In the name of God
Optimization Fundamentals
Final Project; Deadline: Bahman 18

Nonnegative matrix factorization with gradient descent

Let $\mathbf{V} \in \mathbb{R}^{m \times n}$ be a given matrix, and $0 < k < min(m, n)$ be a given positive integer.

The goal of nonnegative matrix factorization (NMF) is to solve

$$J = \min_{\mathbf{W} \geq 0, \mathbf{H} \geq 0} \|\mathbf{V} - \mathbf{W}\mathbf{H}\|_F^2 \tag{1}$$

where $\mathbf{W} \in \mathbb{R}^{m \times k}$, $\mathbf{H} \in \mathbb{R}^{k \times n}$, and $\mathbf{A} \geq 0$ denotes that $A_{ij} \geq 0$ for all i, j.

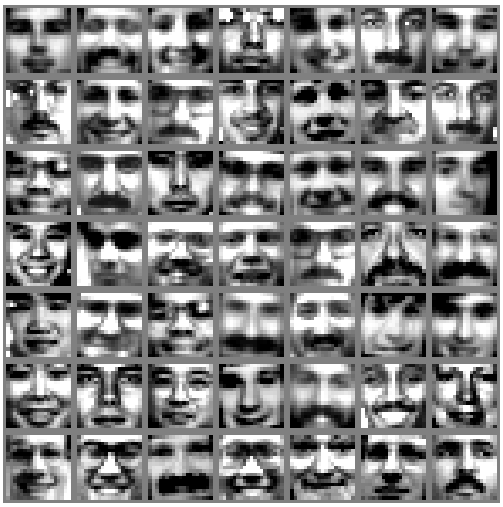
The NMF problem is a nonconvex problem, and therefore it is difficult to solve. However, if we fix $\mathbf{W}$, then the problem is convex in $\mathbf{H}$, and similarly if $\mathbf{H}$ is fixed, then the problem is convex in $\mathbf{W}$. Therefore a commonly used approach is to do alternating minimization: Fix $\mathbf{W}$, optimize J in $\mathbf{H}$, then fix $\mathbf{H}$ optimize J in $\mathbf{W}$ and repeat this process until convergence.

Interestingly, there is an elegant gradient descent algorithm (GDA) for this approach where the learning rate can be set such that GDA always generates feasible solutions, i.e. after the updates the $\mathbf{W}$ and $\mathbf{H}$ matrices remain nonnegative. The update rule is given in this paper Lee & Seung (2001) `http://hebb.mit.edu/people/seung/papers/nmfconverge.pdf` . More information about NMF can be found in `http://www.nature.com/nature/journal/v401/n6755/abs/401788a0.html` .

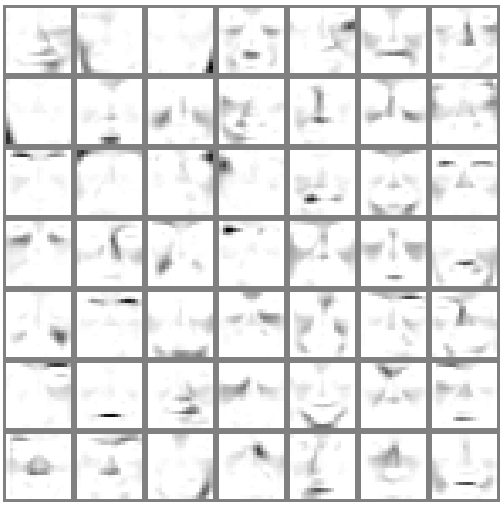
Your task is to implement this NMF algorithm on the CBCL FACE database `http://www.ai.mit.edu/projects/cbcl.old/software-datasets/FaceData2.html` . A sample of the original faces look like

(a) Take the original faces from face.txt , apply the following preprocessing:

For every face that is stored in a column of $\mathbf{V}_i$, normalize it such that the median $\mathbf{m}_i = 0.5$ and the median of the deviation from the median is 0.25, i.e. $\underset{j}{median}(|\mathbf{V}_{ji} - \mathbf{m}_i|)$=0.25. Truncate values above 1 or below $1e-4$ (because we do not want exact zeros either). Now, we have a $\mathbf{V}$ that looks like (sampled columns):



(b) Apply the algorithm Eq(4) in the paper Lee&Seung (2001) to optimize (1). $k = 49$. Initialize every entry in your $\mathbf{W}$ and $\mathbf{H}$ as a random draw from uniform(0,1). Run 300 iterations. Plot the columns of $\mathbf{W}$ similar to the bellow. Also plot the objective value in every iteration. What columns of $\mathbf{W}$ are more important?



(c) Show, following the arguments in Lee & Seung (2001), that the algorithm you have implemented is indeed performing gradient descent steps, with a cleverly chosen step size that ensures the matrices W, H will have nonnegative entries at each step. What is an alternative, still using gradient descent, to ensure that the entries of W, H are nonnegative?

(d) Now load face_test.txt , reconstruct the faces by optimizing $\mathbf{H}$ with your learned $\mathbf{W}$ fixed. What is the mean mean-squared-error of each reconstructed images (the objective value normalized by the number of images)? Do your reconstructed faces look similar to test faces?

To submit for (b) and (d) Submit the plots for parts (b) and (d) to answer the questions asked. Please also submit the following functions (or similar functions if you are using another language) in a zip file.

(a) `function [V_normalized] = normalize_faces(V)`
(b) `function [W, H, objective_val_history] = nmf(V_normalized, k)`
(c) `function [objective_val_history, H] = test_faces(W, testV_normalized)`

[Hint: the function compact_canvas.m may come handy to plot the faces. You may find $\mathbf{V}$ and $\mathbf{V}_{test}$ in face.txt and face_test.txt]