Rank Minimization via Online Learning

Raghu Meka RAGHU@CS.UTEXAS.EDU
Prateek Jain PJAIN@CS.UTEXAS.EDU
Constantine Caramanis CMCARAM@ECE.UTEXAS.EDU
Inderjit S. Dhillon INDERJIT@CS.UTEXAS.EDU
University of Texas at Austin, Austin, TX 78712

Abstract

Minimum rank problems arise frequently in machine learning applications and are notoriously difficult to solve due to the non-convex nature of the rank objective. In this paper, we present the first online learning approach for the problem of rank minimization of matrices over polyhedral sets. In particular, we present two online learning algorithms for rank minimization - our first algorithm is a multiplicative update method based on a generalized experts framework, while our second algorithm is a novel application of the online convex programming framework (Zinkevich, 2003). In the latter, we flip the role of the decision maker by making the decision maker search over the constraint space instead of feasible points, as is usually the case in online convex programming. A salient feature of our online learning approach is that it allows us to give provable approximation guarantees for the rank minimization problem over polyhedral sets. We demonstrate the effectiveness of our methods on synthetic examples, and on the real-life application of low-rank kernel learning.

1. Introduction

Minimizing the rank of matrices restricted to a convex set is an important problem in the field of optimization with numerous applications in machine learning. For instance, many important problems like low-rank kernel learning, feature efficient linear classification, semi-definite embedding (SDE), non-negative matrix approximation (NNMA), etc., can be viewed as rank minimization problems over a polyhedron with additional convex constraints such as a Frobenius norm constraint and/or a semi-definiteness constraint. Even though there has been extensive work on the specific problems mentioned above, the general problem of rank minimization over polyhedral sets is not well understood. In this paper we address the problem of rank minimization when there are a large number of trace constraints along with a few convex constraints that are relatively "easy" in a precise sense defined below.

We now formulate the rank minimization problem we study. Let $A_1, \ldots, A_m \in \mathbb{R}^{n \times n}$, $b_1, \ldots, b_m \in \mathbb{R}$ and let $\mathcal{C} \subseteq \mathbb{R}^{n \times n}$ be a convex set of matrices. Then, consider the following optimization problem which we refer to as RMP (for Rank Minimization over Polyhedron):

$$\begin{aligned} \min \quad & \text{rank}(X) \\ \text{s.t} \quad & \text{Tr}(A_i X) \geq b_i, \quad 1 \leq i \leq m \\ & X \in \mathcal{C}. \end{aligned} \tag{RMP}$$

The set $\mathcal{C}$ will represent the "easy" constraints in the sense that for such a set $\mathcal{C}$, we assume that RMP with a single trace constraint can be solved efficiently. This holds for many typical convex sets $\mathcal{C}$, e.g., the unit ball under any L_p or Frobenius norm, the semi-definite cone, and the intersection of the unit ball with the p.s.d. cone. Furthermore, low-rank kernel learning, SDE and NNMA can all be seen as instantiations of the above general formulation.

The general RMP problem as stated above is non-convex, NP-hard and, as we prove, cannot be approximated well unless $P = NP$. Due to the computational hardness of the problem, much of the previous work has concentrated on providing heuristics, with no guarantees on the quality of the solution. We remark that the recent trace-norm based approach of (Recht et al., 2007) does guarantee an optimal solution for a simplified instance of RMP where only well-conditioned linear equality constraints are allowed. However, it is not clear how to extend their guarantees to the more general RMP problem.

We now list the main contributions of this paper:

- We show that for the RMP problem, the minimum feasible rank cannot be approximated well unless $P =$

Appearing in *Proceedings of the 25^{th} International Conference on Machine Learning*, Helsinki, Finland, 2008.

NP (see Theorem 3.1). To get over this hurdle we introduce a relaxed notion of approximation, where along with approximating the optimal rank we also allow small violations in the constraints. In practice, this relaxed notion is as meaningful as the standard notion of approximation since almost all real-life problems have noisy measurements.

- We provide an algorithm for RMP based on the Multiplicative Weights Update framework of (Plotkin et al., 1991; Arora et al., 2005b) and under the relaxed notion of approximation, we prove approximation guarantees for the algorithm.
- We provide an algorithm for RMP based on the framework of online convex programming (OCP) introduced by (Zinkevich, 2003). We use the OCP framework in a novel way by changing the role of the decision maker to search over the constraints instead of the feasible points, as is usually the case. We prove that under the relaxed notion of approximation, the algorithm provides approximation guarantees. The guarantees obtained using the OCP framework are better than those obtained using the Multiplicative Weights Update framework by a logarithmic factor.
- For a practical application, we apply our methods to the problem of low-rank kernel learning which can be seen as a specific instance of general RMP.

We empirically evaluate our methods on synthetic instances of RMP, where the constraints are chosen randomly. We compare them with the trace-norm heuristic of (Fazel et al., 2001; Recht et al., 2007) and the log-det heuristic of (Fazel et al., 2003), and our experimental results indicate that our methods are significantly faster and give comparable rank solutions to existing methods. We also evaluate the performance of our methods for low-rank kernel learning on UCI datasets. On all the datasets, our algorithms improve the accuracy of the baseline kernel while also significantly decreasing the rank.

2. Related Work and Background

Most existing methods for rank minimization over convex sets are based on relaxing the non-convex rank function to a convex function, e.g., the trace-norm (Fazel et al., 2001; Recht et al., 2007) or the logarithm of the determinant (Fazel et al., 2003). Unfortunately, these heuristics do not have any guarantees on the quality of the solution in general. A notable exception is the work of (Recht et al., 2007), which extends the techniques of (Candès & Tao, 2005) for compressed sensing to rank minimization. (Recht et al., 2007) show that minimizing the trace-norm guarantees an optimal rank solution to a special class of RMP where only well-conditioned linear equalities are allowed. Thus their approach is limited in its applicability and it is not clear how to extend it to general RMP. We also remark that minimizing the trace-norm is computationally expensive, which further limits its applicability.

(Barvinok, 2002) (Chapter V) describes an approximation algorithm for RMP based on random projections and a generalization of the Johnson-Lindenstrauss Lemma, with an approximation guarantee similar to the one provided by our MW algorithm (Section 4.1). However, this approach works only for a special case of RMP where only linear equalities described by p.s.d. matrices are allowed. Furthermore, this approach needs to solve the relaxed RMP problem without the rank constraint which involves solving a large semi-definite programming problem. This maybe undesirable for various real-world applications such as the low-rank kernel learning problem. In contrast, our approaches can be used for a larger class of convex sets $\mathcal{C}$ and are considerably more scalable.

Several specific instances of the general RMP problem have been widely researched in the machine learning community. Examples include low-rank kernel learning, SDE, sparse PCA and NNMA. Most methods for these problems can be broadly grouped into the following two categories: a) methods which drop the rank constraint and use the top k eigenvectors of the solution to the relaxed optimization problem e.g., (Weinberger et al., 2004); b) methods which factor the matrix X in RMP into AB^T and optimize the resultant non-convex problem e.g., (Lee & Seung, 2000; Kim et al., 2007). However, typically these methods do not have any provable guarantees.

We apply our algorithms for the general RMP problem to the low-rank kernel learning problem(Bach & Jordan, 2005; Kulis et al., 2006). Existing methods for this problem do not provide any provable guarantees on the solution and/or assume that the initial kernel has a small rank to begin with. In contrast, a straight forward application of our general RMP framework gives algorithms with provable guarantees on the rank of the learned kernel. Furthermore, we demonstrate that our algorithms can be used to initialize existing methods to obtain better solutions.

Our approaches to RMP are based on two online learning methods - the generalized experts framework as abstracted in (Arora et al., 2005b) and the online convex programming (Zinkevich, 2003), which we now review briefly.

2.1. Multiplicative Weights Update Algorithm

The Multiplicative Weights Update algorithm (MW algorithm) is an adaptation of the Winnow algorithm (Littlestone & Warmuth, 1989) for a generalized experts framework as described in (Freund & Schapire, 1997). This framework was implicitly used by (Plotkin et al., 1991) for solving several fractional packing and covering problems and was formalized and extended to semi-definite programs

in (Arora et al., 2005a). Throughout this work we will follow the presentation of the generalized experts framework as abstracted in (Arora et al., 2005b).

In the generalized experts (GE) framework there is a set of n experts, a set of events $\mathcal{E}$, and a penalty matrix M such that the i-th expert incurs a penalty of $M(i,j)$ for an event $j \in \mathcal{E}$. The penalties are assumed to be bounded and lie in the interval $[-\rho, \rho]$ for a fixed $\rho > 0$. At each time step $t = 1, 2, \ldots,$ an adversary chooses an event $j^t \in \mathcal{E}$ so that the i-th expert incurs a penalty of $M(i, j^t)$. The goal in the GE framework is to formulate a *prediction algorithm* that chooses a distribution $\mathcal{D}^t = (p_1^t, \ldots, p_n^t)$ on the experts at time step t, so that the total expected loss incurred by the prediction algorithm is not much worse than the total loss incurred by the best expert. Formally, the goal of the prediction algorithm is to minimize

$$\sum_{t=1}^{T}\sum_{l=1}^{n} p_l^t M(l, j^t) - \min_i \sum_{t=1}^{T} M(i, j^t).$$

Note that the distribution in round t, $\mathcal{D}^t$, must be chosen without knowledge of the event j^t chosen at time step t. At every step t, the MW algorithm has a weight w_i^t assigned to expert i, and sets the distribution $\mathcal{D}^t = (p_1^t, \ldots, p_n^t)$, where $p_i^t = w_i^t / \sum_j w_j^t$. The MW algorithm then proceeds analogously to the Winnow algorithm and updates the weights at time step $t+1$ to $w_i^{t+1} = w_i^t(1-\delta)^{M(i,j^t)/\rho}$ if $M(i,j^t) \geq 0$ and $w_i^{t+1} = w_i^t(1+\delta)^{M(i,j^t)/\rho}$ if $M(i,j^t) < 0$, where δ is a parameter provided to the algorithm. For our analysis we will use the following theorem.

Theorem 2.1 (Corollary 4 of (Arora et al., 2005b))**.** *Suppose that for all i and $j \in \mathcal{E}$, $M(i,j) \in [-\rho, \rho]$. Let $\epsilon > 0$ be an error parameter and let $\delta = \min\{\frac{\epsilon}{4\rho}, \frac{1}{2}\}$, and $T = \frac{16\rho^2 \ln n}{\epsilon^2}$. Then, the following bound holds for the average expected loss of the MW algorithm*

$$\frac{\sum_{t=1}^{T}\sum_{l=1}^{n} p_l^t M(l, j^t)}{T} \leq \epsilon + \frac{\sum_t M(k, j^t)}{T}, \ \forall k.$$

2.2. Online Convex Programming

The online convex programming (OCP) framework (Zinkevich, 2003; Kalai & Vempala, 2005; Hazan et al., 2006) models various useful online learning problems like industrial production and network routing. The OCP framework involves a fixed convex set K and a sequence of unknown cost functions $f_1, f_2, \ldots : K \to \mathbb{R}$. At each time step t, a decision maker must choose a point $z_t \in K$ and incurs a cost $f_t(z_t)$. However, the choice of z_t must be made with the knowledge of $z_1, \ldots, z_{t-1}$ and $f_1, \ldots, f_{t-1}$ alone i.e., without knowing f_t. The total cost incurred by the algorithm after T steps equals $\sum_t f_t(z_t)$. The objective in OCP is to minimize the *regret* as defined below:

$$R(T) = \sum_{t=1}^{T} f_t(z_t) - \min_{z \in K} \sum_{t=1}^{T} f_t(z). \tag{1}$$

(Zinkevich, 2003) has shown that in the case when the functions f_t are convex and differentiable with bounded gradient, one can achieve a regret of $O(\sqrt{T})$. Let $\|K\| = \max_{z_1, z_2 \in K} \|z_1 - z_2\|$ and $G = \max_{z \in K, t \in \{1,\ldots\}} \| \bigtriangledown f^t(z)\|$, where $\|\cdot\|$ denotes the Euclidean norm (or Frobenius norm if the set K is defined over matrices). Also, assume that $\bigtriangledown f^t$ can be evaluated efficiently at any given point z. Under the above assumptions (Zinkevich, 2003) proposed a Generalized Infinitesimal Gradient Ascent algorithm which achieves a regret of $O((G^2 + \|K\|^2)\sqrt{T})$. The function GIGA in Algorithm 2 describes a slightly modified version of (Zinkevich, 2003)'s algorithm that achieves the following improved regret bound.

Theorem 2.2 (Adaptation of Theorem 1 of (Zinkevich, 2003))**.** *The following bound holds for the regret of the GIGA sub-routine of Algorithm 2 after T rounds,*

$$R(T) \leq G \cdot \|K\| \sqrt{T} \tag{2}$$

Proof sketch: Using the modified step-size in Algorithm 2, the theorem follows from Zinkevich's original proof. □

3. Computational Complexity

As was mentioned in the introduction, RMP is NP-hard in general. Further, by a reduction to the problem of support minimization over convex sets, and using hardness of approximation results from (Amaldi & Kann, 1998) we prove the following hardness result for RMP. A full proof of the following theorem appears in (Meka et al., 2008).

Theorem 3.1. *There exists no polynomial time algorithm for approximating* RMP *within a logarithmic factor unless $P = NP$. Further, assuming $NP \nsubseteq DTIME(n^{poly \log n})$,* RMP *is not approximable within a factor of $2^{\log^{1-\delta} n}$ for every $\delta > 0$; and* RMP *is not approximable within a factor of $2^{\log^{1-\delta} \Delta}$ for every $\delta > 0$, where $\Delta = \max\{\|A_i\|_F + |b_i| : 1 \leq i \leq m\}$* [1].

In view of the above hardness result we introduce a weaker notion of approximation. We believe the relaxed notion of approximation to be of equal use, if not more, as the standard notion of approximation in practice. For an instance of RMP, let $\mathbb{F}(A_1, \ldots, A_m, \boldsymbol{b}, \mathcal{C})$ denote the feasible region, where $\boldsymbol{b} = (b_1, \ldots, b_m)$:

$$\mathbb{F}(A_1, \ldots, A_m, \boldsymbol{b}, \mathcal{C}) = \{X : X \in \mathcal{C}, \mathrm{Tr}(A_i X) \geq b_i, \forall i\}. \tag{3}$$

[1] This hardness result holds even when $\mathcal{C}$ is fixed to be the unit ball under an L_p or Frobenius norm or many other common sets.

Definition 3.1. *Given a function $c : \mathbb{R} \to \mathbb{R}_+$, we say that a matrix $\overline{X}$ is a $(c(\epsilon), \epsilon)$-approximate solution to* RMP *if the following hold:*

$\bar{X} \in \mathbb{F}(A_1, \ldots, A_m, \boldsymbol{b} - \epsilon\mathbf{1}, \mathcal{C})$

$\textit{rank}(\bar{X}) \leq c(\epsilon) \min\{\textit{rank}(X) : X \in \mathbb{F}(A_1, .., A_m, \boldsymbol{b}, \mathcal{C})\}$.

Further, we say that RMP *is $(c(\epsilon), \epsilon)$-approximable, if there exists a polynomial time algorithm that given inputs $A_1, \ldots, A_m, \boldsymbol{b}, \epsilon$, outputs a $(c(\epsilon), \epsilon)$-approximate solution to* RMP.

Thus, along with approximating the minimum feasible rank we also allow a small violation, quantified by ϵ, of the constraints. Note that for $\epsilon = 0$, we recover the normal notion of approximation with an approximation factor of $c(0)$.

4. Methodology

Our approaches to RMP rely on the fact that even though RMP is hard in general, it is efficiently solvable for certain convex sets $\mathcal{C}$ when there is a single trace constraint. For instance, when $\mathcal{C} = \{X : \|X\|_F \leq 1\}$, a RMP problem with a single trace constraint can be solved efficiently using a singular value decomposition of the constraint matrix.

In our approach, we assume the existence of an oracle $\mathcal{O}$ that solves the following RMP problem with a single trace constraint, and returns an optimal X or declares the problem infeasible:

$$\mathcal{O}: \quad \min \text{ rank}(X) \text{ s.t. } \text{Tr}(AX) \geq b, X \in \mathcal{C}. \quad (4)$$

As discussed above, for certain convex sets $\mathcal{C}$, oracle $\mathcal{O}$ solves a non-convex problem. In both our approaches, we exploit this fact by making several queries to the oracle where the trace constraint $\text{Tr}(AX) \geq b$ is obtained by a weighted combination of the original trace constraints. The trick then is to choose the combinations in such a way that after a small number of iterations, we can find a low-rank X that satisfies all the constraints with at most an ϵ-violation.

Based on the above intuition, we give two approaches to solve the RMP problem - one based on the Multiplicative Weights Update algorithm and the other based on online convex programming.

Before we describe our algorithms, we need to introduce additional notation. For an instance of RMP specified by matrices $A_1, \ldots, A_m$, scalars $b_1, \ldots, b_m$ and convex set $\mathcal{C}$, let $D = \max\{\|X\|_F : X \in \mathcal{C}\}$. We assume, without loss of generality, that $D \geq 1$. Recall that $\mathbb{F}((A_1, \ldots, A_m), \boldsymbol{b}, \mathcal{C})$ and $\mathbb{F}((A_1, \ldots, A_m), \boldsymbol{b} - \epsilon\mathbf{1}, \mathcal{C})$ denote the feasibility sets as defined in (3) and $\Delta = \max\{\|A_i\|_F + |b_i| : 1 \leq i \leq m\}$. Further, let k^* be the rank of the optimal solution to RMP. That is,

$$k^* = \min\{\text{rank}(X) : X \in \mathbb{F}((A_1, \ldots, A_m), \boldsymbol{b}, \mathcal{C})\}.$$

Algorithm 1 RMP-MW (Multiplicative Updates)

Require: Constraints (A_i, b_i), $1 \leq i \leq m$, ϵ
Require: Oracle $\mathcal{O}(A, b)$ which solves
$\min \text{ rank}(X) \quad s.t. \quad \text{Tr}(AX) \geq b, \ X \in \mathcal{C}$
Initialize: $w_i^1 = 1, \ \forall i$ and $t = 1$
repeat
  Set $(A^t, b^t) = \sum_i w_i^t (A_i, b_i)$
  if Oracle $\mathcal{O}(A^t, b^t)$ declares infeasibility **then**
    return Problem is infeasible
  else
    Obtain X^t using Oracle $\mathcal{O}(A^t, b^t)$
    Set $M(i, X^t) = \text{Tr}(A_i X^t) - b_i$
    Set $\rho = \max_i M(i, X^t)$
    Set $\boldsymbol{w}^{t+1}$=MultUpdate($\boldsymbol{w}^t, M, \rho, \epsilon$)
  end if
  Set $t = t + 1$
until $t > T$
return $\overline{X} = \sum_t X^t / T$

function $\boldsymbol{w}^{t+1}$=MultUpdate($\boldsymbol{w}^t, M, \rho, \epsilon$)
Set $\delta = \min\{\frac{\epsilon}{4\rho}, \frac{1}{2}\}$
for all $1 \leq i \leq m$ **do**
  if $M(i, X^t) \geq 0$ **then**
    $w_i^{t+1} = w_i^t (1 - \delta)^{M(i, X^t)/\rho}$
  else
    $w_i^{t+1} = w_i^t (1 + \delta)^{-M(i, X^t)/\rho}$
  end if
end for

4.1. Rank Minimization via Multiplicative Weights Update

In this section we present an approach to RMP based on the generalized experts (GE) framework described in Section 2.1. To adapt the GE framework for the RMP problem, we first need to select a set of experts, a set of events and the associated penalties. We associate each RMP constraint $\text{Tr}(A_i X) \geq b_i$ with an expert and let the events correspond to elements of $\mathcal{C}$. The penalty for expert i corresponding to the i-th constraint and event X is then given by $\text{Tr}(A_i X) - b_i$. Note that rather than rewarding a satisfied constraint, we penalize it. This strategy is motivated by the work of (Plotkin et al., 1991; Arora et al., 2005a) and is similar to boosting, where a distribution is skewed towards an example for which the current hypothesis made an incorrect prediction.

We assign weight w_i^t to the i-th expert in the t-th iteration, and initialize the weights $w_i^1 = 1$, for all i. In the t-th iteration we query the oracle $\mathcal{O}$ with $(A^t, b^t) = \sum_i w_i^t (A_i, b_i)$ to obtain a solution $X^{t+1} \in \mathcal{C}$. We then use the Multiplicative Weights Update algorithm as described in function MultUpdate of Algorithm 1 to compute the weights

w_i^{t+1} for the $(t+1)$-st iteration. Algorithm 1 describes our multiplicative update based algorithm for RMP. In the following theorem we prove approximation guarantees for the solution output by Algorithm 1.

Theorem 4.1. *Given the existence of an oracle $\mathcal{O}$ to solve the problem* (4)*, Algorithm 1 outputs an $(O(\frac{\Delta^2 D^2 \log n}{\epsilon^2}), \epsilon)$-approximate solution to* RMP.

Proof. Observe that, if the oracle declares infeasibility at any time step t, the original problem is also infeasible. Hence, we assume that the oracle returns a feasible point X^t at time-step t, for all $1 \leq t \leq T$.

Now, $|\text{Tr}(A_i X) - b_i| \leq \|A_i\|_F \|X\|_F + |b_i| \leq \Delta D$. Thus, the penalties $\text{Tr}(A_i X) - b_i$ lie in the interval $[-\Delta D, \Delta D]$. Since Algorithm 1 uses multiplicative updates to update the weights[2] as in Theorem 2.1, for $T = 16(\Delta D)^2 \log n/\epsilon^2$, we have

$$\frac{\sum_t \sum_j p_j^t [A_j X^t - b_j]}{T} \leq \epsilon + \frac{\sum_t [A_i X^t - b_i]}{T}, \ \forall i,$$

where $p_j^t = w_j^t / \sum_l w_l^t$. Since $\text{Tr}(A^t X^t) \geq b^t), \ \forall t$, the $LHS \geq 0$. Thus, for $\overline{X} = \sum_t X^t/T$ we have

$$\text{Tr}(A_i \overline{X}) \geq b_i - \epsilon, \quad \forall i. \tag{5}$$

We now bound the rank of $\overline{X}$ compared to the optimal value. Let t be such that X_t has the highest rank, say k, among $X_1, \ldots, X_T$. Then, $k^* \geq k$, as for a particular convex combination of (A_i, b_i) the minimum rank possible was k. Thus, $\text{rank}(\overline{X}) \leq kT = O(\frac{(\Delta^2 \cdot D^2 \log n) k^*}{\epsilon^2})$. Using (5) we have that $\overline{X} \in \mathbb{F}((A_1, \ldots, A_m), b - \epsilon \mathbf{1}, \mathcal{C})$. Thus, by Definition 3.1 $\overline{X}$ is an $(O(\frac{\Delta^2 D^2 \log n}{\epsilon^2}), \epsilon)$-approximate solution to RMP. □

The running time of Algorithm 1 is $O(\frac{\Delta^2 D^2 \log n}{\epsilon^2}(T_{\mathcal{O}} + mn^2))$, where $T_{\mathcal{O}}$ denotes the oracle's running time.

4.2. Rank Minimization via OCP

In this section, we present a novel application of online convex programming described in Section 2.2 to obtain an approximate solution to RMP. The intuition behind this approach is similar to that of Section 4.1; in fact this approach can be viewed as a generalization of the approach of Section 4.1.

In the OCP framework one generally associates the convex set K with a feasible region and the cost functions with penalty functions. In our application of OCP to RMP we flip this view and choose K to be the space of convex combinations of the constraints and associate cost functions with feasible points of RMP. In particular, we set $K \subseteq \mathbb{R}^{n \times n} \times \mathbb{R}$ to be the convex hull of $(A_1, b_1), \ldots, (A_m, b_m)$, i.e.,

$$K = \left\{ \sum_i \lambda_i (A_i, b_i) : \sum_i \lambda_i = 1, \lambda_i \geq 0 \ \forall i \right\}.$$

Given a matrix X, we define a cost function $f_X : K \to \mathbb{R}$ by $f_X(A, b) = \text{Tr}(AX) - b$.

We initialize $A^1 = \sum_i A_i/m$ and $b^1 = \sum_i b_i/m$. Given $(A^t, b^t) \in K$ for the t-th iteration, we query the oracle $\mathcal{O}$ with $(A, b) = (A^t, b^t)$ to obtain a solution $X^t \in \mathcal{C}$. We then set the cost function $f^t(A, b) = f_{X^t}(A, b) = \text{Tr}(AX^t) - b$ and use the OCP algorithm (Zinkevich, 2003) as described in function GIGA of Algorithm 2 to compute (A^{t+1}, b^{t+1}) for the $(t+1)$-st iteration. Algorithm 2 describes our OCP based algorithm for RMP. In the following theorem we prove approximation guarantees for the output of Algorithm 2.

Algorithm 2 RMP-OCP (Online Convex Programming)

Require: Constraints (A_i, b_i), $1 \leq i \leq m$, ϵ
Require: Oracle $\mathcal{O}(A, b)$ which solves $\min \text{rank}(X) \quad s.t. \quad \text{Tr}(AX) \geq b, \ X \in \mathcal{C}$
Initialize: $A^1 = \frac{\sum_i A_i}{m}$ and $b^1 = \frac{\sum_i b_i}{m}$, $t = 1$
Set $K = \{\sum_i \lambda_i (A_i, b_i) : \sum_i \lambda_i = 1, \lambda_i \geq 0 \ \forall i\}$
repeat
  if Oracle $O(A^t, b^t)$ declares infeasibility **then**
    return Problem is infeasible
  else
    Obtain X^t using Oracle $\mathcal{O}(A^t, b^t)$
    Define function $f^t(A, b) = \text{Tr}(AX^t) - b$
    Set (A^{t+1}, b^{t+1})=GIGA($(A^t, b^t), f^t(A, b), K, t$)
  end if
  Set $t = t + 1$
until $t > T$
return $\overline{X} = \sum_t X^t/T$

function $\boldsymbol{z^{t+1}}$=GIGA($\boldsymbol{z^t}, f^t(\boldsymbol{z}), K, t$)
Set $\eta_t = \frac{\Delta}{2D\sqrt{t}}$
Set $\boldsymbol{z^{t+1}} = \Pi_K \left(\boldsymbol{z^t} - \eta_t \nabla f^t(\boldsymbol{z^t}) \right)$, where Π_K represents the orthogonal projection onto K

Theorem 4.2. *Given the existence of an oracle $\mathcal{O}$ to solve the problem* (4)*, Algorithm 2 outputs an $(O(\frac{\Delta^2 D^2}{\epsilon^2}), \epsilon)$-approximate solution to* RMP.

Proof. As in Theorem 4.1 we assume that the oracle returns a feasible point at all time steps. Note that using the terminology of Theorem 2.2, $G = \max_{z \in K, t \in \{1, \ldots\}} \| \bigtriangledown$

[2] Our updates are slightly different from those of (Arora et al., 2005b) in that we adaptively choose the width parameter ρ. However, the analysis of (Arora et al., 2005b) is applicable for these updates as well.

$f^t(z)\| \leq D$ and $\|K\| \leq \Delta$. Thus, using Theorem 2.2 we have

$$\sum_{t=1}^{T}(\text{Tr}(A^tX^t)-b^t) \leq \min_{(A,b)\in K}\sum_{t=1}^{T}(\text{Tr}(AX^t)-b)+\Delta D\sqrt{T}.$$

Note that the above LHS ≥ 0 since oracle returns a feasible $X^t,\ \forall t$. Thus, for $T = \Delta^2D^2/\epsilon^2$ and $\overline{X} = \sum_t X^t/T$,

$$\text{Tr}(A\overline{X}) \geq b - \epsilon, \tag{6}$$

for all $(A, b) \in K$. In particular, we have for every i, $\text{Tr}(A_i\overline{X}) \geq b_i - \epsilon$. We now bound the rank of $\overline{X}$ compared to the optimal value. Let t be such that X_t has the highest rank, say k, among $X_1, \ldots, X_T$. Then, we must have $k^* \geq k$, and so we have $\text{rank}(\overline{X}) \leq kT \leq O((\Delta D)^2k^*/\epsilon^2)$. Also, from (6) we have that $\overline{X} \in \mathbb{F}((A_1, \ldots, A_m), b - \epsilon\mathbf{1}, \mathcal{C})$. Thus by Definition 3.1, $\overline{X}$ is a $(O((\Delta^2D^2)/\epsilon^2), \epsilon)$-approximate solution to RMP. □

The running time of Algorithm 2 is $O(\frac{\Delta^2D^2}{\epsilon^2}(T_{\mathcal{O}}+T_{OCP}+mn^2))$, where $T_{\mathcal{O}}$ denotes the running time of the oracle, and T_{OCP} denotes the time taken in each round by the GIGA algorithm of Theorem 2.2.

4.3. Discussion

Oracle: The oracle for solving problem (4) plays a crucial role in both our approaches. As discussed previously, for typical cases of $\mathcal{C}$, like the unit ball under an L_p or Frobenius norm etc., (4) can be solved by the singular value decomposition of A. Further, in the case when the set $\mathcal{C}$ involves a quadratic or ellipsoid constraint we can use the S-procedure (Rockafellar, 1970) to solve (4).

Comparison of the approaches: Our approach to RMP based on Multiplicative Weights Update has a slightly weaker guarantee than the approach based on OCP. This is also confirmed by our experiments where OCP gives better results than the MW approach. However, the MW approach is computationally less intensive as the approach based on OCP involves a projection onto the convex set K. Thus, MW can be used for large scale problems.

Limitations: A drawback of our methods is the dependence on Δ, ϵ in the bounds of Theorems 4.1 and 4.2. This limits the applicability of our methods to problems, such as NNMA, with a large number of non-negativity constraints where the ratio $\frac{\Delta}{\epsilon}$ is typically large. However, our algorithms can be used as a heuristic for such problems and can be used to initialize other methods which require a good low-rank solution for initialization. Also, the lower bounds for the experts framework and boosting suggest that the dependence on Δ, ϵ in our bounds may be optimal for the general RMP problem (Arora et al., 2005b).

5. Low-rank Kernel Learning

In this section we apply both our rank minimization algorithms to the problem of low-rank kernel learning, which involves finding a low-rank positive semi-definite (p.s.d.) matrix that satisfies linear constraints typically derived from labeled data. Due to the rank constraint, this problem is non-convex and is in general hard to solve. As described below, both our online learning approaches can be applied naturally to this problem. We provide provable guarantees on the rank of the obtained kernel.

Formally, the low-rank kernel learning problem can be cast as the following optimization problem:

$$\begin{aligned} \min_K \quad & \|K - K_0\|_F \\ \text{s.t.} \quad & \text{Tr}(S_iK) \leq \ell, \quad \forall\ 1 \leq i \leq |\mathcal{S}|, \\ & \text{Tr}(D_jK) \geq u, \quad \forall\ 1 \leq j \leq |\mathcal{D}|, \\ & \text{rank}(K) \leq r, \quad K \succeq 0, \end{aligned} \tag{7}$$

where $\mathcal{S}$ is a set of pairs of points from the same class that are constrained to have distance less than ℓ. Similarly, $\mathcal{D}$ is a set of pairs of points from different classes that are constrained to have distance greater than u, with $\ell \ll u$. For a similarity constraint matrix S_i, $S_i(i_1, i_1) = S_i(i_2, i_2) = 1,\ S_i(i_1, i_2) = S_i(i_2, i_1) = -1$ and all other entries 0. The dissimilarity constraint matrices D_j can be constructed similarly. Assuming $\|K_0\|_F = 1$, (7) can be reformulated as:

$$\begin{aligned} \min_K \quad & \text{rank}(K) \\ \text{s.t.} \quad & \text{Tr}(S_iK) \leq \ell\ \forall i, \quad \text{Tr}(D_jK) \geq u\ \forall j, \\ & \text{Tr}(KK_0) \geq \beta, \quad \|K\|_F \leq 1, \quad K \succeq 0, \end{aligned} \tag{8}$$

where β is a function of r and can be computed using binary search. Note that (8) is a special case of RMP with the convex set $\mathcal{C}$ being the intersection of the p.s.d. cone and the unit Frobenius ball. Hence, we can use RMP-MW and RMP-OCP to solve (8). Given (A, b) the oracle for both the methods solves:

$$\min_K \text{rank}(K) :\ \text{Tr}(AK) \geq b, \|K\|_F \leq 1,\ K \succeq 0. \tag{9}$$

Let $A = U\Sigma U^T$ be the eigenvalue decomposition of A, and let Λ be a diagonal matrix with just the positive entries of Σ. Then the minimum k s.t. $\sqrt{\sum_{i=1}^{k}\Lambda(i,i)^2} \geq b$ is the solution to (9). This follows from elementary linear algebra. Note that for the oracle solving (9), $T_{\mathcal{O}} = O(n^3)$.

Now, $D = 1$ and $\Delta = O(1 + l^2 + u^2)$ as $\|S_i\|_F = \|D_j\|_F = 2$. Using Theorem 4.1, the RMP-MW algorithm obtains a solution with rank $r \leq O(\frac{1+u^2+l^2}{\epsilon^2}\log n)r^*$ where r^* is the optimal rank. Similarly, RMP-OCP obtains an $\left(O(\frac{1+u^2+l^2}{\epsilon^2}), \epsilon\right)$-approximate solution. In Section 6.2, we present empirical results for RMP-MW and RMP-OCP algorithms on some standard UCI datasets.

6. Experimental Results

We empirically evaluate and compare our algorithms to existing methods for general RMP as well as low-rank kernel learning. For general RMP, we use synthetic examples to compare our methods against the trace-norm heuristic (Recht et al., 2007) and the log-det heuristic (Fazel et al., 2001). The trace-norm heuristic relaxes the rank objective to the trace-norm of the matrix, which is given by the sum of its singular values. Note that the trace-norm of a matrix is a convex function. The log-det heuristic relaxes the rank objective to the log of the determinant of the matrix. For the application of RMP to low-rank kernel learning, we use standard UCI datasets. All the presented results represent the average over 20 runs.

6.1. Synthetic Datasets

First we use synthetic datasets by generating random matrices $A_i \in \mathbb{S}_n$, where $\mathbb{S}_n$ is the set of $n \times n$ symmetric matrices. We also generate a random positive semi-definite matrix $X_0 \in \mathbb{S}_n$ with $\|X_0\|_F \leq 1$, and use the obtained X_0 to generate constraints $\text{Tr}(A_i X) \geq b_i = \text{Tr}(A_i X_0)$. The convex set $\mathcal{C}$ is fixed to be the intersection of the p.s.d cone and the unit ball under the Frobenius norm. We fix the number of constraints to be 200 and the tolerance ϵ for RMP-MW and RMP-OCP to be 5%. We use SeDuMi to implement the trace-norm and log-det heuristics.

In Table 1, we compare the ranks of the solutions obtained by our algorithms against the ones obtained by the trace-norm and log-det heuristics. For small n, both trace-norm and log-det heuristic perform better than RMP-MW and RMP-OCP. Note that since the constraint matrices A_i are random, they satisfy (with high probability) the restricted isometry property used in the analysis of (Recht et al., 2007). However, RMP-OCP outperforms trace-norm heuristic for large n (Table 1, $n = 100$) and RMP-MW performs comparably. We attribute this phenomenon to the Frobenius norm constraint for which the theoretical guarantees of (Recht et al., 2007) are not applicable. Also, both trace-norm and log-det heuristic scale poorly with the problem size and fail to obtain a result in reasonable time even for moderately large n. In contrast, both our algorithms scale well with n, with RMP-MW in particular able to solve problems of sizes up to $n = 5000$.

6.2. Low-rank Kernel Learning

We evaluate the performance of our methods applied to the problem of low-rank kernel learning, as described in Section 5, for k-NN classification on standard UCI datasets. We use two-fold cross validation with $k = 5$. The lower and upper bounds for the similarity and dissimilarity constraints (l, u) are set using the 30-th and 70-th percentiles of the observed distribution of distances between pairs of points. We randomly select a set of $40c^2$ pairs of points for constraints, where c is the number of classes in the dataset. We run both RMP-MW and RMP-OCP for $T = 50$ iterations. Empirically our algorithms significantly outperform the theoretical rank guarantees of Theorems (4.1) and (4.2).

Method\n	50	75	100	200	300
RMP-MW	23.25	11.25	7.3	2	2
RMP-OCP	12.8	7.5	5.3	2	2
Trace-norm	6.8	6.7	6.5	-	-
LogDet	5	4.2	4.0	-	-

Table 1. Rank of the matrices obtained by different RMP methods for varying size of the constraint matrices (n). The number of constraints generated (m) is fixed to be 200. A "-" represents that the method could not find a solution within 3 hours on a 2.6GHz Pentium 4 machine. Note that for large problem sizes, both the trace-norm and the log-det heuristics are not computationally viable. Both our approaches outperform the trace-norm heuristic as the problem size increases.

Dataset\Method	GK	MW	OCP	BK
Musk	80.80 (476)	93.11 (44.1)	**98.15** (61.2)	81.51 (61.2)
Heart	77.44 (267)	91.05 (46.8)	**91.13** (39.5)	83.91 (39.5)
Ionosphere	90.34 (350)	**91.26** (40)	91.17 (27.9)	90.67 (27.9)
Cancer	90.12 (569)	93.14 (82)	91.46 (94)	**93.38** (94)
Scale	66.34 (607)	**73.78** (146)	72.46 (91)	72.11 (91)

Table 2. Accuracies for 5-Nearest Neighbor classification using kernels obtained by different methods. Numbers in parentheses represent the rank of the obtained solution. GK represents Gaussian Kernel ($\sigma = 0.1$), MW represents RMP-MW, OCP represents RMP-OCP and BK represents BurgKernel(Kulis et al., 2006). Overall, RMP-OCP obtains the best accuracy.

Table 2 shows the accuracies achieved by the baseline Gaussian kernel (with $\sigma = 0.1$), RMP-MW, RMP-OCP and the Burg divergence (also called as LogDet divergence) based low-rank kernel learning algorithm (BurgKernel) of (Kulis et al., 2006). It can be seen from the table that both RMP-MW and RMP-OCP obtain a significantly lower rank kernel than the baseline Gaussian kernel. Further, RMP-MW and RMP-OCP achieve a substantially higher accuracy than the Gaussian kernel. Our algorithms also achieve a substantial improvement in accuracy over the BurgKernel method. Note that we iterate our algorithms for fewer iterations compared to the ones suggested by the theoretical bounds, hence few of the constraints maybe unsatisfied. This suggests that these unsatisfied constraints maybe noisy constraints and have small effect on the generalization error. We leave further investigation into gener-

alization error of our methods as a topic for future research.

Note that the BurgKernel method needs to be initialized with a low-rank kernel. Typically, a few top eigenvectors of the baseline kernel are used for this initialization. However, selecting only a few top eigenvectors can lead to a poor initial kernel, especially if the rank of the initial kernel is high. This can further lead to poor accuracy for the BurgKernel method, as indicated by our experiments. Instead, the kernels obtained by our algorithms could be used to *initialize* the BurgKernel algorithm. For example, for the case of the Heart dataset, initialization of BurgKernel algorithm with the low-rank solution obtained by RMP-OCP method achieves an accuracy of 94.29 compared to 83.91 achieved when initialized with the top eigenvectors of the baseline Gaussian kernel. Note that this also improves upon the accuracy achieved by RMP-MW and RMP-OCP.

7. Conclusion

In this paper, we address the general problem of rank minimization over polyhedral sets and in particular the problem of low-rank kernel learning. We show that the problem is hard to approximate within a factor of $2^{\log^{1-\epsilon}\Delta}$ (see Theorem 3.1). Further, we introduce a relaxed notion of approximation and present two novel approaches for solving RMP with provable guarantees. Our first approach is based on the multiplicative weights update framework and provides an $(O(\frac{\Delta^2 D^2}{\epsilon^2}\log n), \epsilon)$-approximate solution. Our second approach is based on online convex programming and provides a tighter bound of $O(\frac{\Delta^2 D^2}{\epsilon^2})$ for the rank of the obtained matrix.

For future work, it would be interesting to see if the hardness of approximation factor of Theorem 3.1 can be improved; we believe it can be improved to $O(\Delta^2)$. Another question of interest is whether the dependence on ϵ in the bounds of Theorems 4.1 and 4.2 can be improved.

The regret bounds of (Zinkevich, 2003) were improved in (Hazan et al., 2006). However, the algorithms of (Hazan et al., 2006) require stronger convexity properties which are not satisfied in our application of OCP to RMP. It would be interesting to see if the linear constraints in RMP can be perturbed to satisfy the strong convexity properties, so that the improved regret bounds of (Hazan et al., 2006) can be used to achieve better bounds for RMP.

Our algorithms to RMP are motivated from an online learning perspective. However, for an optimization problem such as RMP an understanding of the algorithms from an optimization perspective would be highly desirable. In particular, intuitively there seems to be a correspondence between our methods and a primal-dual approach but we were unable to obtain a rigorous connection. We believe that such an understanding would be of importance in obtaining new applications of the online learning approach to solving optimization problems.

References

Amaldi, E., & Kann, V. (1998). On the approximability of minimizing non-zero variables or unsatisfied relations in linear systems. *Theoretical Computer Science, 209*, 237–260.

Arora, S., Hazan, E., & Kale, S. (2005a). Fast algorithms for approximate semidefinite programming using the multiplicative weights update method. *FOCS* (pp. 339–348).

Arora, S., Hazan, E., & Kale, S. (2005b). Multiplicative weights method: a meta-algorithm and its applications. *Survey, http://www.cs.princeton.edu/ arora/pubs/MWsurvey.pdf*.

Bach, F. R., & Jordan, M. I. (2005). Predictive low-rank decomposition for kernel methods. *ICML* (pp. 33–40).

Barvinok, A. (2002). *A course in convexity*. American Mathematical Society.

Candès, E. J., & Tao, T. (2005). Decoding by linear programming. *IEEE Transactions on Information Theory, 51*, 4203–4215.

Fazel, M., Hindi, H., & Boyd, S. (2001). A rank minimization heuristic with application to minimum order system approximation. *American Control Conference, Arlington, Virginia*.

Fazel, M., Hindi, H., & Boyd, S. (2003). Log-det heuristic for matrix rank minimization with applications to Hankel and Euclidean distance matrices. *American Control Conference, Denver, Colorado*.

Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci., 55*, 119–139.

Hazan, E., Kalai, A., Kale, S., & Agarwal, A. (2006). Logarithmic regret algorithms for online convex optimization. *COLT* (pp. 499–513).

Kalai, A. T., & Vempala, S. (2005). Efficient algorithms for online decision problems. *J. Comput. Syst. Sci., 71*, 291–307.

Kim, D., Sra, S., & Dhillon, I. S. (2007). Fast Newton-type methods for the least squares nonnegative matrix approximation problem. *SDM* (pp. 343–354).

Kulis, B., Sustik, M., & Dhillon, I. S. (2006). Learning low-rank kernel matrices. *ICML* (pp. 505–512).

Lee, D. D., & Seung, H. S. (2000). Algorithms for non-negative matrix factorization. *NIPS* (pp. 556–562).

Littlestone, N., & Warmuth, M. K. (1989). The weighted majority algorithm. *FOCS* (pp. 256–261).

Meka, R., Jain, P., Caramanis, C., & Dhillon, I. S. (2008). *Rank minimization via online learning* (Technical Report TR-08-23). The Univ. of Texas at Austin, Dept. of Comp. Sci.

Plotkin, S. A., Shmoys, D. B., & Tardos, E. (1991). Fast approximation algorithms for fractional packing and covering problems. *IEEE Symposium on Foundations of Computer Science* (pp. 495–504).

Recht, B., Fazel, M., & Parrilo, P. (2007). Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *Preprint, http://www.optimization-online.org/DB_HTML/2007/06/1707.html*.

Rockafellar, R. T. (1970). *Convex analysis*. Princeton University Press.

Weinberger, K. Q., Sha, F., & Saul, L. K. (2004). Learning a kernel matrix for nonlinear dimensionality reduction. *ICML*.

Zinkevich, M. (2003). Online convex programming and generalized infinitesimal gradient ascent. *ICML* (pp. 928–936).