Interpretable Robust Recommender Systems with Side Information

Wenyu Chen [1 *] **Zhechao Huang** [1 *] **Jason Cheuk Nam Liang** [1 *] **Zihao Xu** [2 *]

Abstract

In this paper, we propose two methods, namely *Trace-norm regression* (TNR) and *Stable Trace-norm Analysis* (StaTNA), to improve performances of recommender systems with side information. Our trace-norm regression approach extracts low-rank latent factors underlying the side information that drives user preference under different context. Furthermore, our novel recommender framework StaTNA not only captures latent low-rank common drivers for user preferences, but also considers idiosyncratic taste for individual users. We compare performances of TNR and StaTNA on the MovieLens datasets against state-of-the-art models, and demonstrate that StaTNA and TNR in general outperforms these methods.

1. Introduction

The boom of user activity on e-commerce and social networks has continuously fueled the development of recommender systems to most effectively provide suggestions for items that may potentially match user interest. In highly-rated Internet sites such as Amazon.com, YouTube, Netflix, Spotify, LinkedIn, Facebook, Tripadvisor, Last.fm, and IMDb, developing and deploying personalized recommender systems lie at the crux of the services they provide to users and subscribers (Ricci et al., 2015). For example, Youtube, one of the worlds most popular video sites, has deployed a recommender system that updates regularly to deliver personalized sets of videos to users based on their previous or recent activity on site to help users find videos relevant to their interests, potentially keeping users entertained and engaged (Davidson et al., 2010).

Among the vast advancements in deep learning and matrix completion techniques to build recommender systems (Ricci et al., 2015; Singhal et al., 2017), one of the most imperative aspect of research in such area is to identify latent (possibly low-rank) commonalities that drive specific types of user behaviour. For example, (Dong et al., 2017) proposes a deep neural network based matrix factorization approach that uses explicit rating as well as implicit ratings to map user and items into common low-dimensional space. Yet, such variety of low-rank methodologies do not address the impact of idiosyncratic behaviour among buyers, which may potentially skew the overall learned commonalities across user groups.

In this work, we propose two multi-task learning methods to improve performances of recommender systems using contextual side information. We first introduce an approach based on trace-norm regression (TNR) that enables us to extract low-rank latent dimensions underlying the side information that drive user preference according to variations in context, such as item features, user characteristics, time, season, location, etc. This is achieved by introducing a nuclear-norm regularization penalty term in the multi-task regression model, and we highlight that such latent dimensions can be thought of as homogeneous behaviour among particular types of user groups. Furthermore, we propose a novel recommender framework called Stable Trace-norm Analysis (StaTNA) that not only captures latent low-rank common drivers for user preference, but also considers idiosyncratic taste for individual users. This is achieved by, in addition to the low-rank penalty, adding a sparsity regularization term to exploit the sparse nature of heterogeneous behaviour. Finally, we test the performance of StaTNA on the MovieLens datasets against state-of-the-art models, and demonstrate that StaTNA and TNR in general outperforms these methods.

2. Preliminaries and the StaTNA Framework

We first introduce some notation that will be adopted throughout the rest of our work. We let Ω denote the set of all observed entries, and for any matrix $\boldsymbol{Z}$, define $\bar{\boldsymbol{Z}} = \mathcal{P}_\Omega(\boldsymbol{Z})$ as $\bar{\boldsymbol{Z}}_{ij} = \boldsymbol{Z}_{ij}$ if $(i,j) \in \Omega$ and 0 otherwise. We let $\boldsymbol{Y} \in \mathbb{R}^{n\times p}$, be the final output of the recommender system, $\boldsymbol{X} \in \mathbb{R}^{n\times d}$ includes all side information, and $\boldsymbol{L}, \boldsymbol{S} \in \mathbb{R}^{d\times p}$ represent the common and idiosyncratic effects of side information on users. Both $\boldsymbol{L}$ and $\boldsymbol{S}$ can be

[1]Operations Research Center, Massachusetts Institute of Technology [2]Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology. Correspondence to: Wenyu Chen <wenyu@mit.edu>.

Proceedings of the 1^{st} Adaptive & Multitask Learning Workshop, Long Beach, California, 2019. Copyright 2019 by the author(s).

considered as mappings from the side information to the recommender response. To be more specific, we take a movie recommender system for example: n can be the number of movies, p is the number of users, so each entry of $\boldsymbol{Y}$ is the predicted rating of each user for every movie, while $\boldsymbol{X}$ represents the features of each movie where each feature is d dimensional. When a new movie comes in, we apply $\boldsymbol{L}$ or/and $\boldsymbol{S}$ to the movie's feature to predict the rating of each existing user's, and recommend the movie to users with high predicted ratings.

2.1. The Trace-norm Regression (TNR) Approach

Before turning to TNR, we first consider a regularized low-rank solution for large-scale matrix completion problems called Soft-Impute (Mazumder et al., 2010), which sheds light upon the key ideas of trace-norm regression. The Soft-Impute problem is formulated as the following:

$$\min \frac{1}{2}\|\mathcal{P}_\Omega(\boldsymbol{Y}-\boldsymbol{L})\|_F^2 + \lambda_L\|\boldsymbol{L}\|_*,$$

where $\|\cdot\|_F$ denotes the Frobenius norm and $\|\cdot\|_*$ denotes the nuclear norm. In this formulation, we minimize the reconstruction error subject to a bound on the nuclear norm, which serves as a convex relaxation of rank of a matrix and allows us to exploit the low-rank structure of the matrix $\boldsymbol{L}$.

Based on similar ideas, trace-norm regression extends this idea of incorporating regularization on the rank of a matrix in the context of multi-task learning, as it minimizes square loss while penalizing large ranks of the coefficient matrix:

$$\min \frac{1}{2}\|\mathcal{P}_\Omega(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{L})\|_F^2 + \lambda_L\|\boldsymbol{L}\|_*.$$

2.2. Stable Trace-norm Anlaysis (StaTNA)

Similar to our introduction of Soft-Impute and TNR, we first discuss a non contextual model that incorporates both the low-rank matrix $\boldsymbol{L}$ and sparse matrix $\boldsymbol{S}$, namely the stable principal component pursuit (SPCP) (Wright et al., 2013; Zhou et al., 2010; Wright et al., 2009):

$$\min \frac{1}{2}\|\mathcal{P}_\Omega(\boldsymbol{Y}-(\boldsymbol{L}+\boldsymbol{S}))\|_F^2 + \lambda_L\|\boldsymbol{L}\|_* + \lambda_S\|\boldsymbol{S}\|_1,$$

where $\|\cdot\|_1$ denotes the sum of absolute values for all entries of a matrix. $\|\boldsymbol{S}\|_1$ and $\|\boldsymbol{L}\|_*$ models sparsity in $\boldsymbol{S}$ and the low-rank structure in $\boldsymbol{L}$ respectively. To further illustrate some intuition for the choice of such norms, we provide an example in the context of foreground-background separation in video processing. $\boldsymbol{L}$ can be considered as the stationary background, which is low-rank due to the strong correlation between frames; while $\boldsymbol{S}$ can represent foreground objects, which normally occupy only a fraction of the video and hence can be treated as sparse. (Mu et al., 2016) Finally, in light of SPCP, we propose a novel framework called Stable Trace-norm Analysis (StaTNA) by adding contextual side information to consideration:

$$\min \frac{1}{2}\|\mathcal{P}_\Omega(\boldsymbol{Y}-\boldsymbol{X}(\boldsymbol{L}+\boldsymbol{S}))\|_F^2 + \lambda_L\|\boldsymbol{L}\|_* + \lambda_S\|\boldsymbol{S}\|_1.$$

Note that StaTNA can be considered as a combination of trace norm regression and SPCP, and some theoretical aspects are discussed in (Agarwal et al., 2012). The matrix $\boldsymbol{L}$ captures latent homogeneity among preferences of users, such as an Academy Award winning film would be preferable to many users. On the other hand, idiosyncratic tastes of users are embodied in $\boldsymbol{S}$, such as some users particularly like horror movies that involve robots and monsters. Note that $\boldsymbol{S}$ can also be considered as a way to be robust to outliers in user behaviour or preference. In the case where $\boldsymbol{X}$ is the identity matrix, this problem is reduced to SPCP. Also, TNR can be considered as a special case of StaTNA by taking $\lambda_S = \infty$, which explains why StaTNA will in general be more robust compared to TNR.

2.3. Solving TNR and StaTNA with FISTA

In this subsection we will only briefly discuss the methodologies used in this paper to solve TNR and StaTNA, due to space limitations. We first highlight the computational feasibility for both models since methods such as proximal gradient descent (Nesterov, 2013) or (Fast) Iterative Shrinkage-Thresholding Algorithm (FISTA, (Beck & Teboulle, 2009)) can be used to solve these problems with provable (sub)linear convergence rate (Karimi et al., 2016). For StaTNA, we directly apply FISTA to estimate $\boldsymbol{L}$ and $\boldsymbol{S}$, and the procedure is detailed in Algorithm 1 in Appendix A.1. Next, as aforementioned, TNR is a special case for StaTNA, so to solve TNR, we simply take $\lambda_S = \infty$ in Algorithm 1. We also point out that Algorithm 1 may be computationally expensive when the matrix is large. In Appendix A.2 we will propose several modifications to our method for solving TNR and StaTNA which will enable us to improve computational complexity.

3. Experiment

3.1. Dataset

In our work, we consider the MovieLens 100K and MovieLens 1M datasets. The summary for both datasets are shown in Table 1 Note that MovieLens 1M movies do not include all MovieLens 100k movies. In both datasets, each user has rated at least 20 movies, and ratings are whole numbers in the scale of 1-5. In addition, each dataset is associated with a side information matrix whose rows are indexed by movies and includes 1 feature column denoting the movie category (19 categories in total), along with 1128 columns which denote relevant scores of a movie to provided tags.

Dataset	Ratings	Items	Users	Observed
MovieLens 100K	100,000	943	1682	6.30%
MovieLens 1M	1,000,209	6040	3706	4.47%

Table 1. Summary of the MovieLens 100K and MovieLens 1M datasets. **Ratings**: total number of user ratings available; **Items**: total number of movies; **Users**: number of users; **Observed**: percentage of available ratings.

3.2. Experiment Setup

We pre-process side information to obtain two types of side-information matrices. For the first type, we apply one-hot-encoding to the categorical feature and obtain 19 categorical features for each dataset. The final side information matrix is the concatenation of this categorical representation and relevance scores to given tags, which has dimensions $n \times 1147$ where n is the number of movies in each dataset. For the second type, in addition to one-hot-encoding, we apply GloVe word embeddings with $K = 300$ (Pennington et al., 2014) to these categories using an average pooling trick, and result in a 300-dimensional categorical representation vector for each movie. The final side information matrix has dimensions $n \times 1428$. For simplicity, we use the suffix "-G" to denote models trained on the second type of side information processed using GloVe word embedding, while models without this suffix will denote models trained on the first type of side information.

In this work, we perform two experiments: 1. (*Matrix completion with side information*) We fill in missing values in the rating matrix using the pre-processed side information matrix, which is the traditional matrix completion problem. For each dataset, we randomly select 20% of observed values in the rating matrix as the test set, and train TNR & StaTNA, and TNR-G & StaTNA-G on the two types of side-information matrices respectively. We use state-of-the-art models such as SVD, Sparse FC, and GC-MC as our baseline. Here we point out that these baseline models do not incorporate side information, opposed to our TNR and StaTNA models which are trained on side information. Yet, our experimental results will demonstrate that our proposed models, via utilizing side information, will significantly improve performance in this matrix completion task. 2. (*Regression*) We predict the ratings for new movies based on new side information of the movie, which is similar to the traditional regression problem. For each dataset, we train TNR, TNR-G, StaTNA, StaTNA-G on a randomly selected 80% of all movies and apply trained models on the remaining 20%. Furthermore, both experiments involve two hyperparameters, namely λ_L and λ_S, which are tuned using 10-fold cross validation. We use Lasso, Elastic Nets, Multi-task Lasso and Multi-task Elastic Nets (Obozinski et al., 2006) as our baseline models. Note that standard Lasso and standard Elastic Nets are trained for each user independently to predict the user's rating for a given movie. The matrix formulations of these baseline models are shown in Table 6 in Appendix B.2. Finally, we evaluate model performance using mean absolute error (MAE) and root mean square error (RMSE) on the test set.

3.3. Performance Results

	MovieLens 1M		MovieLens 100K	
Method	**RMSE**	**MAE**	**RMSE**	**MAE**
SVD	0.874	0.687	0.934	0.737
SVD++	0.863	0.673	0.921	**0.722**
k-NN	0.922	0.706	0.930	0.733
GC-MC	0.832	-	0.910	-
Sparse FC	0.824	-	**0.890**	-
TNR	0.780	0.600	1.032	0.823
TNR-G	0.778	0.595	1.031	0.825
StaTNA	0.783	0.596	1.034	0.826
StaTNA-G	**0.768**	**0.585**	1.037	0.828

Table 2. MAE and RMSE for test data in Experiment 1 (matrix completion) for baseline models, TNR and StaTNA using MovieLens 100K and MovieLens 100M. Note that TNR-G and StaTNA-G represents TRN and StaTNA trained on side information processed using GloVe word embedding.

Experiment 1: Matrix Completion with Side Information As shown in the out of sample results in Table 2, StaTNa significantly outperforms state-of-the-art models including Sparse FC and GC-MC on MovieLens 1M, resulting in a test set 0.768 RMSE and 0.585 MAE. This illustrates that StaTNA with GloVe embedding applied to side information greatly improves movie rating predictions. Among baseline models, Sparse FC yields the best result for the MovieLens 1M matrix completion problem, but we highlight that Sparse FC relies on complex neural network structure and hence causes many difficulties in model interpretation. Furthermore, for MovieLens 1M, the memory size of weights and multiply accumulate operations (MAC) for Sparse FC are extremely large, amounting to 7.00M and 2.23M. On the other hand, results for StaTNA can be analyzed through investigating the structure in $\boldsymbol{L}$ and $\boldsymbol{S}$, and the memory size of StaTNA weights and MAC are only 3.33M and 0.87M respectively, which imply that StaTNA is more applicable for real-time edge computing.

We notice that both TNR and StaTNA do not perform as well as state-of-the-art models in MovieLens 100K. One explanation is that our models require more training data compared to baseline models to fully capture latent structures of both $\boldsymbol{L}$ and $\boldsymbol{S}$ in order to generalize well on test data. Finally, we also point out that StaTNA converges faster, and in general performs better than TNR, as shown in Figure 1 within Appendix C.1.

Experiment 2: Regression In the second experiment to predict user ratings for movies, as shown in Table 3, for

	MovieLens 1M		MovieLens 100K	
Method	**RMSE**	**MAE**	**RMSE**	**MAE**
Lasso	3.454	3.277	2.851	2.527
Lasso-G	3.512	3.344	2.877	2.558
EN	3.355	3.167	2.863	2.491
EN-G	3.320	3.132	2.887	2.557
MT-Lasso	2.703	2.418	2.859	2.522
MT-Lasso-G	2.681	2.389	2.819	2.519
MT-EN	2.709	2.406	2.886	2.503
MT-EN-G	2.707	2.401	2.921	2.550
TNR	0.782	0.597	1.137	0.911
TNR-G	0.760	0.579	**1.117**	**0.891**
StaTNA	0.752	0.575	1.156	0.918
StaTNA-G	**0.735**	**0.563**	1.141	0.908

Table 3. MAE and RMSE for test data in Experiment 2 (regression) for baseline models (Lasso and Elastic Nets (denoted as EN)), TNR and StaTNA using MovieLens 100K and MovieLens 1M. The prefix "MT" is the abbreviation for "Multi-task".

MovieLens 100K, all StaTNA and TNR models significantly outperform baseline models, while TNR with GloVe embedding yields the best out-of-sample performance. For MovieLens 1M, StaTNA with GloVe embedding results in the best out-of-sample performance compared to all other baseline models including TNR, with and without GloVe embedding. The strong performance of StaTNA and StaTNA-G across both experiments, and especially in MovieLens 1M, indicates that our StaTNA framework provides promising performance guarantees for solving both matrix completion and regression tasks.

3.4. Discussion on Interpretability for StaTNA

As mentioned in earlier sections, we are interested in analyzing particular underlying commonalities in user preferences. We achieve this by investigating the principal components of our estimate of the low-rank matrix $\boldsymbol{L}$, each of which we consider as a common type of user preference. Since our estimated $\boldsymbol{L}$ is of rank 6, we conclude that there are 6 major common types of user preferences, whose component scores (i.e. explained variance percentages) are listed in Table 4, where we observe that the first principal component explains 88.94% of the variability in user ratings. Table 5 shows the top 12 features of highest absolute weights within the first two principal components. The first principal component (PC1) indicates that users' preferences incline towards movies that are more likely to appeal to the general public, for example original movies, movies with great endings, movies that are relevant to cultural issues, etc. The second principle component (PC2) indicates that user preferences are influenced by overall movie quality, such that they do not prefer movies with bad plots (perhaps too predictable) or have bad acting. We highlight that we do not compare values of features across different principal components. For example, in PC1 feature with positive values indicate such features are more preferable, but in PC2 positive values means such features are less preferable, which makes sense since the feature *surprisingly clever* is negative in PC2.

	PC1	PC2	PC3	PC4	PC5	PC6
Var (%)	88.94	5.32	2.16	1.55	1.34	0.69

Table 4. Explained variance percentage for all 6 principal components.

Top	PC1 Features	Weights	PC2 Features	Weights
1	intercept	0.40	predictable	0.23
2	original	0.21	bad plot	0.16
3	catastrophe	0.16	big budget	0.15
4	great	0.15	horrible	0.14
5	surprisingly clever	0.13	plot	0.12
6	mentor	0.12	lame	0.12
7	good	0.12	pg-13	0.12
8	life philosophy	0.12	destiny	0.11
9	dialogue	0.12	bad acting	0.11
10	culture clash	0.11	so bad it's funny	0.11
11	great ending	0.11	surprisingly clever	-0.11
12	runaway	0.11	silly	0.11

Table 5. Top 12 features of highest absolute weights within the first two principal components (PC1 and PC2). Details of other principle components are shown in Table 7 in Appendinx C.2.

4. Future Work and Conclusion

Our methodology to solve TNR and StaTNA (i.e. Algorithm 1 in Appendix A.1) may be computationally expensive when the matrix is large since it requires calling a Singular Value Decomposition (SVD) oracle in each iteration of the algorithm. Hence we propose two alternative methods, a FW-T algorithm and a nonconvex reformulation of the problem, to avoid using an SVD oracle. These are detailed in Appendix A.2. Furthermore, our current studies use side information from only one side, namely movie information. Our StaTNA framework can be extended to incorporate side information for both movies and users:

$$\min \frac{1}{2}\|\mathcal{P}_\Omega(\boldsymbol{Y} - \boldsymbol{X}_M(\boldsymbol{L}_M + \boldsymbol{S}_M) - (\boldsymbol{L}_U + \boldsymbol{S}_U)\boldsymbol{X}_U\|_F^2$$
$$+\lambda_{L_M}\|\boldsymbol{L}_M\|_* + \lambda_{S_M}\|\boldsymbol{S}_M\|_1 + \lambda_{L_U}\|\boldsymbol{L}_U\|_* + \lambda_{S_U}\|\boldsymbol{S}_U\|_1.$$

where U and M denotes users and movies respectively. Moreover, our StaTNA framework is also compatible with neural networks by including nuclear norm and sparse penalties to the objective. We believe that similar formulations will provide us with better performance guarantees, but at the cost of model interpretability.

References

Agarwal, A., Negahban, S., Wainwright, M. J., et al. Noisy matrix decomposition via convex relaxation: Optimal rates in high dimensions. The Annals of Statistics, 40(2): 1171–1197, 2012.

Aravkin, A., Kumar, R., Mansour, H., Recht, B., and Herrmann, F. J. Fast methods for denoising matrix completion formulations, with applications to robust seismic data interpolation. SIAM Journal on Scientific Computing, 36 (5):S237–S266, 2014.

Beck, A. and Teboulle, M. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. SIAM journal on imaging sciences, 2(1):183–202, 2009.

Burer, S. and Monteiro, R. D. Local minima and convergence in low-rank semidefinite programming. Mathematical Programming, 103(3):427–444, 2005.

Cai, J.-F., Candès, E. J., and Shen, Z. A singular value thresholding algorithm for matrix completion. SIAM Journal on Optimization, 20(4):1956–1982, 2010.

Davidson, J., Liebald, B., Liu, J., Nandy, P., Van Vleet, T., Gargi, U., Gupta, S., He, Y., Lambert, M., Livingston, B., et al. The youtube video recommendation system. In Proceedings of the fourth ACM conference on Recommender systems, pp. 293–296. ACM, 2010.

Dong, X., Yu, L., Wu, Z., Sun, Y., Yuan, L., and Zhang, F. A hybrid collaborative filtering model with deep structure for recommender systems. In Thirty-First AAAI Conference on Artificial Intelligence, 2017.

Driggs, D., Becker, S., and Aravkin, A. Adapting regularized low-rank models for parallel architectures. SIAM Journal on Scientific Computing, 41(1):A163–A189, 2019.

Frank, M. and Wolfe, P. An algorithm for quadratic programming. Naval research logistics quarterly, 3(1-2):95–110, 1956.

Gao, W., Goldfarb, D., and Curtis, F. E. ADMM for multiaffine constrained optimization. arXiv preprint arXiv:1802.09592, 2018.

Harchaoui, Z., Juditsky, A., and Nemirovski, A. Conditional gradient algorithms for norm-regularized smooth convex optimization. Mathematical Programming, 152(1-2):75–112, 2015.

Jaggi, M. Revisiting frank-wolfe: Projection-free sparse convex optimization. In ICML (1), pp. 427–435, 2013.

Karimi, H., Nutini, J., and Schmidt, M. Linear convergence of gradient and proximal-gradient methods under the polyak-łojasiewicz condition. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pp. 795–811. Springer, 2016.

Ma, S., Goldfarb, D., and Chen, L. Fixed point and bregman iterative methods for matrix rank minimization. Mathematical Programming, 128(1-2):321–353, 2011.

Mazumder, R., Hastie, T., and Tibshirani, R. Spectral regularization algorithms for learning large incomplete matrices. Journal of machine learning research, 11(Aug): 2287–2322, 2010.

Mu, C., Zhang, Y., Wright, J., and Goldfarb, D. Scalable robust matrix recovery: Frank–wolfe meets proximal methods. SIAM Journal on Scientific Computing, 38(5): A3291–A3317, 2016.

Nesterov, Y. Gradient methods for minimizing composite functions. Mathematical Programming, 140(1):125–161, 2013.

Obozinski, G., Taskar, B., and Jordan, M. Multi-task feature selection. Statistics Department, UC Berkeley, Tech. Rep, 2, 2006.

Pennington, J., Socher, R., and Manning, C. Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), pp. 1532–1543, 2014.

Ricci, F., Rokach, L., and Shapira, B. Recommender Systems: Introduction and Challenges, pp. 1–34. Springer US, Boston, MA, 2015. ISBN 978-1-4899-7637-6. doi: 10.1007/978-1-4899-7637-6_1. URL https://doi.org/10.1007/978-1-4899-7637-6_1.

Singhal, A., Sinha, P., and Pant, R. Use of deep learning in modern recommendation system: A summary of recent works. arXiv preprint arXiv:1712.07525, 2017.

Srebro, N., Rennie, J., and Jaakkola, T. S. Maximum-margin matrix factorization. In Advances in neural information processing systems, pp. 1329–1336, 2005.

Wright, J., Ganesh, A., Rao, S., Peng, Y., and Ma, Y. Robust principal component analysis: Exact recovery of corrupted low-rank matrices via convex optimization. In Advances in neural information processing systems, pp. 2080–2088, 2009.

Wright, J., Ganesh, A., Min, K., and Ma, Y. Compressive principal component pursuit. Information and Inference: A Journal of the IMA, 2(1):32–68, 2013.

Zhou, Z., Li, X., Wright, J., Candes, E., and Ma, Y. Stable principal component pursuit. In 2010 IEEE international symposium on information theory, pp. 1518–1522. IEEE, 2010.

Supplementary Material

A. Algorithms for Solving TNR and StaTNA

A.1. FISTA algorithm

In this section, we discuss the methodologies we use to solve TNR and StaTNA. As mentioned earlier, we use (Fast) Iterative Shrinkage-Thresholding Algorithm (FISTA, (Beck & Teboulle, 2009)) to solve these problems. Before we address the detailed applications of these algorithms in our context to solve TNR and StaTNA, we introduce the following optimization oracles. We define the proximal mapping of the ℓ_1 norm as $\mathcal{T}_\lambda[\boldsymbol{Y}] = \arg\min \frac{1}{2}\|\boldsymbol{X}-\boldsymbol{Y}\|_F^2 + \lambda\|\boldsymbol{X}\|_1$, where $\mathcal{T}_\lambda : \mathbb{R} \to \mathbb{R}$ denotes the soft-thresholding operator $\mathcal{T}_\lambda(x) = x\left(1 - \frac{\lambda}{|x|}\right)^+$, whose extension to matrices is obtained by applying the scalar operator to each element. Moreover, we define the proximal mapping of the nuclear norm (Cai et al., 2010; Ma et al., 2011) as $\mathcal{S}_\lambda[\boldsymbol{Y}] = \arg\min \frac{1}{2}\|\boldsymbol{X}-\boldsymbol{Y}\|_F^2 + \lambda\|\boldsymbol{X}\|_*$, where $\mathcal{S}_\lambda[\boldsymbol{Y}] = \boldsymbol{U}\mathcal{T}_\lambda[\boldsymbol{D}]\boldsymbol{V}^\top$, and $\boldsymbol{Y} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^\top$ is the SVD of matrix $\boldsymbol{Y}$. Now, using these definitions, we detail the algorithm to solve StaTNA in Algorithm 1. Note that one can also initialize $\boldsymbol{L}^0$ in both Algorithm 1 as $\boldsymbol{L}^0 = (\boldsymbol{X}^\top\boldsymbol{X})^\dagger\boldsymbol{X}^\top\mathcal{P}_\Omega(\boldsymbol{Y})$, where † denotes the pseudo-inverse of a matrix.

For StaTNA, we directly apply FISTA to estimate $\boldsymbol{L}$ and $\boldsymbol{S}$, and the procedures are detailed in Algorithm 1. As aforementioned, TNR is a special case for StaTNA, so to solve TNR, we simply take $\lambda_S = \infty$ in Algorithm 1, which forces all $\boldsymbol{S}^k$ and $\hat{\boldsymbol{S}}^k$ to $\boldsymbol{0}$.

Algorithm 1 FISTA for StatNA

```
1: Given $\boldsymbol{Y}, \boldsymbol{X}$ set $\hat{\boldsymbol{L}}^0 = \boldsymbol{L}^0 = \boldsymbol{0}$, $\hat{\boldsymbol{S}}^0 = \boldsymbol{S}^0 = \boldsymbol{0}$, $t^0 = 1$
2: Let $L = \lambda_{\max}(\boldsymbol{X}^\top\boldsymbol{X})$
3: for $k = 0, 1, \ldots$ do
4:     $\boldsymbol{G}^k = -\boldsymbol{X}^\top\mathcal{P}_\Omega(\boldsymbol{Y} - \boldsymbol{X}\hat{\boldsymbol{L}}^k - \boldsymbol{X}\hat{\boldsymbol{S}}^k)$
5:     $\boldsymbol{L}^{k+1} = \mathcal{S}_{\lambda_L/L}[\hat{\boldsymbol{L}}^k - \boldsymbol{G}^k/L]$
6:     $\boldsymbol{S}^{k+1} = \mathcal{T}_{\lambda_S/L}[\hat{\boldsymbol{S}}^k - \boldsymbol{G}^k/L]$
7:     $t^{k+1} = \frac{1+\sqrt{1+4(t^k)^2}}{2}$
8:     $\hat{\boldsymbol{L}}^{k+1} = \boldsymbol{L}^{k+1} + \frac{t^k-1}{t^{k+1}}(\boldsymbol{L}^{k+1} - \boldsymbol{L}^k)$
9:     $\hat{\boldsymbol{S}}^{k+1} = \boldsymbol{S}^{k+1} + \frac{t^k-1}{t^{k+1}}(\boldsymbol{S}^{k+1} - \boldsymbol{S}^k)$
10: end for
11: return $\boldsymbol{L}^{k+1}, \boldsymbol{S}^{k+1}$
```

A.2. Extension on Algorithm Scalability

In Algorithm 1, we call an SVD oracle in each iteration in FISTA to find the proximal mapping of the nuclear norm, which is computationally expensive when the matrix is large, i.e. the number of movie features and users are large. Here we propose two methods to avoid using an SVD oracle.

First, inspired by a scalable algorithm FW-T on SPCP (Harchaoui et al., 2015; Mu et al., 2016), we propose a similar FW-T algorithm to solve StaTNA by replacing the proximal mapping of the nuclear norm with a Frank-Wolfe update in each iteration. To be more specific, we consider the following reformulation of StaTNA:

$$\begin{aligned} \min \quad & \frac{1}{2}\|\mathcal{P}_\Omega(\boldsymbol{Y} - \boldsymbol{X}(\boldsymbol{L}+\boldsymbol{S}))\|_F^2 + \lambda_L t_L + \lambda_S t_S \\ \text{s.t.} \quad & \|\boldsymbol{L}\|_* \le t_L \le U_L \\ & \|\boldsymbol{S}\|_* \le t_S \le U_S, \end{aligned}$$

for some U_L and U_S such that the optimal solution $(\boldsymbol{L}^\star, \boldsymbol{S}^\star)$ is still feasible to the above problem, i.e. $\|\boldsymbol{L}^\star\|_* \le U_L$ and $\|\boldsymbol{S}^\star\|_* \le U_S$. For simplicity, we write the above objective function as $g(\boldsymbol{L}, \boldsymbol{S}, t_L, t_S)$. In each iteration of the FW-T algorithm, we call the following Frank-Wolfe oracle (Algorithm 2) for $\boldsymbol{L}$ and $\boldsymbol{S}$ respectively. We can see that for the Frank-Wolfe update for matrix $\boldsymbol{L}$ only requires to compute the leading singular pairs for a matrix, which can be achieved by computationally cheap power iteration (Jaggi, 2013). In addition, we perform an exact line-search by easily solving a

Algorithm 2 Frank-Wolfe Oracle $\mathcal{F}_\lambda(\boldsymbol{G}, U, \|\cdot\|)$

```
1: $\boldsymbol{D}^\star \in \arg\min_{\|\boldsymbol{D}\|\leq 1}\langle \boldsymbol{D}, \boldsymbol{G}\rangle$
2: if $\lambda \geq -\langle \boldsymbol{D}^\star, \boldsymbol{G}\rangle$ then
3:     return $(\boldsymbol{V}^\star, t^\star) = (\boldsymbol{0}, 0)$
4: else
5:     return $(\boldsymbol{V}^\star, t^\star) = (U\boldsymbol{D}^\star, U)$
6: end if
```

quadratic problem P_k.

$$\begin{aligned} \min \quad & g(\boldsymbol{L}, \boldsymbol{S}, t_L, t_S) \\ \text{s.t.} \quad & \begin{pmatrix} \boldsymbol{L} \\ t_L \end{pmatrix} \in \text{conv}\left\{ \begin{pmatrix} \boldsymbol{0} \\ 0 \end{pmatrix}, \begin{pmatrix} \boldsymbol{L}^k \\ t_L^k \end{pmatrix}, \begin{pmatrix} \boldsymbol{V}_L^k \\ V_{t_L}^k \end{pmatrix} \right\} \\ & \begin{pmatrix} \boldsymbol{S} \\ t_S \end{pmatrix} \in \text{conv}\left\{ \begin{pmatrix} \boldsymbol{0} \\ 0 \end{pmatrix}, \begin{pmatrix} \boldsymbol{S}^k \\ t_S^k \end{pmatrix}, \begin{pmatrix} \boldsymbol{V}_S^k \\ V_{t_S}^k \end{pmatrix} \right\}. \end{aligned} \tag{P_k}$$

The full algorithm for the FW-T is detailed in Algorithm 3.

Algorithm 3 FW-T for StatNA

```
1: Given $\boldsymbol{Y}, \boldsymbol{X}$ set $\boldsymbol{L}^0 = \boldsymbol{S}^0 = \boldsymbol{0}$, $t_L^0 = t_S^0 = 0$; $U_L^0 = g(\boldsymbol{L}^0, \boldsymbol{S}^0, t_L^0, t_S^0)/\lambda_L$, $U_S^0 = g(\boldsymbol{L}^0, \boldsymbol{S}^0, t_L^0, t_S^0)/\lambda_S$
2: Let $L = \lambda_{\max}(\boldsymbol{X}^\top \boldsymbol{X})$
3: for $k = 0, 1, \ldots$ do
4:     $\boldsymbol{G}^k = -\boldsymbol{X}^\top \mathcal{P}_\Omega(\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{L}^k - \boldsymbol{X}\boldsymbol{S}^k)$
5:     $(\boldsymbol{V}_L^k, V_{t_L}^k) = \mathcal{F}_{\lambda_L}(\boldsymbol{G}^k, U_L^k, \|\cdot\|_*)$
6:     $(\boldsymbol{V}_S^k, V_{t_S}^k) = \mathcal{F}_{\lambda_S}(\boldsymbol{G}^k, U_S^k, \|\cdot\|_1)$
7:     $\left(\hat{\boldsymbol{L}}^k, \hat{\boldsymbol{S}}^k, \hat{t}_L^k, \hat{t}_S^k\right) \in \arg\min(P_k)$
8:     $\hat{\boldsymbol{G}}^k = -\boldsymbol{X}^\top \mathcal{P}_\Omega(\boldsymbol{Y} - \boldsymbol{X}\hat{\boldsymbol{L}}^k - \boldsymbol{X}\hat{\boldsymbol{S}}^k)$
9:     $\boldsymbol{S}^{k+1} = \mathcal{T}_{\lambda_2/L}[\hat{\boldsymbol{S}}^k - \hat{\boldsymbol{G}}^k/L]$
10:    $\boldsymbol{L}^{k+1} = \hat{\boldsymbol{L}}^k, t_L^{k+1} = \hat{t}_L^k, t_S^{k+1} = \|\boldsymbol{S}^{k+1}\|_1$
11:    $U_L^{k+1} = g(\boldsymbol{L}^{k+1}, \boldsymbol{S}^{k+1}, t_L^{k+1}, t_S^{k+1})/\lambda_L$
12:    $U_S^{k+1} = g(\boldsymbol{L}^{k+1}, \boldsymbol{S}^{k+1}, t_L^{k+1}, t_S^{k+1})/\lambda_S$
13: end for
14: return $\boldsymbol{L}^{k+1}, \boldsymbol{S}^{k+1}$
```

Second, we propose a nonconvex formulation of the problem suggested by (Aravkin et al., 2014; Srebro et al., 2005):

$$\min \quad \frac{1}{2}\|\mathcal{P}_\Omega(\boldsymbol{Y} - \boldsymbol{X}(\boldsymbol{U}\boldsymbol{V}^\top + \boldsymbol{S}))\|_F^2 + \frac{\lambda_L}{2}(\|\boldsymbol{U}\|_F^2 + \|\boldsymbol{V}\|_F^2) + \lambda_S\|\boldsymbol{S}\|_1,$$

where $\boldsymbol{U} \in \mathbb{R}^{d\times r}$, $\boldsymbol{V} \in \mathbb{R}^{p\times r}$. This problem is nonconvex but smooth, and these two problems are equivalent in the sense that there is a ono-on-one correspondence between the global minima of these two problems (Burer & Monteiro, 2005; Driggs et al., 2019). Since this new formulation has multi-affine structure, according to new results given by (Gao et al., 2018), we can use ADMM to solve this, which is detailed in Algorithm 4.

To apply ADMM on this nonconvex reformulation, we further reformulate the problem as the following

$$\begin{aligned} \min \quad & \frac{1}{2}\|\mathcal{P}_\Omega(\boldsymbol{N})\|_F^2 + \frac{\lambda_L}{2}(\|\boldsymbol{U}\|_F^2 + \|\boldsymbol{V}\|_F^2) + \lambda_S\|\boldsymbol{S}\|_1 \\ \text{s.t.} \quad & \boldsymbol{Y} - \boldsymbol{X}(\boldsymbol{L} + \boldsymbol{R}) - \boldsymbol{N} = \boldsymbol{0}, \\ & \boldsymbol{U}\boldsymbol{V}^\top - \boldsymbol{L} = \boldsymbol{0}, \\ & \boldsymbol{S} - \boldsymbol{R} = \boldsymbol{0} \end{aligned} \tag{1}$$

Algorithm 4 ADMM for nonconvex StaTNA

```
1: Given $\boldsymbol{Y}, \boldsymbol{X}$, set $\boldsymbol{L}^0 = \boldsymbol{S}^0 = \boldsymbol{U}^0 = \boldsymbol{V}^0 = \boldsymbol{N}^0 = \boldsymbol{R}^0 = \boldsymbol{\Lambda}_Y^0 = \boldsymbol{\Lambda}_L^0 = \boldsymbol{\Lambda}_S^0 = \boldsymbol{0}$, set $\rho_Y, \rho_L, \rho_S$
2: for $k = 0, 1, \dots$ do
3:     $\boldsymbol{U}^{k+1} = (\rho_L \boldsymbol{L}^k - \boldsymbol{\Lambda}_L^k)\boldsymbol{V}^k[\boldsymbol{\lambda}_L \boldsymbol{I} + \rho_L(\boldsymbol{V}^k)^\top \boldsymbol{V}^k]^{-1}$
4:     $\boldsymbol{V}^{k+1} = (\rho_L \boldsymbol{L}^k - \boldsymbol{\Lambda}_L^k)^\top \boldsymbol{U}^{k+1}[\boldsymbol{\lambda}_L \boldsymbol{I} + \rho_L(\boldsymbol{U}^{k+1})^\top \boldsymbol{U}^{k+1}]^{-1}$
5:     $\boldsymbol{S}^{k+1} = \mathcal{T}_{\lambda_S/\rho_S}[\boldsymbol{R}^k - \boldsymbol{\Lambda}_S^k/\rho_S]$
6:     $\boldsymbol{L}^{k+1} = [\rho_L \boldsymbol{I} + \rho_Y \boldsymbol{X}^\top \boldsymbol{X}]^{-1}[\rho_L \boldsymbol{U}^{k+1}(\boldsymbol{V}^{k+1})^\top + \boldsymbol{\Lambda}_L^k + \boldsymbol{X}^\top \boldsymbol{\Lambda}_Y^k + \rho_Y \boldsymbol{X}^\top(\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{R}^k - \boldsymbol{N}^k)]$
7:     $\boldsymbol{R}^{k+1} = [\rho_S \boldsymbol{I} + \rho_Y \boldsymbol{X}^\top \boldsymbol{X}]^{-1}[\rho_S \boldsymbol{S}^{k+1} + \boldsymbol{\Lambda}_S^k + \boldsymbol{X}^\top \boldsymbol{\Lambda}_Y^k + \rho_Y \boldsymbol{X}^\top(\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{L}^{k+1} - \boldsymbol{N}^k)]$
8:     $\boldsymbol{N}^{k+1} = \frac{1}{\rho_Y+1}\mathcal{P}_\Omega[\boldsymbol{\Lambda}_Y^k + \rho_Y \boldsymbol{Y} - \rho_Y \boldsymbol{X}(\boldsymbol{L}^{k+1} + \boldsymbol{R}^{k+1})] + \frac{1}{\rho_Y}\mathcal{P}_\Omega^\perp[\boldsymbol{\Lambda}_Y^k + \rho_Y \boldsymbol{Y} - \rho_Y \boldsymbol{X}(\boldsymbol{L}^{k+1} + \boldsymbol{R}^{k+1})]$
9:     $\boldsymbol{\Lambda}_Y^{k+1} = \boldsymbol{\Lambda}_Y^k + \rho_Y[\boldsymbol{Y} - \boldsymbol{X}(\boldsymbol{L}^{k+1} + \boldsymbol{R}^{k+1}) - \boldsymbol{N}^{k+1}]$
10:    $\boldsymbol{\Lambda}_L^{k+1} = \boldsymbol{\Lambda}_L^k + \rho_L[\boldsymbol{U}^{k+1}(\boldsymbol{V}^{k+1})^\top - \boldsymbol{L}^{k+1}]$
11:    $\boldsymbol{\Lambda}_S^{k+1} = \boldsymbol{\Lambda}_S^k + \rho_S(\boldsymbol{S}^{k+1} - \boldsymbol{R}^{k+1})$
12: end for
13: return $\boldsymbol{L}^{k+1}, \boldsymbol{S}^{k+1}$
```

B. Summary for Formulations

B.1. Summary for norms

Norms	Formula	Comments
$\\|\boldsymbol{X}\\|_F$	$\left(\sum_{i,j} x_{ij}^2\right)^{1/2}$	Frobineous norm ℓ_2-norm of vectorized $\boldsymbol{X}$
$\\|\boldsymbol{X}\\|_*$	$\sum_k \sigma_k(\boldsymbol{X})$	Nuclear norm ℓ_1-norm of singular values of $\boldsymbol{X}$
$\\|\boldsymbol{X}\\|_1$	$\sum_{i,j} \|x_{ij}\|$	ℓ_1-norm of vectorized $\boldsymbol{X}$ ℓ_1-norm of ℓ_1-norm of each row of $\boldsymbol{X}$
$\\|\boldsymbol{X}\\|_{21}$	$\sum_i \left(\sum_j x_{ij}^2\right)^{1/2}$	ℓ_1-norm of ℓ_2-norm of each row of $\boldsymbol{X}$

B.2. Summary for models

Framework	Model	Formulation	Measurement	Weight	Regularizer
Regression	Lasso	$\min \frac{1}{2}\\|\mathcal{P}_\Omega(\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{W})\\|_F^2 + \lambda\\|\boldsymbol{W}\\|_1$	$\boldsymbol{X}$	$\boldsymbol{W}$	$\lambda\\|\boldsymbol{W}\\|_1$
	ElasticNet	$\min \frac{1}{2}\\|\mathcal{P}_\Omega(\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{W})\\|_F^2 + \lambda\rho\\|\boldsymbol{W}\\|_1 + \frac{\lambda(1-\rho)}{2}\\|\boldsymbol{W}\\|_F^2$	$\boldsymbol{X}$	$\boldsymbol{W}$	$\lambda\rho\\|\boldsymbol{W}\\|_1 + \frac{\lambda(1-\rho)}{2}\\|\boldsymbol{W}\\|_F^2$
MTR	MT-Lasso	$\min \frac{1}{2}\\|\mathcal{P}_\Omega(\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{W})\\|_F^2 + \lambda\\|\boldsymbol{W}\\|_{21}$	$\boldsymbol{X}$	$\boldsymbol{W}$	$\lambda\\|\boldsymbol{W}\\|_{21}$
	MT-EN	$\min \frac{1}{2}\\|\mathcal{P}_\Omega(\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{W})\\|_F^2 + \lambda\rho\\|\boldsymbol{W}\\|_{21} + \frac{\lambda(1-\rho)}{2}\\|\boldsymbol{W}\\|_F^2$	$\boldsymbol{X}$	$\boldsymbol{W}$	$\lambda\rho\\|\boldsymbol{W}\\|_{21} + \frac{\lambda(1-\rho)}{2}\\|\boldsymbol{W}\\|_F^2$
MC	Soft-Impute	$\min \frac{1}{2}\\|\mathcal{P}_\Omega(\boldsymbol{Y} - \boldsymbol{L})\\|_F^2 + \lambda_L\\|\boldsymbol{L}\\|_*$	$\boldsymbol{X} = \boldsymbol{I}$	$\boldsymbol{W} = \boldsymbol{L}$	$\lambda_L\\|\boldsymbol{L}\\|_*$
	SPCP	$\min \frac{1}{2}\\|\mathcal{P}_\Omega(\boldsymbol{Y} - (\boldsymbol{L} + \boldsymbol{S}))\\|_F^2 + \lambda_L\\|\boldsymbol{L}\\|_* + \lambda_S\\|\boldsymbol{S}\\|_1$	$\boldsymbol{X} = \boldsymbol{I}$	$\boldsymbol{W} = \boldsymbol{L} + \boldsymbol{S}$	$\lambda_L\\|\boldsymbol{L}\\|_* + \lambda_S\\|\boldsymbol{S}\\|_1$
Ours	TNR	$\min \frac{1}{2}\\|\mathcal{P}_\Omega(\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{L})\\|_F^2 + \lambda_L\\|\boldsymbol{L}\\|_*$	$\boldsymbol{X}$	$\boldsymbol{W} = \boldsymbol{L}$	$\lambda_L\\|\boldsymbol{L}\\|_*$
	StatNA	$\min \frac{1}{2}\\|\mathcal{P}_\Omega(\boldsymbol{Y} - X(\boldsymbol{L} + \boldsymbol{S}))\\|_F^2 + \lambda_L\\|\boldsymbol{L}\\|_* + \lambda_S\\|\boldsymbol{S}\\|_1$	$\boldsymbol{X}$	$\boldsymbol{W} = \boldsymbol{L} + \boldsymbol{S}$	$\lambda_L\\|\boldsymbol{L}\\|_* + \lambda_S\\|\boldsymbol{S}\\|_1$

Table 6. Summary for formulations of models. MTR is the abbreviation for multitask regression, while MC is the abbreviations for matrix completion.

C. Additional Stuff

C.1. Experiment Figures (Figures 1 and 2)

C.2. Other Principal Components (Table 7)

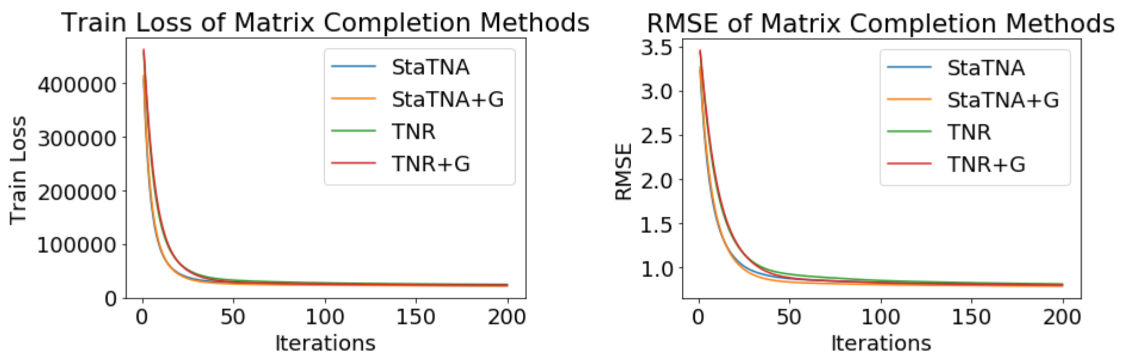

Figure 1. Experiment 1 (matrix completion) for MovieLens 1M. Left: Traning loss vs. number of iterations; Right: Test RMSE vs. number of iterations.

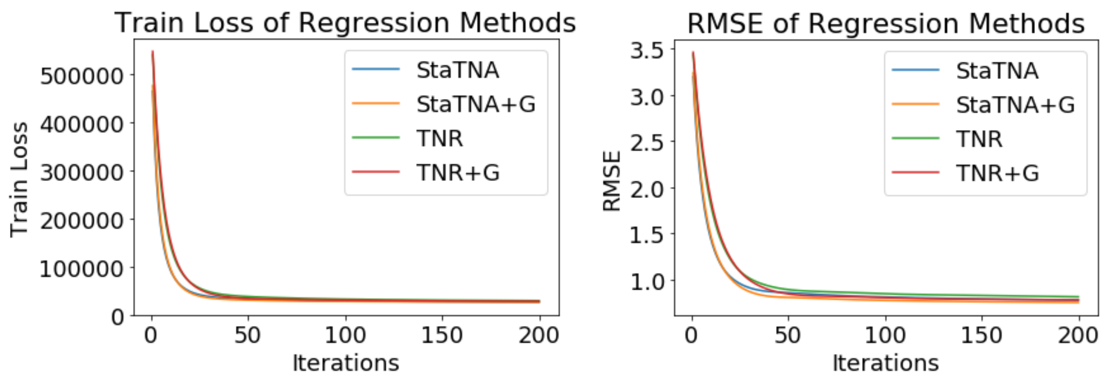

Figure 2. Experiment 2 (regression) for MovieLens 1M. Left: Traning loss vs. number of iterations; Right: Test RMSE vs. number of iterations.

Top	PC3 Features	Weights	PC4 Features	Weights	PC5 Features	Weights	PC6 Features	Weights
1	goofy	-0.14	intercept	-0.16	Drama	-0.15	Adventure	0.16
2	girlie movie	0.14	interesting	0.13	good action	0.11	tense	-0.12
3	brutality	-0.11	great movie	0.13	chase	0.11	Drama	-0.11
4	stupidity	-0.11	intense	0.12	action	0.11	weird	0.11
5	romantic	0.10	narrated	0.11	great movie	0.11	fantasy	0.11
6	sweet	0.10	great acting	0.11	exciting	0.11	fantasy world	0.10
7	family	0.10	imdb top 250	0.11	franchise	0.11	Sci-Fi	0.10
8	violence	-0.10	dark	0.10	series	0.10	mythology	0.10
9	Horror	-0.10	excellent script	0.10	quotable	0.10	imagination	0.10
10	love story	0.10	dark humor	0.10	fighting	0.10	visually stunning	0.09
11	cult classic	-0.10	very good	0.10	hilarious	0.10	suspense	-0.09
12	silly fun	-0.09	drama	0.10	humor	0.10	superheroes	0.09

Table 7. Top 12 features of highest absolute weights within the third to sixth principal components (PC3, PC4, PC5 and PC6)