Exercises on NMF

Alex Williams, 09/05/2017

Notation.

Nonnegative matrix factorization (NMF) considers a matrix dataset $\mathbf{X}$ with dimensions $m \times n$, where all datapoints are nonnegative, i.e.

$$x_{ij} \geq 0, \quad \forall i \in \{1, ..., m\},\ j \in \{1, ..., n\}\,.$$

For brevity, we use $\mathbf{X} \geq 0$ to denote the above condition (this is a common shorthand). A NMF model with r components attempts to solve the following optimization problem:

$$\begin{aligned} \underset{\mathbf{U},\mathbf{V}}{\text{minimize}} \quad & \|\mathbf{X} - \mathbf{U}\mathbf{V}^T\|_F^2 \\ \text{subject to} \quad & \mathbf{U} \geq 0,\, \mathbf{V} \geq 0 \end{aligned}$$

where $\mathbf{U}$ and $\mathbf{V}$ are respectively $m \times r$ and $n \times r$ matrices, where $r < m$ and $r < n$ (to reduce dimensionality). We refer to $\mathbf{U}$ and $\mathbf{V}$ as *factor matrices.*

Exercises.

1. Implement NMF by alternating nonnegative least squares (ANLS). For simplicity, run the algorithm for 30 iterations (you don't need to check for convergence). The update equations for ANLS are:

$$\mathbf{U}_{k+1} \leftarrow \underset{\mathbf{U} \geq 0}{\text{argmin}}\ \|\mathbf{X} - \mathbf{U}\mathbf{V}_k^T\|_F^2$$

$$\mathbf{V}_{k+1} \leftarrow \underset{\mathbf{V} \geq 0}{\text{argmin}}\ \|\mathbf{X} - \mathbf{U}_{k+1}\mathbf{V}^T\|_F^2$$

   where k indexes the iterations. In MATLAB, you can use the `lsqnonneg` function. In Python, you can use the `nnls` function in `scipy.optimize`. Initialize the factor matrices with uniform, random nonnegative numbers.

2. Test your implementation on the synthetic dataset provided to you in `nmfdata.txt`. Use the same data to answer all the questions below.

3. *Scree Plot.* Implement a function that plots the root-mean-square-error (RMSE) of the model as a function of the number of components, r. For each value of r, fit the model from multiple random initializations and plot the RMSE as a separate point. For the provided dataset, is the RMSE sensitive to initialization?

4. *Comparison to truncated SVD.* Modify your scree plot to include a line that plots the RMSE of a truncated SVD/PCA model as a function of r. Compare the performance of NMF to SVD for the synthetic dataset. When you generate this plot from the provided dataset, is the result favorable or unfavorable for NMF?

5. *Similarity Plot.* Implement a function that computes the similarity of two factor matrices. Specifically, your function will take two factor matrices $\mathbf{U}$ and $\mathbf{U}'$ with the same number of components, r, but fit

from different initializations. First, normalize the columns of $\mathbf{U}$ and $\mathbf{U}'$ to be unit Euclidean length. Then, compute the following similarity score between the two factor matrices:

$$\max_{\Pi} \; \frac{1}{r} \operatorname{Tr}\left[\mathbf{U}^T \mathbf{U}' \Pi\right] \, .$$

Here, $\operatorname{Tr}[\cdot]$ denotes the trace of a matrix, and Π is an $r \times r$ permutation matrix. In other words, search over the all permutations of the columns of $\mathbf{U}'$ and return the maximal average cosine similarity with the columns of $\mathbf{U}$. Then, plot the average model similarity as a function of the number of components. After computing and visualizing the similarity for $\mathbf{U}$, as described above, do the same for $\mathbf{V}$ (this should require essentially no extra code).

6. *Visualization and interpretation.* Assume that the rows and columns of the data are in arbitrary order. For example, let's say we're working with gene expression data where each column of $\mathbf{X}$ is a different gene, and each row of $\mathbf{X}$ is a different biological sample (e.g., a tumor derived from a different patient). Implement a function that re-sorts the samples and genes based on the NMF model. Visualize the raw data and the re-sorted data as heatmaps. For the provided synthetic dataset, it should be possible to re-sort the data to reveal clustering within genes and samples.