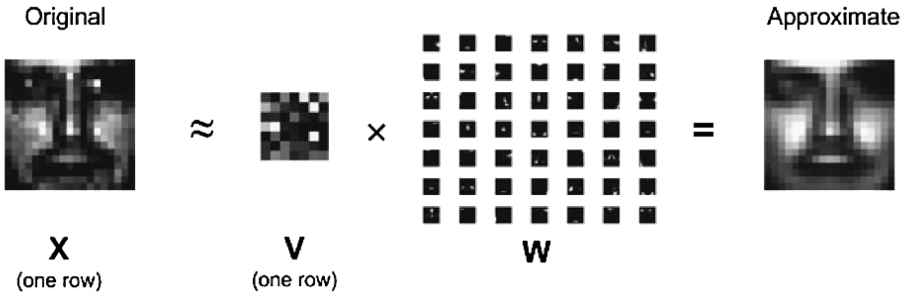
ST790-003, Homework 2 (updated Jan 29 @ 5pm)

Due Wednesday, Feb 11, 2015 @ 11:59PM

Nonnegative Matrix Factorization

Nonnegative matrix factorization (NNMF) was introduced by Lee and Seung (1999, 2001) as an analog of principal components and vector quantization with applications in data compression and clustering. In this homework we consider algorithms for fitting NNMF and high performance computing using graphical processing units (GPUs).



This is a *solo* homework. Discussion with fellow students is allowed (actually encouraged) but you have to write your code and report independently.

1. In mathematical terms, one approximates a data matrix $\boldsymbol{X} \in \mathbb{R}^{m \times n}$ with nonnegative entries x_{ij} by a product of two low-rank matrices $\boldsymbol{V} \in \mathbb{R}^{m \times r}$ and $\boldsymbol{W} \in \mathbb{R}^{r \times n}$ with nonnegative entries v_{ik} and w_{kj}. Consider minimization of the squared Frobenius norm

$$L(\boldsymbol{V}, \boldsymbol{W}) = \|\boldsymbol{X} - \boldsymbol{V}\boldsymbol{W}\|_{\mathrm{F}}^2 = \sum_i \sum_j (x_{ij} - \sum_k v_{ik} w_{kj})^2, \quad v_{ik} \geq 0, w_{kj} \geq 0,$$

which should lead to a good factorization. It is possible to construct a block descent algorithm that hinges on the majorization

$$\left(x_{ij} - \sum_k v_{ik} w_{kj} \right)^2 \leq \sum_k \frac{a_{ikj}^{(t)}}{b_{ij}^{(t)}} \left(x_{ij} - \frac{b_{ij}^{(t)}}{a_{ikj}^{(t)}} v_{ik} w_{kj} \right)^2,$$

where

$$a_{ikj}^{(t)} = v_{ik}^{(t)} w_{kj}^{(t)}, \quad b_{ij}^{(t)} = \sum_k v_{ik}^{(t)} w_{kj}^{(t)},$$

and superscript t indicates the current iteration. Prove this majorization and derive the alternating multiplicative updates

$$v_{ik}^{(t+1)} = v_{ik}^{(t)} \frac{\sum_j x_{ij} w_{kj}^{(t)}}{\sum_j b_{ij}^{(t)} w_{kj}^{(t)}}$$

and

$$w_{kj}^{(t+1)} = w_{kj}^{(t)} \frac{\sum_i x_{ij} v_{ik}^{(t)}}{\sum_i b_{ij}^{(t)} v_{ik}^{(t)}}.$$

2. Implement the algorithm using your preferred language (R, Matlab, Julia, Python, or others). Your function should take arguments $\boldsymbol{X}$ (data), rank r, convergence tolerance, and optional starting point.

3. Database #1 from the MIT Center for Biological and Computational Learning (CBCL) (`http://cbcl.mit.edu`) reduces to a matrix X containing $m = 2,429$ gray-scale face images with $n = 19 \times 19 = 361$ pixels per face. Each image (row) is scaled to have mean and standard deviation 0.25.

   Read in the `nnmf-2429-by-361-face.txt` file and display a couple sample images.

4. Report the run times (on the teaching server) of your function for fitting NNMF on the MIT CBCL face data set at ranks $r = 10, 20, 30, 40, 50$. For ease of comparison (and grading), please start your algorithm with the provided $\boldsymbol{V}^{(0)}$ (first r columns of `V0.txt`) and $\boldsymbol{W}^{(0)}$ (first r rows of `W0.txt`) and stopping criterion

$$\frac{|L^{(t+1)} - L^{(t)}|}{|L^{(t)}| + 1} \leq 10^{-4}.$$

5. Choose an $r \in \{10, 20, 30, 40, 50\}$ and start your algorithm from a different $\boldsymbol{V}^{(0)}$ and $\boldsymbol{W}^{(0)}$. Do you obtain the same objective value and $(\boldsymbol{V}, \boldsymbol{W})$ as in Q4? Explain what you find.

   For the same r, start your algorithm from $v_{ik}^{(0)} = w_{kj}^{(0)} = 1$ for all i, j, k. Do you obtain the same objective value and $(\boldsymbol{V}, \boldsymbol{W})$ as in Q4? Explain what you find.

6. Investigate the GPU capabilities of your language and implement a GPU version of your algorithm. Report the speed gain of your GPU code over CPU code at ranks $r = 10, 20, 30, 40, 50$. Make sure to use the same starting point as in Q4.

7. Plot the basis images (rows of $\boldsymbol{W}$) at rank $r = 50$. What do you find?

References

Lee, D. D. and Seung, H. S. (1999). Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791.

Lee, D. D. and Seung, H. S. (2001). Algorithms for non-negative matrix factorization. In *NIPS*, pages 556–562. MIT Press.