

## CSCI 1520: Algorithmic Aspects of Machine Learning (Spring 2025)

# Coding Assignment 2

Due at 11:59pm ET, Thursday, Mar 20

### Getting Started.

- You can use any programming language for this coding assignment.
- You cannot use packages or functions that directly solve the problem (e.g., `stats/nmf` in Matlab or `sklearn.decomposition.NMF` in Python for this assignment).

**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-occur in a document. Your goal is to find nonnegative matrices  $A$  and  $W$  such that  $M \approx AW$ .

**Input.** You will be given an input file `word_word_correlation`, which contains 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.

Note that the value of  $r = 20$  is fixed. You cannot choose  $r$ .

**Output.** The output file should have  $m + r$  lines, specifying your solution: 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 14 points:

- (3 points) Your code should have good readability and should be well commented.
- (3 points) 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.
- (8 points) You will receive a score of  $(10 - \frac{L}{150})$  where  $L$  is your loss. If the score is lower than 0 or higher than 8, it is set to 0 or 8. In particular, you will receive full credit if  $L \leq 300$ .
- (2 bonus points) You will receive 2 bonus points if your loss is among the smallest 20% of all received submissions.
- We may deduct up to 8 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 <sup>1</sup>. This dataset contains 28592 articles (with over 103 million tokens) selected from verified Good and Featured articles on Wikipedia <sup>2</sup>.

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) <sup>3</sup> and tokens with non-alphabetic characters (e.g., I-95). We then kept the most frequent  $m = 2000$  words and removed all other words.

The input matrix  $M$  is initialized to 0. For each article, we updated  $M_{i,j}$  for all  $(i, j)$  as follows:

$$M_{i,j} \leftarrow M_{i,j} + \Pr[\text{a uniformly randomly chosen pair of words in this article} = (\text{word}_i, \text{word}_j)] .$$

The matrix  $M$  is not divided by the number of articles in the end, as this would only make each entry 28592 times smaller.

**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 possible approach is to use alternating minimization, by repeating the following steps:
  - (1) Fix  $W$  and solve for  $A \in \mathbb{R}_{\geq 0}^{m \times r}$  that minimizes  $\|M - AW\|_F$ .
  - (2) Fix  $A$  and solve for  $W \in \mathbb{R}_{\geq 0}^{r \times m}$  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 set  $A = U\Sigma^{1/2}$  and  $W = \Sigma^{1/2}U^\top$ , and then change all negative entries to 0.
- One could add regularizers to the objective functions.
- Although you cannot use functions that solve NMF, you can study their implementations and then independently write your code. For example, **stats/nmf** in Matlab cites [BBL<sup>+</sup>07].

---

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

<sup>2</sup>See [https://en.wikipedia.org/wiki/Wikipedia:Good\\_articles](https://en.wikipedia.org/wiki/Wikipedia:Good_articles) and [https://en.wikipedia.org/wiki/Wikipedia:Featured\\_articles](https://en.wikipedia.org/wiki/Wikipedia:Featured_articles).

<sup>3</sup>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 open-ended questions. There are no bonus points for these optional 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 happens if we directly apply NMF to  $M'$ ?
- 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<sup>+</sup>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.