FastStep: Scalable Boolean Matrix Decomposition

Miguel Araujo[1,2], Pedro Ribeiro[1], and Christos Faloutsos[2]

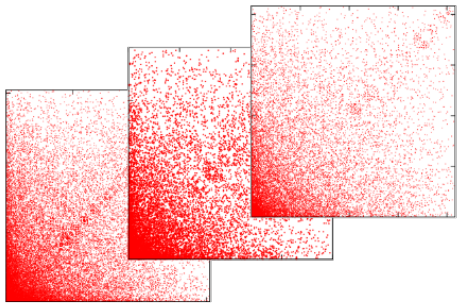
[1] Cracs/INESC-TEC and University of Porto, Porto, Portugal
pribeiro@dcc.fc.up.pt
[2] Computer Science Department, Carnegie Mellon University, Pittsburgh, USA
{maraujo,christos}@cs.cmu.edu

Abstract. Matrix Decomposition methods are applied to a wide range of tasks, such as data denoising, dimensionality reduction, co-clustering and community detection. However, in the presence of boolean inputs, common methods either do not scale or do not provide a boolean reconstruction, which results in high reconstruction error and low interpretability of the decomposition. We propose a novel step decomposition of boolean matrices in non-negative factors with boolean reconstruction. By formulating the problem using threshold operators and through suitable relaxation of this problem, we provide a scalable algorithm that can be applied to boolean matrices with millions of non-zero entries. We show that our method achieves significantly lower reconstruction error when compared to standard state of the art algorithms. We also show that the decomposition keeps its interpretability by analyzing communities in a flights dataset (where the matrix is interpreted as a graph in which nodes are airports) and in a movie-ratings dataset with *10 million* non-zeros.

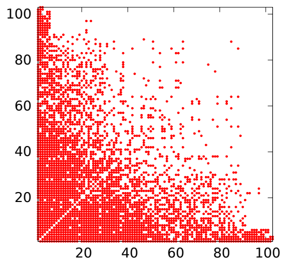
1 Introduction

Given a boolean who-watched-what matrix, with rows representing users and columns representing movies, how can we find an interpretation of the data with low error? How can we find its underlying structure, helpful for compression, prediction and denoising? Boolean matrices appear naturally in many domains (e.g. user-reviews [3] or user-item purchases [16], graphs, word-document co-occurrences [4] or gene-expression datasets [17]) and describing the underlying structure of these datasets is the fundamental problem of community detection [6] and co-clustering [15] techniques.

We address the problem of finding a low-rank representation of a given $n \times m$ boolean matrix $\mathbf{M}$, with small reconstruction error while easily describing $\mathbf{M}$'s latent structure. We propose FastStep, a method for finding a non-negative factorization that, unlike commonly used decomposition methods, yields the best interpretability by combining a **boolean reconstruction** with **non-negative factors**. This combination allows FastStep to find structures that go beyond blocks, providing more realistic representations. Figure 1a showcases three communities (representing 3 venues) in the DBLP dataset that illustrate the important hyperbolic structures found in real data; compare them to the community



(a) **DBLP real communities** - PAKDD, KDD and VLDB.



(b) **FastStep community** - American airports.

Fig. 1: **Realistic hyperbolic structure -** Adjacency Matrices of real communities in `DBLP` and a community found by FastStep.

found by FastStep in Figure 1b representing the American community in the `Airports` dataset.

Using our scalable method, we analyze two datasets of movie ratings and airports flights and show FastStep's interpretability power with intuitively clear and surprising decompositions. As an additional example, Figure 2 illustrates an application of FastStep to the task of community detection. Using route information alone, the world airports are decomposed in 10 factors that clearly illustrate geographical proximity. As we explain in more detail in section 4.3, the communities we find have an arbitrary marginal and do not need to follow a block shape.

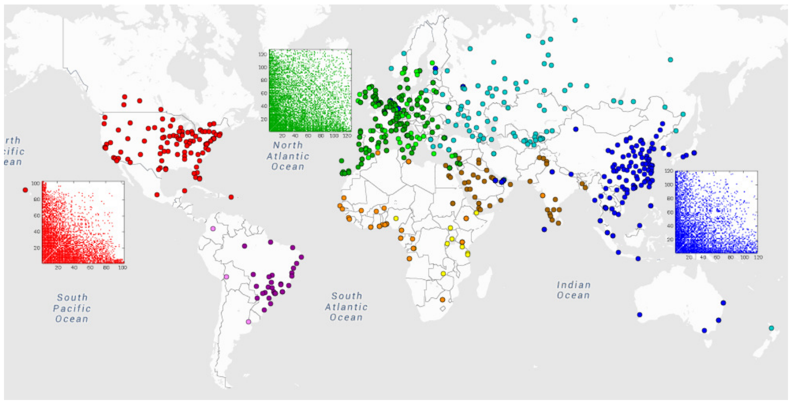


Fig. 2: **Intuitive non-block communities** - Communities automatically found in the `Airports` dataset from flight records. (best viewed in color)

2 Background and Related Work

Real and Non-Negative Matrix Decompositions. In the Singular Value Decomposition (SVD) [7], a real matrix $\mathbf{M}$ is decomposed into $\mathbf{U\Sigma V^T}$ where $\mathbf{U}$ and $\mathbf{V}$ are real orthogonal matrices and $\mathbf{\Sigma}$ is a $k \times k$ non-negative diagonal matrix. While the Eckart-Young theorem [5] proves this to be the best approximation using regular matrix multiplication and real entries, negative values in the factor matrices make it hard to interpret. What does it mean for an element to have a negative score in a component? For non-negative $\mathbf{M}$, Non-Negative Matrix Factorization (NNMF) [9] methods were developed to overcome this problem.

Neither of these methods have clear extensions to the boolean case as the reconstructed matrix is not boolean. One simple idea is rounding or thresholding the reconstructed matrix, but no guarantee can be given on the reconstruction error. Another possibility is thresholding the factor matrices and using boolean algebra in order to obtain a boolean reconstruction, but selecting the appropriate threshold is a difficult problem as a clear cut-off might not exist.

Decomposition of Boolean Matrices. Tao Li [11] proposed an extension of the K-means algorithm to the two-sided case where $\mathbf{M}$ is decomposed into $\mathbf{AXB^T}$ with $\mathbf{A}$ and $\mathbf{B}$ binary, and an alternating least squares method when $\mathbf{X}$ is the identity matrix. Pauli Miettinen showed that Boolean Matrix Factorizations (BMF) methods could achieve lower reconstruction error than SVD in boolean data and proposed an algorithm using association rules (ASSO) which exploits the correlations between columns, but unfortunately it's time complexity is $O(nm^2)$. Zhang et al. [17] proposed two approaches for BMF, one using a penalty in the objective function (BMF-PENALTY) which achieved good results for dense datasets, and an alternative thresholding method (BMF-THRESH) which by thresholding factor matrices is better suited for sparse datasets. None of these methods is scalable and they have the problem of forcing a tiling of the data matrix, as each factor is effectively treated as a block. In particular, the notion of "importance" inside a cluster, which previously existed in NNMF, is now lost and the analysis of the resulting factors is limited. In Logistic PCA (L-PCA), Schein et al. [12] replace PCA's Gaussian assumption with a Bernoulli distribution and fit their new model using an alternating least squares algorithm that maximizes the log-likelihood. Their alternating algorithm has running time $O(nm)$ when applied to a $n \times m$ matrix and therefore does not scale. It is also hard to interpret due to the possibility of negative values in the factors.

Related techniques. There is a strong relationship between boolean matrices and graph data, where matrix decompositions are linked to community detection and graph partitioning, but we would like to refer the reader to a review on spectral algorithms in this area for further details [6]. However, recent work such as the Hyperbolic Community Model [2] has shown the non-uniform nature of real-world communities and has highlighted the need for boolean decomposition methods which do not discard node importance.

An important aspect of fast decomposition methods is their ability to evaluate the reconstruction error $||M - R||_F^2$ in less than quadratic time. Leskovec et al. [10] approximated the log-likelihood of the fit by exploiting the Kronecker nature of their generator. In the Compact Matrix Decomposition [14], Jimeng Sun et al. approximated the sum-square-error (SSE) by sampling a set of rows and columns and scaling the error in the submatrix accordingly.

Table 1 provides a quick comparison of some of the methods discussed in this section. We characterize as *Beyond blocks* methods who do not force a rectangular tiling of the data. *Arbitrary Marginals* refers to a method's ability to represent any marginal in the data (e.g. rectangles, but also triangles or hyperbolic structures). We define *interpretability* as the ability to easily select a subset of elements representing a factor. Given our focus on efficient decompositions, we will limit our comparison in section 4 to scalable methods.

Table 1: Comparison of decomposition methods - **FastStep combines interpretability and beyond block structures for large datasets.**

	FastStep	SVD	NNMF	ASSO	THRESH	HyCoM	L-PCA
Scalability	✓	✓	✓			✓	
Overlapping	✓	✓	✓	✓	✓	✓	✓
Beyond blocks	✓	✓	✓			✓	✓
Boolean Reconstruction	✓			✓	✓	✓	✓
Arbitrary Marginals	✓	✓	✓				✓
Interpretability	✓			✓	✓	✓	

3 Proposed Method

As hinted in the previous section, there are two aspects for a strong interpretability of a boolean matrix decomposition: boolean reconstruction allows clear predictions and explanations of the non-zeros, while the existence non-negative factors establishes the importance of elements and enable the representation of beyond-block structures. In this section, we introduce a new formulation using a step operator that achieves both goals.

3.1 Formal Objective

Let $\mathbf{M}$ be a $n \times m$ boolean matrix. Our goal is to find a $n \times r$ non-negative matrix A and a $m \times r$ non-negative matrix B, so that the product $\mathbf{AB}^T$ is a good approximation of $\mathbf{M}$ after thresholding:

$$\min_{\mathbf{A},\mathbf{B}} ||\mathbf{M} - u_\tau(\mathbf{AB}^T)||_F^2 = \sum_{i,j} \Big(\mathbf{M}_{ij} - u_\tau(\mathbf{AB}^T)_{ij}\Big)^2 \qquad (1)$$

where $|| \cdot ||_F$ is the Frobenius norm and $u_\tau(\mathbf{X})$ simply applies the standard step function to each element X_{ij}:

$$[u_\tau(X)]_{ij} = \begin{cases} 1 & \text{if } X_{ij} \geq \tau \\ 0 & \text{otherwise} \end{cases} \qquad (2)$$

where τ is a threshold parameter. Note that the choice of τ does not affect the decomposition, as matrices $\mathbf{A}$ and $\mathbf{B}$ can always be scaled accordingly.

3.2 Step Matrix Decomposition

The thresholding operator renders the objective function non-differentiable and akin to a binary programming problem. In order to solve it, we will approximate the objective function of equation 1 by a function with similar objective:

$$\min_{\mathbf{A},\mathbf{B}} \sum_{i,j} \log\left(1 + e^{-M_{ij} * \left(\sum_{k=1}^{r} A_{ik}B_{jk} - \tau\right)}\right) \tag{3}$$

where M was transformed so that it has values in $\{-1, 1\}$ by replacing all zeros with -1.

Note that $log(1+e^{-x})$ will tend to zero when x is positive and it will increase when x is negative; the intuition is that this error function will be approximately zero when $M_{i,j}$ and $(\sum_{k=1}^{r} A_{i,k}B_{j,k} - \tau)$ have the same sign and a linear penalty is in place whenever their signs differ.

Given the above formulation, there are several methods for finding **A** and **B** and one possibility is using gradient descent. The gradient is given by

Lemma 1. *Let* $S_{ij} = \sum_{k=1}^{r} A_{ik}B_{jk}$*, then the gradient of the objective function in 3 is given by:*

$$\frac{\partial F}{\partial A_{ik}} = \sum_{j \notin \mathbf{M_i}} \frac{B_{jk}}{1 + e^{\tau - S_{ij}}} - \sum_{j \in \mathbf{M_i}} \frac{B_{jk}}{1 + e^{S_{ij} - \tau}} = \sum_{j=1}^{m} \frac{B_{jk}}{1 + e^{\tau - S_{ij}}} - \sum_{j \in \mathbf{M_i}} B_{jk} \tag{4}$$

Proof. Omitted for brevity.

The update rules for **B** are similar and are also omitted for brevity.

Due to the non-negativity requirement, matrices **A** and **B** are projected after each iteration - this projection is made to a small value ϵ instead of to 0, as $\mathbf{A} = \mathbf{B} = 0$ is a stationary point of the objective function and the algorithm wouldn't improve.

Different gradient descent algorithms and small variations can now be tried. Our experiments indicate that stochastic gradient descent with batches corresponding to factors provides the quickest convergence, as factors quickly converge to different submatrices. Our results also indicate that initializing **A** and **B** to small random numbers provides the best results. Comparing alternative gradient descent methods is out of the scope of this paper.

It should also be noted that τ now impacts the gradient, as the relative error $\left(\frac{\log(1+e^{\tau})}{\log(1+e^{0})}\right)$ of misrepresenting an element increases. However, it is clear that it should be chosen to be the highest possible value in order to improve convergence and to get a sharper decomposition, as long as numerical stability is not compromised. Our implementation uses $\tau = 20$.

Complexity. A straightforward implementation would take $O(TNMR^2)$ time where T is the number of iterations, N and M are the dimensions of the matrix and R is the rank of the decomposition. However, by using additional $O(NM)$ memory, caching and updating $\mathbf{S}$ in each iteration, it can be reduced to $O(TNMR)$.

3.3 FastStep Matrix Decomposition

Unfortunately, the previous algorithm is not adequate for many datasets given its quadratic nature; it grows linearly in $O(NM)$. In many scenarios such as community detection and recommender systems, $\mathbf{M}$ is extremely sparse and algorithms must be linear (or quasilinear) in the number of non-zeros (E). In the following, we describe how to quickly approximate $F(\mathbf{A}, \mathbf{B})$ and the respective gradients of the sparse matrix.

Fast Gradient Calculation. As shown in Equation 4, calculating the gradient exactly requires $O(NM)$ operations per factor because each A_{ik} requires a summation over all elements B_{jk}. Furthermore, there is a $A_{ik}B_{jk}$ term in S_{ij}, which means that this loop cannot be easily unrolled or reused between elements of $\mathbf{A}$. The goal of this subsection is to approximate the gradient of the factor using a number of operations in the order of $O(E)$, the number of non-zeros in the matrix.

Careful analysis of the structure of this summation in the gradient allows us to quickly approximate it. The impact of position (i, j) in factor k is a sigmoid function, scaled by B_{jk} and with parameter S_{ij}. This means that only positions with simultaneously high S_{ij} and B_{jk} have a significant impact on the gradient, which implies that we should first consider pairs (i, j) with high $A_{ik}B_{jk}$, as that correlates well with both metrics.

In other words, Equation 4 can be approximated as

$$\frac{\partial F}{\partial A_{ik}} \simeq \sum_{(i,j)\in P(t)} \frac{B_{jk}}{1 + e^{\tau - S_{ij}}} - \sum_{j\in \mathbf{M_i}} B_{jk} \tag{5}$$

where $P(t)$ is the set of elements of $\mathbf{M}$ that the decomposition "believes" should be reconstructed, i.e. with high $A_{ik}B_{jk}$ for some k. We define r sets of elements $P_k(t)$ that each factor k would like to reconstruct and $P(t) = \bigcup P_k(t)$. The intuition is that, initially, only non-zeros contribute to the gradient so we can quickly calculate it with no error using the second summand of Equation 5. As we iterate, the error will gradually move from the

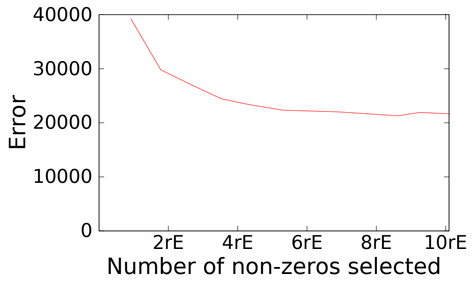


Fig. 3: **A small number of non-zeros approximates the gradient** – quick convergence in the `Airports` dataset.

non-zeros of $\mathbf{M}$ to some of the zeros. However, given $\mathbf{M}$'s sparsity and the symmetry of the error function – the error of misrepresenting a one is the same as misrepresenting a zero – $|P(t)|$ can be kept small and in the order of $O(rE)$; Figure 3 shows the error as the size of $P(t)$ increases.

In order to quickly find the top-t pairs (i, j) with highest $A_{ik}B_{jk}$, let $\mathbf{a_k}$ and $\mathbf{b_k}$ be columns k of matrices $\mathbf{A}$ and $\mathbf{B}$, respectively. After sorting $\mathbf{a_k}$ and $\mathbf{b_k}$, the biggest $A_{ik}B_{jk}$ not currently in P_k can be selected from a very small set of elements along one sort of "diagonal" in the matrix. In particular, it can be shown that element (x, y) should not be added to P_k before both $(x-1, y)$ and $(x, y-1)$ are added, as they would necessarily be at least as big. Therefore, one can keep a priority queue with $O(min(n, m))$ elements and it is possible to select a set of t non-zeros and approximate the gradient of all elements in factor k in $O(t + n \log n + m \log m)$ operations.

Fast Function Evaluation. Given the method currently used to quickly calculate the gradient, one possibility would be to only calculate the error at positions $E + P(t)$. Although fast, some positions of the matrix would never be considered and the algorithm would over-fit, thus it cannot be used to detect convergence.

Therefore, in order to detect convergence and after each iteration of the gradient descent (i.e. after all the batches are completed), we calculate an estimate of the error $\widetilde{F}(\mathbf{A}, \mathbf{B})$ by considering all the non-zeros and a uniform sample of the zeros of the matrix and then scaling the error accordingly. Additionally, in order to decrease the probability of underestimating $F(\mathbf{A}, \mathbf{B})$ and compromising future iterations of the gradient descent, we take the median of 9 simulations.

Complexity. Using the same notation as before, the time complexity is now bounded by the number of non-zeros and $P = |P(t)|$, which as we showed can be $O(rE)$, and the number of samples S to check for convergence. The complexity is now $O(TR(E + P \log(\min(N, M)) + N \log N + M \log M + S))$.

Obtaining clusters from A and B. When a binary answer on whether a given element "belongs" to a factor is desired (e.g. community detection), a clear interpretation exists solely based on the principles of the decomposition:

Definition 1. ***Part of a factor***
A row element i belongs to a factor k if there is non-zero in the reconstructed matrix in row i and if this factor contributed with a weight above $\frac{\tau}{r}$, i.e.:

$$A_{ik} \geq \frac{\tau}{r \max(\mathbf{b_k})} \text{ and } S_{i, \arg\max(\mathbf{b_k})} \geq \tau.$$

We show that this method generates empirically correct clusters in the next section.

Table 2: Datasets used to evaluate FastStep.

Name	Size	Non-zeros	Description
`MovieLens100k`	945×1 684	100 000	User-movie ratings.
`MovieLens10m`	71 568×10 681	10 000 055	User-movie ratings.
`Airports`	7 733×7 733	34 660	Airport to airport flight information.

4 Experimental Evaluation

FastStep was tested on 2 fairly different real-world datasets, see Table 2 for details. `MovieLens100k` and `MovieLens10m` are user-movie ratings datasets made available by `MovieLens` and the `Airports` dataset is a graph made available by `OpenFlights`. Unless otherwise specified, FastStep was run using the default parameters defined in Section 3 and 1 000 000 samples.

We answer the following questions:

Q1. How **scalable** is the fast version of FastStep?

Q2. How does the **reconstruction error** compare to other methods?

Q3. How **effective** and **interpretable** is the FastStep decomposition?

4.1 Scalability

The fast approximation of the gradient has a runtime proportional to the number of non-zeros of the matrix. For the runtime to be reproducible, we took different subsets of the `MovieLens10m` dataset by removing all the ratings of movies produced after a given decade. Please note that the matrix was not resized, resulting in columns (and possibly rows) full of zeros.

Figure 4 shows the execution time of the decomposition for these different matrices. Notice the sub-quadratic running time.

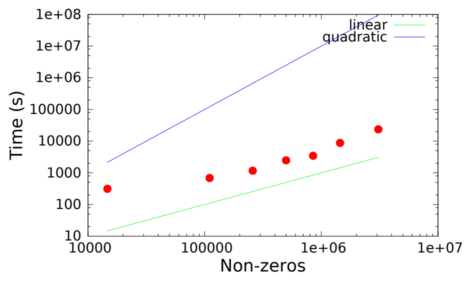


Fig. 4: **Scalability:** the FastStep decomposition has linear running time on the number of non-zeros.

4.2 Low Reconstruction Error

When considering the same number of factors, a lower reconstruction error implies better compression and potentially enables lower-rank representations of the data. Given the boolean nature of **M**, the error function is intuitively easy to represent. Let **M** represent the original dataset and **R** represent the reconstructed matrix, then the error E is given by $E = ||M - R||_F^2$.

Table 3: **The FastStep Decomposition achieves lower squared error** than popular scalable methods.

Dataset	**FastStep**	**SVD**	**NNMF**	**HyCoM**
`Airports`	**21206**	26061	27235	29117
`MovieLens100k`	**68863**	70627	74040	86964

We compared FastStep to other methods that were quasilinear in the number of non-zeros. Table 3 compares the squared error of FastStep, SVD, NNMF and HyCoM in the `MovieLens100k` and `Airports` datasets when using 10 factors. For SVD and NNMF, as arbitrary values such as 0.5 do not guarantee the lowest error, we tried all thresholds and considered the optimal. For FastStep, we selected the lowest error from Figure 3 and its equivalent in the `MovieLens100k` data (which converged after considering only $2rE$ non-zeros). For HyCoM, we considered as error the sum of the edges not represented and the mistakes made inside each community. Among the state of the art methods, we did not compare with non scalable algorithms (L-PCA, ASSO, BMF-Threshold).

However, while comparing the reconstruction error of these methods might be appropriate given the same number of parameters, their expressiveness is not the same given their different characteristics. In this regard, by allowing negative numbers, SVD is at an advantage when compared to the rest of the methods. Please note that common techniques such as the Bayesian Information Criterion (BIC) [13] or the Akaike Information Criterion (AIC) [1] would not provide a fairer comparison because, as the number of parameters is the same, all methods would keep the same relative *rank*. Techniques such as Minimum Description Length (MDL) [8] measure the number of bits required to encode both the error and the model, but it is not clear which method should be used to represent real numbers, especially given that the importance of the bits is not the same - as a result, methods such as HyCoM that uses integer values would greatly benefit.

We can see that FastStep is able to simultaneously achieve a lower reconstruction error while maintaining higher interpretability.

4.3 Discoveries

MovieLens The `MovieLens100k` user-movie dataset was decomposed using a rank-10 decomposition and the factors were clustered as described. Table 4 illustrates the top-5 movies (ranked by score) in three of the factors and shows a grouping by movie theme.

Figure 5 shows 3 clusters and the percentage of movies in each cluster that correspond to a given genre

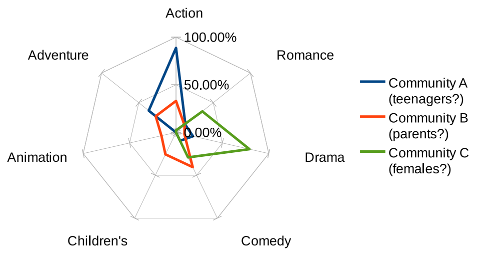


Fig. 5: **MovieLens genre separation**

Table 4: **FastStep is able to automatically group similar movies in the `MovieLens` dataset.** Groups manually labeled according to their highest scoring movies.

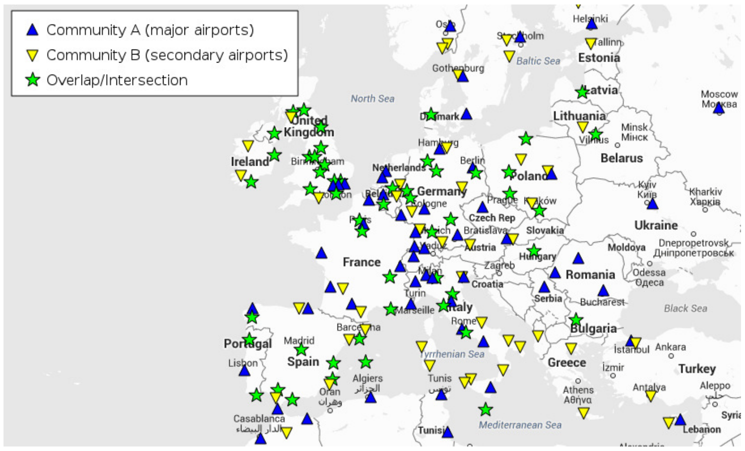
"Action"	**"Romance"**	**"Drama"**
Raiders of the Lost Ark	Picture Perfect	Titanic
The Empire Strikes Back	Addicted to Love	Wag the Dog
Terminator 2: Judgment Day	Bed of Roses	L.A. Confidential
The Terminator	My Best Friend's Wedding	Jackie Brown
Star Trek 3: The Search for Spock	Fly Away Home	Replacement Killers

(movies might have more than one tag, so genres do not sum to 1). We labeled group A as *teenagers* due to the clear prevalence of Action and Adventure movies. In group B, most of the movies rated were in categories of Comedy, Children's, Animation and Adventure; we hypothesize that users rating these movies are parents and labeled the group accordingly. Finally, we labeled group C as *females* due to the Drama, Comedy and Romance movie genres.

Airports The `Airports` dataset is a symmetric matrix representing an undirected unipartite graph, which implies that $\mathbf{B} = \mathbf{A}$ as we are looking for communities. The minimization problem is similar $\left(\min_{\mathbf{A}} ||\mathbf{M} - u(\mathbf{A}\mathbf{A}^T)||_F^2\right)$ and the gradient is omitted for brevity.

Figure 2 shows a geographical plot of the airports in the different communities; some big hubs, such as Frankfurt and Heathrow, appear in multiple communities and were coded with a single color to simplify visualization. Even though no geographical information was used to perform this task, there is a very clear distinction between north American airports, Brazilian airports, European airports, previous French colonies in Africa, Russian airports, Middle-Eastern airports and south-east Asia airports. Additionally, in order to illustrate one of the surprising findings, Figure 6 highlights the two European communities (in blue and yellow) along with the overlapping airports (in green). While it would initially seem that all these airports should be considered the same community, a quick overview makes us realize that they are in fact divided by "major airports" and "secondary airports", usually operated by low-cost companies. The airports with the highest score in the "major airports" community are Barcelona, Munich and Amsterdam, while the airports with the highest score in the "low-cost" group are Girona (85km from Barcelona), Weeze (70km from Dusseldorf) and Frankfurt-Hahn (120km from Frankfurt). We consider these and other surprising findings to be very strong empirical evidence on FastStep's usefulness for these tasks.

Another important improvement of the FastStep decomposition is its ability to reconstruct non-block clusters in the data. Figure 1b shows the adjacency matrix of the American community found in the previous decomposition. As we have non-negative factors, lets explore the additional information available in matrix $\mathbf{A}$. The airports with the highest score correspond to central airports



▲ Community A (major airports)
▽ Community B (secondary airports)
★ Overlap/Intersection

Fig. 6: **Intuitive split of European airports** - FastStep identifies 2 European communities, one with the major international airports (in blue) and the other with secondary airports (in yellow). Overlapping airports appear in green.

in continental United States with hubs from big airlines: Minneapolis, Denver, Chicago, Dallas, Detroit, Houston, etc. Therefore, using this decomposition alone, measures of centrality can be directly obtained.

Finally, the scores of the elements in the communities, when sorted in descending order, closely follow a power-law. This characteristic has been previously observed in ground-truth communities using significantly different ground-truth definitions [2]. Given that no bias was introduced in FastStep, we consider this a strong indicator of its ability to detect realistic structures in graph data.

5 Conclusion

FastStep carefully combines a non-negative decomposition and a boolean reconstruction for the best interpretability of the data. We have shown that it achieves lower reconstruction error than similar methods and have provided strong empirical evidence of its ability to find structural patterns in the data. The main contributions of this work are the following:

1. **New formulation and tractable approximation:** We introduce a novel FastStep Decomposition which exploits thresholding of the reconstructed data in order to minimize the reconstruction error.
2. **Scalable:** A very efficient approximation enables a runtime linear in the number of non-zeros.
3. **Low reconstruction error** when compared to standard methods.
4. **Realistic representation** which relates to nodes in clusters or degree inside communities.
5. **Meaningful and interesting discoveries** in real-world datasets.

Reproducibility Available at `http://cs.cmu.edu/~maraujo/faststep/`.

Acknowledgments Partially funded by the ERDF through the COMPETE 2020 Program and by FCT within project POCI-01-0145-FEDER-006961 and through the CMU|Portugal Program under Grant SFRH/BD/52362/2013. Based upon work supported by the National Science Foundation under Grants No. CNS-1314632 and IIS-1408924, and by a Google Focused Research Award. Any opinions, findings, conclusions or recommendations expressed are those of the authors and do not necessarily reflect the views of the funding parties.

References

1. Akaike, H.: A new look at the statistical model identification. Automatic Control, IEEE Transactions on 19(6), 716–723 (1974)
2. Araujo, M., Günnemann, S., Mateos, G., Faloutsos, C.: Beyond blocks: Hyperbolic community detection. In: ECML PKDD, pp. 50–65 (2014)
3. Bell, R.M., Koren, Y.: Lessons from the netflix prize challenge. ACM SIGKDD Explorations Newsletter 9(2), 75–79 (2007)
4. Dhillon, I.S., Mallela, S., Modha, D.S.: Information-theoretic co-clustering. In: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 89–98. ACM (2003)
5. Eckart, C., Young, G.: The approximation of one matrix by another of lower rank. Psychometrika 1(3), 211–218 (1936), http://dx.doi.org/10.1007/BF02288367
6. Fortunato, S.: Community detection in graphs. Physics Reports 486(3-5), 75–174 (2010)
7. Golub, G., Kahan, W.: Calculating the singular values and pseudo-inverse of a matrix. Journal of the Society for Industrial and Applied Mathematics Series B Numerical Analysis 2(2), 205–224 (1965)
8. Grünwald, P.D.: The minimum description length principle. The MIT Press (2007)
9. Lee, D.D., Seung, H.S.: Learning the parts of objects by non-negative matrix factorization. Nature 401(6755), 788–791 (1999)
10. Leskovec, J., Chakrabarti, D., Kleinberg, J., Faloutsos, C., Ghahramani, Z.: Kronecker graphs: An approach to modeling networks. The Journal of Machine Learning Research 11, 985–1042 (2010)
11. Li, T.: A general model for clustering binary data. In: Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining. pp. 188–197. ACM (2005)
12. Schein, A.I., Saul, L.K., Ungar, L.H.: A generalized linear model for principal component analysis of binary data. In: Proceedings of the 9th international workshop on artificial intelligence and statistics. pp. 14–21 (2003)
13. Schwarz, G., et al.: Estimating the dimension of a model. The annals of statistics 6(2), 461–464 (1978)
14. Sun, J., Xie, Y., Zhang, H., Faloutsos, C.: Less is more: Compact matrix decomposition for large sparse graphs. In: Proceedings of the Seventh SIAM International Conference on Data Mining. vol. 127, p. 366. SIAM (2007)
15. Tanay, A., Sharan, R., Shamir, R.: Biclustering algorithms: A survey. Handbook of computational molecular biology 9(1-20), 122–124 (2005)
16. Vlachos, M., Fusco, F., Mavroforakis, C., Kyrillidis, A., Vassiliadis, V.G.: Improving co-cluster quality with application to product recommendations. In: 23rd ACM Conference on Information and Knowledge Management. pp. 679–688 (2014)
17. Zhang, Z.Y., Li, T., Ding, C., Ren, X.W., Zhang, X.S.: Binary matrix factorization for analyzing gene expression data. Data Mining and Knowledge Discovery 20(1), 28–52 (2010)