Learning Semidefinite Regularizers – An INFORMS Newsletter Feature Article

Yong Sheng Soh

August 10, 2020

1 Learning Regularizers from Data

Regularizers are frequently used to solve ill-posed linear inverse problems. These take the form of penalty functions that are augmented to the objective of an optimization-based approach, and serve the purpose of inducing certain desired structure in the solution.

Consider the following instance where we obtain a vector of measurements $\mathbf{y}$ generated via the following process:

$$\mathbf{y} = F\mathbf{x}^\star + \boldsymbol{\epsilon}.$$

Here, F is known as the *forward map* or the *measurement matrix.* The vector $\mathbf{x}^\star$ is unknown to us, and our goal is to estimate what $\mathbf{x}^\star$ is.

The problem is *ill-posed* if the dimensions of $\mathbf{x}^\star$ exceeds that of $\mathbf{y}$; that is to say, the number of measurements, as it is smaller than the number of unknowns, is insufficient to specify a *unique* solution. However, in many practical applications, some structural side information such as sparsity or smoothness about the solution $\mathbf{x}^\star$ may be known, and incorporating such information may be useful for recovering $\mathbf{x}^\star$.

Regularization techniques are one such class of methods for incorporating such information as a means to address difficulties due to ill-posedness. In our above example, we would estimate $\mathbf{x}^\star$ as the solution of the following optimization instance:

$$\hat{\mathbf{x}} = \underset{\mathbf{x}}{\operatorname{argmin}} \ \frac{1}{2}\|\mathbf{y} - F\mathbf{x}\|_2^2 + \lambda f(\mathbf{x}).$$

Here, we choose f to be an appropriate convex penalty function that induces certain desired in the solution $\hat{\mathbf{x}}$. Choosing f to be convex is particularly useful because the resulting estimator $\hat{\mathbf{x}}$ is the solution of a convex optimization instance, and hence is amenable to computation and analysis.

As an illustration, if $\mathbf{x}^\star$ is known to be a sparse vector, then a prominent choice of regularization function is the L1-norm. If $\mathbf{x}^\star$ is known to have structure as a low-rank matrix, then a prominent choice of regularization function is the matrix nuclear norm (also known as the Schatten-1-norm). If $\mathbf{x}^\star$ is known to have smooth structure, then a popular choice of regularization function is the Total-Variation norm.

Notice in the previous examples that we needed some knowledge concerning the structure of $\mathbf{x}^\star$ before we were able to specify an appropriate choice of regularizer. This observation is true in general – the appropriate choice of regularizer often requires some form of domain specific knowledge. The main question we study in this article (and our paper [12]) is:

> How do we specify an appropriate regularization function f in the *absence* of domain specific knowledge?

This article (and our paper [12]) proposes the following approach:

We *learn* an appropriate choice of regularization function *directly* from data.

As we shall see later on, the regularizers obtained using our framework are specified as convex functions that can be computed efficiently via semidefinite programming, and therefore can be employed in tractable convex optimization approaches for solving inverse problems.

2 Atomic Norms and Facial Geometry

We begin with a more basic question:

What are the properties of a regularizer that make them effective at their task?

The short answer lies in the concept of "sparsity". Let $\mathcal{A} \subset \mathbb{R}^d$ be a (possibly infinite) collection of vectors. The set $\mathcal{A}$ represents elementary building blocks from which we describe signals. We call such a set an *atomic set.* We say that $\mathbf{y} \in \mathbb{R}^d$ has a sparse representation using $\mathcal{A}$ if it is expressible as:

$$\mathbf{y} = \sum_{i=1}^{k} c_i \mathbf{a}_i, \quad \mathbf{a}_i \in \mathcal{A}, c_i \geq 0, \tag{1}$$

for a relatively small number $k \ll d$. For instance, if $\mathcal{A}$ is the collection of signed standard basis vectors, then the types of objects that can be described simply using $\mathcal{A}$ are sparse vectors. Similarly, if $\mathcal{A}$ is the collection of unit-norm rank-one matrices, then the type of objects that can be described simply using $\mathcal{A}$ are low-rank matrices.

The pertinent point we wish to make concerning descriptions based on an atomic set $\mathcal{A}$ is the following: the *atomic norm* – defined as the gauge function of the convex hull of $\mathcal{A}$ – is an effective regularizer for promoting sparsity with respect to $\mathcal{A}$:

$$\|\mathbf{x}\|_{\mathcal{A}} = \inf \{t \, : \, \mathbf{x} \in t \cdot \mathrm{conv}(\mathcal{A})\}. \tag{2}$$

Continuing with our previous examples, if the atomic set $\mathcal{A}$ is the is the collection of signed standard basis vectors, then the associated atomic norm is the L1-norm, and if the atomic set $\mathcal{A}$ is the collection of unit-norm rank-one matrices, then the associated atomic norm is the nuclear norm (for a longer list of examples, see [3]).

Atomic norm regularization is effective because its level set $\mathrm{conv}(\mathcal{A})$ possesses the "correct" geometric structure for promoting sparsity. The basic intuition behind atomic norm regularization is that it tends to favor solutions that are sparsely represented with $\mathcal{A}$ as these have smaller atomic norm evaluations. In other words, atomic norm regularization is useful in practice because it tends to give solutions with the precisely the structure that we seek!

To summarize what we have just noted: if an atomic set for data is known, then the associated atomic norm is an effective choice of regularizer. Stated differently:

To learn an effective *regularizer* for data, it suffices to learn an *atomic* set.

3 Dictionary Learning and Linear Programming-based Regularizers

We proceed to learn a regularizer for data by considering the simplest instantiation of learning an atomic set. Let $\{\mathbf{y}^{(j)}\}_{j=1}^{n}$ be a collection of data-points, and suppose that we wish to learn a *finite* atomic set of the form $\mathcal{A} = \{\pm \mathbf{l}_i\}_{i=1}^q$. Based on the definition we introduced in (1), we require the collection $\{\mathbf{l}_i\}_{i=1}^q$ to satisfy the following property for some $s \ll d$ in order to be an atomic set:

$$\mathbf{y}^{(j)} \approx \sum^{s} x_i^{(j)} \mathbf{l}_i^{(j)}, \qquad \mathbf{l}_i \in \mathcal{A}, x_i^{(j)} \neq 0.$$

By concatenating the atoms into a single matrix $L = \left[\begin{array}{c|c|c} \mathbf{l}_1 & \ldots & \mathbf{l}_q \end{array}\right]$, we may express our learning task as follows:

Given data $\{\mathbf{y}^{(j)}\}_{j=1}^n$, find $L \in \mathbb{R}^q \to \mathbb{R}^d$ and $\{\mathbf{x}^{(j)}\}_{j=1}^n \subset \mathbb{R}^q$ such that

$$\mathbf{y}^{(j)} \approx L\mathbf{x}^{(j)}, \qquad \text{where } \mathbf{x}^{(j)} \text{ is } s\text{-sparse.} \tag{3}$$

Our problem, as some might recognize from the latter form (3), is precisely *dictionary learning.* This has been widely studied in the literature as a procedure for automatically learning sparse representations for data. Without going into the history and development of dictionary learning, the main point we shall make is the following: By applying the definition in (2), the atomic norm obtained from dictionary learning satisfies the following properties:

1. Its level set is the projection of the L1-ball under the linear map L.
2. It is expressible via the following Linear Program (LP):

$$\|\mathbf{x}\|_{\mathcal{A}} \;=\; \inf\{\, \|\mathbf{z}\|_1 \;:\; \mathbf{x} = L\mathbf{z}\,\}.$$

In summary:

Learning a regularizer corresponding to a *finite* atomic set is precisely *dictionary learning*, and it leads to a regularization function that is expressible via *Linear Programming.*

4 Learning Semidefinite Programming-based Regularizers

Our previous point leads to the following natural question: What about learning regularizers that correspond to an *infinite* atomic set? There are a number of reasons one might wish to do so – for one, the collection of such regularizers are far richer than the collection of polyhedral regularizers learned via dictionary learning. Our numerical experiments in the next section will provide additional impetus.

While we were able to learn fairly generic regularizers corresponding to a finite atomic set, there are certain restrictions that arise when we consider learning infinite atomic sets. First, the resulting regularizer we learn needs to be tractable to express – after all, the main purpose for learning regularizers is to deploy it for downstream tasks. Second, we need a computational strategy that permits us to learn such regularizers.

We propose learning a class of regularizers that extends dictionary learning using the ideas of Semidefinite Programming (SDP). SDPs are a powerful generalization of LPs in which we replace non-negativity constraints over the space of vectors with positive semidefinite constraints over the space of symmetric matrices. Specifically, we learn regularizers with the following properties:

1. Its level set is the projection of the nuclear norm ball under an appropriately learned linear map $\mathcal{L} : \mathbb{R}^{q\times q} \to \mathbb{R}^d$.

2. It is expressible via the following SDP ($\|\cdot\|_*$ is the matrix nuclear norm):
$$\|\mathbf{x}\|_{\mathcal{A}} \;=\; \inf\{\,\|Z\|_* \,:\, \mathbf{x} = \mathcal{L}(Z)\,\}.$$

3. The procedure for learning such regularizers corresponds to fitting data as linear projections of low-rank matrices. More specifically:
$$\text{Given data } \{\mathbf{y}^{(j)}\}_{j=1}^{n}, \text{ find } \mathcal{L} \in \mathbb{R}^{q\times q} \to \mathbb{R}^{d} \text{ and } \{X^{(j)}\}_{j=1}^{n} \subset \mathbb{R}^{q\times q} \text{ such that}$$
$$\mathbf{y}^{(j)} \approx \mathcal{L}(X^{(j)}), \qquad \text{where } X^{(j)} \text{ is rank } r. \tag{4}$$

5 Numerical Demonstration

We switch gears to describe a series of numerical experiments that demonstrate the utility of learning infinite set of atoms.

5.1 Representing Natural Image Patches

Our first experiment concerns data compression in which we compare the *representive power* of descriptions as projections of low-rank matrices (learned using our method) with descriptions as projections of sparse vectors (learned using dictionary learning).

The data set $\{\mathbf{y}^{(j)}\}_{j=1}^{6480} \in \mathbb{R}^{64}$ comprises image patches of dimensions 8×8. We apply our framework which we describe in the next section, and we apply the analog of our procedure for dictionary learning. In the left-subplot of Figure 1 we compare the mean squared errors of both types of representations across different representation complexities. Roughly speaking, the representation complexities correspond to the number of parameters required to express each representation (and as such, a more complex representation leads to smaller approximation error). Interestingly, we observe that our approach provides an improvement over dictionary learning for small levels of representation complexity and is comparable at larger levels.

5.2 Denoising Natural Image Patches

Our second experiment compares the performance of using polyhedral and semidefinite regularizers in denoising natural image patches.

We designate the same 6480 data points from the previous experiment as a training set, and we test the performance of the learned regularizers on an additional set of 720 data-points by applying the following proximal operation:
$$\hat{\mathbf{y}}_{\text{denoise}} = \underset{\mathbf{y}\in\mathbb{R}^{64}}{\arg\min} \quad \tfrac{1}{2}\|\mathbf{y}_{\text{obs}} - \mathbf{y}\|_2^2 + \lambda\|\mathbf{y}\|. \tag{5}$$
Here, $\|\cdot\|$ is a regularizer learned on the training set and $\lambda > 0$ is a regularization parameter. In the right sub-plot of Figure 1 we compare the average normalized mean-squared error of the denoised image using a collection of regularizers obtained from our framework and from dictionary learning. The horizontal axis corresponds to the computational complexity of solving the proximal operation (5) using an interior point method – (very) roughly speaking, a polyhedral regularizer with lifting dimension q (the dimension of the sparse vectors) has comparable complexity with a semidefinite regularizer with lifting dimensions $\sqrt{q} \times \sqrt{q}$ (the dimension of the low-rank matrices).

For both types of regularizers the denoising performance improves initially before degrading due to overfitting. More interestingly (and more significantly), given a *fixed* computational budget, our results suggest that semidefinite regularizers provide *better* performance than polyhedral regularizers in denoising image patches in our data set.

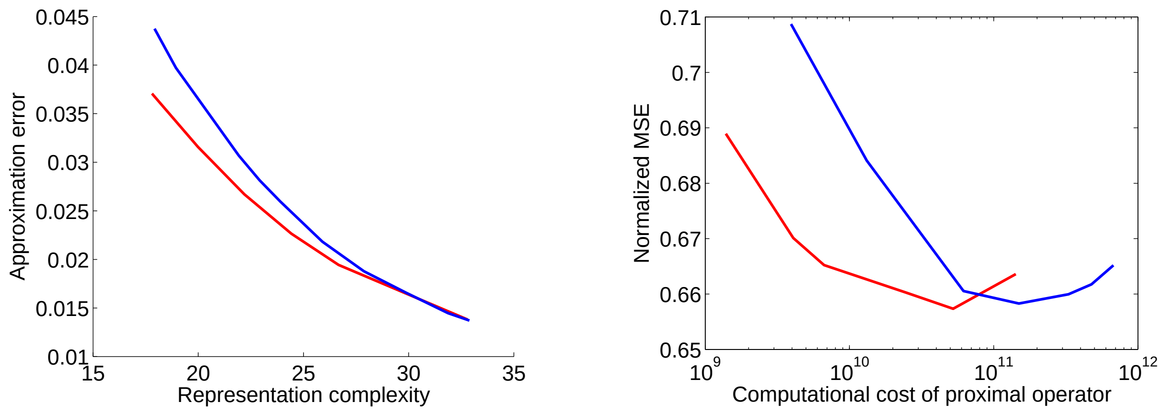

Figure 1: Comparison between dictionary learning (blue) and our approach (red) in representing natural image patches (left); comparison between polyhedral (blue) and semidefinite (right) regularizers in denoising natural image patches (right).

6 An Alternating Minimization Based Approach

Our procedure for learning semidefinite regularizers is based on alternating updates in which we keep one factor fixed (either the linear projection map $\mathcal{L}$ or the low-rank matrices $\{X^{(j)}\}$) while updating the other.

6.1 Dictionary Learning

We begin by discussing prior algorithms for dictionary learning (3).

Updating the sparse vectors $\{\mathbf{x}^{(j)}\}$. In the first step, we fix the linear map $L \in \mathbb{R}^q \to \mathbb{R}^d$ and we seek sparse vectors $\mathbf{x}^{(j)}$ such that $\mathbf{y}^{(j)} \approx L\mathbf{x}^{(j)}$. Although this problem is NP-hard in general, there are a number of tractable heuristics that are useful in practice and succeed in certain instances. The most prominent method is based on applying the L1-norm as a penalty function to induce sparsity:

$$\hat{\mathbf{x}} \in \underset{\mathbf{x}\in\mathbb{R}^q}{\operatorname{argmin}} \ \frac{1}{2}\|\mathbf{y} - L\mathbf{x}\|_2^2 + \lambda\|\mathbf{x}\|_1.$$

An alternative is based on an *iterative hard thresholding* procedure that attempts to find vector $\mathbf{x}$ of fixed sparsity s such that $\mathbf{y} \approx L\mathbf{x}$ [1,6]. This procedure proceeds by iteratively between performing an unconstrained update in the vector $\mathbf{x}$ followed by a thresholding step in which all but the top s entries in magnitude are set to zero.

Updating the linear map L. In the second step, we fix the vectors $\{\mathbf{x}^{(j)}\}_{j=1}^n$ and the objective is to update the linear map L. This step can be performed, for instance, via a least squares update, which is often referred to as the *Method of Optimal Descent* in the literature.

Normalization. The dictionary learning problem as it is stated contains an element of ambiguity: Let $M \in \mathbb{R}^{q\times q}$ be any matrix obtained as the product of a signed permutation and a diagonal matrix with positive entries. Given a factorization of the data as $\mathbf{y}^{(i)} = L\mathbf{x}^{(i)}$ where $\mathbf{x}^{(i)}$ is a sparse vector, the following specifies an equally valid factorization:

$$\mathbf{y}^{(i)} = (L \circ M)(M^{-1}\mathbf{x}^{(i)}).$$

The most typical approach to addressing such ambiguities is to scale each column of the linear map L to be of unit Euclidean-norm. Doing so specifies L up to a signed permutation of its columns.

6.2 From Dictionary Learning to Learning Semidefinite Regularizers

We describe an algorithm for (4) by generalizing each step in the previous section.

Updating the low-rank matrices $\{X^{(j)}\}_{j=1}^n$. In the first step, we fix the linear map $\mathcal{L} : \mathbb{R}^{q\times q} \to \mathbb{R}^d$, and seek low-rank matrices $\{X^{(j)}\}_{j=1}^n$ such that $\mathbf{y}^{(j)} \approx \mathcal{L}(X^{(j)})$. This problem is also NP-hard in general, but a number of tractable heuristics have been developed. One prominent method is based on applying the matrix nuclear norm as a penalty function to induce low-rank structure in solutions [2,4,11]:

$$\hat{X} = \underset{X\in\mathbb{R}^{q\times q}}{\arg\min} \quad \frac{1}{2}\|\mathbf{y} - \mathcal{L}(X)\|_2^2 + \lambda\|X\|_\star. \tag{6}$$

An alternative is based on an *iterative singular value projection* procedure that attempts to find a matrix X of fixed rank r such that $\mathbf{y} \approx \mathcal{L}(X)$. This procedure can be viewed as a natural generalization of the iterative soft thresholding procedure. It alternates between performing a least squares update and a projection step onto the subset of matrices with rank at most r [7,9].

Updating the linear map $\mathcal{L}$. In the second step, we fix the low-rank matrices $\{X^{(j)}\}_{j=1}^n$ and update the linear map $\mathcal{L}$. This step can be performed via a least squares update in precisely the same fashion as we did for dictionary learning.

Normalization. The normalization step is the trickiest part to generalize, and so we proceed with extra care.

First, let us formally state what we require of our normalization scheme: (i) given a generic linear map, the normalization scheme specifies a *unique* choice of regularizer, and (ii) we have an algorithm for normalizing any generic linear map.

Our proposal is as follows:

Definition. Let $\mathcal{L} : \mathbb{R}^{q\times q} \to \mathbb{R}^d$ be a linear map, and let $\mathcal{L}_i \in \mathbb{R}^{q\times q},\ i = 1,\ldots,d$ be the component linear functionals of $\mathcal{L}$. Then $\mathcal{L}$ is said to be *normalized* if $\sum_{i=1}^d \mathcal{L}_i\mathcal{L}_i{}' = qI$ and $\sum_{i=1}^d \mathcal{L}_i{}'\mathcal{L}_i = qI$.

We can perform a simple sanity check to see that the proposed normalization scheme reduces to scaling columns when we specialize our framework to dictionary learning: Restrict the matrices $\{X^{(j)}\}_{j=1}^n$ as well as the linear functionals of $\mathcal{L}_i$ to be diagonal, and consider the linear map L whose rows are diagonal entries of $\mathcal{L}_i$.

How do we go about showing that our proposed normalization scheme satisfies the two criteria laid out above? The basic idea lies in characterizing the ambiguities that exist in our problem. Consider an operator $\mathsf{M} : \mathbb{R}^{q\times q} \to \mathbb{R}^{q\times q}$ of the form $\mathsf{M}(X) = W_1XW_2$ or $\mathsf{M}(X) = W_1X'W_2$, where $W_1, W_2 \in \mathbb{R}^{q\times q}$ is any pair of invertible matrices. Notice that such maps are invertible and they preserve the rank of any input matrix. Given a factorization of the data $\mathbf{y}^{(i)} = \mathcal{L}(X^{(i)})$ where $X^{(i)}$ are low-rank matrices, we can insert such a map M to obtain alternative factorizations of the form $\mathbf{y}^{(i)} = \mathcal{L}\circ\mathsf{M}(\mathsf{M}^{-1}(X^{(i)}))$. By applying a result due to Marcus and Moyls [10], one can in fact show that maps of these sort are the *only* sources of ambiguities.

In fact, it turns out that, given a generic linear map $\mathcal{L}$, computing an invertible operator M of the form $\mathsf{M}(X) = W_1XW_2$ or $\mathsf{M}(X) = W_1X'W_2$ so that $\mathcal{L}\circ\mathsf{M}$ is normalized closely relates to a very fascinating and peculiar problem known as the *Operator Sinkhorn Iteration* (OSI). The OSI was initially developed and studied by Gurvits in the fields of quantum physics and theoretical computer science [8], and has deep (and highly surprising) implications in algorithmic questions concerning non-commutative polynomials [5].

It is perhaps simpler to describe what the OSI is in terms of its classical analog: Given a square matrix with positive entries, scale the row and columns so that the resulting matrix has row and columns sum equal to one. Now, we can trivially scale the rows so that the row sums are equal to one, and likewise for columns – the surprise is that, if we proceed to alternate between scaling rows and columns, we would actually converge to the solution! This procedure is known as Matrix Sinkhorn Scaling, and OSI is precisely its Operator analog.

7 Local Linear Convergence

Our main technical result is a local linear convergence guarantee. We state a more intuitive version of our result based on a random ensemble first, and remark on the deterministic version of our result later:

Theorem 7.1. *Suppose that the data is generated as* $\mathbf{y}^{(j)} = \mathcal{L}^\star(X^{(j)\star})$, $j = 1, \ldots, n$, *where*

1. $\mathcal{L}^\star : \mathbb{R}^{q\times q} \to \mathbb{R}^d$ *is a linear map drawn from the normal distribution and subsequently normalized,*

2. $X^{(j)\star} = U^{(j)} V^{(j)\prime}$, *where* $U^{(j)}$ *and* $V^{(j)}$ *are partial orthogonal matrices of size* $q \times r$ *drawn from the uniform distribution.*

Suppose that $d \gtrsim rq$. *Then our algorithm is* locally linearly convergent*; that is, when initialized a linear map that is suitably close to* $\mathcal{L}^\star$ *(up to an equivalence class), converges to a linear map that specifies the same regularizer as* $\mathcal{L}^\star$*.*

The deterministic result relaxes the above criteria as follows: (i) In reality, we only require $\mathcal{L}^\star$ to satisfy a *restricted isometry property* (RIP) (in addition to being normalized) in that it is approximately isometric when restricted to the set of low-rank matrices. The RIP is a standard ingredient in the compressed sensing literature that guarantees the success of the singular value thresholding procedure as well as convex relaxations based on the nuclear norm. (ii) We need the ensemble $\{X^{(j)\star}\}_{j=1}^n$ to be sufficiently "diverse" in that the column and row spaces of $X^{(j)\star}$ are sufficiently isotropic.

8 Concluding Remarks

Learning data representations based on sparsity is a concept that is familiar to many – one of the surprises of this article is that we are actually able to learn analogous data representations based on *projections of low-rank matrices.*

This article raises the broader question of learning an infinite atomic set. In fact, one needs to view the contribution of this article as one of laying the computational framework for learning a specific type of infinite atomic set – there still remains numerous conceptual questions that lie ahead of us. For instance, when is it desirable to learn infinite atomic sets? Are there certain characteristics of data that should inform the appropriate type of representation, either as projections of sparse vectors or as projections as low-rank matrices? Could we, for instance, look to symmetries? Many instances of data possess some form of group symmetry or invariances – could it be that data with certain types of group symmetries are more naturally represented, say, via SDPs?

Finally, we wish to express our gratitude to the INFORMS Optimization Society and the Prize Committee for recognizing our work and for the opportunity to write this article!

References

[1] T. Blumensath and M. E. Davies. Iterative Hard Thresholding for Compressed Sensing. *Applied and Computational Harmonic Analysis*, 27:265–274, 2009.

[2] E. J. Candès and B. Recht. Exact Matrix Completion via Convex Optimization. *Foundations of Computational Mathematics*, 9(6):717–772, 2009.

[3] V. Chandrasekaran, B. Recht, P. A. Parrilo, and A. S. Willsky. The Convex Geometry of Linear Inverse Problems. *Foundations of Computational Mathematics*, 12(6):805–849, 2012.

[4] M. Fazel. *Matrix Rank Minimization with Applications.* PhD thesis, Department of Electrical Engineering, Stanford University, 2002.

[5] A. Garg, L. Gurvits, R. Oliveira, and A. Wigderson. A Deterministic Polynomial Time Algorithm for Non-Commutative Rational Identity Testing with Applications. In *IEEE 57th Annual Symposium on Foundations of Computer Science*, 2016.

[6] R. Garg and R. Khandekar. Gradient Descent with Sparsification: An Iterative Algorithm for Sparse Recovery with Restricted Isometry Property. *International Conference on Machine Learning*, pages 337–344, 2009.

[7] D. Goldfarb and S. Ma. Convergence of Fixed-Point Continuation Algorithms for Matrix Rank Minimization. *Foundations of Computational Mathematics*, 11:183–210, 2011.

[8] L. Gurvits. Classical Complexity and Quantum Entanglement. *Journal of Computer and Systems Sciences*, 69(3):448–484, 2004.

[9] P. Jain, R. Meka, and I. S. Dhillon. Guaranteed Rank Minimization via Singular Value Projection. In *Advances in Neural Information Processing Systems*, 2009.

[10] M. Marcus and B. N. Moyls. Transformations on Tensor Product Spaces. *Pacific Journal of Mathematics*, 9(4):1215–1221, 1959.

[11] B. Recht, M. Fazel, and P. A. Parrilo. Guaranteed Minimum-Rank Solutions of Linear Matrix Equations via Nuclear Norm Minimization. *SIAM Review*, 52(3):471–501, 2010.

[12] Y. S. Soh and V. Chandrasekaran. Learning semidefinite regularizers. *Foundations of Computational Mathematics*, 19:375–434, 2019.