NONNEGATIVE MATRIX FACTORIZATION METHODS

RYAN BLAKE STORY

Abstract. In this paper we will discuss Nonnegative Matrix Factorization (NMF), a low-dimensional approximate parts based representation of a data matrix. We will discuss constraints to an orthogonal subspace on such a factorization. Orthogonal constraints ameliorate clustering performance and construct sparse parts-based representations of data.

Contents

1. Nonnegative Matrix Factorization 1
2. Nonnegative Matrix Factorization on Orthogonal Subspace 2
3. Convergence of NMFOS Multiplicative Update Rules 4
4. Sparseness 6
5. Matlab Implementation 7
6. Experiment Results 8
7. Conclusion 9
8. Acknowledgements 10
References 11

1. Nonnegative Matrix Factorization

The motivation behind Nonnegative Matrix Factorization (NMF) is to find a parsimonious representation of a data set to elucidate latent traits. NMF seeks a positive low-dimensional approximation to an original data matrix. Given data consisting of \( n \) measurements of \( p \) positive real valued variables, any \( p \)-dimensional measurement vector \( \mathbf{v}^t (t = 1, \ldots, n) \) can be approximated by

\[
\mathbf{v}^t \approx \sum_{i=1}^{k} \mathbf{w}_i \mathbf{h}_i^t,
\]

where we choose \( k \ll p, n \). We call each \( p \)-dimensional vector \( \mathbf{w}_i \) a basis vector, and each \( k \)-dimensional vector \( \mathbf{h}_i^t \) a coefficient vector or sample with entries \( h_i^t \). The samples describe how the base vectors are fractionally combined to reconstruct an approximation of the original data.
\[
\mathbf{v}^t = \begin{bmatrix}
v_1^t \\
v_2^t \\
\vdots \\
v_p^t
\end{bmatrix} \approx \begin{bmatrix}
w_{11} \\
w_{21} \\
\vdots \\
w_{p1}
\end{bmatrix} h_1^t + \begin{bmatrix}
w_{12} \\
w_{22} \\
\vdots \\
w_{p2}
\end{bmatrix} h_2^t + \ldots + \begin{bmatrix}
w_{1k} \\
w_{2k} \\
\vdots \\
w_{pk}
\end{bmatrix} h_k^t = w_{11} h_1^t + w_{21} h_2^t + \ldots + w_{pk} h_k^t
\]

**Definition 1.1.** Given an input data matrix \( V = [\mathbf{v}^1, \mathbf{v}^2, \ldots, \mathbf{v}^n] \in \mathbb{R}^{p \times n} \) with measurement vectors as columns, Nonnegative Matrix Factorization (NMF) seeks two nonnegative matrices, \( W \in \mathbb{R}^{p \times k} \) with basis vectors as columns and \( H \in \mathbb{R}^{k \times n} \) with samples as columns, such that \( V \approx WH \) where \( k \ll p, n \).

Or more explicitly, we can write \( V \approx WH \) as

\[
\begin{bmatrix}
v_1^1 & v_2^1 & \ldots & v_n^1 \\
v_1^2 & v_2^2 & \ldots & v_n^2 \\
\vdots & \vdots & \ddots & \vdots \\
v_1^p & v_2^p & \ldots & v_n^p
\end{bmatrix} \approx
\begin{bmatrix}
w_{11} & \ldots & w_{1k} \\
w_{21} & \ldots & w_{2k} \\
\vdots & \ddots & \vdots \\
w_{p1} & \ldots & w_{pk}
\end{bmatrix}
\begin{bmatrix}
h_1^1 & \ldots & h_1^n \\
h_2^1 & \ldots & h_2^n \\
\vdots & \ddots & \vdots \\
h_k^1 & \ldots & h_k^n
\end{bmatrix}
\]

Our choice of \( W \) and \( H \) is made in a way that minimizes the error between the reconstruction of the data matrix, \( WH \), and the original data matrix, \( V \). We will consider the NMF optimization problem based on the Euclidean distance metric

\[
\min_{W,H} \| V - WH \|^2 \quad \text{s.t.} \quad W, H \succeq 0,
\]

where \( \| \cdot \| \) is the Frobenius norm, \( \| X \|_F = \sqrt{\text{tr}(XX^T)} \).

Lee and Seung [4] proved the convergence of the objective function under the following multiplicative update rules:

\[
W := W \odot (VH^T) \odot (WHH^T),
\]

\[
H := H \odot (W^TV) \odot (W^TWH),
\]

where \( \odot \) and \( \odot \) are componentwise multiplication and division, respectively. We will explore the derivation and the proof of convergence of multiplicative update rules for Nonnegative Matrix Factorization on Orthogonal Subspace, and then observe applications.

### 2. Nonnegative Matrix Factorization on Orthogonal Subspace

The purpose of Nonnegative Matrix Factorization on Orthogonal Subspace is to find solutions where one factor matrix is semi-orthogonal. Depending on the dimensions of the factor matrix, either the rows or columns will form orthonormal
vectors. This gives the advantage of reducing overlap in parts-based representations. Determining which orthogonal constraint is appropriate depends on the purpose of the experiment.

**Definition 2.1.** Given an input data matrix \( V = [v^1, v^2, \ldots, v^p] \in \mathbb{R}^{p \times n} \), Nonnegative Matrix Factorization on Orthogonal Subspace (NMFOS) seeks two nonnegative matrices \( W \in \mathbb{R}^{p \times k}, H \in \mathbb{R}^{k \times n} \), such that \( V \approx WH \), where \( k \ll p, n \) and either \( \|W^T W - I\| \) or \( \|HH^T - I\| \) is minimized.

(Li, Wu, Peng [5]) Rather than placing an orthogonality constraint on \( W \) or \( H \), with NMFOS we subsume the production of orthogonal factor matrices within the optimization objective function.

\[
F = \min_{W,H \geq 0} \|V - WH\|^2 + \lambda \|W^T W - I\|^2, \\
F = \min_{W,H \geq 0} \|V - WH\|^2 + \lambda \|HH^T - I\|^2,
\]

where \( I \) is the \( k \times k \) identity matrix. The parameter \( \lambda \geq 0 \) controls the orthogonality of \( W \) and \( H \) directly. From this point on we will consider the case for the constraint on \( H \) since the calculations are similar for \( W \).

To derive the multiplicative update rules for NMFOS, we will first rewrite the objective function for the constraint on \( H \) using the definition of the Frobenius norm.

\[
J = \text{tr}((V - WH)(V - WH)^T) + \lambda \text{tr}((HH^T - I)(HH^T - I)^T) \\
= \text{tr}(VV^T) - 2\text{tr}(VH^T W^T) + \text{tr}(WHH^T W^T) + \lambda \text{tr}(HH^T HH^T) \\
- 2\lambda \text{tr}(HH^T) + \lambda \text{tr}(I).
\]

By letting \( \phi_{p \times k} \) and \( \psi_{k \times n} \) be the Lagrange multipliers for the nonnegativity constraints on \( W \) and \( H \), respectively, we define the Lagrange \( L \) as

\[
L = J + \text{tr}(\phi W^T) + \text{tr}(\psi H^T).
\]

We find the partial derivatives of \( L \) with respect to \( W \) and \( H \). In general, we have the following derivative of a scalar by matrix forms,

\[
\frac{\partial \text{tr}(AX)}{\partial X} = \frac{\partial \text{tr}(XA)}{\partial X} = A^T \\
\frac{\partial \text{tr}(AX^T)}{\partial X} = \frac{\partial \text{tr}(X^TA)}{\partial X} = A \\
\frac{\partial \text{tr}(AXBX^TC)}{\partial X} = B^TX^T A^TC^T + BXTCA,
\]

of which implementation retrieves,
L_W = -2VH^T + 2WHH^T + \phi,
L_H = -2W^TV + 2W^TWH + 4\lambda HHH^TH - 4\lambda H + \psi.

By using the Karush-Kuhn-Tucker conditions (Kuhn [3]) we have

0 = -2(VH^T)_{ij}w_{ij} + 2(WHH^T)_{ij}w_{ij}
0 = -2(W^TV + 2\lambda H)_{ij}h_{ij}^j + 2(W^TWH + 2\lambda HHH^TH)_{ij}h_{ij}^j.

We use these equations to solve update rules for $w_{ij}$ and $h_{ij}^j$.

\[ w_{ij}(s+1) = \frac{(VH^T)_{ij}}{(WHH^T)_{ij}} w_{ij}(s) \]
\[ h_{ij}^j(s+1) = \frac{(W^TV + 2\lambda H)_{ij}}{(W^TWH + 2\lambda HHH^TH)_{ij}} h_{ij}^j(s). \]

Finally, the multiplicative update rules can be written,

\[ W = W \odot ((VH^T) \odot (WHH^T)), \]
\[ H = H \odot ((W^TV + 2\lambda H) \odot (W^TWH + 2\lambda HHH^TH)). \]

Similarly, we find the multiplicative update rules for the constraints on $W$ are

\[ W = W \odot ((VH^T + 2\lambda W) \odot (WHH^T + 2\lambdaWW^TW)), \]
\[ H = H \odot ((W^TV) \odot (W^TWH)). \]

3. Convergence of NMFOS Multiplicative Update Rules

We will now justify the convergence of these multiplicative update rules.

**Definition 3.1.** $G(h, h')$ is an auxiliary function for $F(h)$ if the following conditions are satisfied:

\[ G(h, h') \geq F(h), \quad G(h, h) = F(h). \]

Where we take $h$ and $h'$ as distinct arguments.

**Lemma 3.1.** If $G$ is an auxiliary function, then $F$ is nonincreasing under the update rule

\[ h(s+1) = \arg \min_h F(h(s)). \]

**Proof 1. (Lemma 4.1)**

\[ F(h(s+1)) \leq G(h(s+1), h(s)) \leq G(h(s), h(s)) = F(h(s)). \]
To help us prove the objective function is nonincreasing, we consider an arbitrary element $h^i_j$ of $H$. The first and second order derivatives of the objective function with respect to $h^i_j$ are

\[
J'_{ij} = -2(W^TV)_{ij} + 2(W^TWH)_{ij} + 4\lambda(HH^T H)_{ij} - 4\lambda h^i_j,
\]

\[
J''_{ij} = 2[(W^TW)_{ii} + 4\lambda((H^TH)_{jj} + (h^i_j)^2) - 1].
\]

**Lemma 3.2.** The function

\[
G(h, h(s)) = J_{ij}(h^i_j(s)) + J'_j(h^i_j(s))(h - h^i_j(s))
\]

\[
+ \frac{((W^TW)_{ii} + 2\lambda((H^TH)_{jj} + (h^i_j)^2) \pm \delta)}{h^i_j(s)}(h - h^i_j(s))^2
\]

is an auxiliary function for $J_{ij}$ for choice of $\delta$ and if all diagonal elements of $HH^T$ are close to 1.

**Proof 2.** (Lemma 4.2) We expand $J_{ij}(h)$ with a Taylor series.

\[
J_{ij}(h) = J_{ij}(h^i_j(s)) + J'_j(h^i_j(s))(h - h^i_j(s))
\]

\[
+ [(W^TW)_{ii} + 4\lambda((H^TH)_{jj} + (h^i_j)^2) \pm \epsilon](h - h^i_j(s))^2.
\]

Since $(HH^T)_{jj} \approx 1$, this can be rewritten as

\[
J_{ij}(w) = J_{ij}(h^i_j(s)) + J'_j(h^i_j(s))(h - h^i_j(s)) + [(W^TW)_{ii} + 4\lambda(h^i_j)^2 \pm \epsilon](h - h^i_j(s))^2.
\]

Where $\epsilon \approx 0$. We also observe

\[
(W^TW)_{ii} = \sum_{p=1}^k (W^TW)_{ip}h^i_p(s) \geq (W^TW)_{ii}h^i_i(s),
\]

\[
(HH^T H)_{ij} = \sum_{p=1}^k (H H^T H)_{ip}h^i_p(s) \geq (H^TH)_{ij}h^i_i(s)
\]

\[
\geq (h^i_i)^2h^i_i(s).
\]

If we choose $\delta = \epsilon$ it is clear that $G(h, h) = J_{ij}(h)$, and since we have

\[
\frac{(W^TW)_{ij}}{h^i_j(s)} \geq (W^TW)_{ii}
\]

\[
\frac{(HH^T H)_{ij}}{h^i_j(s)} \geq (h^i_i)^2,
\]

we conclude $G(h, h^i_j(s)) \geq J_{ij}(h)$. 

**Theorem 3.1.** By normalizing each column vector of $H$ to unitary Euclidean length, the objective function

$$F = \min_{W,H \geq 0} \|V - WH\|^2 + \lambda \|HH^T - I\|^2$$

is a nonincreasing function under the multiplicative update rules

$$W = W \odot ((VH^T) \odot (WHH^T)),$$

$$H = H \odot ((W^TV + 2\lambda H) \odot (W^TWH + 2\lambda HH^TH)).$$

**Proof 3.** The proof of the Theorem 4.1 need only show the convergence of the update rule for $H$. This convergence is a consequence of lemmas 4.1 and 4.2. If

$$\frac{\partial G(h,h_i^j(s))}{\partial h} = 0,$$

then

$$h_i^j(s + 1) = \frac{(W^TV + 2\lambda H)_{ij}}{(W^TWH + 2\lambda HH^TH)_{ij}} h_i^j(s).$$

Similarly, by juxtaposing the roles of $H$ with $W$, we can show the convergence of the objective function with constraints on $W$ under the appropriate update rules.

**4. Sparseness**

A vector is considered 'sparse' if only a few entries are significantly non-zero. It is desirable for parts-based representations to be sparse. To measure sparseness, we form a mapping $s : \mathbb{R}^n \to [0,1]$ defined by

$$s(x) = \frac{\sqrt{n} - \sum |x_i|}{\sqrt{n} - 1},$$

where $x$ is a vector in $\mathbb{R}^n$ (Hoyer, 2004 [2]). An example of a non-sparse vector is a vector with all components being the same number.

$$\alpha = \begin{bmatrix} \alpha \\ \alpha \\ \vdots \\ \alpha \end{bmatrix}$$

In which case,

$$s(\alpha) = \frac{\sqrt{n} - \sum |\alpha|}{\sqrt{n} - 1} = \frac{\sqrt{n} - n\alpha}{\sqrt{n} - 1} = \frac{\sqrt{n} - \sqrt{n}}{\sqrt{n} - 1} = 0.$$
The most sparse vector will have only one nonzero element. For example,

\[
\beta = \begin{bmatrix}
\beta \\
0 \\
\vdots \\
0
\end{bmatrix}.
\]

In which case,

\[
s(\beta) = \sqrt{n - \frac{\beta + 0 + \ldots + 0}{\sqrt{\beta^2 + 0 + \ldots + 0}}} = \frac{\sqrt{n} - \frac{\beta}{\sqrt{\beta^2}}}{\sqrt{n} - 1} = \frac{\sqrt{n} - 1}{\sqrt{n} - 1} = 1.
\]

In general, sparse vectors have sparsity close to 1, and non-sparse vectors have sparsity close to 0. It is useful to gauge how parsimonious a representation is by measuring the sparsity.

5. Matlab Implementation

To observe application of NMFOS we will adjust Matlab code provided by Hoyer [2] to minimize the NMFOS objective function. In particular, we will compare results of NMF and NMFOS performed on the CBCL and ORL face data bases.

There are 2429 .pgm files in the CBCL data set, each image is 19 × 19 pixels. There are 400 .pgm files in the ORL data set, each image is 92 × 112 pixels. We take each image and vectorize it. We can interpret the vectorization as forming a vector of the features in each image. So \( V_{p \times n} \) will have \( p \) features as rows and \( n \) images as columns. Within the cbcl.m and orl.m files the images are read in and vectorized.

In Matlab, the command \( B = \text{reshape}(A,m,n) \) makes \( B \) an \( m \times n \) matrix whose elements are formed columnwise from \( A \). In the cbcl.m file, we loop through the images in the cbcl-face-database and store each image as a column of our data matrix \( V \). This requires reshaping each 19 × 19 image into a 361-dimensional column vector. Our final data matrix \( V \) will then have dimension 361 × 2429. With respect to our notation, \( p = 361 \) is the dimension of each vectorized image and \( n = 2429 \) is the number of images in the cbcl-face-database.

The Matlab command \( \text{imresize}(A, \text{scale}, \text{`bilinear'}) \) rescales the dimension of the matrix by \( \text{scale} \), and \text{`bilinear'} gives a pixel value that is average of pixels in a 2 × 2 neighborhood. In the orl.m file, we loop through the images in orl-faces, resize each image to half original dimensions taking average values of pixel neighborhoods, and then vectorizing the images and storing them as columns of our data matrix \( V \). We reduce each 92 × 112 image to a 46 × 56 image, and then reshape this image into a 2576-dimensional column vector. Our final data matrix \( V \) will have dimension
2576 × 400, with \( p = 2576 \) is the dimension of each vectorized image and \( n = 400 \) is the number of images in orl-faces.

We seek to make a low rank approximation of these data matrices. We let \( k \) be the number of components or parts we wish to observe. We can interpret \( W \) as having \( k \) \( p \)-dimensional component vectors as columns where \( p \) represents the number of features. We can interpret \( H \) as having \( n \) \( k \)-dimensional prominence vectors as columns with the weights of each component as entries.

For the CBCL data we choose \( k = 49 \), so \( W \) will have 361 features by 49 parts. In \text{visual.m} we reshape the column vectors of \( W \) into 49 19 × 19 pixel parts-based images. For the ORL data we choose \( k = 25 \), so \( W \) will have 2576 features by 25 parts. In \text{visual.m} we reshape the column vectors of \( W \) into 25 46 × 56 pixel parts-based images. In both cases, the entries of \( H \) describe the prominence of the features for each part.

We conduct this experiment using the Euclidean optimizations for NMF, NMFOS for semi-orthogonal \( H \), and NMFOSW for semi-orthogonal \( W \). In both NMFOS cases, we will use the same orthogonallity control parameter as used in Li, Wu, and Peng [5], \( \lambda = 7 \).

6. Experiment Results

The experiment is to compare feature extraction performance on the CBCL and ORL face databases between NMF, NMFOSH (orthogonallity penalty on \( H \)), and NMFOSW (orthogonality penalty on \( W \)). We want to compare feature meaningfulness, sparseness, complexity, and reconstruction MSE. The point of the experiment is to find a good representation of features in the data set. Refer to the feature extractions in Figures 1 and 2. For NMF and NMFOSH we see similar results. We can see parts based representations of faces. For NMFOSW, those features may not be so clear. We mostly have small clusters and blobs. \( W \) contains the parts-based images, so it could be that orthogonalizing the actual images causes these uninterpretable clusters and blobs. Orthogonalizing the prominence of the features retrieves meaningful representations. For this reason we reduce our experiment to comparing NMF and NMFOSH. Figures 3 and 4 compare the sparseness between NMF and NMFOSH. In Figure 3 it appears NMFOSH has a slightly more sparse \( H \) matrix. In Figure 4 the sparseness of \( H \) in both seems about the same. For NMF on cbcl we ran 405 iterations with a run time of 68 seconds and MSE 8772.61. NMFOSH on cbcl ran 400 iterations in 134.2 seconds with MSE 8864.48. NMFOSH took a significantly longer time to execute and had larger reconstruction error. For NMF on orl we ran 410 iterations with a run time of 52.7 seconds with MSE 147150.89. NMFOSH on orl ran 415 iterations with a run time of 59.8 seconds and MSE 147130.20. In this case the two methods performed similarly.
Figure 1. The first block of images is the feature extraction with NMF on cbcl. The second is with NMFOSH on cbcl. The third is NMFOSW on cbcl.

Figure 2. The first block of images is the feature extraction with NMF on orl. The second is with NMFOSH on orl. The third is NMFOSW on orl.

Figure 3. The first image is sparseness measurement for NMF on cbcl, and the second is for NMFOSH. For this experiment we are less concerned with the top bar. The middle bar measures the sparseness of each column vector in $W$. The bottom bar measures the sparseness of each row vector in $H$.

7. Conclusion

In conclusion, NMFOS appears to not perform significantly better than NMF, but has higher computational complexity. However, these experiments were performed with only a single orthogonality parameter. Further investigation into differing parameters is needed. We should expect better sparseness with higher parameter values. There is also opportunity to experiment with a mixed orthogonal
Figure 4. The first image is sparseness measurement for NMF on cbcl, and the second is for NMFOSH. For this experiment we are less concerned with the top bar. The middle bar measures the sparseness of each column vector in $W$. The bottom bar measures the sparseness of each row vector in $H$.

8. Acknowledgements

I would like to thank Dr. Chen for giving me this research opportunity and for several helpful discussions. I would also like to thank Dr. Wang and Dr. Chang for their constructive criticism and insights into the mathematics behind Nonnegative Matrix Factorization.
References


