Non-Negative Tensor Factorization with RESCAL

Denis Krompaß[1], Maximilian Nickel[1], Xueyan Jiang[1], and Volker Tresp[1,2]

[1] Department of Computer Science, Ludwig Maximilian University, Oettingenstraße 67, 80538 Munich, Germany,
Denis.Krompass@campus.lmu.de
[2] Corporate Technology, Siemens AG, Otto-Hahn-Ring 6, 81739 Munich, Germany,
Volker.Tresp@siemens.com

Abstract. Non-negative data is generated by a broad selection of applications today, e.g in gene expression analysis or imaging. Many factorization techniques have been extended to account for this natural constraint and have become very popular due to their decomposition into interpretable latent factors. Generally relational data like protein interaction networks or social network data can also be seen as naturally non-negative. In this work, we extend the RESCAL tensor factorization, which has shown state-of-the-art results for multi-relational learning, to account for non-negativity by employing multiplicative update rules. We study the performance via these approaches on various benchmark datasets and show that a non-negativity constraint can be introduced by losing only little in terms of predictive quality in most of the cases but simultaneously increasing the sparsity of the factors significantly compared to the original RESCAL algorithm.

Keywords: RESCAL, non-negative matrix factorization, non-negative tensor factorization, Kullback-Leibler-Divergence, relational learning, multiplicative updates, sparsity

1 Introduction

When mining non-negative datasets like document collections, gene expression data or imaging data, matrix and tensor decompositions are often employed for low rank approximations of the original data matrix/tensor to perform tasks like clustering [1] or predicting items that might be interesting for a user in a recommendation environment [2]. Other interesting applications arise in the field of image processing e.g. compression. As the original data is non-negative, a non-negative constraint on the computed factors of the decomposition seem also very natural. In contrast to e.g. SVD based decompositions a non-negative decomposition results in an additive representation that tends to be naturally sparse and therefore memory can be saved when storing the factors. Another big benefit of non-negative factors come from their interpretability. This benefit has been demonstrated in various applications like text-processing [3] or image processing

[4], where the factors could be interpreted as latent topics or as structures seen in the image like eyes, ears or noses, respectively.

But the benefits of a non-negative constraint on the factorization also have serious drawbacks as the decomposition is generally not unique and therefore tend to converge into local minima. As a consequence, the initialization of the factor matrices becomes critical. In the past, different methods have been proposed for initializing one or more of the factor matrices instead of initializing them randomly (see [3] for a summary and [5]). It is also possible to additionally enforce uniqueness by taking the solution that is the sparsest.

In the past, non-negative matrix factorization (NMF) has been extended to commonly used tensor decompositions like PARAFAC/CP [6][7] and TUCKER[8] by employing multiplicative update rules as introduced by [9]. As relational datasets from protein interaction or social networks also impose a natural non-negativity constraint, we propose to also extend NMF to the RESCAL model [10]. RESCAL is a three-way-tensor factorization model that has been shown to have very good results in various canonical relational learning tasks like link prediction, entity resolution and collective classification [10]. One main feature of RESCAL is that it can exploit a collective learning effect when used on relational data.

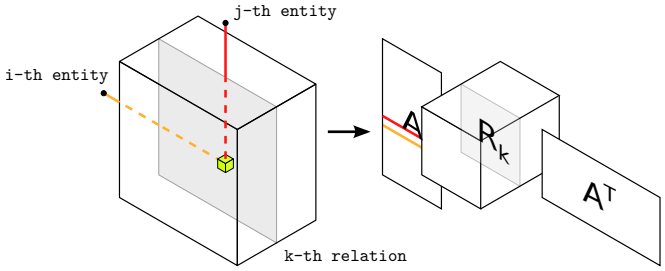


Fig. 1. Graphical illustration of the RESCAL tensor factorization into the factor matrix A and the core tensor $\mathcal{R}$ [11].

The RESCAL decomposition decomposes a tensor $\mathcal{X}$ of shape $n \times n \times m$, where each of the m slices of $\mathcal{X}$ can be seen as a adjacency-matrix between n entities for each of the m relations, into a factor matrix A of shape $n \times r$ and a core tensor R of shape $r \times r \times m$ (figure 1). Where r is the number of latent factors. In terms of frontal slices, each of the k frontal slices of $\mathcal{X}$ is factorized into:

$$X_k = AR_kA^T, for\ k = 1, ..., m$$

The decomposition is computed by minimizing the regularized least-squares cost function

$$C_{LS}(\mathcal{X}, A, \mathcal{R}) = f_{LS}(\mathcal{X}, A, \mathcal{R}) + f_{L2}(A, \mathcal{R}) \tag{1}$$

$$with\ f_{LS}(\mathcal{X}, A, \mathcal{R}) = \sum_k \|X_k - AR_kA^T\|_F^2$$
$$f_{L2}(A, \mathcal{R}) = \lambda_A\|A\|_F^2 + \lambda_R \sum_k \|R_k\|_F^2$$

in an Alternating Least Squares (ALS) fashion with update functions for A and R_k:

$$A \leftarrow \left[\sum_k X_kAR_k^T + X_k^TAR_k\right]\left[\sum_k R_kA^TAR_k^T + R_k^TA^TAR_k + \lambda_A\mathbf{I}\right]^{-1}$$
$$R \leftarrow (Z^TZ + \lambda_R\mathbf{I})^{-1}Z^T\mathbf{vec}(X_k)$$
$$with\ Z = (A \otimes A)$$

This update functions obviously do not guarantee non-negative factors because of the inverse.

The contribution of this paper will be as follows: We derived and implemented non-negative tensor decompositions based on the RESCAL model by employing multiplicative update rules for A and $\mathcal{R}$ with respect to least-squares and Kullback-Leibler divergence based cost functions including regularization and normalization of factor matrix A as introduced by [9], [12] and [13]. Additionally we derived update rules for including attribute information of entities as proposed by [11]. The derived models are compared against the original RESCAL model without non-negative constraint using the Kinship, Nations and UMLS multirelational datasets. We show that the non-negative extensions result into much sparser factors that have comparably good results in most of the cases with respect to predictive capabilities.

2 Methods

In the next sections, $X \bullet Y$ and $\frac{X}{Y}$ will denote elementwise matrix-multiplication and division. XY will represent the regular matrix multiplication. Futhermore will $\mathcal{R}$ or $\mathcal{X}$ represent tensors where R, X will represent matrices. The matrix $\mathbf{J}^{ij}$ is defined as a single entry matrix, where $\mathbf{J}_{ij} = 1$ and zero otherwise. The matrix $\mathbf{E}^{n\times m}$ represents a $n \times m$ matrix filled with ones and $\mathbf{1}$ a row vector of ones.

For deriving non-negative updates for A and R_k, we use multiplicative update rules as proposed in [9], giving the general form of:

$$\theta_i = \theta_i \left(\frac{\frac{\partial C(\theta)^-}{\partial \theta_i}}{\frac{\partial C(\theta)^+}{\partial \theta_i}}\right)^{\alpha} \tag{2}$$

Where $C(\theta)$ represents a cost function of the non-negative variables θ and $\frac{\partial C(\theta)^-}{\partial \theta_i}$ and $\frac{\partial C(\theta)^+}{\partial \theta_i}$ are the negative and positive parts of the derivative of $C(\theta)$ [13].

2.1 NN updates for RESCAL with least-squares cost function

The partial derivatives of (1) with respect to A and $\mathcal{R}$ respectively are:

$$\begin{aligned}\frac{\partial C_{LS}(\mathcal{X}, A, \mathcal{R})}{\partial A} =& -2\left(\sum_k X_k A R_k^T + X_k^T A R_k\right) \\ &+ 2\left(\sum_k A R_k A^T A R_k^T + A R_k^T A^T A R_k\right) + \lambda_A A \\ \frac{\partial C_{LS}(\mathcal{X}, A, \mathcal{R})}{\partial R_k} =& -2\left(A^T X_k A\right) + 2\left(A^T A R_k A^T A + \lambda_R R_k\right)\end{aligned}$$

Inserting this into (2) we get the updates for A and R_k in matricized form:

$$A \leftarrow A \bullet \frac{\sum_k X_k A R_k^T + X_k^T A R_k}{A\left(\left[\sum_k R_k A^T A R_k^T + R_k^T A^T A R_k\right] + \lambda_A \mathbf{I}\right)} \tag{3}$$

$$R_k \leftarrow R_k \bullet \frac{A^T X_k A}{A^T A R_k A^T A + \lambda_R R_k} \tag{4}$$

2.2 Kullback-Leibler-divergence cost function

The generalized Kullback-Leibler-divergence cost function for RESCAL with L_1 regularization is given by:

$$C_{KL}(\mathcal{X}, A, \mathcal{R}) = f_{KL}(\mathcal{X}, A, \mathcal{R}) + f_{L1}(A, \mathcal{R}) \tag{5}$$

$$\begin{aligned}with\ f_{KL}(\mathcal{X}, A, \mathcal{R}) &= \sum_{ijk}\left(X_{ij} log \frac{X_{ij}}{(A R_k A^T)_{ij}} - X_{ij} + (A R_k A^T)_{ij}\right) \\ f_{L1}(A, \mathcal{R}) &= \lambda_A \|A\|_1 + \lambda_R \sum_k \|R_k\|_1\end{aligned}$$

The elementwise partial derivatives of (5) with respect to A and R_k respectively are:

$$\begin{aligned}\frac{\partial C_{KL}(\mathcal{X}, A, \mathcal{R})}{\partial A_{ia}} =& -\left[\sum_{jk} \frac{X_{ijk}}{(A R_k A^T)_{ij}} (J^{ia} R_k A^T)_{ij} + \frac{X_{jik}^T}{(A R_k^T A^T)_{ji}} (A R_k J^{ai})_{ji}\right] \\ &+ \left[\sum_{jk} (J^{ia} R_k A^T)_{ij} + (A R_k J^{ai})_{ji}\right] + \lambda_A \\ \frac{\partial C_{KL}(\mathcal{X}, A, \mathcal{R})}{\partial R_{rtk}} =& -\left[\sum_{ij} (A J^{rtk} A^T)_{ij} \frac{X_{ijk}}{(A R_k A^T)_{ij}}\right] + \left[\sum_{ij} (A J^{rtk} A^T)_{ij}\right] + \lambda_R\end{aligned}$$

Inserting this into (2) we get the elementwise updates for A and R_k:

$$A_{ia} \leftarrow A_{ia} \frac{\sum_{jk} \frac{X_{ijk}}{(AR_kA^T)_{ij}}(J^{ia}R_kA^T)_{ij} + \frac{X^T_{jik}}{(AR_k^TA^T)_{ji}}(AR_kJ^{ai})_{ji}}{\left[\sum_{jk}(J^{ia}R_kA^T)_{ij} + (AR_kJ^{ai})_{ji}\right] + \lambda_A}$$

$$R_{rtk} \leftarrow R_{rtk} \frac{\sum_{ij} \frac{X_{ijk}}{(AR_kA^T)_{ij}}(AJ^{rtk}A^T)_{ij}}{\left[\sum_{ij}(AJ^{rtk}A^T)_{ij}\right] + \lambda_R}$$

In matricized form the updates equal:

$$A \leftarrow A \bullet \frac{\sum_k \frac{X_k}{AR_kA^T}AR_k^T + \frac{X_k^T}{AR_k^TA^T}AR_k}{\sum_k \mathbf{E}^{n\times n}(AR_k^T + AR_k) + \lambda_A\mathbf{E}^{n\times r}} \tag{6}$$

$$R_k \leftarrow R_k \bullet \frac{A^T\frac{X_k}{AR_kA^T}A}{A^T\mathbf{E}^{n\times n}A + \lambda_R\mathbf{E}^{n\times r}} \tag{7}$$

2.3 Integrating L1 Regularization with Normalization of A

In 2004, [12] proposed an algorithm for sparse NMF by penalizing the right factor matrix of the decomposition by the L_1 norm while keeping the column vectors of the left factor matrix normalized to unit length ($\widetilde{W}_{ir} = \frac{W_{ir}}{\|W_r\|_F}$). Originally this algorithm was proposed for a least squares cost function but was also derived for a generalized Kullback-Leibler divergence cost function by [13]. Applying this idea to the RESCAL cost functions based on least squares and KL-divergence leads to the following updates for A and R_k, respectively:

For a least squares cost function with L_1 penalty on R:

$$C^*_{LS}(\mathcal{X}, A, \mathcal{R}) = f_{LS}(\mathcal{X}, \widetilde{A}, \mathcal{R}) + f_{L1}(\mathcal{R}) \tag{8}$$

$$with\ f_{LS}(\mathcal{X}, \widetilde{A}, \mathcal{R}) = \sum_k \|X_k - \widetilde{A}R_k\widetilde{A}^T\|_F^2$$

$$f_{L1}(\mathcal{R}) = \lambda_R \sum_k \|R_k\|_1$$

The updates equal:

$$A \leftarrow \widetilde{A} \bullet \frac{\sum_k \widetilde{B}_k + \widetilde{A}\ diag\left(\mathbf{1}\left[\widetilde{C}_k \bullet \widetilde{A}\right]\right)}{\sum_k \widetilde{C}_k + \widetilde{A}\ diag\left(\mathbf{1}\left[\widetilde{B}_k \bullet \widetilde{A}\right]\right)} \tag{9}$$

$$R_k \leftarrow R_k \bullet \frac{\widetilde{A}^T X_k \widetilde{A}}{\widetilde{A}^T\widetilde{A}R_k\widetilde{A}^T\widetilde{A} + \lambda_R\mathbf{E}^{r\times r}} \tag{10}$$

where

$$\widetilde{B}_k = X_k \widetilde{A} R_k^T + X_k^T \widetilde{A} R_k, \quad \widetilde{C}_k = \widetilde{A} \left(R_k \widetilde{A}^T \widetilde{A} R_k^T + R_k^T \widetilde{A}^T \widetilde{A} R_k \right)$$

For a generalized KL-divergence based cost function we get:

$$C^*_{KL}(\mathcal{X}, A, \mathcal{R}) = f_{KL}(\mathcal{X}, \widetilde{A}, \mathcal{R}) + f_{L1}(\mathcal{R}) \tag{11}$$

The updates equal:

$$A \leftarrow \widetilde{A} \bullet \frac{\sum_k \widetilde{E}_k + \widetilde{A}\ diag\left(\mathbf{1}\left[\widetilde{F}_k \bullet \widetilde{A}\right]\right)}{\sum_k \widetilde{F}_k + \widetilde{A}\ diag\left(\mathbf{1}\left[\widetilde{E}_k \bullet \widetilde{A}\right]\right)} \tag{12}$$

$$R_k \leftarrow R_k \bullet \frac{\widetilde{A}^T \frac{X_k}{\widetilde{A} R_k \widetilde{A}^T} \widetilde{A}}{\widetilde{A}^T \mathbf{E}^{n \times n} \widetilde{A} + \lambda_R \mathbf{E}^{n \times r}} \tag{13}$$

where

$$\widetilde{E}_k = \frac{X_k}{\widetilde{A} R_k \widetilde{A}^T} \widetilde{A} R_k^T + \frac{X_k^T}{\widetilde{A} R_k^T \widetilde{A}^T} \widetilde{A} R_k, \quad \widetilde{F}_k = \mathbf{E}^{n \times n} \widetilde{A} (R_k^T + R_k)$$

2.4 Integrating entity attributes

In [11] an algorithm for efficiently including attributes of entities was proposed. For this reason, the original cost function was extended by a decomposition of the attributes matrix D into A and V ($D \approx AV$). Additionally a regularization penalty on V was added:

$$C_{LS_{attr}} = f_{LS_{attr}}(D, A, V) + f_{L2}(V) \tag{14}$$

$$with\ f_{LS_{attr}}(D, A, V) = \|D - AV\|_F^2$$
$$f_{L2}(V) = \|V\|_F^2$$

The idea can be used to extend the previous update rules such that they also include the information of entity attributes. The updates for R will stay untouched but the updates of A have to be modified. Update rules for V can be easily derived from [9].

Updates for LS cost function with L_2-regularization:

$$A \leftarrow A \bullet \frac{\left[\sum_k X_k A R_k^T + X_k^T A R_k\right] + D V^T}{A(\left[\sum_k R_k A^T A R_k^T + R_k^T A^T A R_k\right] + \lambda_A \mathbf{I} + V V^T)} \tag{15}$$

$$V \leftarrow V \bullet \frac{A^T D}{(A^T A + \lambda_V \mathbf{I}) V} \tag{16}$$

Updates for LS cost function with normalization of A and L_1-regularization on $\mathcal{R}$ and V:

$$A \leftarrow \widetilde{A} \bullet \frac{\widetilde{T_1} + DV^T + \widetilde{A}\ diag\left(\mathbf{1}\left[\widetilde{A}VV^T \bullet \widetilde{A}\right]\right)}{\widetilde{T_2} + \widetilde{A}VV^T + \widetilde{A}\ diag\left(\mathbf{1}\left[DV^T \bullet \widetilde{A}\right]\right)} \tag{17}$$

$$V \leftarrow V \bullet \frac{A^T D}{A^T AV + \lambda_V \mathbf{E}^{r\times m}} \tag{18}$$

where

$$\widetilde{T_1} = \sum_k \widetilde{B}_k + \widetilde{A}\ diag\left(\mathbf{1}\left[\widetilde{C}_k \bullet \widetilde{A}\right]\right), \quad \widetilde{T_2} = \sum_k \widetilde{C}_k + \widetilde{A}\ diag\left(\mathbf{1}\left[\widetilde{B}_k \bullet \widetilde{A}\right]\right)$$

Updates for KL-divergence with L_1-regularization:

$$A \leftarrow A \bullet \frac{\left[\sum_k \frac{X_k}{AR_kA^T} AR_k^T + \frac{X_k^T}{AR_k^TA^T} AR_k\right] + \frac{D}{AV}V^T}{\left[\sum_k \mathbf{E}^{n\times n}(AR_k^T + AR_k)\right] + \lambda_A \mathbf{E}^{n\times r} + \mathbf{E}^{n\times m}V^T} \tag{19}$$

$$V \leftarrow V \bullet \frac{A^T \frac{D}{AV}}{A^T\mathbf{E}^{n\times m} + \lambda_V \mathbf{E}^{r\times m}} \tag{20}$$

Updates for KL-divergence with normalization of A and L_1-regularization on $\mathcal{R}$ and V:

$$A \leftarrow \widetilde{A} \bullet \frac{\widetilde{T_3} + \frac{D}{\widetilde{AV}}V^T\widetilde{A}\ diag\left(\mathbf{1}\left[\widetilde{A}^T\mathbf{E}^{n\times m} \bullet \widetilde{A}\right]\right)}{\widetilde{T_4} + \widetilde{A}^T\mathbf{E}^{n\times m}\widetilde{A}\ diag\left(\mathbf{1}\left[\frac{D}{\widetilde{AV}}V^T \bullet \widetilde{A}\right]\right)} \tag{21}$$

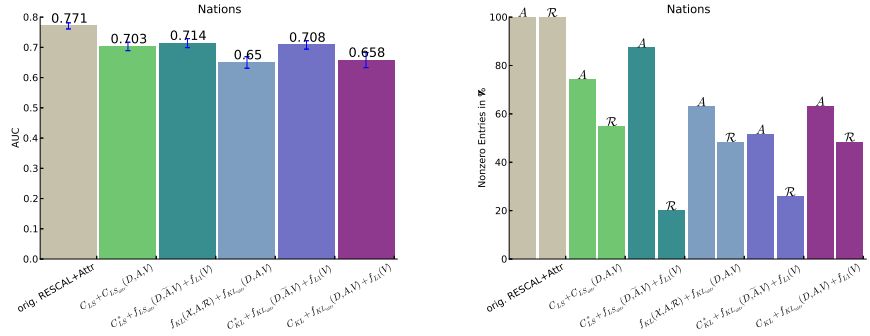
$$V \leftarrow V \bullet \frac{\widetilde{A}^T \frac{D}{\widetilde{AV}}}{\widetilde{A}^T\mathbf{E}^{n\times m} + \lambda_V \mathbf{E}^{r\times m}} \tag{22}$$

where

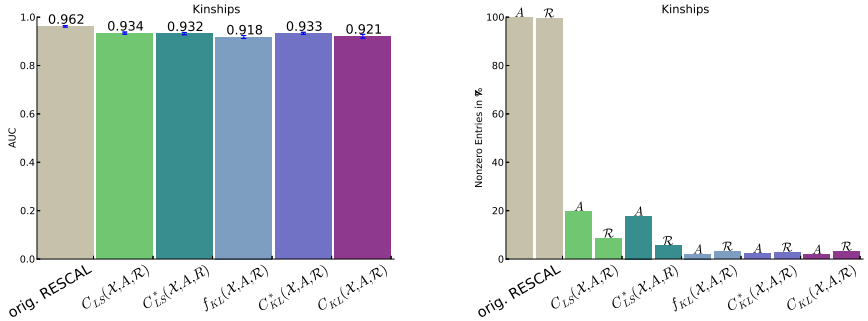
$$\widetilde{T_3} = \sum_k \widetilde{E}_k + \widetilde{A}\ diag\left(\mathbf{1}\left[\widetilde{F}_k \bullet \widetilde{A}\right]\right), \quad \widetilde{T_4} = \sum_k \widetilde{F}_k + \widetilde{A}\ diag\left(\mathbf{1}\left[\widetilde{E}_k \bullet \widetilde{A}\right]\right)$$

3 Experiments

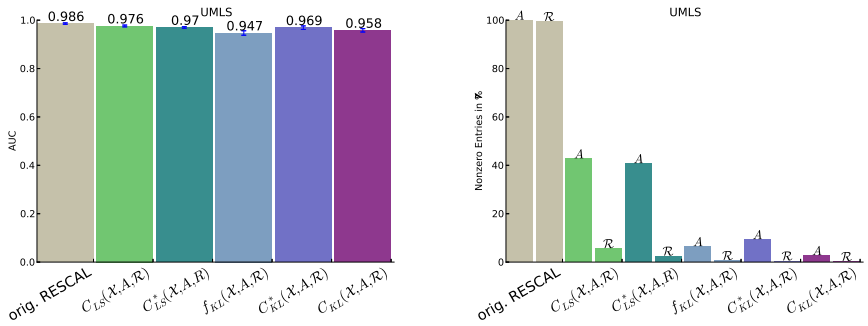
We conducted experiments on three different multi-relational datasets to evaluate the performance of the different non-negative extension of RESCAL:
Kinship $104 \times 104 \times 26$ multi-relational data that consist of several kinship relations within the Alwayarra tribe.
Nations $14 \times 14 \times 56$ multi-relational data that consist of relations between nations (treaties, immigration, etc). Additionally the dataset contains attribute information for each entity.



(a) Nations with attributes



(b) Kinships



(c) UMLS

Fig. 2. The plots on the left show the area under the precision-recall curve results of the different non-negative models and the original version of RESCAL on the Nations (a), Kinships (b) and UMLS (c) datasets. The plots on the rights show the amount of nonzero entries in the factor matrix A and the core tensor $\mathcal{R}$ respectively. For the description of the models, the notations from the methods section were used. From left to right: 1) Original version of RESCAL. 2) LS cost function with L_2-regularization on A and $\mathcal{R}$. 3) LS cost function with normalized A and L_1-regularization on $\mathcal{R}$. 4) KL-divergence cost function without regularization. 5) KL-divergence cost function with normalized A and L_1-regularization on $\mathcal{R}$. 6) KL-divergence cost function with L_1-regularization on A and $\mathcal{R}$. In case of the Nations dataset the attributes of the entities were included.

UMLS $135 \times 135 \times 49$ Multi-relational data that consist of biomedical relationships between categorized concepts of the Unified Medical Language System (UMLS).

For evaluation of the different methods, we performed a 10-fold cross-validation and initialized the factor matrix A by using the initial factors calculated by the Non-negative Double Singular Value Decomposition method (NNDSVD) [5] for all datasets. The sets for the cross-validation were chosen by randomly selecting entries throughout the whole tensor. In each iteration, the nonzero entries in the test-set were set to zero for training and predicted afterwards. The results were evaluated by using the area under the precision-recall curve. The sparsity of the factor matrices was inferred by defining every value that is smaller than 1.0e-9 and greater than -1.0e-9 as zero. The sparsity for each model was determined by taking the mean sparsity for A and $\mathcal{R}$ over all ten cross-validation iterations.

In figure 2 the results on the three datasets, Kinships, Nations and UMLS are shown. In case of the Kinships and UMLS dataset, the non-negative realizations of RESCAL are quite close to the AUC results of the original RESCAL model (left plots). RESCAL achieves 0.962 in Kinships and 0.986 in the UMLS dataset, where the non-negative least-squares approach with L_2 regularization has an AUC of 0.934 and 0.976, respectively, closely followed by the two normalized (A) variations. In case of the Nations dataset, the difference is more severe. RESCAL achieves an AUC of 0.771, where the best non-negative model achieves only 0.714. Between the different non-negative approaches, there is not much difference in AUC. Only the update functions for the KL-divergence without regularization seem to perform a little bit worse than the other non-negative approaches. Regarding the sparsity of the factor matrix A and the core tensor $\mathcal{R}$ the difference between the non-negative decompositions and RESCAL are more clear. As expected, the decomposition of the original RESCAL algorithm is almost completely dense (density: 99.9% for A and 99.8% for $\mathcal{R}$ in all datasets). It can be seen that in the case of the non-negative decompositions, the sparsity of the factor matrix A and the core tensor $\mathcal{R}$ increases dramatically with the size of the original matrix (and the factor matrices respectively). Taking the regularized KL-divergence based results for instance (rightmost), the density between the Nations and the UMLS dataset drops for A from 62% to 2.7% and for $\mathcal{R}$ from 48% to 0.2%. By comparing the results of the least-squares and KL-divergence based factorizations where A is kept normalized during minimization, it also seems that in the KL-divergence based cost functions sparsity of the factors is more enforced during minimization, especially for A. When factorizing the UMLS dataset, A has a density of about 40% (about 20% in Kinships) in the least-squares case, but at most 9% (2.5% in the Kinships) in the KL-divergence case.

4 Conclusion

We demonstrated, that the principles of non-negative matrix factorization can be extended to the RESCAL model for relational learning tasks, by employing

multiplicative update rules. We showed that the non-negative constraint can be introduced into the model with little loss in terms of predictive quality of the low rank approximations of the original data tensor but with a significant gain in sparsity of the latent factor representation. Additionally we presented different update rules for the factor matrices based on least-squares or the Kullback-Leibler divergence cost functions.

It has to be noted here, that running time and convergence rate in terms of number of iterations of the different proposed decompositions compared to the original RESCAL model have not been discussed yet. We have to point out, that we observed that the convergence through the multiplicative update rules tend to have more iterations and have clearly longer running times until convergence, even with non-random initialization of the factor matrices through the NNDSVD algorithm. A more accurate analysis of this issue is planned in the near future.

References

1. Wang, F, Li, P, König, AC: Efficient Document Clustering via Online Nonnegative Matrix Factorizations *In Proceedings* of SDM'11, 908–919 (2011)
2. Kohen, Y: The BellKor Solution to the Netflix Grand Prize. (2009)
3. Langville, AN, Meyer, CD, Albright R: Initializations for the Nonnegative Matrix Factorization. *KDD* (2006)
4. Lee, DD, Seung, HS: Learning the parts of objects by non-negative matrix factorization. *Nature*, 401, 788–791 (1999)
5. Boutsidis, C., Gallopoulos, E: SVD-based initialization: A head start for nonnegative matrix factorization. Pattern Recognition. 41(4), 1350–1362 (2008)
6. Harshman, RA: Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multi-modal factor analysis. *UCLA Working Papers in Phonetics*, 16, 1–84 (1970)
7. Carroll, JD, Chang, JJ: Analysis of individual differences in multidimensional scaling via an N-way generalization of "Eckert-Young" decomposition. *Psychometrika*, 35, 283–319 (1970)
8. Tucker, LR: Some Mathematical notes on three-mode factor analysis. *Psychometrika*, 31, 279–311 (1966)
9. Lee, D.D., Seung, H.S.: Algorithms for Non-negative Matrix Factorization. In *NIPS*, 556–562 (2000)
10. Nickel, M., Tresp, V., Kriegel, HP: A Three-Way Model for Collective Learning on Multi-Relational Data. In *Proceedings of the 28th International Conference on Machine Learning* (2011)
11. Nickel, M., Tresp, V., Kriegel, HP: Factorizing YAGO. Scalable Machine Learning for Linked Data. In *Proceedings of the 21st International World Wide Web Conference* (WWW2012) (2012)
12. Eggert, J., Körner, E.: Sparse coding and NMF. In *Neural Networks*, volume 4, pages 2529–2533 (2004)
13. Mørup, M., Hanson, L.K.: Algorithms for Sparse Non-negative Tucker decomposition. Neural Comput. 20(8), 2112–31 (2008)