On Causal Matrix Completion

6.867 Final Document

Lucas Camelo Sa Florian Juengermann Víctor Quintas-Martínez

December 7, 2021

1 Introduction

We focus our efforts on the challenging problem of matrix completion. Formally, we are given a matrix of latent outcome variables $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ and a matrix of indicators $\boldsymbol{D} \in \{0,1\}^{m \times n}$ for whether each entry is revealed. We have access to noisy observations $\widetilde{\boldsymbol{Y}}$:

$$\widetilde{Y}_{ij} = \begin{cases} A_{ij} + \epsilon_{ij} & \text{if } D_{ij} = 1, \text{ with } \epsilon_{ij} \sim \mathcal{N}(0, \sigma_\epsilon^2) \\ \star & \text{otherwise.} \end{cases} \tag{1}$$

The question is: given $\widetilde{\boldsymbol{Y}}$, how can we retrieve $\boldsymbol{A}$? This minimalistic setting serves as a model for a wide variety of applications. A recurring example in literature is the Netflix recommendation problem, where each entry $\widetilde{Y}_{ij}$ is the rating user i gave to movie j. The full matrix is not observed, since not all users watch (and rate) all movies. However, a recommendation system would like to infer the unknown A_{ij} from the observed data on how similar users to i rated movie j, and how user i rated similar movies to j.

In order to complete $\widetilde{\boldsymbol{Y}}$, it is necessary to make assumptions about the underlying structure of $\boldsymbol{A}$ and to specify a model of missingness for $\boldsymbol{D}$. For the former, most of the existing literature assumes that $\boldsymbol{A}$ is (approximately) low-rank. As for the latter, the early literature focused on the "missing completely at random" (MCAR) case, where each entry of matrix $\boldsymbol{D}$ is drawn i.i.d. from a Bernoulli distribution. In this setting, result by Candès and Recht [2008] shows that if the number of revealed entries is reasonably large, then it is possible to retrieve the exact latent matrix with high probability. A variety of algorithms to complete MCAR data exists in the literature: e.g. `PMF` [Salakhutdinov and Mnih, 2008], `SVD` [Funk, 2006], `SVD++` [Koren, 2008], `softImpute` [Hastie et al., 2015] and `KNN` [Lee et al., 2016].

Nonetheless, the MCAR scenario is quite unrealistic in most practical applications. For instance, in the Netflix example, a user who does not enjoy horror films is unlikely to watch — and therefore rate — any particular movie of the genre. A more general framework considers "missing not at random" (MNAR) data.

The MNAR setting is challenging in that, if the dependence between $\boldsymbol{D}$ and $\widetilde{\boldsymbol{Y}}$ is ignored, estimates will be badly biased [Schnabel et al., 2016]. At the same time, the pattern of missingness can be leveraged to reveal information about $\boldsymbol{A}$ (e.g. which users have similar tastes). Agarwal et al. [2021] proposes an algorithm to deal with MNAR data, Synthetic Nearest Neighbors (`SNN`), that we will take as our starting point for this project. We describe this algorithm in detail in the next section. Other recent algorithms that are robust to some limited form of MNAR include `MaxNorm` [Cai and Zhou, 2016], `ExpoMF` [Liang et al., 2016], and `WTN` [Srebro and Salakhutdinov, 2010].

2 Synthetic Nearest Neighbors

We begin by describing Agarwal et al. [2021]'s K-Synthetic Nearest Neighbors (`SNN`) algorithm in more detail. Let (i,j) be a missing entry in $\widetilde{\boldsymbol{Y}}$. Define the *neighborhood rows* as

$\mathrm{NR}(j) = \{a \in [m] : D_{aj} = 1\}$ (i.e. rows where entries in column j are observed) and the *neighborhood columns* as $\mathrm{NC}(i) = \{b \in [n] : D_{ib} = 1\}$. (columns where entries in row i are not missing). Among those, we select a subset of *anchor rows* AR and *anchor columns* AC corresponding to a fully observed submatrix $\boldsymbol{S}$. Denote the columns in row i corresponding to AC by q, and the rows in column j corresponding to AR by x.

SNN works as follows:

STEP 1 For a missing entry (i, j), find the corresponding anchor matrix $\boldsymbol{S}$.

STEP 2 The submatrix $\boldsymbol{S}$ is split row-wisely in K disjoint blocks $\boldsymbol{S}^{(k)}$, $k \in [K]$.

STEP 3 For each of these blocks, a Principal Component Regression (PCR) of q on $\boldsymbol{S}^{(k)}$ is estimated, yielding a coefficient $\widehat{\beta}^{(k)}$. A prediction for entry (i, j) using synthetic NN k is then obtained as $\langle x^{(k)}, \widehat{\beta}^{(k)} \rangle$.

STEP 4 Finally, the K predictions are averaged.

The method is much more clearly understood with an image. Figure 1, taken from Agarwal et al. [2021], depicts the method graphically.

Figure 1: Graphical depiction of the SNNalgorithm (from Agarwal et al., 2021)

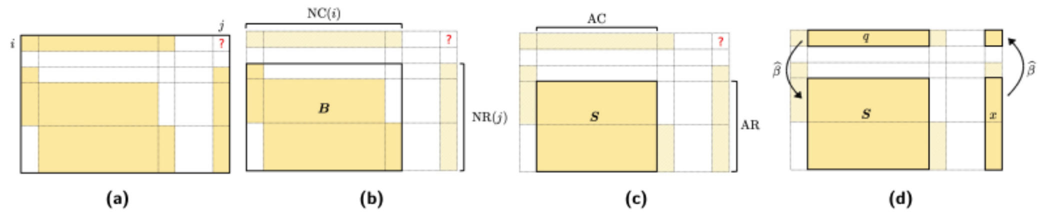


Figure 5: We visually depict the various quantities needed to define the SNN algorithm. Figure 5a depicts a particular sparsity pattern in our matrix $\boldsymbol{Y}$ with entry (i, j) missing. Figure 5b depicts $\mathrm{NR}(j)$ and $\mathrm{NC}(i)$. Figure 5c depicts AR, AC, and $\boldsymbol{S}$. Figure 5d depicts the SNN algorithm with $K = 1$; for $K > 1$, we partition the rows in S into K mutually disjoint sets.

Contribution In this project, we take the SNN algorithm as a baseline, and explore three questions. First, we notice that the problem of finding a fully observed anchor matrix $\boldsymbol{S}$ is computationally expensive and slow. In Section 3, we investigate faster alternatives to finding $\boldsymbol{S}$. Second, Agarwal et al. [2021] use PCR as a way to estimate missing entries using q, x and $\boldsymbol{S}$, but in some settings, other regression methods may be preferred. Section 4 considers two alternatives, Ridge and LASSO. We conduct ablation studies to assess the effect of varying certain hyperparameters, and we vary the experimental design to see which method works best under different data generating processes. Finally, Section 5 evaluates SNN in a real movie recommendation dataset, which we see as a complement to Agarwal et al. [2021]'s evaluation in synthetic experiments.

3 Faster Anchor Matrix Search

STEP 1 of the SNN algorithm is to find a fully observed anchor matrix $\boldsymbol{S}$. $\boldsymbol{S}$ is a complete submatrix of the matrix $\boldsymbol{B}$ that is induced by the neighboring rows and columns $NR(j), NC(i)$ (see Figure 1). Finding a complete submatrix is equivalent to finding a biclique in the bipartite graph induced by the rows and columns of the matrix. However, the edge maximum biclique problem is NP-complete (Peeters [2003]). To find a submatrix with optimal size, Agarwal et al. [2021] use an algorithm that enumerates all bicliques that are maximal, i.e. that are not contained in any larger biclique. As there can be exponentially many such bicliques, the runtime is inevitably exponential. In practice, using an 80×80 matrix and finding bicliques in 20×20 matrices already takes a couple of minutes.

In this work, we explore two alternative methods for finding anchor matrices: using an ensemble of multiple random, non-optimal anchor matrices and using an incomplete submatrix for the estimation.

Anchor Matrix Ensemble The inherent problem in the biclique search is that there are exponentially many bicliques that are maximal. But a biclique being maximal does not necessarily entail that the induced submatrix is useful for our purpose. For example, if we select a single movie and all the viewers of that movie, this may form a maximal submatrix, meaning it is not possible to add another movie or user without changing the other set. Still, this submatrix will not be helpful for our completion task. However, we hypothesize that in practice, if we combine estimates from multiple maximal bicliques, the expected result is reasonably good. In Figure B1 in the Appendix, we provide empirical support for this claim.

Incomplete Anchor Matrix As an alternative to finding a complete submatrix of $\boldsymbol{B}$, we propose using common matrix completion methods to complete $\boldsymbol{B}$ and then using it as an anchor matrix. This way, we avoid the computationally expensive biclique search altogether. In this work, we complete missing entries with column or row averages. We can either use the row/column average of the entire matrix $\widetilde{\boldsymbol{Y}}$ or only consider the neighbor matrix $\boldsymbol{B}$.

Experiments We evaluate our two approaches on the limited MNAR movie recommendation setting from Agarwal et al. [2021] (see Appendix A for a more detailed description). We find that the original approach yields the highest accuracies but takes 10 and 350 times longer than the ensemble method with 5 anchors and the incomplete anchor method on the medium-sized dataset. On a 160×160 matrix, the original approach times out, while the ensemble and incomplete matrix approach yield similar results, wherein the latter one is faster by a factor of 26. Table B1 and Figure B2a in the Appendix contain the exact results.

We also analyze how many anchor matrices we need in the ensemble to receive a good estimate. Even with 20 anchor matrices, we do not achieve the accuracy of the original method, but using 5 seems to be a good compromise between runtime and accuracy for our use case (see Figure B2b in the Appendix for a visualization).

In the general MNAR recommendation dataset as defined in Appendix A.2, there are very few maximal bicliques so that the ensemble approach yields similar results to the original one, just with a longer runtime. The incomplete anchor matrix approach performs worse than in the limited MNAR case as it does not capture the information in the missingness. Additionally, we observe that using the entire matrix averages versus averages from the neighbor matrix $\boldsymbol{B}$ is slightly more beneficial in the limited MNAR case but significantly worse in the MNAR case (exact results are shown in Table B2).

4 Alternative Regression Models

Step 3 of the `SNN` algorithm uses a PCR of q on $\boldsymbol{S}^{(k)}$ to construct a prediction for the missing entry as $\langle x^{(k)}, \widehat{\beta}^{(k)} \rangle$. The intuition for this is as follows: you want to predict user i's rating of movie j using a linear combination of ratings for the same movie by similar users for whom these are observed, x. In the MNAR case, similarity is based on the pattern of missingness, since that can be informative about users' and movies' underlying characteristics. The weights in this linear combination, $\widehat{\beta}$, are learnt from a regression of i's observed ratings, q, on $\boldsymbol{S}$.

Agarwal et al. [2021] provide a theoretical justification for this approach in the form of a low-rank latent factor model $\boldsymbol{Y} = \boldsymbol{U}\boldsymbol{V}^T + \boldsymbol{E}$ with $\mathbb{E}[\boldsymbol{E} \mid \boldsymbol{U}, \boldsymbol{V}, \boldsymbol{D}] = 0$. In their Theorem 1, they show that for any sufficiently large subset $\mathcal{I} \subset \mathrm{NR}(j)$, entry A_{ij} is identified as $A_{ij} = \sum_{\ell \in \mathcal{I}} \beta_\ell \mathbb{E}[\widetilde{Y}_{\ell j} \mid \boldsymbol{U}, \boldsymbol{V}, \boldsymbol{D}]$ for some vector β. The main finite-sample consistency result, Theorem 2, gives the rate of convergence. How fast $\widehat{A}_{ij}$ converges to A_{ij} will be thightly linked to $\mathbb{E}\|\widehat{\beta} - \beta\|^2$, the MSE rate of convergence of the regression estimator.

In some settings it is plausible that other regression estimators may have better properties than PCR. For example, if we believe that β is very sparse (in the movie rating example, this would mean that user i can be approximated by a small subset of other users), then LASSO would seem an attractive alternative. In this project we want to explore two alternatives to PCR: Ridge and LASSO regression. Although giving formal rate of convergence results for these estimators is beyond the scope of this paper, we evaluate this alternatives with different choices of hyperparameters and under different experimental designs to investigate when each method may be most appropriate.

Ablation Studies First, we investigate the effect of the choice of hyperparameters on accuracy for the three regression methods considered (PCR, Ridge and LASSO). We run experiments based on the general MNAR design of Agarwal et al. [2021] described in Appendix A, and we vary the regularization strength and the number of neighbors. We consider both absolute and adaptive choices of parameters.

In terms of regularization, it seems that low levels of regularization work best for all methods. The results are in Table B7. In the PCR case, the absolute regularization parameter λ refers to the minimum eigenvalue threshold (we keep principal components with eigenvalue above said threshold); in the Ridge and LASSO case, this corresponds to the respective penalties. For adaptive regularization we consider $\gamma \times$optimal threshold for PCR, where the optimal threshold is given by Donoho and Gavish [2013], and $\gamma \times |\mathrm{AR}|/|\mathrm{AC}|$ for Ridge and Lasso. The best results for all methods are obtained with very small regularization, either $\lambda = 0.0001$ or $\gamma = 0.0001$. Under those conditions, Ridge and LASSO (RMSE 0.027 ± 0.003 for both) perform slightly better than PCR (RMSE 0.037 ± 0.001).

The results for the number of neighbors are in Table B4. We consider either a fixed number of neighbors K, or an adaptive choice $\min\{1, \lfloor|\mathrm{AR}|/\kappa\rfloor\}$. It seems that using more than 1 neighbor actually worsens performance.

Design Studies We also study how the different methods perform when varying the parameters of the design. The experiments are again based on the general MNAR design of Agarwal et al. [2021] described in Appendix A.

The first parameter we experiment with is the Dirichlet concentration parameter α of the distribution of the weights on core movies. The idea is the following: when α is very low, weights will tend to be very close to 0 or 1, so we can expect β in the representation of A_{ij} to be sparser; conversely, if α is large, all weights in β should be similar in magnitude. For that reason, we would have expected LASSO to perform best when α is low, and Ridge or PCR to perform best when α is high. The results are in Table B5. Surprisingly, all methods rank similarly for different values of α (the differences in MAE and RMSE seem to be statistically insignificant).

The second parameter we experiment with is the approximate rank r (the dimension of the latent variables). All methods perform worse when $r = 20$ is large. LASSO and Ridge perform equally well when $r = 2, 5$ is small or moderate. In contrast, PCR performs comparably well only if $r = 5$, otherwise it has a substantially larger MAE and RMSE than Ridge and LASSO. An intuition for why this happens is that, for PCR to recover A_{ij} accurately, we would like to keep exactly r principal components (that is also why this method is most sensitive to regularization, as can be seen in Table B7). Hence, in practice Ridge and LASSO seem more attractive alternatives.

5 Evaluation on Real-World Data

As part of our efforts to evaluate Agarwal et al. [2021]'s approach as well as our proposed variations, we attempted to run the techniques developed above on real world data. We chose the well-studied MovieLens dataset introduced by Harper and Konstan [2015].

The dataset consists of movie ratings collected on the movielens.org platform between 1996 and 2018. We analyzed more than $100,000$ ratings distributed across more than $9,000$ movies and 600 different users. Additionally, each user had rated more than 20 movies and allowed ratings were valued from 0 to 5 with .5 increments.

Since both training and testing had to be performed on the same set, k-fold cross-validation arises as a natural choice for evaluating our results. More specifically, we partition our ratings across $K = 10$ disjoint subsets, and then separately predict each subset's elements based on the other $K - 1$ subsets. We then average the RMSEs over all K tests as our evaluation metric.

We then attempted to run the original method from Agarwal et al. [2021] against this dataset. As discussed in previous sections, the original method proved to be computationally infeasible; we were not able to predict a single rating with our computational resources and such a large dataset. Ensembling anchor matrices helps but is not ideal either. As discussed previously, instead of searching exhaustively for the optimal anchor matrix, we can randomly find several maximal anchor matrices and then average over the resulting predicted ratings. Ensembling 10 maximal matrices with ridge regression yields an RMSE of 1.031, but at 12 seconds per predicted rating, which would become infeasible in a large dataset.

Apart from ensembling, another alternative method of obtaining maximal anchor matrices consists of completing the matrix $\boldsymbol{B}$ by setting missing values to column averages, as explained in previous sections. With this method, both ridge regression, lasso, and PCR behaved fairly similarly. Adjusting the regression coefficients in the ridge regression method managed to bring RMSE down to 0.917 (Lasso and PCR had RMSEs of 0.964 and 0.944, respectively). The three techniques took between 28 and 36 iterations/second, with Lasso being the slowest and PCR the fastest.

We compared our results against several naive baseline methods for prediction. One example consists of predicting a movie's rating by using its most frequent score. This yielded an RMSE of 1.02. Similar results can be obtained by using the user's most frequent rating (RMSE=1.115) and applying simple heuristics to choose a linear combination of anchor rows (RMSE=1.021). Still, this means our methods produce an average 10% improvement in accuracy against naive baselines.

The data suggests it might be difficult to push the RMSE too far below 1 without some degree of sophistication. We went over pairs of users and tried to make sense of the actual proximity of their respective ratings versus how much intersection there was between their lists of rated movies. Specifically, for each pair of users (i, j), we compute $S(i, j) = \frac{\langle D_i, D_j \rangle}{\langle D_i, D_i \rangle}$, where D_i is the indicator vector of movies user i rated — i.e. $S(i, j)$ is the fraction of movies i rated that j also rated. We then plotted $S(i, j)$ versus RMSE(i, j), the rooted mean squared difference between ratings of movies both i and j rated. What we observed was that as $S(i, j)$ approaches 1, RMSE(i, j) approaches 1 too — i.e., even when j watches most of the movies i watches, j's ratings are still, on average, 1 away from i.

6 Conclusion

Matrix completion in the MNAR framework is a challenging problem with important applications, such as movie recommender systems. In this project, we try to expand on the `SNN` algorithm of Agarwal et al. [2021] in three ways.

First, we point out the computational infeasibilty of the current approach on large datasets and propose using ensembles of non-optimal anchor matrices as well as simple completion heuristics to speed up the anchor matrix search. We achieve a significant reduction in runtime, unfortunately at the cost of accuracy.

Second, we evaluate using Ridge and LASSO regression in the predictive step, instead of PCR. We find that these methods have better and more stable performance in a variety of experimental designs. Additionally, it seems that using more than one synthetic neighbor deteriorates accuracy, and, in our setting, lower levels of regularization work best.

Finally, we apply all the methods we developed to a large real-world dataset of movie recommendations. The algorithm from Agarwal et al. [2021] becomes computationally prohibitive even when replacing anchor matrices with ensembling. However, by completing our anchor matrices via heuristics, we were able to run our different regression methods and obtained a 10% improvement in accuracy against naive baselines.

Team Contributions

The three authors contributed equally, where Florian Juengermann was responsible for the anchor method search described Section 3, Víctor Quintas-Martínez analyzed the alternative regression models in Section 4, and Lucas Camelo Sa evaluated the approach on real-world data in Section 5. Sections 1, 2 and 6 were written collaboratively.

References

Anish Agarwal, Munther Dahleh, Devavrat Shah, and Dennis Shen. Causal matrix completion. *arXiv pre-print 2109.15154*, 2021.

T Tony Cai and Wen-Xin Zhou. Matrix completion via max-norm constrained optimization. *Electronic Journal of Statistics*, 10(1):1493–1525, 2016.

Emmanuel J. Candès and Benjamin Recht. Exact matrix completion via convex optimization. 2008.

David L Donoho and Matan Gavish. The optimal hard threshold for singular values is $4/\sqrt{3}$, 2013.

Simon Funk. Netflix update: Try this at home. 2006. URL `https://sifter.org/simon/journal/20061211.html`.

F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4):1–19, 2015.

Trevor Hastie, Rahul Mazumder, Jason D Lee, and Reza Zadeh. Matrix completion and low-rank SVD via fast alternating least squares. *The Journal of Machine Learning Research*, 16 (1):3367–3402, 2015.

Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 426–434, 2008.

Christina E Lee, Dogyoon Song, Yihua Li, and Devavrat Shah. Blind regression: Nonparametric regression for latent variable models via collaborative filtering. *Advances in Neural Information Processing Systems*, 29:2155–2163, 2016.

Dawen Liang, Laurent Charlin, James McInerney, and David M Blei. Modeling user exposure in recommendation. In *Proceedings of the 25th international conference on World Wide Web*, pages 951–961, 2016.

René Peeters. The maximum edge biclique problem is np-complete. *Discrete Applied Mathematics*, 131(3):651–654, 2003.

Ruslan Salakhutdinov and Andriy Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *Proceedings of the 25th International Conference on Machine Learning*, pages 880–887, 2008.

Tobias Schnabel, Adith Swaminathan, Ashudeep Singh, Navin Chandak, and Thorsten Joachims. Recommendations as treatments: Debiasing learning and evaluation. In *international conference on machine learning*, pages 1670–1679. PMLR, 2016.

Nathan Srebro and Ruslan Salakhutdinov. Collaborative filtering in a non-uniform world: Learning with the weighted trace norm. *arXiv preprint arXiv:1002.2780*, 2010.

Appendix

A Experimental Design

Our experimental designs are drawn from Agarwal et al. [2021]'s recommendation system synthetic examples for comparability. To fix ideas, suppose that entry (i, j) in these experiments represents how user i rates movie j (ratings are numbers from 1 to 5). In the base case, following their design, we consider an 80×80 matrix $\boldsymbol{A}$.[1]

A.1 Limited MNAR

For the limited MNAR design, we divide both users and movies into a "core" subset (of size 20) and a "standard" subset. The real-world phenomenon that this is trying to capture is that most users will watch only a few movies, but a few users will tend to watch (and rate) many of them. The entries of the matrix are generated based on a latent factor model with a low-rank structure and re-scaled to take values in $[1, 5]$.

For core users, a user latent factor $\boldsymbol{U}_0 \in \mathbb{R}^{20\times r}$ is drawn from a standard Gaussian distribution. For standard users, we construct their user latent factor as a linear combination $\boldsymbol{U}_1 = \boldsymbol{B}\boldsymbol{U}_0 \in \mathbb{R}^{60\times r}$, where $\boldsymbol{B} \in \mathbb{R}^{60\times 20}$ is i.i.d. sampled from a Dirichlet distribution with all concentration parameters equal to α. In the base case, we take $r = 5$ and $\alpha = 0.01$.[2] We draw the movie latent factor for core movies $\boldsymbol{V}_0$ and for standard movies $\boldsymbol{V}_1$ in an analogous manner. Next, the underlying rating matrix is constructed as $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{V}^T \in \mathbb{R}^{80\times 80}$, where $\boldsymbol{U} = [\boldsymbol{U}_0, \boldsymbol{U}_1]$ and $\boldsymbol{V} = [\boldsymbol{V}_0, \boldsymbol{V}_1]$. Finally, the entries of $\boldsymbol{A}$ are re-scaled and clipped to lie within $[1, 5]$.

On the other hand, the model for the propensity matrix $\boldsymbol{P}$, where $p_{ij} = \Pr(D_{ij} = 1)$,, is as follows:

$$p_{ij} = \begin{cases} \kappa_{ij}\alpha_{ij}^{A_{ij}-1}, & \text{if } A_{ij} \in [1, 2.3], \\ \kappa_{ij}\alpha_{ij}^{5-A_{ij}}, & \text{if } A_{ij} \in (2.3, 5]. \end{cases}$$

As Agarwal et al. [2021], we set $\alpha_{ij} = 0.7$ if i is a core user and j is a core movie, $\alpha_{ij} = 0.35$ if i is a core user and j is a standard movie or vice versa, or $\alpha_{ij} = 0.1$ if both are standard. The κ_{ij} are chosen to guarantee an expected number of observations of 90% for core users and movies, 70% for core users and standard movies or vice versa, or 5% for standard users and movies.

This design tries to capture two real-life features of movie recommendation systems. On the one hand, the distinction between "core" and "standard" users and movies represents the fact that a few users will tend to watch a wide variety of movies, whereas the rest of users will watch mostly only the most popular films. At the same time, a few films will be seen and rated by a majority of users. On the other hand, the model for the propensity matrix is MNAR (missing not at random), as controlled by α_{ij}: the smaller α_{ij}, the more likely it will be that only extreme ratings (1 or 5) will be reported, which captures the fact that, in reality, people tend to rate only films that they really like or dislike.

A.2 General MNAR

The rating matrix is drawn in a similar form to the limited MNAR case, with the difference that all users are core users (so $\boldsymbol{U} \in \mathbb{R}^{80\times r}$ is drawn from a standard Gaussian distribution) and there are 30 core movies, $\boldsymbol{V}_0 \in \mathbb{R}^{30\times r}$ drawn from a standard Gaussian, $\boldsymbol{V}_1 = \boldsymbol{B}\boldsymbol{V}_0 \in$

[1]Source code to reproduce our results is avalailable at `https://github.com/florianjuengermann/on-causal-matrix-completion`

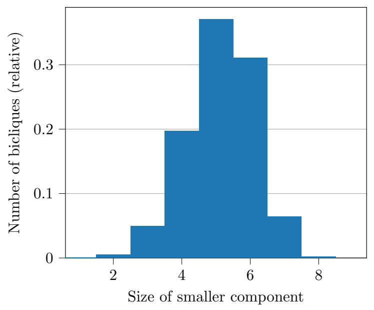
[2]Agarwal et al. [2021] do not report the value of α that they used, but we chose it by eyeballing to obtain a distribution of samples that mimics theirs.

$\mathbb{R}^{50\times r}$ with $\boldsymbol{B}$ i.i.d. Dirichlet with all concentration parameters set to α. In the base case, again, we consider $r = 5$ and $\alpha = 0.01$ Then, $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{V}^T \in \mathbb{R}^{80\times 80}$, where $\boldsymbol{V} = [\boldsymbol{V}_0, \boldsymbol{V}_1]$.

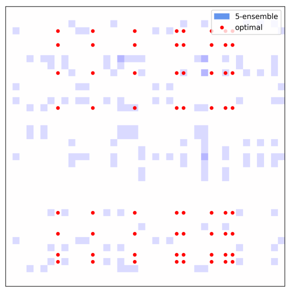
The main difference is in the construction of $\boldsymbol{D}$, which is deterministic (given $\boldsymbol{U}$) rather than stochastic. To give a real like analogy, Agarwal et al. [2021] interpret r as the number of "movie genres," entry (i, k) of $\boldsymbol{U}$ represents how much user i likes genre k, and entry (j, k) for $\boldsymbol{V}$ represents how much film j falls into genre k. Then, they set $D_{ij} = 1$ for all users i and all core movies j (core movies are "blockbusters" that everyone watches and rates). For the non core movies, let $k^u(i) = \arg\max_{k\in[r]} U_{ik}$ (i's favorite genre) and $k^m(i) = \arg\max_{k\in[r]} V_{jk}$ (the genre that matches movie j the most). They set $D_{ij} = 1$ if and only if $k^u(i) = k^m(j)$ (i.e. users only rate movies in their favorite genre). This is a very form of MNAR, since some entries are never observed.

B Additional Tables and Figures

B.1 Anchor Matrix Search

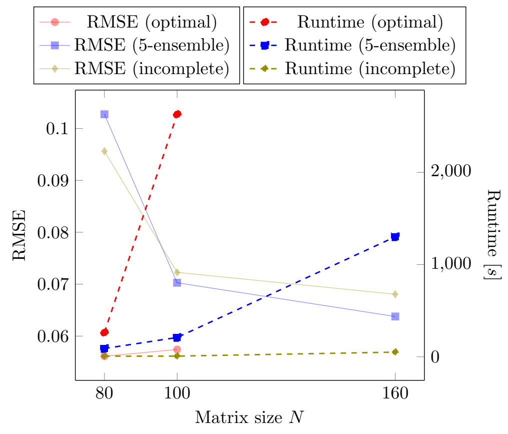


(a)

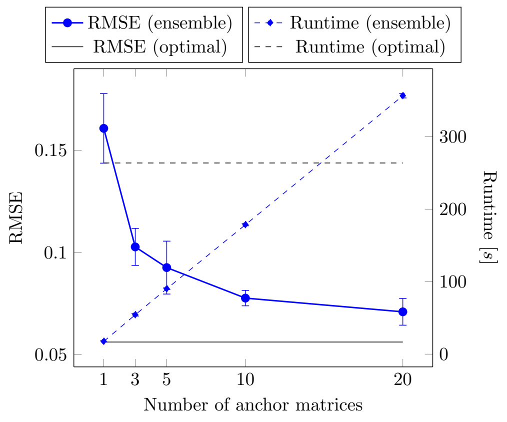


(b)

Figure B1: Distribution of maximal bicliques for a random 40×40 matrix. (a) shows the size distribution according to the smaller dimension. (b) shows an overlay of 5 randomly sampled bicliques compared to one optimal sized biclique. The optimal bicliques cover 8 rows and 8 columns, but if we select 5 random maximal bicliques, we cover a larger portion of the matrix than the single 8-by-8 matrix.



(a)



(b)

Figure B2: RMSEs and runtimes on the limited MNAR recommendation system for different anchor matrix finding methods. (a) shows the baseline with a single but optimal anchor matrix, the randomized approach with 5 matrices, and the incomplete matrix approach for matrices different sizes $N \times N$. (b) shows how the number of anchor matrices effects the second approach (for $N = 80$). All test use the Ridge regression model with parameter $\lambda = 0.001$.

Table B1: RMSEs and runtimes on the limited and general MNAR recommendation system of different sizes $N \times N$ for different anchor matrix finding methods: the baseline with a single but optimal anchor matrix, the randomize ensemble with 5 matrices, and the incomplete matrix approach. For all test, we use the Ridge regression model with parameter $\lambda = 0.001$.

	optimal		5-ensemble		incomplete	
	time [s]	RMSE	time [s]	RMSE	time [s]	RMSE
N	**Limited MNAR**					
80	264	0.06	90	0.10	6	0.10
100	2630	0.06	209	0.07	7	0.07
160	-		1301	0.06	50	0.07
N	**General MNAR**					
80	23	0.03	111	0.03	3	0.12
100	54	0.04	275	0.04	4	0.12
160	-		3664	0.03	45	0.11

Table B2: RMSE of the incomplete anchor matrix approach with different way of calculating the row or column averages for completion. For the row "complete", we use the averages from the entire matrix, for row "submatrix" we use the averages from the neighboring matrix $\boldsymbol{B}$. Result are the means and standard deviations of 5 runs using the Ridge estimator on a $N = 80$ matrix.

	Limited MNAR	General MNAR
complete	0.07 ±0.1	0.20 ±0.01
submatrix	0.08 ±0.1	0.12 ±0.00

B.2 Alternative Regression Models

B.2.1 Ablation

Table B3: Effect of regularization. We keep the number of neighbors $K = 1$ fixed and use the general MNAR design. See main text for the definition of λ and γ

	General MNAR					
	PCR		Ridge		LASSO	
	MAE	RMSE	MAE	RMSE	MAE	RMSE
λ	**Absolute Regularization**					
0.0001	0.007 ± 0.001	0.037 ± 0.009	0.006 ± 0.000	0.027 ± 0.003	0.007 ± 0.000	0.027 ± 0.003
0.001	0.007 ± 0.001	0.037 ± 0.009	0.006 ± 0.000	0.027 ± 0.003	0.010 ± 0.000	0.030 ± 0.002
0.01	0.006 ± 0.001	0.027 ± 0.003	0.008 ± 0.000	0.028 ± 0.003	0.020 ± 0.000	0.047 ± 0.001
0.1	0.009 ± 0.001	0.031 ± 0.003	0.011 ± 0.000	0.032 ± 0.003	0.086 ± 0.002	0.159 ± 0.003
0.5	0.022 ± 0.001	0.051 ± 0.004	0.016 ± 0.000	0.040 ± 0.002	0.233 ± 0.003	0.425 ± 0.007
1	0.031 ± 0.006	0.072 ± 0.015	0.019 ± 0.000	0.047 ± 0.002	0.277 ± 0.003	0.510 ± 0.005
γ	**Adaptive Regularization**					
0.0001	0.007 ± 0.001	0.037 ± 0.009	0.006 ± 0.000	0.027 ± 0.003	0.007 ± 0.000	0.027 ± 0.003
0.001	0.007 ± 0.001	0.037 ± 0.009	0.006 ± 0.000	0.027 ± 0.003	0.009 ± 0.000	0.029 ± 0.002
0.01	0.007 ± 0.001	0.028 ± 0.003	0.007 ± 0.000	0.028 ± 0.003	0.016 ± 0.000	0.040 ± 0.002
0.1	0.021 ± 0.004	0.077 ± 0.024	0.010 ± 0.000	0.031 ± 0.003	0.054 ± 0.001	0.104 ± 0.001
0.5	0.063 ± 0.011	0.216 ± 0.039	0.013 ± 0.000	0.036 ± 0.002	0.170 ± 0.004	0.311 ± 0.008
1	0.096 ± 0.011	0.277 ± 0.028	0.016 ± 0.000	0.040 ± 0.002	0.239 ± 0.004	0.437 ± 0.009

Table B4: Effect of number of neighbors. We keep adaptive regularization $\gamma = 0.0001$ fixed and use the general MNAR design. See main text for the definition of K and κ

	General MNAR					
	PCR		Ridge		LASSO	
	MAE	RMSE	MAE	RMSE	MAE	RMSE
K	**Absolute Number of Neighbors**					
1	0.023 ± 0.007	0.083 ± 0.027	0.009 ± 0.001	0.032 ± 0.003	0.018 ± 0.001	0.044 ± 0.003
2	0.102 ± 0.011	0.259 ± 0.032	0.013 ± 0.002	0.039 ± 0.005	0.020 ± 0.002	0.048 ± 0.005
5	0.370 ± 0.007	0.667 ± 0.008	0.114 ± 0.003	0.225 ± 0.004	0.114 ± 0.003	0.226 ± 0.004
κ	**Adaptive Number of Neighbors**					
20	0.023 ± 0.007	0.083 ± 0.027	0.009 ± 0.001	0.032 ± 0.003	0.018 ± 0.001	0.044 ± 0.003
10	0.024 ± 0.007	0.083 ± 0.026	0.009 ± 0.001	0.032 ± 0.003	0.018 ± 0.001	0.044 ± 0.003
5	0.216 ± 0.015	0.407 ± 0.025	0.021 ± 0.001	0.051 ± 0.002	0.026 ± 0.001	0.058 ± 0.002

B.2.2 Design

Table B5: Effect of α. We keep the number of neighbors $K = 1$ and adaptive regularization $\gamma = 0.0001$ fixed, and use the general MNAR design

	PCR		Ridge		LASSO	
	MAE	RMSE	MAE	RMSE	MAE	RMSE
α	**Dirichlet Concentration Param.**					
0.001	0.088 ± 0.018	0.198 ± 0.066	0.068 ± 0.003	0.123 ± 0.003	0.070 ± 0.003	0.126 ± 0.003
1	0.018 ± 0.002	0.039 ± 0.003	0.018 ± 0.002	0.039 ± 0.003	0.021 ± 0.001	0.042 ± 0.002
10	0.016 ± 0.002	0.039 ± 0.004	0.014 ± 0.002	0.036 ± 0.003	0.016 ± 0.001	0.034 ± 0.003

Table B6: Effect of r. We keep the number of neighbors $K = 1$ and adaptive regularization $\gamma = 0.0001$ fixed, and use the general MNAR design

	PCR		Ridge		LASSO	
	MAE	RMSE	MAE	RMSE	MAE	RMSE
r	**Dimension of Latent Variables**					
2	0.281 ± 0.036	0.623 ± 0.080	0.054 ± 0.004	0.121 ± 0.006	0.055 ± 0.004	0.124 ± 0.006
5	0.061 ± 0.004	0.111 ± 0.005	0.061 ± 0.004	0.109 ± 0.004	0.062 ± 0.004	0.112 ± 0.004
20	0.435 ± 0.010	0.713 ± 0.010	0.277 ± 0.006	0.462 ± 0.011	0.277 ± 0.006	0.462 ± 0.011

B.3 Real-World Data

Figure B3: Plotting mean squared error vs fraction of movies watched. For a fixed user i, another user j's ratings will be 1 away from i's, even when j has rated most of i's movies. This explains why naive algorithms can only go so far.

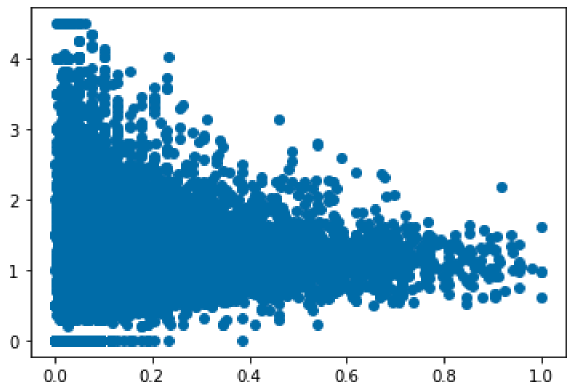


Table B7: Effect of different choices of regularization parameters. Higher parameters yield better results, contradicting the synthetic setting. Ridge regression also outperforms LASSO slightly.

	Ridge		LASSO	
	MAE	RMSE	MAE	RMSE
λ	**Regularization factor**			
0.001	1.006	1.549	1.009	1.473
0.01	0.85	1.19	0.834	1.135
0.1	0.718	0.974	0.709	0.969
0.5	0.678	0.917	0.712	0.964
1	0.680	0.917	0.721	0.972