NLA project report.
Nonnegative matrix factorization for link prediction in directed complex networks using PageRank and asymmetric link clustering information

Valerii Baianov, Dmitrii Leshchev, Gleb Mezentsev

December 2020

1 Team members and their contributions

- Valerii Baianov
  - Sparse implementation of algorithm
  - Algorithm pipeline implementation
  - Collecting datasets and spliting graph edges function
- Dmitrii Leshchev
  - Implementation of local structure information gathering
  - Derivatives comparison and new updating rules calculation
  - Comparison of received metrics (our) and article ones for the datasets
- Gleb Mezentsev
  - Implementation of global structure information gathering
  - Vectorization of implementation of local structure information gathering
  - Comparison of computational time and quality metrics of our version and article one.

2 Formal problem statement

An directed complex network can be denotes by a graph G(V,E), where $V = \{v_i\}_{i=1}^n$ and E are sets of nodes and links, respectively. Multiple links and self-loops are not allowed here. We use $A = [a_{ij}]_{n\times n}$ to represents the adjacent matrix of graph G, where $a_{ij} = 1$ if an link exists between nodes v_i and v_j, and $a_{ij} = 0$ otherwise. Clearly, A is asymmetric matrix, namely $a_{ij} \neq a_{ji}$.

Let U stand for the universal set of all possible links. Then $U - E$ is set of nonexistent links. The goal of link prediction is to find missing links or predict possible links in the future. To validate the algorithm accuracy, we randomly divide the observed link set E into two parts: the training ET and the probe set EP. The former is considered as given information and the latter is only used for testing.

Clearly, $E^T \cap E^P = \phi$ and $E^T \cup E^P = E$

Basic nonnegative matrix factorization model: nonnegative matrix A aims to find two nonnegative matrices $U \in R_+^{n\times K}, V \in R_+^{n\times K}$ whose product can well approximate original matrix A. NMF can be formulated as:

$$A \approx UV^T \quad s.t. \quad U \geq 0, V \geq 0$$

Here, K is the dimension of latent space, A denotes the original matrix,U denotes the basic matrix, V denotes the coefficient matrix. Due to $K < n$ dimension reduction is achieved and a lower dimensional representation of A in a K-dimensional space is given by V. This decompose aims to solve the following F-norm optimization problem:

$$\mathcal{L}_{NMF} = \min_{U\geq 0, V\geq 0} \|A - UV^T\|_F^2 \tag{1}$$

where $U \in R_+^{n\times K}$ *and* $V \in R_+^{n\times K}$ denotes the base matrix and coefficient matrix, respectively. $\|\cdot\|_F$ is F-norm.

For NMF-AP, which is proposed in this article, authors use both local and global structures information, so the final F-norm optimization problem is the following:

$$\min_{U\geq 0, V\geq 0, W\geq 0} \mathcal{L} = \mathcal{L}_{local} + \gamma\mathcal{L}_{global} + \beta(\|U\|_F^2 + \|V\|_F^2) \tag{2}$$

or in more detail

$$\min_{U\geq 0, V\geq 0, W\geq 0} \mathcal{L} = \|Y \circ (A - UV^T)\|_F^2 + \gamma\|C - UWU^T\|_F^2 + \beta(\|U\|_F^2 + \|V\|_F^2, \tag{3}$$

, where $Y = 1 + \alpha S$.
The way of achieving this formula will be described in the next section.

3 Ideas and mathematical description of the algorithm

To collect local information authors use two matrices S_0 and S_m. S_0 is just an adjacency matrix and S_m is the asymmetric link clustering coefficient score, which is defined via asymmetric link clustering coefficients.

The asymmetric link clustering coefficients can be defined according to the literature as follows:

$$S_m^{iz} = \frac{|\Gamma_{in}(i) \cap \Gamma_{out}(z)|}{k_{out} - 1} \tag{4}$$

where $|\Gamma_{in}(i)\Gamma_{out}(z)|$ represents the number of common neighbors and $k_{out}(j)$ represents the out degree of j. Obviously, $S_m^{iz} \neq S_m^{zi}$

Through Eq. (4), the similarity of nodes v_i and v_j can be obtained as defined follow:

$$S_m^{ij} = \sum_{z \in \Gamma_{in}(i) \cap \Gamma_{out}(z)} S_m^{iz} \tag{5}$$

It is obvious that S_m^{ij} is not equal to S_m^{ji} from the definition of Eq. (5). This is exactly same as that in Eq. (5), S_m^{ji} can be as defined follow:

$$S_m^{ji} = \sum_{z \in \Gamma_{in}(j) \cap \Gamma_{out}(z)} S_m^{jz} \tag{6}$$

The final asymmetric clustering link score matrix is constructed by taking the maximum of Eqs. (5) and (6).

$$S_m = \max\{S_m^{ij}, S_m^{ji}\} \tag{7}$$

To extract more useful local structural information, the final similarity matrix $S = S_m + S_0$ is obtained by S_0 and S_m, where S_m and S_0 complement each other. Specifically, S is added as an indicator matrix to the Eq. (1) as defined below:

$$\mathcal{L}_{local} = \min_{U \geq 0, V \geq 0} ||(1 + \alpha S) \circ (A - UV^T)||_F^2 \tag{8}$$

Real-world networks are often so sparse that the observed links only account for a small portion. Therefore, only capturing the local information is not sufficient. Authors introduce the PageRank algorithm to obtain the global topological information.

Thus, authors adopt PageRank algorithm to calculate the influence score of node. The influence score of node is defined as follows:

$$c_i = \frac{1}{N}(1-p) + p * \sum_{j=1}^{N} \frac{a_{ij}}{k_j^{out}} * c_j \tag{9}$$

where, c_i represents the influence score of the ith node and $c \in [0, 1]$, p is the damping coefficient and empirically set $p = 0.85$, k_j^{out} indicates the out-degree of the jth node and a_{ji} is adjacency matrix of any directed network. Then set C is the influence score matrix of nodes, which is constructed as:

$$C_{ij} = \begin{cases} c_i, a_{ij} \neq 0 \\ 0, a_{ij} = 0 \end{cases} \tag{10}$$

Therefore, C contains all the global information of the observed network. To fully exploit the global information, authors further map C to the low-dimensional latent space U, and authors obtain the following minimizing loss function.

$$\mathcal{L}_{global} = \min_{U \geq 0, W \geq 0} ||C - UWU^T||_F^2 \tag{11}$$

where $U \in R_+^{n \times K}, W \in R_+^{k \times K}$ are nonnegative. Each row U_i in the base matrix U can be viewed as a low-dimensional latent representation of the corresponding node v_i, which denotes the influence information of node. W provides the degree of freedom to make the factorization accurate.

By integrating the local and global information, the authors propose a unified model NMF-AP for link prediction. The optimized objective function is expressed as follows:

$$\min_{U\geq 0,V\geq 0,W\geq 0} \mathcal{L} = \mathcal{L}_{local} + \gamma\mathcal{L}_{global} + \mathcal{L}_{reg} \tag{12}$$

where $\circ$ denotes the hadamard product, β is a positive parameter that prevent over-fitting, α and γ are a positive parameter that control the contribution of local and global information, respectively. By seeking an optimal value of $\mathcal{L}$, we can simultaneously preserve both local and global structure information.

For the sake of simplicity, let $Y = 1 + \alpha S$, we rewrite the Eq. (12):

$$\min_{U\geq 0,V\geq 0,W\geq 0} \mathcal{L} = ||Y \circ (A - UV^T)||_F^2 + \gamma||C - UWU^T||_F^2 + \beta(||U||_F^2 + ||V||_F^2) \tag{13}$$

Since the optimization problem in Eq. (13) is not convex in three variables U,V and W, it is infeasible to find the global minimum. Here we use the lagrangian multiplication rule algorithm that iteratively updates U with fixed V and W, V with fixed U and W and W with fixed U and V, which guarantees the objective function values do no increase with iterations. According to matrix trace properties: $Tr(A) = Tr(A^T)$, $Tr(AB) = Tr(BA)$, we can rewrite Eq. (13) as follows

$$\begin{aligned}\mathcal{L} = Tr[\ (Y \circ (A - UV^T))(Y \circ (A - UV^T))^T] +& \\ \gamma Tr[\ (C - UWU^T)(C - UWU^T)^T]\ &+ \beta(Tr(UU^T) + Tr(VV^T))\end{aligned} \tag{14}$$

Authors introduce a lagrange multiplier matrix $\Phi = [\phi_{nk}] \in R^{n\times K}$, $\Psi = [\psi_{nk}] \in R^{n\times K}$ and $\Theta = [\theta_{kk}] \in R^{K\times K}$ for the constraint $U \geq 0$, $V \geq 0$ and $W \geq 0$ then we rewrite Eq. (14) as follows:

$$\begin{aligned}\mathcal{L} = Tr[\ (Y \circ (A - UV^T))(Y \circ (A - UV^T))^T] +& \\ \gamma Tr[\ (C - UWU^T)(C - UWU^T)^T]\ &+ \beta(Tr(UU^T) \\ &+ Tr(VV^T)) + Tr(\Phi U^T) + Tr(\Psi V^T) + Tr(\Theta W^T)\end{aligned} \tag{15}$$

Updating U: to update U with V and W fix, setting the derivative of $\mathcal{L}$ with respect to U, authors get:

$$\frac{\partial \mathcal{L}}{\partial U} = -Y \circ AV + Y \circ (UV^T)V + \beta U - \gamma CUW^T + \gamma UWU^TUUW^T + \Phi \tag{16}$$

By the Karush-Kuhn-Tucker(KKT) condition $\phi_{nk}u_{nk} = 0$, authors have:

$$[-Y \circ AV + Y \circ (UV^T)V + \beta U - \gamma CUW^T + \gamma UWU^TUUW^T]U_{nk} = 0 \tag{17}$$

According to Eq. (17), authors present the following updating rule.

$$U_{nk} \leftarrow U_{nk}\frac{[Y \circ AV + \gamma CUW^T]_{nk}}{[Y \circ (UV^T)V + \gamma UWU^TUUW^T]_{nk}} \tag{18}$$

Updating V: to update V with U and W fix, setting the derivative of $\mathcal{L}$ with respect to V, authors get:

$$\frac{\partial \mathcal{L}}{\partial V} = -(Y \circ A)^TU + (Y \circ (UV^T))^TU + \beta V + \Psi \tag{19}$$

By the Karush-Kuhn-Tucker(KKT) condition $\psi_{nk}v_{nk} = 0$, authors have:

$$[-(Y \circ A)^TU + (Y \circ (UV^T))^TU + \beta V + \Psi]V_{nk} = 0 \tag{20}$$

According to Eq. (20), authors present the following updating rule.

$$V_{nk} \leftarrow V_{nk}\frac{[(Y \circ A)^TU]_{nk}}{[(Y \circ (UV^T))^TU + \beta V]_{nk}} \tag{21}$$

Updating W: to update W with U and V fix, setting the derivative of $\mathcal{L}$ with respect to W, authors get:

$$\frac{\partial \mathcal{L}}{\partial W} = -U^T C U + U^T U W U^T U + \Theta \tag{22}$$

By the Karush-Kuhn-Tucker(KKT) condition $\theta_{kk} w_{kk} = 0$, authors have:

$$[-U^T C U + U^T U W U^T U + \Theta] W_{kk} = 0 \tag{23}$$

According to Eq. (23), authors present the following updating rule.

$$W_{kk} \leftarrow W_{kk} \frac{[U^T C U]_{kk}}{[U^T U W U^T U]_{kk}} \tag{24}$$

We acquire new the basic matrix U and the feature matrix V by minimizing Eq. (14)). Finally, we calculate the similarity score of original network with $A = UV^T$

However, we found the typo in the updating rule for U. The denominator has the following term:

$$\gamma U_{n\times k} W_{k\times k} U^T_{k\times n} U_{n\times k} U_{n\times k} W^T_{k\times k} \quad = \quad \gamma (U_{n\times k} W_{k\times k} U^T_{k\times n} U_{n\times k})(U_{n\times k} W^T_{k\times k}) \quad = \quad \gamma F_{n\times k} S_{n\times k} \tag{25}$$

As can be seen, it's impossible to multiply rectangular matrices with equal dimensions. Maybe, it's just a typo and we should drop the extra U. But we decided to recalculate the derivatives:

$$\frac{\partial \mathcal{L}}{\partial U} = 2\beta U - (2(Y \circ (A - UV^T) \circ Y)V + 2\gamma(C - UWU^T)UW^T + 2\gamma(C^T - UW^T U^T)UW) \tag{26}$$

$$\frac{\partial \mathcal{L}}{\partial V} = 2\beta V - 2(Y^T \circ (A^T - VU^T) \circ Y^T)U \tag{27}$$

$$\frac{\partial \mathcal{L}}{\partial W} = -2\gamma U^T (C - UWU^T) U \tag{28}$$

Using the Karush-Kuhn-Tucker(KKT) condition again, we get new updating rules:

$$U_{nk} \leftarrow U_{nk} \frac{[(Y \circ A \circ Y)V + \gamma(CUW^T + C^T UW)]_{nk}}{[(Y \circ (UV^T) \circ Y)V + \gamma(UWU^T UW^T + UW^T U^T UW) + \beta U]_{nk}} \tag{29}$$

$$V_{nk} \leftarrow V_{nk} \frac{[((Y \circ A)^T \circ Y^T)U]_{nk}}{[((Y \circ (UV^T))^T \circ Y^T)U + \beta V]_{nk}} \tag{30}$$

$$W_{kk} \leftarrow W_{kk} \frac{[U^T C U]_{kk}}{[U^T U W U^T U]_{kk}} \tag{31}$$

We did not find an explanation for the exclusion of some terms, so we made experiments with both our formulas and the paper ones. The comparison of computational time and metric values will be given in the next section.

Now lets consider another ideas we used to speed up the process and reduce memory usage during working on a project

1. Firstly, we wanted to rewrite formulas for S_m (4) - (7) in the matrix format in order to do computations faster.

   (a) Lets start with optimal computing S_m^{iz} (4).

   $$S_m^{iz} = \frac{|\Gamma_{in}(i) \cap \Gamma_{out}(z)|}{k_{out} - 1}$$

   It is obvious that we have a number of paths of length 2 from node z to node i in numerator and a sum of a z-th column of an adjacency matrix minus 1 in denominator. We want to define

S_m^{iz} only for the pairs of nodes having a directed edge from z to i, so the final matrix formula for element of matrix S_m^{iz} looks like follows.

$$s_m^{iz} = a_{zi} \frac{A^2}{\sum_{k=1}^{n} a_{zk} - 1}$$

Or for the whole matrix:

$$S_m^{iz} = A^T \circ (A^2 \times Diag(\frac{1}{\sum_{k=1}^{n} a_{zk} - 1}))$$

(b) We also want to rewrite the formula (5) for S_m^{ij} in the matrix form.

$$S_m^{ij} = \sum_{z \in \Gamma_{in}(i) \cap \Gamma_{out}(z)} S_m^{iz}$$

A way to do that is to create a boolean mask in tensor form, which would indicate which elements should be preserved in the sum for each pair of i and j. If we had such tensor, then the S_m^{ij} could be computed as sum over one of the axis of the elementwise product of two tensors, where one of the tensors is our mask and the other is tensor produced by broadcasting of S_m^{iz} (Fig. 1).

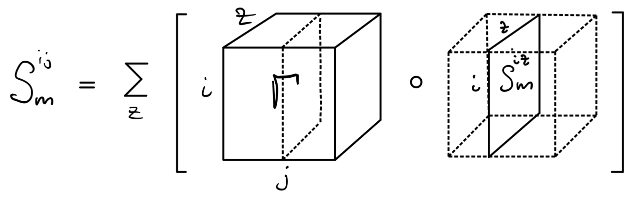


Figure 1: Tensor representation of S_m^{ij}

Let's show how to construct such tensor Γ. γ_{ijz} should be equal to 1, if node z is a neighbour of node i and node j at the same time. Also let's note that z is a neighbour of i if either $a_{iz} = 1$ or $a_{zi} = 1$, so we can write down Γ in the following way (Fig. 2) (orientation of matrices inside the tensors denote the axis in which we broadcast them).

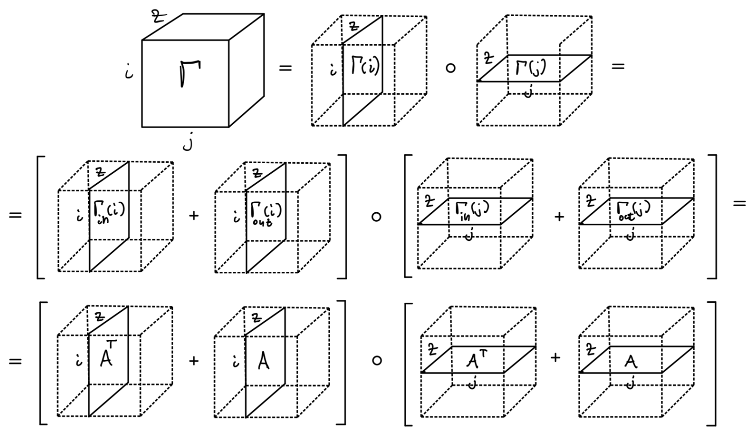


Figure 2: Mask tensor Γ

Now we can write down the formulas for S_m^{ij} and see, that they can be rewritten without forming any tensors (Fig. 3).

The final formula for computing S_m^{ij} looks like the following.

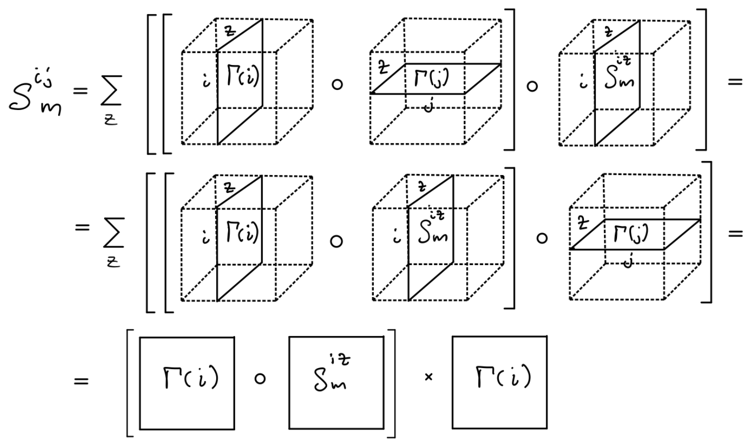


Figure 3: Calculation of S_m^{ij} without forming tensors

$$S_m^{ij} = ((A + A^T) \circ S_m^{iz})(A + A^T)$$

It can be shown in the same way that the formula for S_m^{ji} looks similar.

$$S_m^{ji} = (A + A^T)((A + A^T) \circ S_m^{T\,iz}) = S_m^{T\,ij}$$

2. Secondly, we do not form the matrix $Y = 1 + \alpha S$ directly, not to break sparsity of matrix S by adding all ones and multiply 1 and αS on matrices separately. By doing that, we receive the following update formulas.

   $U_{nk} \leftarrow U_{nk} \frac{[(A+2\alpha S\circ A+\alpha^2 S\circ S\circ A)V+\gamma(CUW^T+C^TUW)]_{nk}}{[(UV^TV+(2\alpha S\circ(UV^T))V+(\alpha^2 S\circ(UV^T)\circ S)V)+\gamma(UWU^TUW^T+UW^TU^TUW)+\beta U]_{nk}}$

   $V_{nk} \leftarrow V_{nk} \frac{[(A^T+2\alpha(A\circ S)^T+\alpha^2(A\circ S)^T\circ S^T)U]_{nk}}{[VU^TU+(2\alpha S^T\circ(VU^T))U+((\alpha^2 S^T\circ S^T)\circ(VU^T))U+\beta V]_{nk}}$

   $W_{kk} \leftarrow W_{kk} \frac{[U^TCU]_{kk}}{[U^TUWU^TU]_{kk}}$

3. Thirdly, we are changing an order of matrix multiplication in updating formulas for U (formulas (18) and (29)). More specifically, we are talking about the term $UW^TU^TUW^T$.

   The complexity of the direct multiplication UW^TU^TUW is $O(kn^2)$

   If we change the order as follows $U(W^T(U^TU)W)$ the complexity will be $O(nk^2)$.

   As the dimension of matrices is $U_{n\times k}, W_{k\times k}$ and $k < n$, we reduce the complexity.

Algorithm NMF-AP itself is the following:

```
Input:
A: adjacency matrix of directed network;
K: dimension of latent space;
Niter: maximum number of iterations;
Paramenters: $\alpha, \beta, \gamma$;
Output:
Similarity score matrix A
1: Divide A into training set $E_T$ and probe set $E_P$
2: Randomly initialize U, V, W
3: Preserve the local information according to (8)
4: Preserve the global information according to (11)
5: For t=1:iter do
6: Update U according to (18)
7: Update V according to (21)
```

```
8: Update W according to (24)
9: Get U and V after convergence;
10: endfor
11: Compute probability matrix for link prediction $A = UV^T$
```

Computational complexity analysis (from the paper):
The computational cost of the NMF-AP algorithm uses the multiplicative updating rules to optimize the objective function. In each iteration, updating U according to (18) requires $O(n^2 * K * N_{iter})$, and updating V according to (21) requires $O(n^2 * K * N_{iter})$. Therefore, the total time complexity of NMF-AP is $O(n^2)$.
Computational complexity analysis (for our formulas): If comparing exact number of operations, our formulas have more matrix multiplications, so the total number is higher. However, in big-O notations we have the same $O(n^2 * K * N_{iter})$ for updating formulas of U (29) and V(30). Therefore, the total time complexity of NMF-AP with our updating formulas is still $O(n^2)$.

4 Experiments description and results.

A description of the graphs we worked with can be seen in the table (1) below. $|V|$ is the number of nodes and $|E|$is the number of links. CC is the clustering coefficient of node. $< k >$ is the average degree, $< d >$ is the average distance, k^{in}_{max} and k^{out}_{max} are max in-degree and max out-degree, respectively.

Network	$\|V\|$	$\|E\|$	k^{in}_{max}	k^{out}_{max}	CC	$< k >$	$< d >$
Celegans	306	2345	134	39	0.1685	15.3268	2.8838
Wikivote	7115	201524	1065	963	0.1409	56.6476	7.8357
SG	1059	4918	89	232	0.1488	24.8040	0.2115
SciMet	3084	10,412	121	105	0.0757	6.7523	0.1343
OpenFlights	2939	30,501	236	237	0.4380	20.7560	3.9970

Table 1: Graphs description table

As we found typos and inexplicable for us transmissions of terms in formulas, on which the algorithm is based, we compared both variations of the algorithms in further experiments.
We began with comparing computational time of algorithm with formulas from the paper (Original), with our updating formulas(New(dense)) and with our sparse implementation(New(sparse)). As we can see from Figure (4) implementation of the original algorithm is slightly faster then dense version of our version (due to less number of operations). Sparse version, is faster only for one of the datasets. But the main idea of sparsity is saving memory, which is achieved (we made few experiments with larger graphs, and sparse version worked, while dense one failed due to an excess of a memory limit)

Authors compared four metrics, AUC(the area under the receiver operating characteristic curve), Precision, F-Score and AUPR for the accuracy measurement

AUC can be interpreted as the probability that a missing links has higher score than a nonexistence links. In practice, among n independent comparison, we randomly pick a missing link and a nonexistence link at each time to compare their score. If n_1 times the missing link having a higher score and n_2 times they have the same score, the AUC value is:

$$AUC = \frac{n_1 + 0.5 * n_2}{n}$$

Precision is calculated using the predictions on the non-observed (during training) links (existing links from probe set + non-existing links). The L non-observed links with the highest predicted probabilities are selected (in our case we selected L equal to the number of links in the probe set) and the precision is then defined as the ratio of relevant selected objects (actual links) to the all selected objects within those L.

$$Precision = \frac{L_r}{L}$$

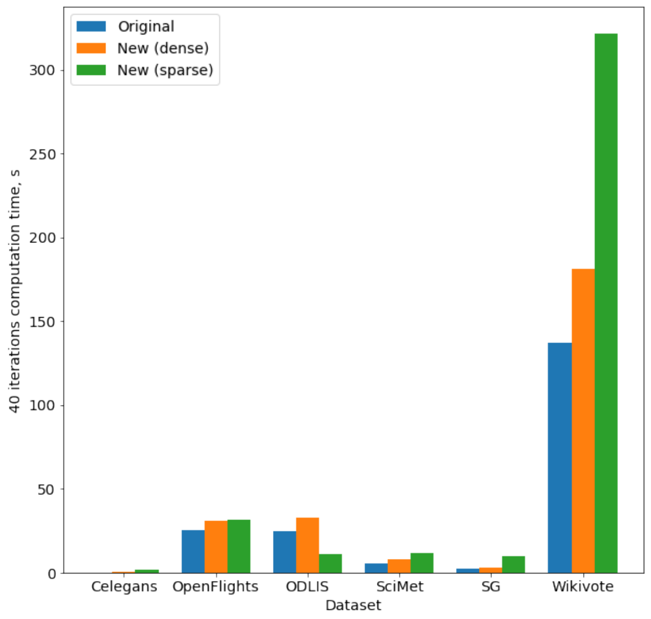


Figure 4: Computational time comparison

Recall is defined in a similar way. It is a ratio of relevant selected objects within selected in the same way set of objects to the total number of relevant objects in the probe set.

$$Recall = \frac{L_r}{R}$$

F-score then is defined by precision and recall.

$$F - score = \frac{2 \cdot Recall \cdot Precision}{Recall + Precision}$$

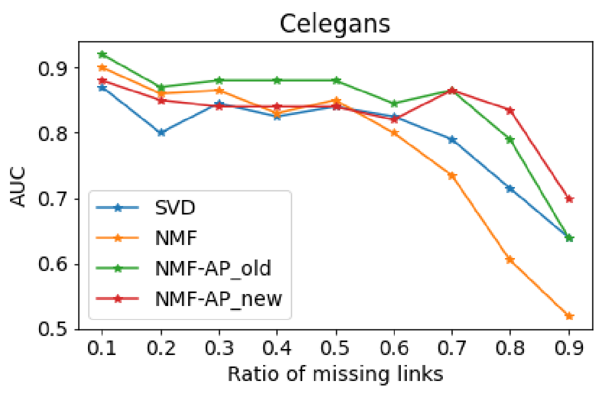
AUPR: Precision-Recall curves are threshold curves. Each point corresponds to a different score threshold with a different Precision and Recall value. AUPR is an area under the curve obtained this way.

We compared AUC metric value for four algorithms NMF-AP_old (from the paper), NMF-AP_new (implementation with our updating formulas), basic NMF and SVD. From Figure 5 we can see, that we have the same results as in the paper. NMF-AP_old outperformes both SVD and NMF. NMF-AP_new is initially a bit worse, but for higher values of uncertainty (ration of missing links > 0.7) it outperforms even NMF-AP_old.

For four another datasets: OpenFlights, SciMet, SG and Wikivote we got the same results.NMF-AP_old outperformes other methods, as in the paper, NMF-AP_new shows better AUC for hight values of ratio. Figure (6)

We also compared AUPR, however, results differ from the paper. In our experiments for all datasets SVD and NMF gave better results, than our implementations. The description of precision and recall and, as consequence, of AUPR wasn't clear from the article, so we guess, that the difference is caused by different ways of computing this metrics. Resulting plots can be seen on figures 7 and 8

After comparing different algorithms on the same datasets we also compared different datasets on the algorithms we implemented (NMF-AP_old and NMF-AP_new). We measured AUC score for different values of K (dimension of latent space) for the datasets we were discussing before. The result is illustrated on Figure (9). In some cases they are similar to the paper one (values for Openflights, orders of Scimet and



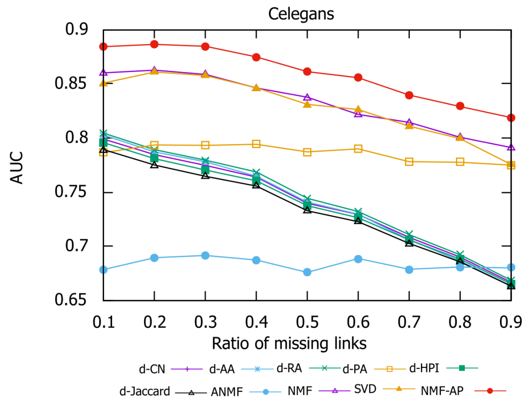


Figure 5: Our implementation (left) compared to plot from the paper (right) (AUC)

lower results for celegans). An implementation with our formulas has similar results for Openflights and and Celegans, bur Scimets AUC is a bit better in our case. Also, both our implementations show the highest AUC on Wikivote, while in the paper values are lower. It can be caused by randomness (cause the way of calculating AUC is not standard in this case.

5 List of references.

1. Nonnegative matrix factorization for link prediction in directed complex networks using PageRank and asymmetric link clustering information (Original article)
2. Asymmetric link clustering coefficients (paper)

6 Link to the public GitHub repository with code to reproduce your results.

GitHub repository (The code could be run just with ”Run all” in order to reproduce results, however, it would take a few hours, because there are quite a few experiments with several graphs, algorithms, and algorithm parameters)

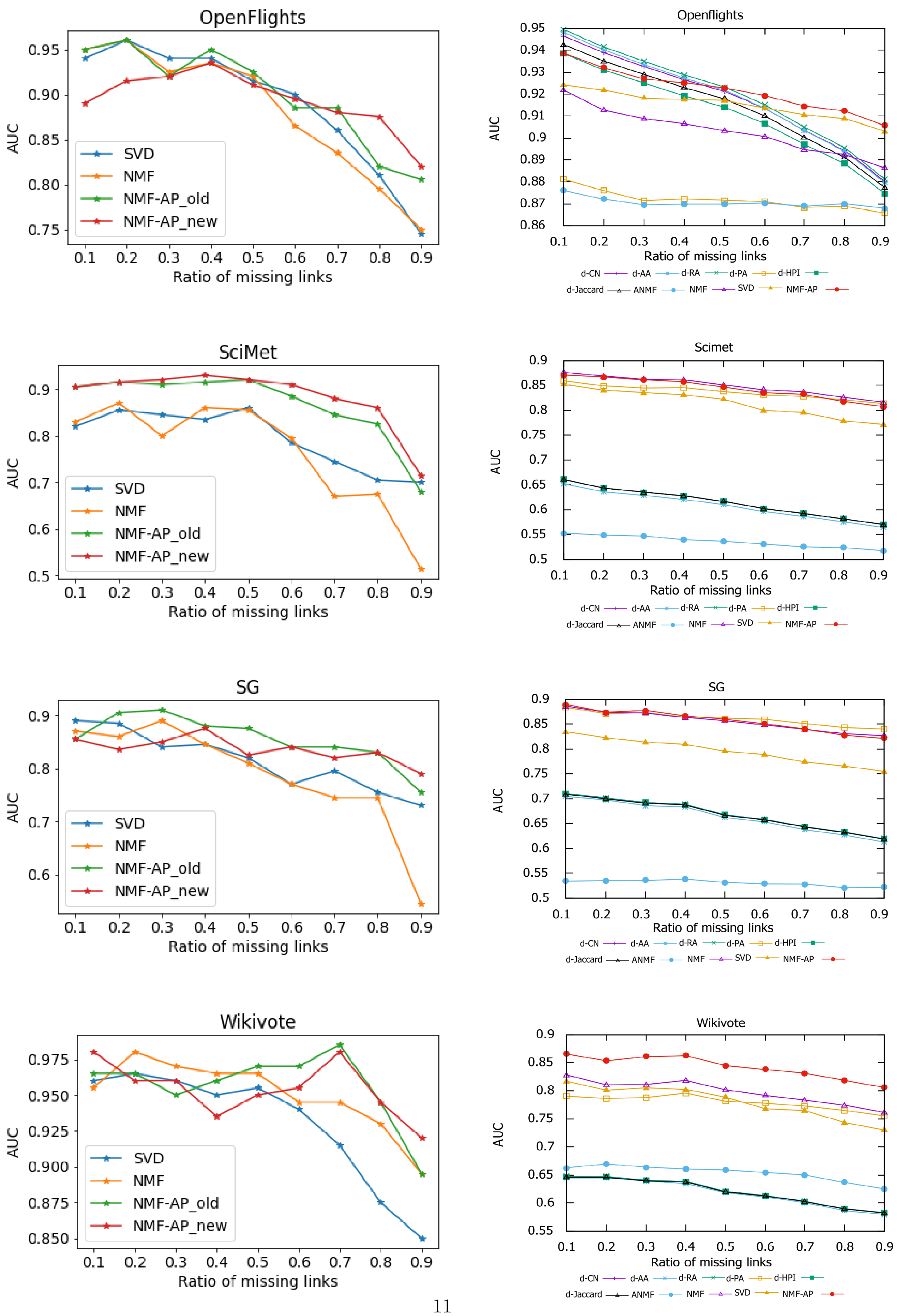


Figure 6: Comparing AUC on other datasets

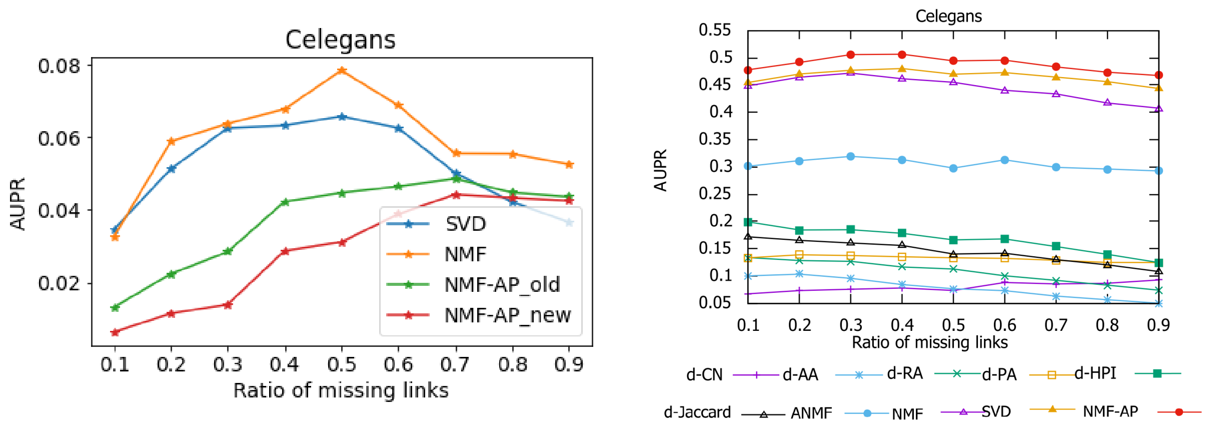


Figure 7: Our implementation (left) compared to plot from the paper (right) (AUPR)

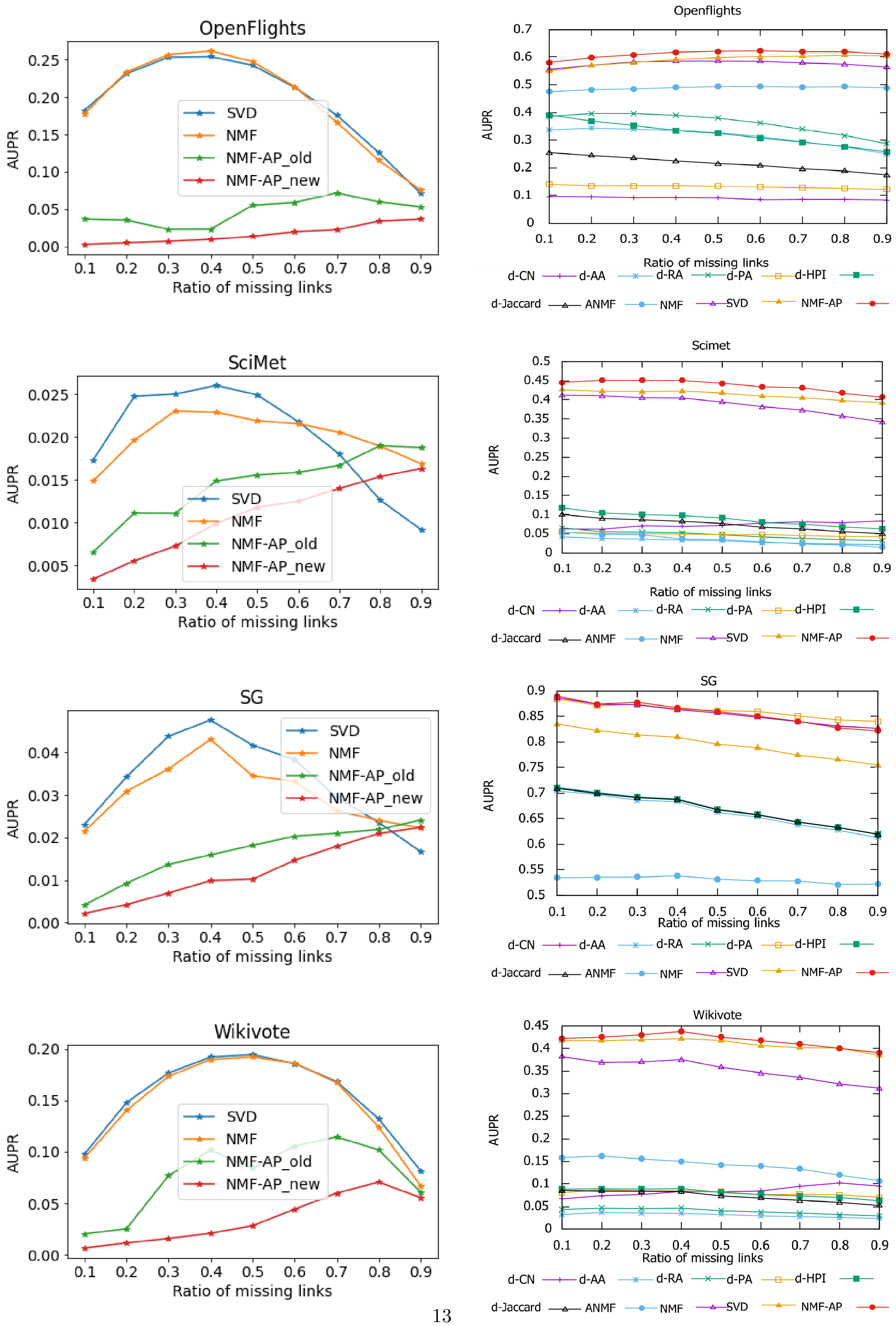


Figure 8: Comparing AUPR on other datasets

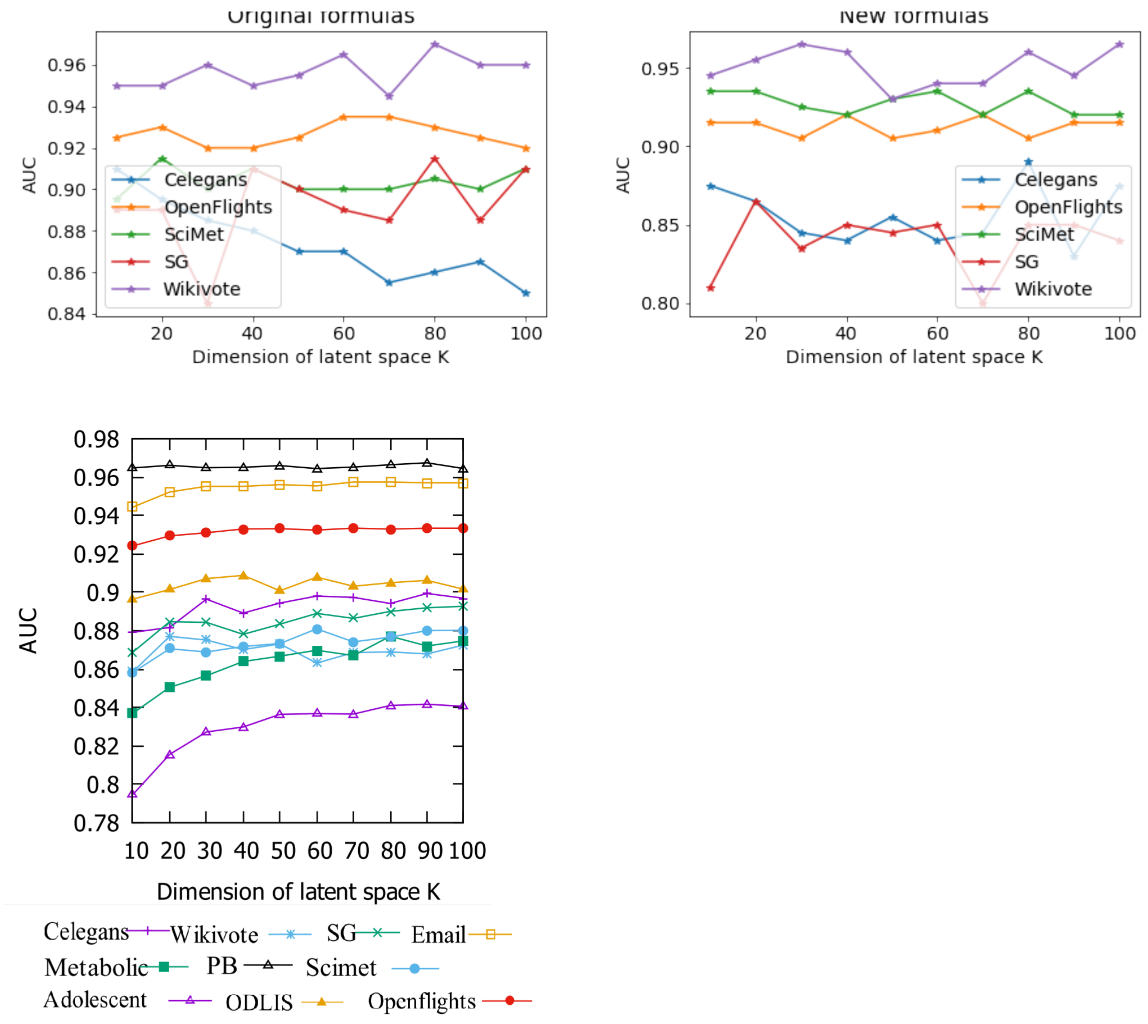


Figure 9: Comparing different implementations AUC on all datasets