The output file should have $m + r$ lines, specifying your solutions: a matrix $A \in \mathbb{R}_{\geq 0}^{m \times r}$ followed by a matrix $W \in \mathbb{R}_{\geq 0}^{r \times m}$. Each of the first $m$ lines should have $r$ nonnegative real numbers and each of the next $r$ lines should have $m$ nonnegative real numbers, separated by a single space.

**Submission.**

- Your submission should consist of exactly 3 files:
  1. An output file `nmf_ans` in the specified format.
  2. A text file (e.g., `.cpp`, `.py`) containing your source code.
  3. A `.pdf` file providing a detailed explanation of your approach.
- We may ask you to show us that running the submitted code does produce the submitted output file.

**Evaluation.** Let $M$ be the input matrix. Let $A$ and $W$ be the output matrices.

The loss of your solution is defined as

$$L = \|M - AW\|_F^2 .$$

**Grading.** This assignment will be graded out of 6 points:

- (1 point) Your code should have good readability and should be well commented.
- (1 point) Your explanation `pdf` must be typed (e.g., MS Word or LaTeX). You should give an overview of your ideas and approach in the first 2 pages. Material beyond the first 2 pages will be read at the discretion of the instructor/TAs.
- (4 points) You will receive a score of $(5 - \frac{L}{300})$ where $L$ is your loss. If the score is lower than 0 or higher than 4, it is set to 0 or 4. In particular, you will receive full credit if your loss is 300 or lower.
- (1 bonus point) you will receive 1 bonus point if your loss is among the smallest 20% of all received submissions.
- We may deduct up to 4 points for any formatting error in your output (including but not limited to, not naming the output file `nmf_ans`, not outputting exactly $m + r$ lines, not outputting exactly $2mr$ numbers, or having negative values in your output).

**Dataset.** The input file was obtained from the WikiText Dataset introduced in [MXBS17]. Specifically, the `WikiText-103` word level dataset was used [1]. This dataset contains 28592 articles (with over 103 million tokens) selected from verified Good and Featured articles on Wikipedia [2].

For this assignment, we processed the `WikiText-103` dataset as follows: We converted all letters to lowercase and removed stopwords (e.g., the, my, is) [3] and tokens with non-alphabetic characters (e.g., I-95). We then extracted the most frequent $m = 2000$ words and removed all other words.

The input matrix $M$ is initialized to 0. For each article, we add the following to $M_{i,j}$:

$$\Pr[\text{a uniformly random chosen pair of words} = (\text{word}_i, \text{word}_j)] \ .$$

The matrix $M$ is not normalized by the number of articles in the end, as this would only scale down every input number by 28592.

**Remarks/Hints.** You are free to use any algorithms for NMF. Due to this reason, the following hints may not apply to your solution.

- One could use alternating minimization by repeating these steps:
  (1) Fix $W$ and find $A \in \mathbb{R}_{\geq 0}^{m \times r}$ that minimizes $\|M - AW\|_F$.
  (2) Fix $A$ and find $W \in \mathbb{R}_{\geq 0}^{r \times m}$ that minimizes $\|M - AW\|_F$.
- One could initialize $A$ and $W$ using the SVD of $M$. Suppose $M = U\Sigma U^\top$. A simple approach is to consider $A = U\Sigma^{1/2}$ and $W = \Sigma^{1/2}U^\top$ and then change all negative entries to 0.
- One could add regularizers to the objective functions.
- While you cannot use functions that solve NMF, you can study their implementations and then independently write your code. For example, `stats/nnmf` in Matlab refers to [BBL+07].

[1] See `https://blog.salesforceairesearch.com/the-wikitext-long-term-dependency-language-modeling-dataset/`, available under the Creative Commons Attribution-ShareAlike License.

[2] See `https://en.wikipedia.org/wiki/Wikipedia:Good_articles` and `https://en.wikipedia.org/wiki/Wikipedia:Featured_articles`.

[3] We used the stopword list from the MALLET topic model package `https://mimno.github.io/Mallet/index`.

**Optional Tasks.** After computing the NMF, you can explore the following questions. There are no bonus points for these tasks.

- How can we compute the word-by-topic matrix $A'$? Recall that $A'_{j,i} = \Pr[\text{word}_j \mid \text{topic}_i]$.
- What are the most frequent words for each topic? Can you infer the topics from these words? The index-to-word mapping is provided in the file `list_of_words`. (If you did not compute $A'$, you can try examining $A$ or a normalized version of $A$.)
- Does $A$ satisfy the separability assumption? If yes, what are the anchor words for each topic?
- Compute the word-by-document matrix $M'$ from the dataset. What if we directly compute an NMF of $M'$?
- Which topics appear most frequently? Which topics tend to co-occur in an article?
- Which articles are similar to each other (based on topic modeling)?

## References


[BBL+07] M. W. Berry, M. Browne, A. N. Langville, V. P. Pauca, and R. J. Plemmons. Algorithms and applications for approximate nonnegative matrix factorization. *Comput. Stat. Data Anal.*, 52(1):155–173, 2007.

[MXBS17] S. Merity, C. Xiong, J. Bradbury, and R. Socher. Pointer sentinel mixture models. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*. OpenReview.net, 2017.