Fast Optimization of Non-Negative Matrix Tri-Factorization: Supporting Information

Andrej Čopar, Blaž Zupan, and Marinka Zitnik

Faculty of Computer and Information Science, University of Ljubljana, Ljubljana, SI 1000, Slovenia
Department of Molecular and Human Genetics, Baylor College of Medicine, Houston, TX 77030, USA
Department of Computer Science, Stanford University, Stanford, CA 94305, USA
andrej.copar@fri.uni-lj.si

OVERVIEW

An established method for non-negative matrix tri-factorization is based on multiplicative update rules. For completion, the derivations of these rules are summarized in Section A. Update rules for three alternative are derived here and represent one of the main contributions of this paper: Section B shows alternating least squares derivations, Section C projected gradient derivations, and Section D coordinate descent derivations. In Section E we show the equivalence between Quasi-Newton and ALS optimization technique for NMTF. Section F shows how the proposed approaches can be used to reconstruct previously unseen data into the same latent space. Section G shows results of mini-batch NMTF variants.

A Derivation of Multiplicative Update Rules

Multiplicative update rules for non-negative matrix tri-factorization (NMTF) and an orthogonal variant were introduced (Long et al., 2005, Ding et al., 2006). We focus on the non-orthogonal variant (Long et al., 2005) to make the approach comparable with alternative optimization techniques without added constraints. Both methods and all derivations in the following sections are based on the squared Frobenius norm objective function:

\[ D_{\text{Fro}}(X||USV^T) = \|X - USV^T\|_{2,\text{Fro}}^2, \]  

where \( X \) presents the input data and \( U, S \) and \( V \) are latent factors. Squared frobenius norm can be written as \( \|A\|^2 = Tr(A^T A) \), therefore \( D_{\text{Fro}} \) equals to:

\[ D_{\text{Fro}}(X||USV^T) = Tr(X^T X - 2X^T USV^T + VS^T U^T USV^T). \]  

Its partial derivatives with respect to each factor \( U, V, S \) are:

\[ \frac{\partial F}{\partial U} = -2XVS^T + 2USV^T VS, \]  
\[ \frac{\partial F}{\partial V} = -2X^T US + 2VS^T U^T US, \]  
\[ \frac{\partial F}{\partial S} = -2U^T XV + 2U^T USV^T V. \]  

Using the Karush-Kuhn Tucker (KKT) complementarity conditions, we can find a static point. The KKT condition for factor \( U \) is as follows:

\[ \frac{\partial F}{\partial U} U_{ik} = 0. \]  

This condition holds if \( \frac{\partial F}{\partial U} \) reached a static point (the derivative is zero):

\[ (USV^T VS^T - XVS^T)_{ik} U_{ik} = 0, \]  
\[ U_{ik} = \frac{U_{ik} (XVS^T)_{ik}}{(USV^T VS^T)_{ik}}. \]
Similarly, we take the KKT complementarity condition for $V$ and derive the second update rule:

$$\frac{\partial F}{\partial V_{ik}} V_{ik} = 0,$$

$$(VS^T U^T US - X^T US)_{ik} V_{ik} = 0,$$  \hspace{1cm} (7)

$$V_{ik} = V_{ik} - \frac{(X^T US)_{ik}}{(VS^T U^T US)_{ik}}.$$ 

The third update rule is computed with respect to factor $S$:

$$\frac{\partial F}{\partial S_{ik}} S_{ik} = 0,$$

$$(U^T USV^T V - U^T XV) S_{ik} = 0,$$  \hspace{1cm} (8)

$$S_{ik} = S_{ik} \frac{U^T XV}{U^T USV^T V}.$$ 

These rules can be expressed in matrix form, where $\odot$ denotes Hadamard product and symbol $\oslash$ denotes Hadamard division:

$$U \leftarrow U \odot (XVS^T \oslash USV^T VS^T),$$  \hspace{1cm} (9)

$$V \leftarrow V \odot (X^T US \oslash VS^T U^T US),$$  \hspace{1cm} (10)

$$S \leftarrow S \odot (U^T XV \oslash U^T USV^T V).$$  \hspace{1cm} (11)

**B Derivation of Alternating Least Squares**

Let us calculate the gradient of function Eq. (1) with respect to each factor matrix $U$, $V$ and $S$. We follow the derivations in Eq. (3), (4) and (5) and simplify the equations:

$$\frac{\partial F}{\partial U} = XVS^T - USV^T VS^T,$$

$$\frac{\partial F}{\partial V} = X^T US - VS^T U^T US,$$  \hspace{1cm} (12)

$$\frac{\partial F}{\partial S} = U^T XV - U^T USV^T V.$$

We equate the gradient for each factor to zero $\frac{\partial F}{\partial U} = 0$. After this step, we calculate the inverse to get the update rule for $U$:

$$USV^T VS^T = XVS^T,$$

$$U = XVS^T (S^T VS^T)^{-1}.$$  \hspace{1cm} (13)

This step can introduce negative values, so we must force non-negativity by assigning all negative elements with zero:

$$U = [XVS^T (S^T VS^T)^{-1}]_+.$$ 

Following the derivative for $V$ from Eq. (4), we can similarly formulate update rule for $V$. We equate the gradient with zero $\frac{\partial F}{\partial V} = 0$, calculate the inverse and force non-negativity:

$$S^T U^T USV^T = S^T U^T X,$$

$$V^T = (S^T U^T US)^{-1} S^T U^T X,$$  \hspace{1cm} (14)

$$V = X^T US (S^T U^T US)^{-1},$$

$$V = [X^T US (S^T U^T US)^{-1}]_+.$$ 

Derivative for $S$ is shown in equation (5). We set the gradient $\frac{\partial F}{\partial S}$ to zero and force non-negativity constraints:

$$U^T USV^T V = U^T XV,$$

$$S = [(U^T U)^{-1} (U^T XV) (V^T V)^{-1}]_+.$$  \hspace{1cm} (15)
C Derivation of Projected Gradients

Projected gradient methods are based on the general gradient descent scheme, where we take the variable $X$ and create a step towards the descent direction $P$ scaled with the learning rate $\eta$:

$$X \leftarrow X - \eta P.$$  \hspace{1cm} (16)

We derive projected gradient algorithm for NMTF, for the squared Frobenius norm objective function (1). We add projection to the non-negative values $\lceil \cdot \rceil_+$ to enforce non-negativity of the factors in case in crosses into negative values.

Projected gradient methods follow this form:

$$U \leftarrow \lceil U - \eta_u P_U \rceil_+,$$
$$S \leftarrow \lceil S - \eta_s P_S \rceil_+,$$
$$V \leftarrow \lceil V - \eta_v P_V \rceil_+,\hspace{1cm} (17)$$

where $P_U$, $P_V$ and $P_S$ are descent directions, $\eta_u$, $\eta_v$, and $\eta_s$ are learning rates. Descent directions are defined with the following form:

$$P_U = D_U \odot \frac{\partial F}{\partial U},$$
$$P_V = D_V \odot \frac{\partial F}{\partial V},$$
$$P_S = D_S \odot \frac{\partial F}{\partial S}.\hspace{1cm} (18)$$

The scaling factors $D_U$, $D_V$, and $D_S$ are set the following way:

$$D_U = U \odot (USV^T VS^T),$$
$$D_V = V \odot (VS^T U^T US),$$
$$D_S = S \odot (U^T USV^T V).\hspace{1cm} (19)$$

The scaling factor was chosen based on a study which compares convergence analysis of four different scaling factors (Merritt, 2005). The choice of scaling factor is inspired by existing projected gradient algorithm for non-negative matrix factorization (Lin, 2007). The learning rate or step size parameter is dynamically chosen using the same form as a related study for classical non-negative matrix factorization (Lin, 2007):

$$\eta_u = Tr((SV^T V)(S^T P^T_U P_U)).$$

We calculate the descent direction for each factor matrix $U$ using the descent direction for $U$ and the partial derivative (3):

$$P_U = \langle U \odot USV^T VS^T \rangle \odot (X - USV^T)VS^T,$$
$$P_U = U - U \odot USV^T VS^T \odot XVS^T.\hspace{1cm} (20)$$

We define learning rate for $V$ as follows:

$$\eta_v = Tr((SP^T_V P_V)(SU^T U)).$$

We calculate the descent direction for $V$ by inserting the $D_S$ and derivative in Eq. (4) into the Eq. (19):

$$P_V = \langle V \odot VS^T U^T US \rangle \odot (X - USV^T)U^T US,$$
$$P_V = V - V \odot VS^T U^T US \odot X^T US.\hspace{1cm} (21)$$

We define learning rate for $S$ as follows:

$$\eta_s = Tr((U^T UP_S)(VS^TP^T_V)).$$

The descent direction for $S$ is computed by inserting the $D_S$ and derivative in Eq. (5) into the Eq. (19):

$$P_S = \langle S \odot U^T USV^T V \rangle \odot U^T (X - USV^T)V,$$
$$P_S = S - S \odot U^T USV^T V \odot U^T XV.\hspace{1cm} (22)$$
We insert the descent directions and learning rates in Eq. (18). The resulting update rules for $U$ are:

$$
P_u = U - U \odot (USV^T V S^T) \odot (XVS^T),$$

$$\eta_u = \frac{\sum (P_u \odot (USV^T V S^T - XVS^T))}{Tr((SV^T V)(S^TP_u T P_u))},$$

$$U \leftarrow [U - \eta_u P_u]_+. $$

The following update rules define the procedure for updating factor $V$:

$$P_v = V - V \odot (VS^T U S) \odot (X^T US),$$

$$\eta_v = \frac{\sum (P_v \odot (VS^T U S - X^T US))}{Tr((SP_v T P_v)(S^T U^T U))},$$

$$V \leftarrow [V - \eta_v P_v]_+. $$

Resulting update rules for factor $S$ are:

$$P_s = S - S \odot (U^T USV^T V) \odot (U^T XV),$$

$$\eta_s = \frac{\sum (P_s \odot (U^T USV^T V - U^T XV))}{Tr((U^T U P_s)(V^T V P_s T))},$$

$$S \leftarrow [S - \eta_s P_s]_+. $$

### D Derivation of Coordinate Descent

First, we define the following residual matrices:

$$X^i = X - USV^T + u_i(SV^T)_i,$$

$$X^j = X - USV^T + (US)_j v_j^T,$$

$$X^{(i,j)} = X - USV^T + (u_is_j v_j^T).$$

The idea is to iterate over this set of subproblems for $i \in \{1 \ldots k_1\}$ and $j \in \{1 \ldots k_2\}$:

$$F_U^{(i)}(u_i) = ||X^i - u_i(SV^T)_i||^2 = ||X^i - u_i S_v V^T||^2,$$

$$F_V^{(j)}(v_j) = ||X^j - (US)_j v_j^T||^2 = ||X^j - Us_j v_j^T||^2,$$

$$F_S^{(i,j)}(s_{i,j}) = ||X^{(i,j)} - u_is_j v_j^T||^2.$$  

We use $||A||_2 = Tr(A^T A)$ and $(AB)_i = Ab_i$, where $b_i$ is $i$-th column of $B$. First, we derive the update rule for $F_U^{(i)}(u_i)$:

$$F_U^{(i)}(u_i) = Tr(X^{(i)}^T X^{(i)} - 2X^{(i)^T} u_is_i V^T + V s_i^T u_i s_i V^T),$$

$$\frac{\partial F}{\partial u_i} = -2X^{(i)^T} V s_i^T + 2u_is_i V^T V s_i^T.$$ 

Then, we equate the derivative $\frac{\partial F}{\partial u_i}$ to zero:

$$X^s V s_i^T = u_is_i V^T V s_i^T,$$

$$u_i = \frac{X^s V s_i^T}{s_i V^T V s_i^T}.$$ 

Update for the $F_V^{(j)}(v_j)$ subproblem:

$$F_V^{(j)}(v_j) = Tr(X^{(j)}^T X^{(j)} - 2X^{(j)^T} Us_j v_j^T + v_j s_j^T Us_j v_j^T),$$

$$\frac{\partial F}{\partial v_j} = -2X^{(j)^T} Us_j + 2v_j s_j^T Us_j.$$ 

4/9
We equate the derivative \( \frac{\partial F}{\partial v_{ij}} \) to zero:

\[
X^T U s_j = v_{ij} s_{ij}^T U^T U s_j, \tag{33}
\]

\[
v_{ij} = \frac{X^T U s_j}{s_{ij}^T U^T U s_j}.
\]

Finally, we find the static point for the \( F_s^{(ij)}(s_{ij}) \) subproblem:

\[
F_s^{(ij)}(s_{ij}) = Tr(X^{(i,j)^T} X^{(i,j)} - 2 X^{(i,j)^T} u_{i,s_{ij}} v_{j}^T + v_{ij} s_{ij}^T u_{i,s_{ij}} v_{j}^T),
\]

\[
\frac{\partial F}{\partial s_{ij}} = -2 u_{i}^T X^{(i,j)} v_{j} + 2 u_{i}^T u_{i,s_{ij}} v_{j}^T v_{j}.
\]

We equate the derivative \( \frac{\partial F}{\partial s_{ij}} \) to zero:

\[
u_{ij} X^{(i,j)} v_{j} = u_{i}^T u_{i,s_{ij}} v_{j}^T v_{j}, \tag{35}
\]

\[
s_{ij} = u_{i}^T X^{(i,j)} v_{j} - u_{i}^T u_{i,s_{ij}} v_{j}^T v_{j}.
\]

For efficient implementation, we replace residual matrices \( X' \), \( X' \) and \( X^{i,j} \) with definitions in equations (26), (27) and (28). We also enforce non-negativity with projection to non-negative values. The resulting update rules are:

\[
u_i = u_i + \left[ \frac{(X V S')_i - (U S V T')_i}{s_i V' V s_i^T} \right]_+, \tag{36}
\]

\[
v_j = v_j + \left[ \frac{(X^T U S)_j - (V S^T U T V)_j}{s_j^T U^T U s_j} \right]_+, \tag{37}
\]

\[
s_{ij} = s_{ij} + \left[ \frac{(U^T X V)_{ij} - (U^T U S V T V)_{ij}}{u_{ij}^T u_{i}^T v_{j}^T v_{j}} \right]_+. \tag{38}
\]

These rules appear in the additive form, where the step size \( \eta \) is implicitly defined within the update rules:

\[
u_i = u_i + \left[ \eta \left( \frac{\partial F}{\partial U} \right) \right]_+. \tag{39}
\]

From the equation (36) and (33) we can observe the step size:

\[
\eta = 1/(s_i V' V s_i^T).
\]

Similarly, the step sizes for \( v_j \) and \( s_{ij} \) can be observed:

\[
\eta = 1/(s_j^T U^T U s_j), \tag{41}
\]

\[
\eta = 1/(u_{ij}^T u_{i}^T v_{j}^T v_{j}). \tag{42}
\]

We observed that such step size is effective in preventing the values to cross into negative space. The loss of information associated with projection into non-negative space is reduced, which keeps convergence stable throughout the factorization procedure.

**E Derivation of Quasi-Newton update rules**

In this section we show the equivalence of quasi-Newton update rules and ALS. The quasi-Newton update rules are formulated as follows, where \( H_U \), \( H_V \), and \( H_S \) are Hessians of each factors:

\[
\text{vec}(U) \leftarrow \left[ \text{vec}(U) - H_U^{-1} \text{vec} \left( \frac{\partial F}{\partial U} \right) \right]_+, \tag{43}
\]

\[
\text{vec}(V) \leftarrow \left[ \text{vec}(V) - H_V^{-1} \text{vec} \left( \frac{\partial F}{\partial V} \right) \right]_+ \tag{44}
\]

\[
\text{vec}(S) \leftarrow \left[ \text{vec}(S) - H_S^{-1} \text{vec} \left( \frac{\partial F}{\partial S} \right) \right]_+. \tag{45}
\]
Let us calculate the gradient and the Hessian of function Eq. (1) with respect to U:

\[
\frac{\partial F}{\partial U} = XVS^T - USV^T VS^T, \tag{46}
\]

\[
H_U = SV^T VS^T \otimes I. \tag{47}
\]

The Kronecker product is shown with \( \otimes \), the vec-operator is shown with vec, and I is identity matrix. By placing the gradient and the Hessian into the equation (43), we get:

\[
\text{vec}(U) = [\text{vec}(U) - (SV^T VS^T \otimes I)^{-1} \text{vec}(USV^T VS^T - XVS^T)]_+, \tag{48}
\]

which by converting from vec-operator to matrix form, using the rule vec(ABC) = (C^T \otimes A) vec(B) simplifies to:

\[
U = [U - (USV^T VS^T - XVS^T)(SV^T VS^T)^{-1}]_+. \tag{49}
\]

Further, we derive the gradient and the Hessian matrix of the factor V:

\[
\frac{\partial F}{\partial V} = X^T US - VS^T U^T US, \tag{50}
\]

\[
H_V = S^T U^T US \otimes I. \tag{51}
\]

We place the gradient and Hessian into equation (44) and convert to matrix form:

\[
V = [V - (VS^T U^T US - X^T US)(S^T U^T US)^{-1}]_+. \tag{52}
\]

The gradient and the Hessian of the S matrix are:

\[
\frac{\partial F}{\partial S} = U^T X V - U^T USV^T V, \tag{53}
\]

\[
H_S = V^T V \otimes U^T U. \tag{54}
\]

We place the gradient and the Hessian into the equation (45).

\[
\text{vec}(S) = [\text{vec}(S) - (V^T V \otimes U^T U)^{-1} \text{vec}(USV^T VS^T - XVS^T)]_+, \tag{55}
\]

We use the following rule to convert from vector to the matrix form: vec(ABC) = (C^T \otimes A) vec(B), where A = (U^T U)^{-1}, C = (V^T V)^{-1}, and B = \frac{\partial F}{\partial S}. Note that (V^T V)^T = V^T V since V^T V is symmetric.

\[
S = [S - (U^T U)^{-1}(U^T USV^T V - U^T XV)(V^T V)^{-1}]_+. \tag{56}
\]

Let us simplify the equations (49), (52) and (56), using the distributive rule \((A + B)C = AC + BC\):

\[
U = [U - U + XVS^T (SV^T VS^T)^{-1}]_+, \tag{57}
\]

\[
V = [V - V + X^T US(S^T U^T US)^{-1}]_+, \tag{57}
\]

\[
S = [S - S + (U^T U)^{-1}(U^T XV)(V^T V)^{-1}]_+. \tag{57}
\]

These rules are equivalent to the ALS update rules. Note that similar was already shown for NMF by Cichocki.¹

**F Reconstruction on testing data**

In this section we study reconstruction error of different optimization approaches on held-out data. We split the data X row-wise into two parts: \(X_{train}\) is composed of the first 80 percent and \(X_{new}\) composed of the remaining 20 percent of the data. Then, we run the factorization on \(X_{train}\), resulting in \(U_{train}\), S and V. We transform the remaining data \(X_{new}\) into the latent space defined by S and V, such that the following objective function is minimized:

\[
D_{Fro}(X_{new}|U_{new}SV^T) = ||X_{new} - U_{new}SV^T||_2^2. \tag{58}
\]

We keep the S and V factors from the training step and initialize \(U_{new}\) with random values. We iteratively apply updates on \(U_{new}\), while S and V remain unchanged. Figure A shows the objective value of four optimization techniques according to Eq. 58 at factorization rank \(k_1, k_2 = 20\). Each experiment is repeated ten times until the convergence criterion with parameter \(\varepsilon = 6\) is reached.
**Figure A.** Convergence of matrix factorization procedure on held-out (test) data of the factorization system that was pre-trained on the held-in (training) data. The experiments were repeated ten times, each time with different random sampling. Solid lines show the convergence of the run with the best-end objective value at convergence point, and the highlighted area shows the span of objective values. MUR, multiplicative update rules; ALS, alternating least squares; PG, projected gradients; COD, coordinate descent.
Figure B. Reconstruction of testing data $X_{\text{new}}$ with respect to factorization rank. The lowest objective function at convergence is shown as a function of factorization rank ($k_1 = k_2 = k_i \in \{10, 20, \ldots, 100\}$). The highlighted area shows the span from lowest to highest objective value across ten repeated runs. MUR, multiplicative update rules; ALS, alternating least squares; PG, projected gradients; COD, coordinate descent.

The convergence using pre-trained model is orders of magnitude faster than training from scratch (Fig. 2 in the main text). Consistent with the optimization on training data, we observe that Alternating least squares and Coordinate descent are fastest, Projected gradient is slower and Multiplicative updates is the slowest. ALS approach did not converge on Coil20 training data and was therefore not used on testing data.

Figure B shows the lowest objective value from multiple runs (solid lines). We compare the optimization function at convergence for factorization rank $k \in \{10, 20, \ldots, 100\}$. We can observe that for majority of datasets, all methods converge to a similar solution (Coil20, STRING, Mutations, Newsgroups). Coordinate descent and Alternating least squares appear more sensitive to random initialization on AlphaDigit and MovieLens datasets. The difference between best and worst solution increases as we increase factorization rank. In such cases, Coordinate descent needs to be repeated a few times in order to ensure results comparable to Projected gradients and Multiplicative update rules. Alternatively, using a different initialization technique we may overcome this drawback.

G Stochastic mini-batch updates

We have implemented the stochastic mini-batch approach for NMTF. In each iteration we randomly permute the dataset into $b$ batches such that each row is included in exactly one batch. We iterate over all batches and update the dataset using a subset of
the data $X$, and a subset of $U$ factor: $U_i$, where $i \in \{1, 2, \ldots, b\}$. We split the data into $b = 20$ batches. The objective function is evaluated over the entire dataset.

We show the convergence of mini-batch versions together with its non-batch counterparts in Figure C. Each experiment was run for a maximum of 1000 iterations. Mini-batch versions of ALS (dashed yellow line) and COD (dashed red line) do not converge. Projected gradient mini-batch (dashed green line) and MUR mini-batch (dashed blue line) approaches show faster convergence at the beginning of some experiments (AlphaDigit, Coil20, MovieLens). The final solution is in all cases worse than the non-batch counterpart and large oscillations (larger than $\epsilon = 0.01$) make it difficult to determine stopping criteria.

![Figure C. Convergence of mini-batch optimization techniques on six datasets using four NMTF optimization methods (continuous lines) and four mini-batch variants (dashed transparent lines). MUR_MB: mini-batch multiplicative updates; PG_MB: mini-batch projected gradients; COD_MB: mini-batch coordinate descent; ALS_MB: mini-batch ALS.](image)

References

1. Cichocki A, Zdunek R, Phan AH, Amari Si. Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation. John Wiley & Sons; 2009.

2. Langville, A., Meyer, C., Albright, R. Initializations for the nonnegative matrix factorization. KDD 2006 Philadelphia, PA USA 12 (2006).