CSCI 1952-Q: Algorithmic Aspects of Machine Learning (Spring 2023)

Coding Assignment 2

Due at 2:30pm ET, Monday, Apr 24

Getting Started.

- You are free to use any programming language of your choice.
- You may use resources on the web and you are encouraged to discuss with other students. You must write your code independently and acknowledge who you discussed with and what resources you used. Failure to provide such acknowledgment is considered a violation of academic integrity.

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 related to the probability that word i and word j appear together in a document. Your goal is to find two nonnegative matrices A and W such that $M \approx AW$.

You cannot use functions/packages that directly solve NMF (e.g., `stats/nnmf` in Matlab or `sklearn.decomposition.NMF` in Python).

Input. You are given an input file `word_word_correlation`. This input file specifies an $m \times m$ matrix and the desired rank r of the factorization. The first line of this file contains two integers m and r. This is followed by m lines, where each line contains m nonnegative real numbers.

Note that you cannot choose the rank of the factorization. You must use the specified 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. Each of the next r lines should have m nonnegative real numbers.

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, and
  3. a `pdf` file containing a detailed explanation of your approach.
- We may ask you to show us that running the submitted code does produce the submitted output file.

Loss. Let M be the input matrix. Let A and W be the matrices in your output.

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 your approach in the first 2 pages. Material beyond the first 2 pages will be read at the sole discretion of the instructor/TAs.
- (4 points) You will get a score of $(5 - \frac{L}{300})$ where L is the loss of your solution as defined earlier. 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 $m + r$ lines, not outputting $2mr$ numbers, or having negative numbers in your output).

Dataset. The input file was calculated from the WikiText Long Term Dependency Language Modeling Dataset [MXBS17]. More specifically, the `WikiText-103` word level dataset was used [1]. We merged the training, validation, and test datasets. This dataset contains 28592 articles (with over 100 million tokens) chosen from the set of verified Good and Featured articles on Wikipedia [2].

For this assignment, we processed the `WikiText-103` dataset as follows. First we converted all letters to lowercase and selected the most frequent $m = 2000$ words in all articles, ignoring 524 stopwords (e.g., "the", "my", "is") [3] and all tokens that contain non-alphabetic characters (e.g., "I-95"). We kept only these m words and removed all other words from the articles.

The input matrix M is initialized to 0, and then for every article, we add the following to $M_{i,j}$:

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

We did not normalize M by the number of articles in the end (because it would only make every input number 28592 times smaller).

Remarks/Hints. You are free to choose which algorithms to implement for NMF. Due to this reason, the following hints may not apply to your solution.

- One could use alternating minimization by repeating the following steps:

  (1) Fix W. Find $A \in \mathbb{R}_{\geq 0}^{m \times r}$ that minimizes $\|M - AW\|_F$.

  (2) Fix A. Find $W \in \mathbb{R}_{\geq 0}^{r \times m}$ that minimizes $\|M - AW\|_F$.

[1] See `https://blog.salesforceairesearch.com/the-wikitext-long-term-dependency-language-modeling-dataset/`. This dataset is 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`.

- 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/packages that directly solve NMF, you can take a look at their implementations to see what algorithms they use. For example, `stats/nnmf` in Matlab includes a reference to [BBL+07].

Optional Tasks. After computing the NMF, you can explore the following questions on this dataset. There are no bonus points for answering these questions.

- How can we compute the word-by-topic matrix A'? Recall that $A'_{j,i} = \Pr[\text{word}_j \mid \text{topic}_i]$.
- Which words occur most frequently for each topic? Can you guess the topics based on these words? (If you did not do the previous step, you can simply look at A or a row-normalized version of A.) The index-to-word mapping is provided in the file `list_of_words`.
- Does A satisfy the separability assumption? If so, what are the anchor words for each topic?
- Compute the word-by-document matrix M' and try to write it as $A'W'$. How does this compare to directly computing an NMF for M'?
- Which topics appear most frequently? Which topics often appear together 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.