Big Data is Low Rank

Madeleine Udell

May 3, 2019

Abstract

This column defends the assertion that big data is low rank and considers implications for data scientists and opportunities for optimizers.

1 Introduction

Low rank models have demonstrated effective performance in a wide range of data science applications. In this column, we survey a few of the more surprising applications of low rank models in data science, introduce a mathematical explanation for their effectiveness, and survey optimization approaches and challenges in fitting these models.

We'll start by giving a flavor of the challenges in these data science applications. Suppose you have collected information (*features*) on a number of distinct entities (*examples*). Often these examples are people. Features might include

- the opinions of every respondent to a survey,
- the purchase and browsing history for each customer who has visited an e-commerce website,
- the financial history of a credit card applicant, or
- the medical record of every patient at a hospital,

in addition to demographic characteristics.

In other applications, examples might be the following:

- Securities in a financial model. Here, features might include stock performance, accounting metrics, and indicators of environmental stewardship and of sound corporate governance.
- Samples of tumors from different individuals. Here, features might include immunological markers, size, location, vascularization, and indicators of key mutations.
- Geolocations. Here, features might include local demographics or daily weather over the past year.
- Datasets or problem instances. Here, features might include performance of various algorithms and heuristics to fit the dataset or solve the problem.

In the applications above, the features can be numeric, Boolean, ordinal, or categorical. Even among numeric features, the data can be on wildly different scales or follow very different statistical distributions. Some data may be corrupted with gross errors: survey respondents may lie or misunderstand the question, data may be improperly coded, or doctors inputting their patients' data may make a mistake in haste. Medical records contain instances of babies born weighing hundreds of pounds, and credit card records contain applicants whose credit score was 999 (out of 800).

Furthermore, many entries may be missing: questions skipped on surveys, patients who died or dropped out of a panel study, new questions added several years into the study, medical tests deemed unnecessary, sensors that failed, concentrations below a machine's sensitivity threshold, locations covered by clouds, eyes covered by sunglasses, algorithms that took too long to run. Notice that *whether or not* each entry is missing is always observed. The pattern of missingness may itself be informative about the value of the entry, or it may not be.

Other information is sometimes available. For example, the data may have internal structure (vector, matrix, tensor), Perhaps each observation is associated with some relevant covariates or is known to have been recorded at a particular time. The data may not be available all at the same time, but rather as a stream of observations.

Data analysts may be interested in a variety of related questions. Can we impute missing data and denoise observed data? Which features are correlated? Which examples are similar? How many *effective* features are present, and how many are just noise? How can this dataset best be used to predict some other quantity of interest for each example? If the dataset spans a long time period, have the statistics of the data changed during that period and, if so, when? Can we learn from the dataset without snooping into the future? When the dataset is large, developing efficient algorithms to answer these questions is important.

This column will discuss optimization methods to answer these questions (and more) by identifying low rank structure in the dataset. These techniques have been studied for over a century, and the literature is correspondingly large. We cannot hope to provide a comprehensive survey here. Instead, we present a few surprising applications of these methods, introduce the mathematics of low rank models, discuss optimization methods to fit these models, and examine why these techniques are effective for such a wide variety of problems.

2 Model

Suppose that the data is collected into a table A with m rows (one for each example) and n columns (one for each feature). The value of the jth feature for the ith example is written as A_{ij}. Some entries may also be *missing* or unobserved. Define $\Omega \subseteq [m] \times [n]$ to be the set of observed entries.

Low rank models make one simple—and seemingly strong—assumption about the data. They posit that every example $i = 1, \ldots, m$ can be represented by a low-dimensional vector $x_i \in \mathbf{R}^k$ and that every feature $j = 1, \ldots, n$ can be represented by a low-dimensional vector $y_j \in \mathbf{R}^k$ so that

$$x_i^T y_j \approx A_{ij} \tag{1}$$

for every observation $(i, j) \in \Omega$. We call these vectors the low-dimensional *representations* of each example and feature.

Notice that the left-hand side of Eq. (1) is a number, while the right-hand side can be Boolean, ordinal, or categorical. What can "$\approx$" mean in this case? The solution we adopt here is to choose a loss function ℓ and to define $\approx$ so that

$$\ell(x_i^T y_j, A_{ij}) \text{ is small} \iff x_i^T y_j \approx A_{ij}.$$

We'll discuss a few common loss functions in Section 6; for a more extensive review, see (UHZB16) or the software package `LowRankModels.jl`.

Collecting the representations into matrices

$$\begin{bmatrix} X \end{bmatrix} = \begin{bmatrix} —x_1— \\ \vdots \\ —x_m— \end{bmatrix}, \quad \begin{bmatrix} \quad Y \quad \end{bmatrix} = \begin{bmatrix} | & & | \\ y_1 & \cdots & y_n \\ | & & | \end{bmatrix},$$

we see the parameter k controls the rank of the matrix XY.

To fit a low rank model, we seek representations $x_i \in \mathbf{R}^k$ for each example $i = 1, \ldots, m$ and $y_j \in \mathbf{R}^k$ for each feature $j = 1, \ldots, n$ to minimize the sum of the losses over the observed entries,

$$\sum_{(i,j)\in\Omega} \ell(x_i^T y_j, A_{ij}). \tag{2}$$

Sometimes this loss is minimized together with a regularizer that controls the complexity of the learned representations. We'll discuss algorithms for this problem in Section 5.

Let's first remark on how a low rank model can address the data science challenges posed above:

- To accommodate data tables with heterogeneous entries, use different loss functions for each kind of observation.
- To impute or denoise observations, use the inner product $x_i^T y_j$ to predict the observed value. When the observations come from some restricted domain $\mathcal{A}$, the loss function $\ell : \mathbf{R} \times \mathcal{A} \to \mathbf{R}$ maps the number $x_i^T y_j$ to a prediction $\hat{A}_{ij}$ via $\hat{A}_{ij} = \operatorname{argmin}_a \ell(x_i^T y_j, a)$. Notice that the domain of ℓ ensures $\hat{A}_{ij} \in \mathcal{A}$.
- To determine which features are correlated or which examples are similar, compare their low-dimensional representations.
- To assess the effective dimension (rank) k, use cross-validation: leave out some of the observations as a validation set, fit a model to the remaining observations for each value of k under consideration, and choose the value of k that minimizes the loss on the validation set.
- When items are not missing at random, treating each observation equally can lead to a biased estimate. Instead, estimate the propensity of observing an entry and weight observations by the inverse propensity score to achieve a consistent estimate (SSS$^+$16).
- Low rank models use a flexible optimization formulation that easily accommodates covariates (FM13, PU17, RWJ$^+$18)

- It's straightforward to design an optimization procedure to avoid snooping from time series data. Suppose that each example i is observed at time i. Use a block coordinate descent or block coordinate minimization algorithm. The key to prevent snooping is to order the blocks so that the representation learned for example i never depends on observations from future times $i+1, i+2, \ldots$ For more detail, see the recent review (CBL18).
- The representations x_i of each example $i = 1, \ldots, n$ can be used as features in other learning models. The main advantages are that these representations are lower dimensional, real-valued, and fully observed, in contrast with the original features. It is also possible to learn representations that simultaneously fit the observed features well and perform well for a supervised task; see, for example, (RGC$^+$08).

3 Applications

We first review four different applications of low rank models, drawn from the author's research. These applications are meant to give a flavor of the wide variety of problem domains in which low-rank structure appears and to indicate the kinds of challenges these techniques can address.

Medical informatics. Medical treatments succeed when they correctly identify which patient would benefit from a given treatment. In order to learn personalized treatments from observational data, one necessary first step is to identify patients with similar "phenotypes": those with similar symptoms, similar comorbidities, and (we hope) similar responses to each treatment. Identifying clusters of similar patients from medical records is difficult: observations are heterogeneous and may be very sparse. Nevertheless, low rank models have been used successfully to impute missing data and to identify groups of similar patients (SLW$^+$16).

Automated machine learning. In automated machine learning (AutoML), the goal is to quickly identify an algorithm (together with its hyperparameters) that will perform well on a new dataset. Yang et al. (YAKU18) propose to learn which algorithms will accurately fit a new dataset by using a low rank model. Here, examples are datasets, and each feature is the performance of a particular algorithm. Observations are made by running an algorithm on a dataset. The first (slow) step is to collect observations by running many algorithms on many datasets. Surprisingly, the resulting table of observations has a spectrum that decays rapidly. The second (fast) step determines the best algorithm for a new dataset: Yang et al. (YAKU18) suggest running a small number of (fast, informative) algorithms on the new dataset. These observations can be used to impute the performance of all other algorithms and to choose the algorithm(s) with the best predicted performance. The resulting AutoML method is competitive with the state of the art in automated machine learning.

Understanding categorical variables. High-dimensional categorical variables often stymie data analysis: using a standard one-hot encoding inflates the number of variables and can result in overfitting. Low rank models can be used to to embed these high-dimensional categoricals into a low-dimensional vector space. Fu and Udell (FU) show how to use this approach to reduce the dimension of the feature "zip code"(with 32,989 nominal values) to a ten-dimensional vector. Substituting the zip code feature by these low-rank representations of the zip code allows for better predictions of labor code violations by businesses in each zip code compared with standard approaches.

Causal inference. To correctly identify the causal effect of a treatment on an outcome from observational data, one must control for possible confounders: other covariates that may influence both the treatment and the outcome. However, controlling for more and more (noisy) covariates increases the variance of the model; worse, some covariates may not be observed for all examples. Instead, Kallus et al. (KMU18) suggest controlling for *latent* confounders by fitting a low rank model to the covariates. The low rank representations of the covariates are identified as the latent confounders. Empirically, controlling for these latent confounders improved the accuracy of *every* causal inference method tested (KMU18).

4 Why Low Rank?

Why do low rank models perform so well in such a wide variety of problems? A data table can be well approximated by a low rank model if there are

- a small number of latent features for each row and
- a small number of latent features for each column
- so that entries of the data table are approximately (functions of) the inner product of the row latent features with the column latent features.

Two elements of this story are surprising. Why should row and column latent features be low dimensional? And why should they interact linearly to form the data?

A more general (and perhaps more plausible) model is to choose some high-dimensional row and column latent features $\alpha_i \in \mathcal{A} \subseteq \mathbf{R}^N$ and $\beta_j \in \mathcal{B} \subseteq \mathbf{R}^N$ (where N may be large), and some arbitrary function $g : \mathbf{R}^N \times \mathbf{R}^N \to \mathbf{R}$, and to model entries as $g(\alpha_i, \beta_j)$. We call such a model a latent variable model.

Now we can say why the effectiveness of low rank models should not be a surprise. Under very general conditions on g, $\mathcal{A}$, and $\mathcal{B}$,[1] the matrix with entries $g(\alpha_i, \beta_j)$ is approximately low rank. More concretely, consider the problem of approximating a matrix $A \in \mathbf{R}^{m\times n}$ by a lower-rank matrix X so that the difference between X and A is no greater than ϵ on each entry. How does rank of this ϵ-approximation to A change with m and n? Udell and Townsend (UT18) show that the optimal value of the problem

$$\begin{array}{ll} \text{minimize} & \operatorname{rank}(X) \\ \text{subject to} & \|X - A\|_\infty \le \epsilon \end{array}$$

grows as $\mathcal{O}(\log(m+n)/\epsilon^2)$. That is, the rank of the matrix A grows much less quickly than its dimension. Hence large enough datasets have low relative rank: big data is low rank.

The idea of the proof is simple. For each α, expand g around $\beta = 0$ by its Taylor series

$$\begin{aligned} & g(\alpha,\beta) - g(\alpha,0) \\ = \;& \langle \nabla g(\alpha,0), \beta\rangle + \langle \nabla^2 g(\alpha,0), \beta\beta^\top\rangle + \dots \\ = \;& \begin{bmatrix} \nabla g(\alpha,0) \\ \operatorname{vec}(\nabla^2 g(\alpha,0)) \\ \vdots \end{bmatrix}^\top \begin{bmatrix} \beta \\ \operatorname{vec}(\beta\beta^\top) \\ \vdots \end{bmatrix}, \end{aligned}$$

where we have collected terms depending on α and on β into two vectors. Notice that we have approximated $g(\alpha, \beta)$ by an inner product. Since g is analytic, we can achieve an approximation with error ϵ by truncating the expansion after $O(\log(1/\epsilon))$ terms.

If α and β are themselves low dimensional (for example, univariate), this immediately gives a low rank factorization of A. Otherwise, apply the Johnson-Lindenstrauss lemma (JL84) to reduce the dimension of the vectors. Udell and Townsend (UT18) use a variant of the lemma that bounds the error in the inner product:

Lemma 1 (Variant of the Johnson–Lindenstrauss lemma (UT18)). Consider $x_1, \dots, x_n \in \mathbf{R}^N$. Pick $0 < \epsilon < 1$ and set $r = \lceil 8(\log n)/\epsilon^2 \rceil$. A linear map $Q : \mathbf{R}^N \to \mathbf{R}^r$ exists such that for all $1 \le i, j \le n$,

$$\left| x_i^T x_j - x_i^T Q^T Q x_j \right| \le \epsilon \left(\|x_i\|^2 + \|x_j\|^2 - x_i^T x_j \right). \tag{3}$$

The technical conditions on the function g and the sets $\mathcal{A}$ and $\mathcal{B}$ guarantee that the right-hand side of Eq. (3) is bounded by a constant, finishing the proof.

5 Fitting Low Rank Models

5.1 PCA

When observations are numeric, it's traditional to measure error with the quadratic loss $\ell(u, a) = (u - a)^2$, which makes the optimization problem Eq. (2) particularly easy to solve when every entry is observed. In this case, Eq. (2) is known as *principal components analysis* (PCA):

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^m \sum_{j=1}^n (A_{ij} - Z_{ij})^2 = \|A - Z\|_F^2 \\ \text{subject to} & \operatorname{rank}(Z) \le k. \end{array} \tag{4}$$

Here, we introduce the variable $Z = XY \in \mathbf{R}^{m\times n}$; representations $X \in \mathbf{R}^{m\times k}$ and $Y \in \mathbf{R}^{k\times n}$ can be recovered by using any (rank-revealing) factorization of the matrix Z.

Optimization problems with rank constraints are in general challenging. This problem is the one exception: it can be solved easily by using the singular value decomposition (SVD). Let $\sigma_1, \dots, \sigma_k$ be the first k singular values of A, and let $u_1, \dots, u_k$ and $v_1, \dots, v_k$ be the first k left and right singular vec-

[1] It suffices for the sets $\mathcal{A}$ and $\mathcal{B}$ to be bounded and for g to be analytic with bounded derivatives. Udell and Townsend (UT18) show that the same result holds under more general conditions.

tors, respectively. A solution to Eq. (4) is

$$Z^\star = \sum_{i=1}^{k} \sigma_i u_i v_i^T,$$

which is unique so long as the singular values are. The top few singular values are particularly easy to compute: indeed, in Hotelling's 1933 paper introducing PCA, he computes them by hand using power iteration (Hot33).

5.2 Missing Entries

The solution is not as straightforward when some entries are missing. One simple heuristic is to guess a value for each of the missing entries and then solve the resulting (completely observed) problem with PCA. Often, missing values are replaced by the column mean. This heuristic is easy to understand or to code, but it does not work when a large fraction of the entries are missing.

Two general strategies—convex and nonconvex—are used for fitting low rank models with many missing entries. These methods are often described for quadratic losses, but they work well for any convex differentiable loss.

Convex methods. Convex methods model the product $Z = XY \in \mathbf{R}^{m\times n}$ rather than modeling the low-dimensional representations explicitly, and they do not explicitly constrain the rank of the model. Instead, they introduce a regularizer or constraint on Z—usually involving the nuclear norm $\|Z\|_*$, which is the 1-norm of the singular values of Z—to encourage a low-rank solution. The resulting problem is a large-scale semidefinite program:

$$\text{minimize} \quad \sum_{(i,j)\in\Omega} (A_{ij} - Z_{ij})^2 + \lambda\|Z\|_* \tag{5}$$

with variable $Z \in \mathbf{R}^n$. Algorithms include interior point methods (for small problems), proximal gradient methods, and conditional gradient methods.

While interior point methods scale poorly, first-order methods are more promising: the gradient of the loss term in Eq. (5) is sparse, and so it is easy to manipulate even for very large problems. Proximal gradient methods (MHT10, MGC11, CCS10) compute the SVD of an $m \times n$ matrix at each iteration: this step can be prohibitively slow in high dimensions. The conditional gradient method (CGM) for Eq. (5) avoids the SVD (Jag13); instead, it moves in the direction of the top singular vector of the gradient. As an added benefit, the rank of the iterate is bounded by the number of iterations. The storage requirements of CGM can be further reduced by sketching the decision variable, which gives an optimal memory algorithm for Eq. (5) (YUTC17).

Nonconvex methods. Nonconvex methods search over the matrices X and Y and thereby (implicitly) constrain the rank of the product XY. The resulting problem is nonconvex and may have local minima and other suboptimal stationary points. Algorithms include gradient descent, alternating minimization, alternating proximal gradient methods, and manifold optimization.

Some variants allow the dimension k to vary in order to avoid or escape bad stationary points (JBAS10). This strategy exploits the connection between the convex and nonconvex problem formulation. Let's describe one such method. When progress of the nonconvex method slows, map the nonconvex iterate $(X, Y) \in \mathbf{R}^{m\times k} \times \mathbf{R}^{k\times n}$ to the matrix $Z = X^T Y \in \mathbf{R}^{m\times n}$. Consider Z as an iterate of CGM, and take one step of the CGM method to form Z'. This step increases the rank of the iterate by at most one: $\text{rank}(Z') \le \text{rank}(Z)+1$. Factor Z' as $Z' = X'Y'$ to obtain a new iterate $(X', Y') \in \mathbf{R}^{m\times k+1} \times \mathbf{R}^{k+1\times n}$ to use in the nonconvex method, and continue.

Methods such as manifold optimization (BMAS14) guarantee convergence to a second-order stationary point. Furthermore, all such points are optimal for Eq. (5) if $k = O(\sqrt{n})$ (BVB18). Unfortunately this result is tight: for smaller k, second-order critical points are not generically optimal (WW18).

Statistical guarantees. When observations are generated from a true low rank matrix via a simple statistical model (e.g., with entrywise Gaussian noise), one can prove that both convex and nonconvex optimization methods recover the true matrix as the number of observations increases, for appropriate choices of the parameters. Examples of this approach include (CP09, CR08, GLM16, GAGG13, KU16, KMO10, NW11); see (CLC18) for a recent review of nonconvex recovery results. Convex methods and manifold optimization (for large enough k) provide guaranteed solutions for the convex relaxation Eq. (5). When the data generating distribution is unknown, however, all methods should be regarded as heuristics for minimizing Eq. (2).

6 Loss Functions

Choosing the right loss function can substantively improve the imputation error of the model (ABKKW18, SLW+16). In fact, this approach can even result in smaller squared error! The loss function induces a nonlinear mapping from the parameter $z = x_i^T y_j$ to the imputed value $\hat{A}_{ij}$ via $\hat{A}_{ij} = \operatorname{argmin}_a \ell(z, a)$, so the resulting matrix need not be low rank. Anderson-Bergman et al. (ABKKW18) show examples of real datasets for which a low rank model of rank k, fit by using a data-driven loss function, induces imputations $\hat{A}$ that improve on the square error of the best rank-k model:

$$\sum_{(i,j)\in\Omega} (\hat{A}_{ij} - A_{ij})^2 \leq \inf_{\operatorname{rank}(Z)\leq k} \sum_{(i,j)\in\Omega} (Z_{ij} - A_{ij})^2.$$

What loss function to pick? A common choice in the theoretical literature is to consider a parametric noise distribution, often in the exponential family, and choose as a loss function the negative log likelihood of the noise distribution. This approach has the advantage of offering provable guarantees, but it can be difficult to validate the choice of noise model for real data. Instead, a recent suggestion is to learn the noise distribution from the data (ABKKW18, HL12). Unfortunately, these loss functions are often significantly more challenging to optimize.

We may also pick a loss function by considering our qualitative goals in fitting the model. When the goal is to predict individual entries of a matrix, it's often more natural to measure the mean absolute error of a model, rather than the mean square error, and so we'd measure error in the (entrywise) 1-norm. Or we may want a model that fits *every* entry of the matrix well; in this case, we'd want to minimize the maximum absolute error.

If the data takes values from a discrete set (e.g., $\{0, 1\}$ or $\{1, 2, 3, 4, 5\}$), it's natural to round the entries from our low rank model so that imputed values have the same domain. In this case, we may want to know how often the (rounded) model gets the answer *right* or *wrong* and so measure the *misclassification error*: the number of entries $x_i^T y_j$ for $(i, j) \in \Omega$ that don't round to A_{ij}.

When the loss functions are not differentiable or are not convex, the corresponding methods (using, for example, subgradients in place of gradients) lack guarantees and tend to work much more poorly. When the loss functions are not continuous (like the misclassification error) or are not separable entrywise (like the maximum absolute error), the problem is even more severe. For example, finding a rank-1 matrix that minimizes the maximum absolute error to a set of observations is NP hard (GS17).

Various heuristics to solve these problems have been proposed (GS17, UT18). In practice, a common strategy is to replace these error metrics by functions that are easier to optimize: in place of the infinity norm, quadratic loss (UT18); in place of the 1-norm, elementwise Huber loss (UHZB16); in place of misclassification loss, hinge loss or ordinal hinge loss (SRJ04, UHZB16). The best model is chosen by fitting the surrogate loss function for a range of different parameters and choosing the one that produces the lowest error with respect to the original loss function. Interestingly, under strong statistical assumptions, even simple algorithms like gradient descent applied to the nonconvex formulation—still with quadratic objective!—can be proven to control entrywise error (MWCC17).

7 Conclusion

Low rank models can be used to answer a wide variety of questions in data science. They can adapt to perversities in the data including missing and heterogeneous entries, and they perform well across a diverse array of problems. Numerous optimization methods are available to fit low rank models, some with provable statistical recovery guarantees and others that guarantee convergence to the solution of a particular relaxation, Eq. (5). These methods yield useful results in practice. Minimizing the original rank-constrained objective, Eq. (2), for general data distributions, is more challenging. Moreover, substantial room exists to improve optimization heuristics for fitting low rank models involving nonsmooth or discontinuous loss functions.

References

[ABKKW18] Clifford Anderson-Bergman, Tamara G Kolda, and Kina Kincher-Winoto. XPCA: Extending PCA for a combination of discrete and continuous variables. *arXiv preprint arXiv:1808.07510*, 2018.

[BMAS14] Nicolas Boumal, Bamdev Mishra, P-A Absil, and Rodolphe Sepulchre. Manopt, a matlab toolbox for optimization on manifolds. *The Journal of Machine Learning Research*, 15(1):1455–1459, 2014.

[BVB18] Nicolas Boumal, Vladislav Voroninski, and Afonso S Bandeira. Deterministic guarantees for Burer-Monteiro factorizations of smooth semidefinite programs. *arXiv preprint arXiv:1804.02008*, 2018.

[CBL18] Y Chi, L Balzano, and YM Lu. A modern perspective on streaming pca and subspace tracking: The missing data case. *Proceedings of the IEEE*, 2018.

[CCS10] Jian-Feng Cai, Emmanuel J Candès, and Zuowei Shen. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 20(4):1956–1982, 2010.

[CLC18] Yuejie Chi, Yue M Lu, and Yuxin Chen. Nonconvex optimization meets low-rank matrix factorization: An overview. *arXiv preprint arXiv:1809.09573*, 2018.

[CP09] E. Candès and Y. Plan. Matrix completion with noise. *CoRR*, abs/0903.3131, 2009.

[CR08] E. Candès and B. Recht. Exact matrix completion via convex optimization. *CoRR*, abs/0805.4471, 2008.

[FM13] William Fithian and Rahul Mazumder. Flexible low-rank statistical modeling with side information. *arXiv preprint arXiv:1308.4211*, 2013.

[FU] Anqi Fu and Madeleine Udell. Census labor law violations. `https://github.com/h2oai/h2o-3/blob/master/h2o-r/demos/rdemo.census.labor.violations.large.R`. Accessed: 2019-01-13.

[GAGG13] S. Gunasekar, A. Acharya, N. Gaur, and J. Ghosh. Noisy matrix completion using alternating minimization. In *Machine Learning and Knowledge Discovery in Databases*, pages 194–209. Springer, 2013.

[GLM16] Rong Ge, Jason D Lee, and Tengyu Ma. Matrix completion has no spurious local minimum. In *Advances in Neural Information Processing Systems*, pages 2973–2981, 2016.

[GS17] Nicolas Gillis and Yaroslav Shitov. Low-rank matrix approximation in the infinity norm. *arXiv preprint arXiv:1706.00078*, 2017.

[HL12] Fang Han and Han Liu. Semiparametric principal component analysis. In *Advances in Neural Information Processing Systems*, pages 171–179, 2012.

[Hot33] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24(6):417, 1933.

[Jag13] Martin Jaggi. Revisiting Frank-Wolfe: Projection-free sparse convex optimization. In *International Conference on Machine Learning*, pages 427–435, 2013.

[JBAS10] Michel Journée, Francis Bach, P-A Absil, and Rodolphe Sepulchre. Low-rank optimization on the cone of positive semidefinite matrices. *SIAM Journal on Optimization*, 20(5):2327–2351, 2010.

[JL84] William B Johnson and Joram Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. *Contemporary mathematics*, 26(189-206):1, 1984.

[KMO10] R. H. Keshavan, A. Montanari, and S. Oh. Matrix completion from a few entries. *IEEE Transactions on Information Theory*, 56(6):2980–2998, 2010.

[KMU18] Nathan Kallus, Xiaojie Mao, and Madeleine Udell. Causal inference with noisy and missing covariates via matrix factorization. In *Advances in Neural Information Processing Systems*, 2018.

[KU16] N. Kallus and M. Udell. Revealed preference at scale: Learning personalized preferences from assortment choices. In *The 2016 ACM Conference on Economics and Computation*, New York, NY, USA, 2016. ACM.

[MGC11] Shiqian Ma, Donald Goldfarb, and Lifeng Chen. Fixed point and bregman iterative methods for matrix rank minimization. *Mathematical Programming*, 128(1-2):321–353, 2011.

[MHT10] Rahul Mazumder, Trevor Hastie, and Robert Tibshirani. Spectral regularization algorithms for learning large incomplete matrices. *Journal of machine learning research*, 11(Aug):2287–2322, 2010.

[MWCC17] Cong Ma, Kaizheng Wang, Yuejie Chi, and Yuxin Chen. Implicit regularization in nonconvex statistical estimation: Gradient descent converges linearly for phase retrieval, matrix completion and blind deconvolution. *arXiv preprint arXiv:1711.10467*, 2017.

[NW11] S. Negahban and M. Wainwright. Estimation of (near) low-rank matrices with noise and high-dimensional scaling. *The Annals of Statistics*, 39(2):1069–1097, 2011.

[PU17] Mihir Paradkar and Madeleine Udell. Graph-regularized generalized low rank models. In *CVPR Workshop on Tensor Methods in Computer Vision*, 2017.

[RGC+08] Irina Rish, Genady Grabarnik, Guillermo Cecchi, Francisco Pereira, and Geoffrey J Gordon. Closed-form supervised dimensionality reduction with generalized linear models. In *Proceedings of the 25th international conference on Machine learning*, pages 832–839. ACM, 2008.

[RWJ+18] Geneviève Robin, Hoi-To Wai, Julie Josse, Olga Klopp, and Éric Moulines. Low-rank interaction with sparse additive effects model for large data frames. In *Advances in Neural Information Processing Systems*, pages 5501–5511, 2018.

[SLW+16] A. Schuler, V. Liu, J. Wan, A. Callahan, M. Udell, D. Stark, and N. Shah. Discovering patient phenotypes using generalized low rank models. *Pacific Symposium on Biocomputing (PSB)*, 2016.

[SRJ04] N. Srebro, J. Rennie, and T. Jaakkola. Maximum-margin matrix factorization. In *Advances in Neural Information Processing Systems*, volume 17, pages 1329–1336, 2004.

[SSS+16] Tobias Schnabel, Adith Swaminathan, Ashudeep Singh, Navin Chandak, and Thorsten Joachims. Recommendations as treatments: Debiasing learning and evaluation. *arXiv preprint arXiv:1602.05352*, 2016.

[UHZB16] M. Udell, C. Horn, R. Zadeh, and S. Boyd. Generalized low rank models. *Foundations and Trends in Machine Learning*, 9(1), 2016.

[UT18] Madeleine Udell and Alex Townsend. Why are big data matrices approximately low rank? *SIAM Mathematics of Data Science (SIMODS), to appear*, 2018.

[WW18] Irène Waldspurger and Alden Waters. Rank optimality for the Burer-Monteiro factorization. *arXiv preprint arXiv:1812.03046*, 2018.

[YAKU18] Chengrun Yang, Yuji Akimoto, Dae Won Kim, and Madeleine Udell. OBOE: Collaborative filtering for automl initialization. In *NIPS Workshop on Automated Machine Learning*, 2018.

[YUTC17] A. Yurtsever, M. Udell, J. A. Tropp, and V. Cevher. Sketchy decisions: Convex low-rank matrix optimization with optimal storage. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2017.