Low-rank approximations for predicting voting behaviour *

Aldo Porco
Universitat Pompeu Fabra
Barcelona, Spain
aldo.porco@upf.edu

Andreas Kaltenbrunner
Eurecat
Barcelona, Spain
kaltenbrunner@gmail.com

Vicenç Gómez
Universitat Pompeu Fabra
Barcelona, Spain
vicen.gomez@upf.edu

Abstract

We consider the task of predicting behaviour in voting networks, where nodes correspond to candidates/voters and edge signs correspond to positive/negative votes. Our approach is based on recommender systems and learns latent features from the networked data that are used subsequently to train a classifier. The latent features are learnt by solving a nuclear norm regularized least-squares problem. We evaluate our method and show better classification performance than previous approaches that used ad-hoc features in several real-world datasets. We discuss the applicability of the method to more general signed networks and possible extensions.

Keywords: *Social Networks, Recommender Systems, Link Prediction, Edge Polarity Prediction, Latent Features, Nuclear Norm.*

1 Introduction

The vast increase of available digital information comes with many challenges for their analysis. Very often, this information can be modelled as a network, such as the friendship network in Facebook or the follower network in Twitter. Network analysis tools can thus provide a methodology to unravel properties and patterns of behaviour hidden in the data [15].

Our interest in this paper lies in modelling and predicting social behaviour in voting systems. We consider an election as a network in which nodes denote individuals (who can be both voters or candidates) and signed edges represent assessments for one another. We use the underlying structure of the network to predict voting activity. A very simple example of a voting network is shown in Figure 1. In this toy example, node j has incoming negative votes from h and i, and an outgoing negative vote for k. Node k has another vote, this one positive, from node i and has not participated in any election as a voter.

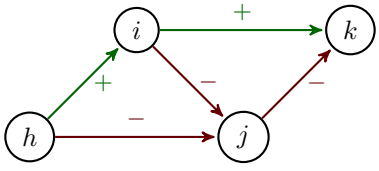

Figure 1: A small voting network example. Nodes indicate both candidates and voters when incoming and outgoing links, repectively, are considered.

*NIPS 2015 Workshop: Networks in the Social and Information Sciences.

Concepts from social status theory have been used to predict the polarity (or sign) of a vote [9, 18]. According to this theory, a positive edge between two nodes, x and y, suggests that x has higher status than y. Conversely, if the edge is negative, then y has a lower status than x. Building on this work, Leskovec et al. [17] developed a method for predicting the sign of an edge using two types of *ad-hoc* features computed from structural properties of the network. This approach results in very good predicting performance and allows to relate the learned models with social theories of balance and status [4].

Other approaches combine additional information besides the network structure to address the task of predicting the existence or sign of an edge. For example, Lee et al. [16] uses information about the communication graph, the one formed by the interchange of messages between the users (voters). They found interesting insights such as users tend to evaluate candidates that they previously had contact with. Furthermore, they found evidence indicating that a node's choices are influenced by their closest contacts. More recently, West et al. [24] analysed two social networks in which votes are accompanied with a brief text that complements the decision. They combined the network structure with sentiment analysis (an estimate of the attitude or polarity of the writer expressed in the text) and they were able to show that the mixed method outperforms the pure ones (only network oriented, or only sentiment analysis oriented).

Contrary to the aforementioned works that are based on *ad-hoc* features, we are interested in *learning* automatically the features from the data and use them later to make predictions. Our methodology is mainly inspired by Recommender Systems (RS) [21]. In RS, the task is to predict the rating or preference that a user would give to a product. In our case, our goal is to predict an evaluation (preference) of a voter (the user) for a particular candidate (the product). The advantage of using this approach is that typical issues and methodology that appear naturally in the task of predicting voting behaviour, have been traditionally addressed in the literature of recommender systems. For example, the cold-start problem (how to predict the rating of a new product) or the use of sparsity as prior knowledge.

In the next section, we introduce the mathematical framework followed by the experimental results. Finally, we conclude and discuss future work.

2 Problem Definition

We will focus on two main tasks: attendance prediction and polarity prediction, giving more attention to the latter one. Let $G = (V, E)$ be a signed directed graph, where V are the nodes (representing users) and E are the edges (representing votes). We define the adjacency matrix A_{ij} such as $A_{ij} = 1$ if $(i, j) \in E$ and zero otherwise and denote the sign of an edge as $s(x, y) \in \{-1, 1\}$, for $(x, y) \in E$. The two tasks considered in this work are:

Attendance prediction : to predict whether a voter x will participate in the election of candidate y.

Polarity prediction : to determine the polarity of the vote, given that x attends to y election.

Note that both tasks are challenging when little information is available. In this situation, one has to make model assumptions on the underlying graph that are key in order to have a well-defined and tractable problem at the same time.

3 General approach

To solve the tasks described in the previous section, we take a *Recommender Systems* RS approach. The goal of RS is to predict a user's preference over products given a previous record of ratings. This type of information can be viewed as a signed network in which nodes can be users or products and edges are evaluations from users about products. RS typically use a matrix M of size $n \times p$ where the n rows represent the users, the p columns represent the items and the values M_{ij} represent the score of user i to product j. Such structure is called utility matrix [21].

Many RS rely on computing some measure of users similarity, usually based on ratings, to recommend products to users [21]. The result of a query for user's i preference over a product j is the weighted sum of i's nearest neighbours' ratings for j. This strategy is called *heuristic-based* or

memory-based. In this work, we take a different, model-based approach, in which the model (in our case a feature model and a classifier) is built from the dataset of user ratings. The features of the classifier are learned by solving a low-rank matrix approximation problem. These features are subsequently used in a supervised learning setting to learn the parameters of a classifier.

More precisely, our method consists of the following steps:

1. Construct a matrix M of voters and candidates representing the voting network
2. Find a low-dimensional space that summarizes the network by a proper factorization of M
3. Train a classifier using the features previously computed

In most cases, the number of products p is orders of magnitude larger than the average number of ratings per user. As a consequence, the utility matrix has a lot of missing values, which makes the task of finding similar voting patterns between users difficult. This condition suffered by the utility matrix, namely sparsity, is one of the main problems faced by RS. The most common way to address it is by making a model assumption that the underlying true matrix (which is unknown and we try to estimate) is indeed sparse and has many missing entries. As we will see, this has proven to be very useful in practice.

A special trait of voting networks is that nodes can have two types of roles: they can appear as voters and candidates at the same time. For instance, the node j in Figure 1. This difference can be used to address the cold-start problem.

3.1 Feature Selection Step: Low-Rank Approximation of the Utility Matrix

Our approach first learns features from the data. This step can be stated as learning an unknown parameter, namely a high dimensional matrix $\tilde{M}_{n\times p}$, for which we are only given a very few observations. This problem is ill-posed and, to make it well defined, we assume that $\tilde{M}$ lies in a much lower dimensional manifold, meaning that it can be well represented by a low-rank matrix, i.e. $\tilde{M} \sim U_{n\times k}V_{k\times p}$ where $k \ll \min(n, p)$.

The problem of finding an approximation that minimizes the difference between the original matrix and the rank optimized one is called *low-rank approximation*. For this problem, the truncated Singular Value Decomposition (SVD) provides a solution. SVD factorizes a matrix M into three matrices $M = U\Sigma V^T$ such that Σ is a diagonal matrix whose non-zero cells are called singular values. The greater the singular value is, the more predictive power the dimension (rows in U and V) has. The rank of the matrix M is the number of nonzero singular values.

To obtain the aforementioned approximation $\tilde{M}$, one can minimize the sum of squares error with the given data using a regularizing term that penalizes high rank solutions. The optimization problem is

$$\min_{\tilde{M}} \sum_{M_{i,j}\neq 0} (M_{ij} - \tilde{M}_{ij})^2 + \lambda \text{rank}(\tilde{M}), \qquad \tilde{M} = U\Sigma V^T, \tag{1}$$

where the value of λ controls the importance of having a low-rank solution, i.e. for high values of λ, the solutions obtained have a very small rank whereas for small values of λ, the solutions are allowed to have a higher rank.

The optimization problem of equation (1) is combinatorially hard because of the form of the penalty term (the number of non-zero eigenvalues) [22]. A possible way to make it tractable is to replace the penalty by the nuclear norm of $\tilde{M}$, denoted as $\|\tilde{M}\|_*$, which is the sum of the eigenvalues of $\tilde{M}$

$$\min_{\tilde{M}} \sum_{M_{i,j}\neq 0} (M_{ij} - \tilde{M}_{ij})^2 + \lambda \|\tilde{M}\|_*. \tag{2}$$

Equation (2) is thus an effective convex relaxation of the original problem [3, 6]. The resulting problem is called *nuclear norm regularized least-squares problem*.

This formulation is equivalent to the one derived for sparse linear regression [5, 10]. The task in sparse linear regression is to learn the parameters of a linear model to predict the values of a response

variable given n samples, each one consisting of a p-dimensional input vector. For $p \gg n$, the standard least-squares solution requires the inversion of a matrix that is not well defined. A solution to this problem is to add a penalty to the least-squares error function, as in Equation (2). The ℓ_1-norm penalty on the linear weights (the sum of the absolute values) represents a convex relaxation of the ℓ_0-norm constraint (the number of absolute non-zero values). The ℓ_1-norm penalty (Lasso formulation [23]) is a good compromise between predictive power, interpretability and tractability. The ℓ_0-norm penalty (Spike-and-Slab model [7]) finds which are the relevant features directly, which is a combinatorial, non-convex problem [12].

Several methods have been derived recently for the nuclear norm regularized least-squares problem, e.g., the singular value thresholding (SVT) [2], soft-impute [20], accelerated proximal gradient approach [11]. In our work, we use the soft-impute method, introduced by Mazumder et al. [20]. The computational complexity of this method is linear in the size of the M. The parameters of this model are the regularization parameter λ and an specific *maximum rank* ($r_{\max}$) of the approximation $\tilde{M}$.

3.2 Classification Step

In this work, we differentiate between three different feature spaces that use the previous factorization $\tilde{M} = U\Sigma V^\top$.

- *Pure Recommender System* (PureRS) strategy: in this case, the feature of a voter and a candidate are given by the ith row of matrix U and the jth column of $V^\top$, respectively. The PureRS strategy therefore uses vectors $(U_{i,:}, V_{j,:}, y_{i,j})$ as training examples to learn a classifier, where $y_{i,j}$ denotes either voting assistance of voter i to the election of candidate j or vote polarity of voter i to candidate j, depending on the task under consideration.
- *Network Recommender System* (NetRS) strategy: The NetRS feature space incorporates the i's features as a candidate and j's features as a voter. Thus, the training examples are vectors of the form $(U_{i,:}, V_{i,:}, U_{j,:}, V_{j,:}, y_{i,j})$. In terms of the underlying network, this corresponds to account for some edges that the PureRS method would ignore. More precisely, U's dimensions synthesize ongoing edges, while V's dimensions synthesize incoming ones. When constructing a sample corresponding to edge (i, j) on Figure 1, the PureRS method would ignore the i incoming edges, like (h, i), and j outgoing ones, like (j, k), as Figure 2 depicts.

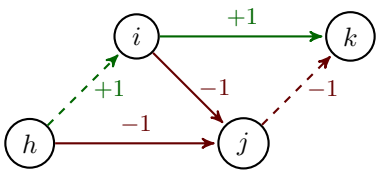

Figure 2: Example of missing (one degree) structure information when constructing $s_{i,j} = -1$ sample using the PureRS approach. Dashed edges correspond to the omitted part of the graph.

These feature spaces are used to learn different classifiers, which are task specific. As classifiers, we have used logistic regression and random forests.

4 Experimental Results

We now present some results on different voting networks, as well as other signed networks. We start showing some results of the performance of the soft-impute algorithm in terms of error in the matrix completion task and then we show the results on predicting voting polarity and participation.

4.1 Experimental Setup

We tested our algorithm in a voting dataset, the Wikipedia Request-for-Adminship (*Wiki-Rfa*) dataset [19]. We also experimented with other two datasets that consist of signed networks. They are described below. In general, class imbalance was handled by sub-sampling from the positive votes

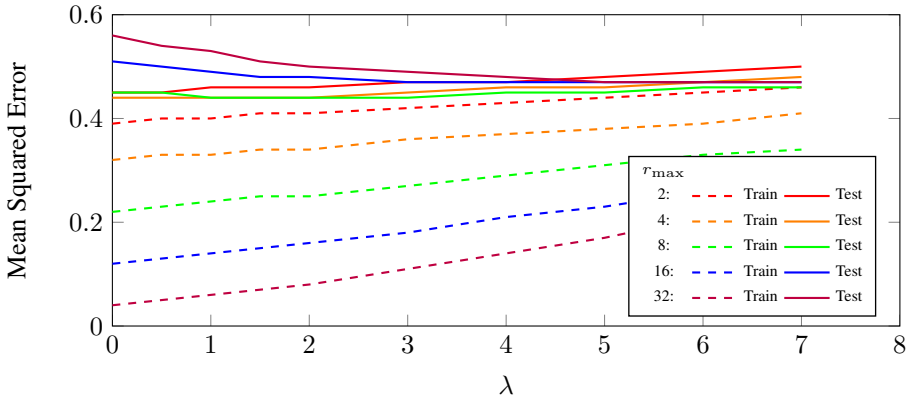

Figure 3: Mean square error between the data matrix M and the *Soft-Impute* approximation $\tilde{M}$ on the *Wiki-Rfa* dataset, for different values of the maximum ranks $r_{\max}$ and the regularizer λ.

class in the training set, after the optimization described in section 3.1. To evaluate the method, we partition randomly the data and use 10-fold cross validation over the entire set of votes. We simulate realistic conditions by reporting performance over the whole testing dataset.

Wikipedia is a free online encyclopedia where articles (almost all) can be edited by any person as long as they follow certain policies. More precisely, there are users called Admins that have been granted special permissions to be able to enforce the site's rules. Moreover, a Request for Adminship on Wikipedia is a deliberation process by which people cast a (not mandatory) vote to promote a user to *Admin*. The data collected by such process induces a signed network in which nodes are Wikipedia users and votes are links between them that can be positive $(+1)$, negative (-1) or neutral (0). The corresponding data we use in this study was collected from 2003 until May 2013 and consists on $11,402$ users that form $189,004$ distinct voter-candidate pairs from which 76% are positive. We preprocessed the data by removing all neutral (0) votes, as done in [16]. In case of multiple votes (repeated voter-candidate pair), the last one was used.

Slashdot, a technology news website created in 1997 that publishes frequently short news posts and allows its readers to comment on them. This website has been analyzed in several studies [8, 13, 14]. Slashdot allowed its members to categorize each other as *friend* or *foe*, meaning that they like or dislike respectively another user. Such behaviour generates a voting network where nodes are users and links are tagged with their assessment by each other: friend $(+1)$ or foe (-1). The website link tagging feature, called *Slashdot Zoo* was introduced in 2002, and it was crawled in February 2009. It has 82.144 nodes and $549,202$ edges (77% positive).

Epinions: is a recommendation network to help people make purchase decisions by reading user product reviews. Users could express their trust between each other. This information was used to choose which item reviews were shown to a specific user. We consider the directed trust graph of evaluations, with the nodes representing the users and the links representing the trust values, $+1$ or -1, depending if one user tagged another as trusted or untrusted, respectively. The website is now closed, and the information crawled correspond to all of its data. The corresponding network has $131,828$ nodes and $841,372$ edges (85% positive).

4.1.1 Results of Feature Selection

We first analyse the performance of the feature selection step as a function of both the regularization parameter λ and the maximum rank ($r_{\max}$) of $\tilde{M}$ [1] Figure 3 shows this performance in terms of the mean square error for the Wiki-Rfa dataset (the other datasets behave similarly). Dashed and continuous lines correspond to errors in the training and testing set, respectively.

[1] We used *Soft-Impute* R library
`https://cran.r-project.org/web/packages/softImpute/index.html`

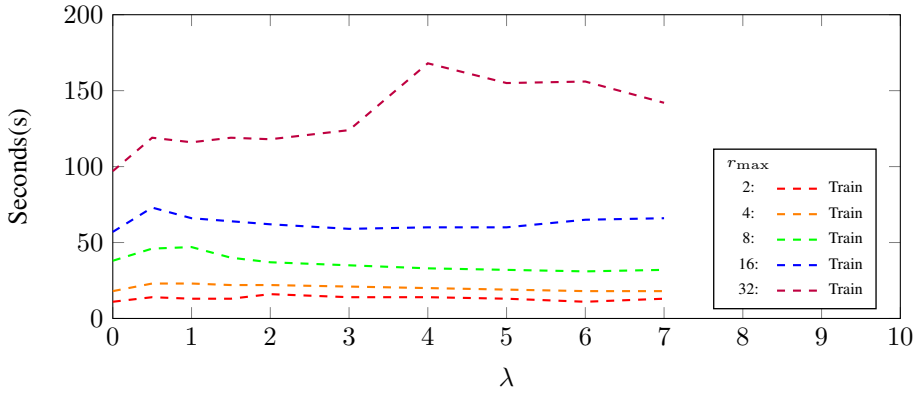

Figure 4: *Soft-Impute* execution time on the *Wiki-Rfa* dataset using different matrix ranks ($r_{\max}$) $\{2, 4, 8, 16, 32, 64\}$ for different values of the regularizer λ.

As we can see, for small values of the maximum rank ($r_{\max}$), both training and testing errors grow as we increase λ. On the contrary, for large values of $r_{\max}$, the training and testing error curves differ in their profiles. While training error increases with λ, testing error decreases. In addition, the difference between training and testing error also grows proportionally to $r_{\max}$. This behaviour can be understood in terms of bias and variance tradeoff. The maximum rank parameter $r_{\max}$ essentially determines the model complexity (the effective number of parameters to be learned). For large values of $r_{\max}$, we observe strong overfitting (small training errors and large testing errors) with a difference that is reduced as we increase the regularizer λ. This situation corresponds to having a model that is too expressive (large variance) and essentially fits the noise of the training set. On the contrary, for small values of $r_{\max}$, both training and testing curves show a similar profile and attain similar errors, indicating sufficient data to learning a correct model. Most of the error in this case can be attributed to the model bias.

Figure 4 shows running time for different parameter settings. We observe, on one hand, that the running time is roughly independent of λ for sufficiently small $r_{\max}$. On the other hand, we observe a (seemingly linear) increase as a function of the maximum rank $r_{\max}$. Therefore, for a big dataset it is recommended to use small values of $r_{\max}$, as we show later.

4.2 Results on Polarity Prediction

We now show some results on the polarity prediction task. The task is to predict the sign of a vote (positive or negative) of a participant, given that the voter participates in that election (the edge exists). In this case, the matrix M takes binary entries M_{ij} that are $+1$ if voter i voted favourably for candidate j, and -1 otherwise. In the feature selection step, null votes as well as missing votes are treated as missing values. In the classification step, an example is defined to belong to the class 1 for positive votes and 2 for negative votes.

To evaluate classification performance, we use two measures: the area under the ROC curve (AUC/ROC) and the accuracy. The AUC/ROC measures the probability that a randomly chosen positive sample will be ranked higher than a randomly chosen negative one. The accuracy is just the number of correctly classified examples divided the total number of examples, in the testing dataset. We show results for the random forest classifier only, since it generally outperforms logistic regression in our evaluation.

Tables 1 and 2 show results comparing the NetRS strategy with the PureRS strategy. We can see that the classification results are very good, and notice a small (but still significant) improvement of the NetRS strategy over the PureRS one, suggesting a benefit of incorporating the network features. Comparing the computational time between both strategies (Table 2), we observe that the NetRS approach roughly doubles the training time of the PureRS approach, due to the increased feature space.

Datasets	NetRS strategy (Testing set)		PureRS strategy (Testing set)	
	AUC/ROC	Accuracy	AUC/ROC	Accuracy
WikiRfa	**0.90**	**0.81**	0.89	0.80
Slashdot	**0.92**	**0.86**	0.89	0.79
Epinions	**0.97**	**0.96**	0.96	0.85

Table 1: Polarity prediction task results: comparison between the NetRS and the PureRS strategies using random forest classifiers. The table contains averaged results over ten runs of 10-fold cross validation. The data was shuffled for every run (10). Parameters $\lambda = 2, r_{\max} = 4$. In all cases, the standard deviation was smaller than 10^{-2}.

Datasets	Low Rank Approx.	NetRS strategy	PureRS strategy
Wiki-Rfa	23 (1)	140 (6)	64 (1)
Slashdot	166 (28)	838 (118)	421 (85)
Epinions	248 (39)	687 (111)	335 (65)

Table 2: Computation time (in seconds) for the polarity prediction task. The table contains the mean (standard deviation in parenthesis) of the time required in each of the two steps of our method for both NetRS and PureRS strategies, on the same experiments as in Table 1. The NetRS roughly doubles the training time of the PureRS, due to the increased feature space.

It is interesting to compare these results with the existing literature. For example, for the Wiki-RfA dataset, West et al. [24] reported AUC/ROC $= 0.82$ for 75% of evidence using the network-based approach (without sentiment analysis) and AUC/ROC $= 0.89$ using their full model (combining network and sentiment analysis) Remarkably, our proposed method using the NetRS strategy outperforms their network-based approach and shows comparable results to the full model, but *without incorporating* any textual (content-based) features, as shown in Table 1.

Another related study can be found in Lee et al. [16] where, in addition to the voting network, they use additional ad-hoc features from the communication graph (talk pages) that are not considered in our Wiki-RfA dataset. Their initial results show AUC/ROC $= 0.87$, which we improve using both PureRS and NetRS strategies, as table 1 indicates. Furthermore, they boost their initial results to AUC/ROC $= 0.90$ when the additional features were used. In this case, our NetRS strategy obtains comparable results *without incorporating* the communication graph.

Finally, we also obtain comparable or better results to the ones reported by Leskovec et al. in [17], although the comparison here is a bit more problematic, since they restrict their analysis to the nodes that have smaller values of the so-called embedness. Without restricting the node embedness, their reported performances are very similar to ours when compared with table 1 (see Figure 1 in [17] for details).

4.3 Results on Attendance Prediction

We now show some results on the attendance prediction task. The task is to predict whether a voter will participate, i.e. vote, on a given election. As before, the matrix M takes binary entries M_{ij} that are $+1$ if voter i voted favourably for candidate j, and -1 otherwise. In the feature selection step, null votes as well as missing votes are treated as missing values.

The difference with the previous task is in the classification step: here we consider all pairs as samples (including missing entries), whereas previously we only used those for which there was an entry in the utility matrix. Notice that, the dataset largely differs in size between the two tasks. In this case, since all edges are positive samples and missing ones become negative ones, the samples correspond to *every pair* of users, which results in a huge dataset (approximately 10^8 training samples) for the Wiki-RfA dataset.

We show some preliminary results for this task in Table 3. The method performs better than the baseline (0.5 AUC/ROC). Comparing the type of classifiers, we observe that random forest outperformed logistic regression, as before. We can also compare our approach to the one of Lee et al. [16].

Datasets	$r_{\max}$	AUC/ROC				Fit Time(s)	
		Random Forest		Logistic Regression		Random Forest	Logistic Regression
		Train	Test	Train	Test		
Wiki-Rfa	2	0.999	0.917	0.845	0.841	412s	3s
	4	1.000	0.942	0.838	0.834	531s	5s

Table 3: *Link Prediction* task classification and speed performace results using the *NetRS* strategy. The table contains the outcome of one 10-fold cross validarion run for each value of *Soft-Impute*'s max rank $mr \in \{2, 4\}$, using $\lambda = 2$ as regularization parameter.

Unlike the previous results in polarity prediction, our method performs slightly better when adding more latent features, i.e. increasing the low-rank approximation max rank parameter ($r_{\max}$).

5 Conclusions and Future Work

Motivated by the theory of recommender systems, we have developed a novel framework for predicting voting behaviour in social networks. Our approach learns a latent feature space from the networked data and uses it to train a classifier. The latent features are learned by solving a nuclear norm regularized least-squares problem. The method can then be applied not only to voting networks, but to general signed networks.

We have evaluated our method in two tasks (attendance and polarity prediction) in three different datasets Wiki-RfA, Slashdot and Epinions, and shown better or comparable performance results to existing approaches that use ad-hoc designed features. The presented approach, thanks to the methods used in its different steps, is able to scale up to very large datasets.

We are currently exploring two directions: the first one, is to follow a hybrid strategy. As it is done in recommender systems, one can use additional, content-based features [1]. This may help to address the cold start problem, since those features can be a good source of prior information.

Another interesting direction is related with the interpretability of the obtained features. The resulting factorization can be interpreted as the degree of membership of a user to a particular community. This interpretation suggest that such features can be particularly good to determine communities in networks. Finally, it is interesting to notice that it is shown how to use the features to predict node interaction values, but they can also be used for predicting a single node's attribute or behaviour.

References

[1] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions, 2005.

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

[3] E. J. Candès and T. Tao. The power of convex relaxation: Near-optimal matrix completion. *Information Theory, IEEE Transactions on*, 56(5):2053–2080, 2010.

[4] D. Cartwright and F. Harary. Structural balance: a generalization of Heider's theory. *Psychological review*, 63(5):277–293, 1956.

[5] J. Fan, F. Han, and H. Liu. Challenges of big data analysis. *National science review*, 1(2):293–314, 2014.

[6] M. Fazel. *Matrix rank minimization with applications*. PhD thesis, 2002.

[7] E. I. George and R. E. McCulloch. Variable selection via gibbs sampling. *Journal of the American Statistical Association*, 88(423):881–889, 1993.

[8] V. Gómez, A. Kaltenbrunner, and V. López. Statistical analysis of the social network and discussion threads in slashdot. In *Proceeding of the 17th international conference on World Wide Web (WWW 2008)*, pages 645–654, New York, NY, USA, 2008. ACM.

[9] R. Guha, R. Kumar, P. Raghavan, and A. Tomkins. Propagation of trust and distrust. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pages 403–412, 2004.

[10] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

[11] S. Ji and J. Ye. An accelerated gradient method for trace norm minimization. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 457–464, New York, NY, USA, 2009. ACM.

[12] H. J. Kappen and V. Gómez. The variational garrote. *Machine Learning*, pages 1–26, 2013.

[13] J. Kunegis, A. Lommatzsch, and C. Bauckhage. The slashdot zoo: mining a social network with negative edges. In *Proceedings of the 18th international conference on World wide web*, pages 741–750. ACM, 2009.

[14] C. Lampe and E. Johnston. Follow the (slash) dot: effects of feedback on new members in an online community. In *Proceedings of the 2005 international ACM SIGGROUP conference on Supporting group work*, pages 11–20. ACM, 2005.

[15] D. Lazer, A. S. Pentland, L. Adamic, S. Aral, A. L. Barabasi, D. Brewer, N. Christakis, N. Contractor, J. Fowler, M. Gutmann, et al. Life in the network: the coming age of computational social science. *Science (New York, NY)*, 323(5915):721, 2009.

[16] J. B. Lee, G. Cabunducan, F. G. C. Cabarle, R. Castillo, and J. A. Malinao. Uncovering the Social Dynamics of Online Elections. *Journal of Universal Computer Science*, 18(4):487–506, 2012.

[17] J. Leskovec, D. Huttenlocher, and J. Kleinberg. Predicting positive and negative links in online social networks. In *Proceedings of the 19th international conference on World wide web - WWW '10*, page 641, New York, New York, USA, Apr. 2010. ACM Press.

[18] J. Leskovec, D. Huttenlocher, and J. Kleinberg. Signed networks in social media. In *Proceedings of the 28th international conference on Human factors in computing systems - CHI '10*, page 1361, 2010.

[19] J. Leskovec and A. Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.

[20] R. Mazumder, T. Hastie, and R. Tibshirani. Spectral Regularization Algorithms for Learning Large Incomplete Matrices. *The Journal of Machine Learning Research*, 11:2287–2322, Mar. 2010.

[21] A. Rajaraman and J. D. Ullman. Mining of Massive Datasets. *Lecture Notes for Stanford CS345A Web Mining*, 67:328, 2011.

[22] N. Srebro and T. Jaakkola. Weighted low-rank approximations. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 720–727, 2003.

[23] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[24] R. West, H. S. Paskov, J. Leskovec, and C. Potts. Exploiting Social Network Structure for Person-to-Person Sentiment Analysis, Oct. 2014.