

Large-Scale Convex Minimization with a Low-Rank Constraint

Shai Shalev-Shwartz

Alon Gonen

School of Computer Science and Engineering, The Hebrew University of Jerusalem, ISRAEL

Ohad Shamir

Microsoft Research New-England, USA

SHAIS@CS.HUJI.AC.IL

ALONGNN@GMAIL.COM

OHADSH@MICROSOFT.COM

Abstract

We address the problem of minimizing a convex function over the space of large matrices with low rank. While this optimization problem is hard in general, we propose an efficient greedy algorithm and derive its formal approximation guarantees. Each iteration of the algorithm involves (approximately) finding the left and right singular vectors corresponding to the largest singular value of a certain matrix, which can be calculated in linear time. This leads to an algorithm which can scale to large matrices arising in several applications such as matrix completion for collaborative filtering and robust low rank matrix approximation.

1. Introduction

Our goal is to approximately solve an optimization problem of the form:

$$\min_{A: \text{rank}(A) \leq r} R(A), \quad (1)$$

where $R: \mathbb{R}^{m \times n} \rightarrow \mathbb{R}$ is a convex and smooth function. This problem arises in many machine learning applications such as collaborating filtering (Koren et al., 2009), robust low rank matrix approximation (Ke & Kanade, 2005; Croux & Filzmoser, 1998; A. Baccini & Falguerolles, 1996), and multiclass classification (Amit et al., 2007). The rank constraint on A is non-convex and therefore it is generally NP-hard to solve Equation (1) (this follows from (Natarajan, 1995; Davis et al., 1997)).

Appearing in *Proceedings of the 28th International Conference on Machine Learning*, Bellevue, WA, USA, 2011. Copyright 2011 by the author(s)/owner(s).

In this paper we describe and analyze an approximation algorithm for solving Equation (1). Roughly speaking, the proposed algorithm is based on a simple, yet powerful, observation: instead of representing a matrix A using $m \times n$ numbers, we represent it using an infinite dimensional vector λ , indexed by all pairs (u, v) taken from the unit spheres of \mathbb{R}^m and \mathbb{R}^n respectively. In this representation, low rank corresponds to sparsity of the vector λ .

Thus, we can reduce the problem given in Equation (1) to the problem of minimizing a vector function $f(\lambda)$ over the set of sparse vectors, $\|\lambda\|_0 \leq r$. Based on this reduction, we apply a greedy approximation algorithm for minimizing a convex vector function subject to a sparsity constraint. At first glance, a direct application of this reduction seems impossible, since λ is an infinite-dimensional vector, and at each iteration of the greedy algorithm one needs to search over the infinite set of the coordinates of λ . However, we show that this search problem can be cast as the problem of finding the first leading right and left singular vectors of a certain matrix.

After describing and analyzing the general algorithm, we show how to apply it to the problems of matrix completion and robust low-rank matrix approximation. As a side benefit, our general analysis yields a new sample complexity bound for matrix completion. We demonstrate the efficacy of our algorithm by conducting experiments on large-scale movie recommendation data sets.

1.1. Related work

As mentioned earlier, the problem defined in Equation (1) has many applications, and therefore it was studied in various contexts. A popular approach is to use the trace norm as a surrogate for the rank (e.g. (Fazel et al., 2002)). This approach is closely related to the idea of using the ℓ_1 norm as a surrogate for spar-

sity, because low rank corresponds to sparsity of the vector of singular values and the trace norm is the ℓ_1 norm of the vector of singular values. This approach has been extensively studied, mainly in the context of collaborating filtering. See for example (Cai et al., 2008; Candes & Plan, 2010; Candès & Recht, 2009; Keshavan et al., 2010; Keshavan & Oh, 2009).

While the trace norm encourages low rank solutions, it does not always produce sparse solutions. Generalizing recent studies in compressed sensing, several papers (e.g. (Recht et al., 2007; Cai et al., 2008; Candes & Plan, 2010; Candès & Recht, 2009; Recht, to appear)) give recovery guarantees for the trace norm approach. However, these guarantees rely on rather strong assumptions (e.g., it is assumed that the data is indeed generated by a low rank matrix, that certain incoherence assumptions hold, and for matrix completion problems, it requires the entries to be sampled uniformly at random). In addition, trace norm minimization often involves semi-definite programming, which usually does not scale well to large-scale problems.

In this paper we tackle the rank minimization directly, using a greedy selection approach, without relying on the trace norm as a convex surrogate. Our approach is similar to forward greedy selection approaches for optimization with sparsity constraint (e.g. the MP (Mallat & Zhang, 1993) and OMP (Pati et al., 2002) algorithms), and in particular we extend the fully corrective forward greedy selection algorithm given in (Shalev-Shwartz et al., 2010)). We also provide formal guarantees on the competitiveness of our algorithm relative to matrices with small trace norm.

Recently, (Lee & Bresler, 2010) proposed the ADMiRA algorithm, which also follows the greedy approach. However, the ADMiRA algorithm is different, as in each step it first chooses $2r$ components and then uses SVD to revert back to a r rank matrix. This is more expensive than our algorithm which chooses a single rank 1 matrix at each step. The difference between the two algorithms is somewhat similar to the difference between the OMP (Pati et al., 2002) algorithm for learning sparse vectors, to CoSaMP (Needell & Tropp, 2009) and SP (Dai & Milenkovic, 2008). In addition, the ADMiRA algorithm is specific to the squared loss while our algorithm can handle any smooth loss. Finally, while ADMiRA comes with elegant performance guarantees, these rely on strong assumptions, e.g. that the matrix defining the quadratic loss satisfies a rank-restricted isometry property. In contrast, our analysis only assumes smoothness of the loss function.

The algorithm we propose is also related to Hazan’s algorithm (Hazan, 2008) for solving PSD problems,

which in turns relies on Frank-Wolfe algorithm (Frank & Wolfe, 1956) (see Clarkson (Clarkson, 2008)), as well as to the follow-up paper of (Jaggi & Sulovský, 2010), which applies Hazan’s algorithm for optimizing with trace-norm constraints. There are several important changes though. First, we tackle the problem directly and do not enforce neither PSDness of the matrix nor a bounded trace-norm. Second, our algorithm is “fully corrective”, that is, it extracts all the information from existing components before adding a new component. These differences between the approaches are analogous to the difference between Frank-Wolfe algorithm and fully corrective greedy selection, for minimizing over sparse vectors, as discussed in (Shalev-Shwartz et al., 2010). Finally, while each iteration of both methods involves approximately finding leading eigenvectors, in (Hazan, 2008) the quality of approximation should improve as the algorithm progresses while our algorithm can always rely on the same constant approximation factor.

2. The GECO algorithm

In this section we describe our algorithm, which we call Greedy Efficient Component Optimization (or GECO for short). Let $A \in \mathbb{R}^{m \times n}$ be a matrix, and without loss of generality assume that $m \leq n$. The SVD theorem states that A can be written as $A = \sum_{i=1}^m \lambda_i u_i v_i^T$, where u_1, \dots, u_m are members of $\mathcal{U} = \{u \in \mathbb{R}^m : \|u\| = 1\}$, v_1, \dots, v_m comes from $\mathcal{V} = \{v \in \mathbb{R}^n : \|v\| = 1\}$, and $\lambda_1, \dots, \lambda_m$ are scalars. To simplify the presentation, we assume that each real number is represented using a finite number of bits, therefore the sets \mathcal{U} and \mathcal{V} are finite sets.¹ It follows that we can also write A as $A = \sum_{(u,v) \in \mathcal{U} \times \mathcal{V}} \lambda_{u,v} uv^T$, where $\lambda \in \mathbb{R}^{|\mathcal{U} \times \mathcal{V}|}$ and we index the elements of λ using pairs $(u, v) \in \mathcal{U} \times \mathcal{V}$. Note that the representation of A using a vector λ is not unique, but from the SVD theorem, there is always a representation of A for which the number of non-zero elements of λ is at most m , i.e. $\|\lambda\|_0 \leq m$ where $\|\lambda\|_0 = |\{(u, v) : \lambda_{u,v} \neq 0\}|$. Furthermore, if $\text{rank}(A) \leq r$ then there is a representation of A using a vector λ for which $\|\lambda\|_0 \leq r$.

Given a (sparse) vector $\lambda \in \mathbb{R}^{|\mathcal{U} \times \mathcal{V}|}$ we define the cor-

¹This assumption greatly simplifies the presentation but is not very limiting since we do not impose any restriction on the amount of bits needed to represent a single real number. We note that the assumption is not necessary and can be waived by writing $A = \int_{(u,v) \in \mathcal{U} \times \mathcal{V}} uv^T d\lambda(u, v)$, where λ is a measure on $\mathcal{U} \times \mathcal{V}$, and from the SVD theorem, there is always a representation with λ which is non-zero on finitely many points.

Algorithm 1 GECO

```

1: Input: Convex-smooth function  $R : \mathbb{R}^{m \times n} \rightarrow \mathbb{R}$  ;
   rank constraint  $r$  ; tolerance  $\tau \in [0, 1/2]$ 
2: Initialize:  $U = \emptyset, V = \emptyset$ 
3: for  $i=1, \dots, r$  do
4:    $(u, v) = \text{ApproxSV}(\nabla R(UV^T), \tau)$ 
5:   Set  $U = [U, u]$  and  $V = [V, v]$ 
6:   Set  $B = \arg\min_{B \in \mathbb{R}^{i \times i}} R(UBV^T)$ 
7:   Calculate SVD:  $B = PDQ^T$ 
8:   Update:  $U = UPD, V = VQ$ 
9: end for
    
```

responding matrix to be

$$A(\lambda) = \sum_{(u,v) \in \mathcal{U} \times \mathcal{V}} \lambda_{u,v} uv^T.$$

Note that $A(\lambda)$ is a linear mapping. Given a function $R : \mathbb{R}^{m \times n} \rightarrow \mathbb{R}$, we define a function

$$f(\lambda) = R(A(\lambda)) = R\left(\sum_{(u,v) \in \mathcal{U} \times \mathcal{V}} \lambda_{u,v} uv^T\right).$$

It is easy to verify that if R is a convex function over $\mathbb{R}^{m \times n}$ then f is convex over $\mathbb{R}^{|\mathcal{U} \times \mathcal{V}|}$ (since f is a composition of R over a linear mapping). We can therefore reduce the problem given in Equation (1) to the problem

$$\min_{\lambda \in \mathbb{R}^{|\mathcal{U} \times \mathcal{V}|} : \|\lambda\|_0 \leq r} f(\lambda). \quad (2)$$

While the optimization problem given in Equation (2) is over an arbitrary large space, we next show that a forward greedy selection procedure can be implemented efficiently. The greedy algorithm starts with $\lambda = (0, \dots, 0)$. At each iteration, we first find the vectors (u, v) that maximizes the magnitude of the partial derivative of $f(\lambda)$ with respect to $\lambda_{u,v}$. Assuming that R is differentiable, and using the chain rule, we obtain:

$$\frac{\partial f(\lambda)}{\partial \lambda_{u,v}} = \langle \nabla R(A(\lambda)), uv^T \rangle = u^T \nabla R(A(\lambda)) v,$$

where $\nabla R(A(\lambda))$ is the $m \times n$ matrix of partial derivatives of R with respect to the elements of $A(\lambda)$. The vectors u, v that maximizes the magnitude of the above expression are the left and right singular vectors corresponding to the maximal singular value of $\nabla R(A(\lambda))$. Therefore, even though the number of elements in $\mathcal{U} \times \mathcal{V}$ is very large, we can still perform a greedy selection of one pair $(u, v) \in \mathcal{U} \times \mathcal{V}$ in an efficient way.

In some situations, even the calculation of the leading singular vectors might be too expensive. We therefore allow approximate maximization, and denote by

$\text{ApproxSV}(\nabla R(A(\lambda)), \tau)$ a procedure² which returns vectors for which

$$u^T \nabla R(A(\lambda)) v \geq (1 - \tau) \max_{p,q} p^T \nabla R(A(\lambda)) q.$$

Let U and V be matrices whose columns contain the vectors u and v we aggregated so far. The second step of each iteration of the algorithm sets λ to be the solution of the following optimization problem:

$$\min_{\lambda \in \mathbb{R}^{|\mathcal{U} \times \mathcal{V}|}} f(\lambda) \text{ s.t. } \text{supp}(\lambda) \subseteq \text{span}(U) \times \text{span}(V), \quad (3)$$

where $\text{supp}(\lambda) = \{(u, v) : \lambda_{u,v} \neq 0\}$, and $\text{span}(U), \text{span}(V)$ are the linear spans of the columns of U, V respectively.

We now describe how to solve Equation (3). Let s be the number of columns of U and V . Note that any vector $u \in \text{span}(U)$ can be written as Ub_u , where $b_u \in \mathbb{R}^s$, and similarly, any $v \in \text{span}(V)$ can be written as Vb_v . Therefore, if the support of λ is in $\text{span}(U) \times \text{span}(V)$ we have that $A(\lambda)$ can be written as

$$\begin{aligned} A(\lambda) &= \sum_{(u,v) \in \text{supp}(\lambda)} \lambda_{u,v} (Ub_u)(Vb_v)^T \\ &= U \left(\sum_{(u,v) \in \text{supp}(\lambda)} \lambda_{u,v} b_u b_v^T \right) V^T. \end{aligned}$$

Thus, any λ whose support is in $\text{span}(U) \times \text{span}(V)$ yields a matrix $B(\lambda) = \sum_{u,v} \lambda_{u,v} b_u b_v^T$. The SVD theorem tells us that the opposite direction is also true, namely, for any $B \in \mathbb{R}^{s \times s}$ there exists λ whose support is in $\text{span}(U) \times \text{span}(V)$ that generates B (and also UBV^T). Denote $\tilde{R}(B) = R(UBV^T)$, it follows that Equation (3) is equivalent to the following unconstrained optimization problem $\min_{B \in \mathbb{R}^{s \times s}} \tilde{R}(B)$. It is easy to verify that \tilde{R} is a convex function, and therefore can be minimized efficiently. Once we obtain the matrix B that minimizes $\tilde{R}(B)$ we can use its SVD to generate the corresponding λ .

In practice, we do not need to maintain λ at all, but only to maintain matrices U, V such that $A(\lambda) = UV^T$.

² An example of such a procedure is the power iteration method, which can implement ApproxSV in time $O(N \log(n)/\tau)$, where N is the number of non-zero elements of $\nabla R(A(\lambda))$. See Theorem 3.1 in (Kuczyński & Woźniakowski, 1992). Our analysis shows that the value of τ has a mild effect on the convergence of GECO, and one can even choose a constant value like $\tau = 1/2$. This is in contrast to (Hazan, 2008; Jaggi & Sulovský, 2010) which require the approximation parameter to decrease when the rank increases. Note also that the ApproxEV procedure described in (Hazan, 2008; Jaggi & Sulovský, 2010) requires an additive approximation, while we require a multiplicative approximation.

A summary of the pseudo-code is given in Algorithm 1. The runtime of the algorithm is as follows. Step 4 can be performed in time $O(N \log(n)/\tau)$, where N is the number of non zero elements of $\nabla R(UV^T)$, using the power method (see Footnote 2). Since our analysis (given in Section 3) allows τ to be a constant (e.g. $1/2$), this means that the runtime is $O(N \log(n))$. The runtime of Step 6 depends on the structure of the function R . We specify it when describing specific applications of GECCO in later sections. Finally, the runtime of Step 7 is at most r^3 , and step 8 takes $O(r^2(m+n))$.

2.1. Variants of GECCO

2.1.1. HOW TO CHOOSE (u, v)

GECCO chooses (u, v) to be the leading singular vectors, which are the maximizers of $u^T \nabla R(A) v$ over unit spheres of \mathbb{R}^m and \mathbb{R}^n . Our analysis in the next section guarantees that this choice yields a sufficient decrease of the objective function. However, there may be a pair (u, v) which leads to an even larger decrease in the objective value. Choosing such a direction can lead to improved performance. We note that our analysis in the next section still holds, as long as the direction we choose leads to a larger decrease in the objective value, relative to the increase we can get from using the leading singular vectors. In Section 6 we describe a method that finds better directions.

2.1.2. ADDITIONAL REPLACEMENT STEPS

Each iteration of GECCO increases the rank by 1. In many cases, it is possible to decrease the objective by replacing one of the components without increasing the rank. If we verify that this replacement step indeed decreases the objective (by simply evaluating the objective before and after the change), then the analysis we present in the next section remains valid. We now describe a simple way to perform a replacement. We start with finding a candidate pair (u, v) and perform steps 5–7 of GECCO. Then, we approximate the matrix B by zeroing its smallest singular value. Let \hat{B} denote this approximation. We next check if $R(U\hat{B}V^T)$ is strictly smaller than the previous objective value. If yes, we update U, V based on \hat{B} and obtain that the rank of UV^T has not been increased while the objective has been decreased. Otherwise, we update U, V based on B , thus increasing the rank, but our analysis tells us that we are guaranteed to sufficiently decrease the objective. If we restrict the algorithm to perform at most $O(1)$ attempted replacement steps between each rank-increasing iteration, then its runtime guarantee is only increased by an $O(1)$ factor, and all the convergence guarantees remain valid.

2.1.3. ADDING SCHATTEN NORM REGULARIZATION

In some situations, rank constraint is not enough for obtaining good generalization guarantees and one can consider objective functions $R(A)$ which contains additional regularization of the form $h(\lambda(A))$, where $\lambda(A)$ is the vector of singular values of A and h is a vector function such as $h(x) = \|x\|_p^2$. For example, if $p = 2$, this regularization term is equivalent to Frobenius norm regularization of A . In general, adding a convex regularization term should not pose any problem. A simple trick to do this is to orthonormalize the columns of U and V before Step 6. Therefore, for any B , the singular values of B equal the singular values of UBV^T . Thus, we can solve the problem in Step 6 more efficiently while regularizing B instead of the larger matrix UBV^T .

2.1.4. OPTIMIZING OVER DIAGONAL MATRICES B

Step 6 of GECCO involves solving a problem with i^2 variables, where $i \in \{1, \dots, r\}$. When r is small this is a reasonable computational effort. However, when r is large, Steps 6 – 7 can be expensive. For example, in matrix completion problems, the complexity of Step 6 can scale with r^6 . If runtime is important, it is possible to restrict B to be a diagonal matrix, or in other words, we only optimize over the coefficients of λ corresponding to U and V without changing the support of λ . Thus, in step 6 we solve a problem with i variables, and Step 7 is not needed. It is possible to verify that the analysis we give in the next section still holds for this variant.

3. Analysis

In this section we give a competitive analysis for GECCO. The first theorem shows that after performing r iterations of GECCO, its solution is not much worse than the solution of *all* matrices \bar{A} , whose trace norm³ is bounded by a function of r . The second theorem shows that with additional assumptions, we can be competitive with matrices whose rank is at most r . The proofs can be found in the long version of this paper.

To formally state the theorems we first need to define a smoothness property of the function f .

Definition 1 (smoothness) *We say that f is β -smooth if for any λ and $(u, v) \in \mathcal{U} \times \mathcal{V}$ we have*

$$f(\lambda + \eta e^{u,v}) \leq f(\lambda) + \eta \frac{\partial f(\lambda)}{\partial \lambda_{u,v}} + \frac{\beta \eta^2}{2},$$

³The trace norm of a matrix is the sum of its singular values.

where $\mathbf{e}^{u,v}$ is the all zeros vector except 1 in the coordinate corresponds to (u, v) . We say that R is β -smooth if the function $f(\lambda) = R(A(\lambda))$ is β -smooth.

Theorem 1 Fix some $\epsilon > 0$. Assume that GECO (or one of its variants) is run with a β -smooth function R , a rank constraint r , and a tolerance parameter $\tau \in [0, 1)$. Let A be its output matrix. Then, for all matrices \bar{A} with

$$\|\bar{A}\|_{\text{tr}}^2 \leq \frac{\epsilon(r+1)(1-\tau)^2}{2\beta}$$

we have that $R(A) \leq R(\bar{A}) + \epsilon$.

The previous theorem shows competitiveness with matrices of low trace norm. Our second theorem shows that with additional assumptions on the function f we can be competitive with matrices of low rank as well. We need the following definition.

Definition 2 (strong convexity) Let $I \subset \mathcal{U} \times \mathcal{V}$. We say that f is σ -strongly-convex over I if for any λ_1, λ_2 whose support⁴ is in I we have

$$f(\lambda_1) - f(\lambda_2) - \langle \nabla f(\lambda_2), \lambda_1 - \lambda_2 \rangle \geq \frac{\sigma}{2} \|\lambda_1 - \lambda_2\|_2^2.$$

We say that R is σ -strongly-convex over I if the function $f(\lambda) = R(A(\lambda))$ is σ -strongly-convex over I .

Theorem 2 Assume that the conditions of Theorem 1 hold. Then, for any \bar{A} such that

$$\text{rank}(\bar{A}) \leq \frac{\epsilon(r+1)(1-\tau)^2 \sigma}{4\beta R(0)}.$$

and such that R is σ -strongly-convex over the singular vectors of \bar{A} , we have that $R(A) \leq R(\bar{A}) + \epsilon$.

We discuss the implications of these theorems for several applications in the next sections.

4. Application I: Matrix Completion

Matrix completion is the problem of predicting the entries of some unknown target matrix $Y \in \mathbb{R}^{m \times n}$ based on a random subset of observed entries, $E \subset [m] \times [n]$. For example, in the famous Netflix problem, m represents the number of users, n represents the number of movies, and $Y_{i,j}$ is a rating user i gives to movie j . One approach for learning the matrix Y is to find a matrix A of low rank which approximately agrees with Y on the entries of E (in mean squared

error terms). Using the notation of this paper, we would like to minimize the objective

$$R(A) = \frac{1}{|E|} \sum_{(i,j) \in E} (A_{i,j} - Y_{i,j})^2,$$

over low rank matrices A .

We now specify GECO for this objective function. It is easy to verify that the (i, j) element of $\nabla R(A)$ is $2(A_{i,j} - Y_{i,j})$ if $(i, j) \in E$ and 0 otherwise. The number of non-zero elements of $\nabla R(A)$ is at most $|E|$, and therefore Step 4 of GECO can be implemented using the power method in time $O(|E| \log(n))$. Given matrices U, V , let u_i be the i 'th row of U and v_j be the j 'th row of V . We have that the (i, j) element of the matrix UBV^T can be written as $\langle \text{vec}(u_i^T v_j), \text{vec}(B) \rangle$, where vec of a matrix is the vector obtained by taking all the elements of the matrix column wise. We can therefore rewrite $R(UBV^T)$ as $\frac{1}{|E|} \sum_{(i,j) \in E} (\langle \text{vec}(u_i^T v_j), \text{vec}(B) \rangle - Y_{i,j})^2$, which makes Step 6 of GECO a vanilla least squares problem over at most r^2 variables. The runtime of this step is therefore bounded by $O(r^6 + |E|r^2)$.

4.1. Analysis

To apply our analysis for matrix completion we first bound the smoothness parameter.

Lemma 1 For matrix completion the smoothness parameter is at most $2/|E|$.

Proof For any u, v and i, j we can rewrite $(A_{i,j} + \eta u_i v_j - Y_{i,j})^2$ as

$$(A_{i,j} - Y_{i,j})^2 + 2(A_{i,j} - Y_{i,j}) \eta u_i v_j + \eta^2 u_i^2 v_j^2.$$

Taking expectation over $(i, j) \in E$ we obtain:

$$f(\lambda + \eta \mathbf{e}^{u,v}) \leq f(\lambda) + \eta \nabla_{u,v} f(\lambda) + \eta^2 \frac{1}{|E|} \sum_{(i,j) \in E} u_i^2 v_j^2.$$

Since $\sum_{(i,j) \in E} u_i^2 v_j^2 \leq \sum_i u_i^2 \sum_j v_j^2 = 1$, the proof follows. ■

Our general analysis therefore implies that for any \bar{A} , GECO can find a matrix with rank $r \leq O(\|\bar{A}\|_{\text{tr}}^2 / (\epsilon|E|))$, such that $R(A) \leq R(\bar{A}) + \epsilon$.

Let us now discuss the implications of this result for the number of observed entries required for predicting the entire entries of Y . Suppose that the entries E are sampled i.i.d. from some unknown distribution $D \in \mathbb{R}^{m \times n}$, $D_{i,j} \geq 0$ for all i, j and $\sum_{i,j} D_{i,j} = 1$.

⁴The support of λ is the set of (u, v) for which $\lambda_{u,v} \neq 0$.

Denote the generalization error of a matrix A by

$$F(A) = \sum_{i,j} D_{i,j}(A_{i,j} - Y_{i,j})^2.$$

Using generalization bounds for low rank matrices (e.g. (Srebro et al., 2005)), it is possible to show that for any matrix A of rank at most r we have that with high probability⁵

$$|F(A) - R(A)| \leq \tilde{O}(\sqrt{r(m+n)/|E|}).$$

Combining this with our analysis for GECCO, and optimizing ϵ , it is easy to derive the following:

Corollary 1 Fix some matrix \bar{A} . Then, GECCO can find a matrix A such that with high probability over the choice of the entries in E

$$F(A) \leq F(\bar{A}) + \tilde{O}\left(\left(\frac{\|\bar{A}\|_{\text{tr}}\sqrt{m+n}}{|E|}\right)^{2/3}\right).$$

Without loss of generality assume that $m \leq n$. It follows that if $\|\bar{A}\|_{\text{tr}}$ is order of \sqrt{mn} then order of $n^{3/2}$ entries are suffices to learn the matrix Y . This matches recent learning-theoretic guarantees for distribution-free learning with the trace norm (Shalev-Shwartz & Shamir, 2011).

5. Application II: Robust Low Rank Matrix Approximation

A very common problem in data analysis is finding a low-rank matrix A which approximates a given matrix Y , namely solving $\min_{A: \text{rank}(A) \leq r} d(A, Y)$, where d is some discrepancy measure. For simplicity, assume that $Y \in \mathbb{R}^{n \times n}$. When $d(A, V)$ is the normalized Frobenius norm $d(A, V) = \frac{1}{n^2} \sum_{i,j} (A_{i,j} - Y_{i,j})^2$, this problem can be solved efficiently via SVD. However, due to the use of the Frobenius norm, this procedure is well-known to be sensitive to outliers.

One way to make the procedure more robust is to replace the Frobenius norm by a less sensitive norm, such as the l_1 norm $d(A, V) = \frac{1}{n^2} \sum_{i,j} |A_{i,j} - Y_{i,j}|$ (see for instance (A. Baccini & Falguerolles, 1996), (Croux & Filzmoser, 1998), (Ke & Kanade, 2005)). Unfortunately, there are no known efficient algorithms to obtain the global optimum of this objective function, subject to a rank constraint on A . However, using

⁵To be more precise, this bound requires that the elements of A are bounded by a constant. But, since we can assume that the elements of Y are bounded by a constant, it is always possible to clip the elements of A to the range of the elements of Y without increasing $F(A)$.

our proposed algorithm, we can efficiently find a low-rank matrix which approximately minimizes $d(A, V)$. In particular, we can apply it to any convex discrepancy measure d , including robust ones such as the l_1 norm. The only technicality is that our algorithm requires d to be smooth, which is not true in the case of the l_1 norm. However, this can be easily alleviated by working with smoothed versions of the l_1 norm, which replace the absolute value by a smooth approximation. One example is a Huber loss, defined as $L(x) = x^2/2$ for $|x| \leq 1$, and $L(x) = |x| - 1/2$ otherwise.

Lemma 2 The smoothness parameter of $d(A, Y) = \frac{1}{n^2} \sum_{i,j} L(A_{i,j} - Y_{i,j})$, where L is the Huber loss, is at most $1/n^2$.

Proof It is easy to verify that the smoothness parameter of $L(x)$ is 1, since $L(x)$ is upper bounded by the parabola $x^2/2$, whose smoothness parameter is exactly 1. Therefore,

$$\begin{aligned} L(A_{i,j} + \eta u_i v_j - Y_{i,j}) &\leq L(A_{i,j} - Y_{i,j}) \\ &\quad + \eta L'(A_{i,j} - Y_{i,j}) u_i v_j + \frac{\eta^2}{2} u_i^2 v_j^2. \end{aligned}$$

Taking the average over all entries, this implies that

$$f(\lambda + \eta e^{u,v}) \leq f(\lambda) + \eta \nabla_{u,v} f(\lambda) + \frac{\eta^2}{n^2} \sum_{i,j} u_i^2 v_j^2.$$

Since the last term is at most η^2/n^2 , the result follows. ■

We therefore obtain:

Corollary 2 Let $d(A, Y)$ be the Huber loss discrepancy as defined in Lemma 2. Then, for any matrix \bar{A} , GECCO can find a matrix A with $d(A, Y) \leq d(\bar{A}, Y) + \epsilon$ and $\text{rank}(A) = O(\frac{\|\bar{A}\|_{\text{tr}}^2}{n^2 \epsilon})$.

6. Experiments

We evaluated GECCO for the problem of matrix completion by conducting experiments on three standard collaborative filtering datasets: MovieLens100K, MovieLens1M, and MovieLens10M⁶. The different datasets contain $10^5, 10^6, 10^7$ ratings of 943, 6040, 69878 users on 1682, 3706, 10677 movies, respectively. All the ranking are integers in $1 - 5$. We partitioned each data set into training and testing sets as done in (Jaggi & Sulovsky, 2010).

We implemented GECCO while applying two of the variants described in Section 2.1 as we explain in details

⁶Available through www.grouplens.org

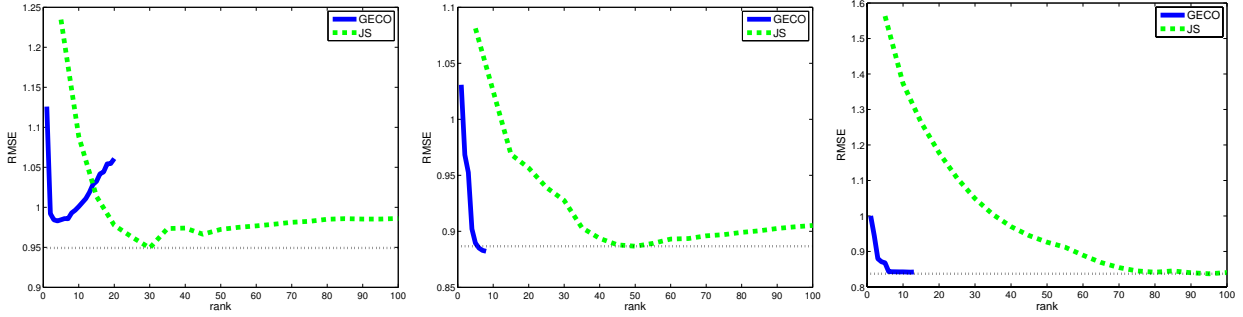


Figure 1. Root Mean Squared Error on the test set as a function of the rank. The horizontal line corresponds to the minimal error achieved by JS. Left: MovieLens100k, Middle: MovieLens1M, Right: MovieLens10M.

below. The first variant (see Section 2.1.1) tries to find update vectors (u', v') which leads to a larger decrease of the objective function relatively to the leading singular vectors (u, v) of the gradient matrix $\nabla R(A)$. Inspired by the proof of Theorem 1, we observe that the decrease of the objective function inversely depends on the smoothness of the scalar function $R(A + \eta uv^t)$. We therefore would like to find a pair which on one hand has a large correlation with $\nabla R(A)$ and on the other hand yields a smooth scalar function $R(A + \eta uv^t)$. The smoothness of $R(A + \eta uv^t)$ is analyzed in Lemma 1 and is shown to be at most $\frac{2}{|E|}$. Examining the proof lines more carefully, we see that for balanced vectors, i.e. $u_i = \pm \frac{1}{\sqrt{m}}, v_j = \pm \frac{1}{\sqrt{n}}$, we obtain a lower smoothness parameter of $\frac{2}{mn}$. Thus, a possible good update direction is to choose u, v that maximizes $u^T \nabla R(A) v$ over vectors of the form $u_i = \pm \frac{1}{\sqrt{m}}, v_j = \pm \frac{1}{\sqrt{n}}$. This is equivalent to maximizing $u^T \nabla R(A) v$ over the ℓ_∞ balls of \mathbb{R}^m and \mathbb{R}^n , which is unfortunately known to be NP-hard. Nevertheless, a simple alternate maximization approach is easy to implement and often works well. That is, fixing some u , we can see that $v = \text{sign}(u^T \nabla R(A)) / \sqrt{n}$ maximizes the objective, and similarly, fixing v we have that $u = \text{sign}(\nabla R(A) v) / \sqrt{m}$ is optimal. We therefore implement this alternate maximization at each step and find a candidate pair (u', v') . As described in section Section 2.1.1, we compare the decrease of loss as obtained by the leading singular vectors, (u, v) , and the candidate pair mentioned previously, (u', v') , and update using the pair which leads to a larger decrease of the objective. We remind the reader that although (u', v') are obtained heuristically, our implementation is still provably correct and our guarantees from Section 3 still hold.

In addition we performed the additional replacement steps as described in Section 2.1.2. For that purpose, let q be the number of times we try to perform additional replacement steps for each rank. Each replace-

ment attempt is done using the alternate maximization procedure described previously. After utilizing q attempts of additional replacement steps, we force an increase of the rank. In our experiments, we set $q = 20$. Finally, we implemented the ApproxSV procedure using 30 iterations of the power iteration method.

We compared GECO to a state-of-the-art method, recently proposed in (Jaggi & Sulovský, 2010), which we denote as the JS algorithm. JS, similarly to GECO, iteratively increases the rank by computing a direction that maximizes some objective function and performing a step in that direction. See more details in Section 1.1. In Figure 1, we plot the root mean squared error (RMSE) on the test set as a function of the rank. As can be seen, GECO decreases the error much faster than the JS algorithm. This is expected — see again the discussion in Section 1.1. We observe that GECO achieves slightly larger test error on the small data set, slightly smaller test error on the medium data set, and the same error on the large data set. On the small data set, GECO starts to overfit when the rank increases beyond 4. The JS algorithm avoids this overfitting by constraining the trace-norm, but also starts overfitting after around 30 iterations. On the other hand, on the medium data, the trace-norm constraint employed by the JS algorithm yields a higher estimation error, and GECO, which does not constrain the trace-norm, achieves a smaller error. In any case, GECO achieves very good results while using a rank of at most 10.

7. Discussion

GECO is an efficient greedy approach for minimizing a convex function subject to a rank constraint. One of the main advantages of GECO is that each of its iterations involves running few (precisely, $O(\log(n))$) iterations of the power method, and therefore GECO scales to large matrices. In future work we intend to

apply GECCO to additional applications such as multiclass classification and learning fast quadratic classifiers.

Acknowledgements

This work emerged from fruitful discussions with Tomer Baba, Barak Cohen, Harel Livyatan, and Oded Schwarz. The work is supported by the Israeli Science Foundation grant number 598-10.

References

- A. Baccini, P. Besse and Falguierolles, A. A l1-norm pca and a heuristic approach. In E. Diday, Y. Lechevalier and Opitz, P. (eds.), *Ordinal and Symbolic Data Analysis*, pp. 359–368. Springer, 1996.
- Amit, Yonatan, Fink, Michael, Srebro, Nathan, and Ullman, Shimon. Uncovering shared structures in multiclass classification. In *International Conference on Machine Learning*, 2007.
- Cai, J.F., Candes, E.J., and Shen, Z. A singular value thresholding algorithm for matrix completion. *preprint*, 2008.
- Candes, E.J. and Plan, Y. Matrix completion with noise. *Proceedings of the IEEE*, 98(6):925–936, 2010. ISSN 0018-9219.
- Candès, E.J. and Recht, B. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9(6):717–772, 2009. ISSN 1615-3375.
- Clarkson, K.L. Coresets, sparse greedy approximation, and the Frank-Wolfe algorithm. In *Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, pp. 922–931, 2008.
- Croux, C. and Filzmoser, P. Robust factorization of a data matrix. In *COMPASTAT, Proceedings in Computational Statistics*, 1998.
- Dai, W. and Milenkovic, O. Subspace pursuit for compressive sensing: Closing the gap between performance and complexity, 2008.
- Davis, G., Mallat, S., and Avellaneda, M. Greedy adaptive approximation. *Journal of Constructive Approximation*, 13:57–98, 1997.
- Fazel, M., Hindi, H., and Boyd, S.P. A rank minimization heuristic with application to minimum order system approximation. In *American Control Conference, 2001. Proceedings of the 2001*, volume 6, pp. 4734–4739. IEEE, 2002. ISBN 0780364953.
- Frank, M. and Wolfe, P. An algorithm for quadratic programming. *Naval Res. Logist. Quart.*, 3:95–110, 1956.
- Hazan, Elad. Sparse approximate solutions to semidefinite programs. In *Proceedings of the 8th Latin American conference on Theoretical informatics*, pp. 306–316, 2008.
- Jaggi, M. and Sulovský, M. A simple algorithm for nuclear norm regularized problems. In *ICML*, 2010.
- Ke, Q. and Kanade, T. Robust l1 norm factorization in the presence of outliers and missing data by alternative convex programming. In *CVPR*, 2005.
- Keshavan, R.H. and Oh, S. Optspace: A gradient descent algorithm on the grassman manifold for matrix completion. *Arxiv preprint arXiv:0910.5260 v2*, 2009.
- Keshavan, R.H., Montanari, A., and Oh, S. Matrix completion from a few entries. *Information Theory, IEEE Transactions on*, 56(6):2980–2998, 2010. ISSN 0018-9448.
- Koren, Yehuda, Bell, Robert M., and Volinsky, Chris. Matrix factorization techniques for recommender systems. *IEEE Computer*, 42(8):30–37, 2009.
- Kuczyński, J. and Woźniakowski, H. Estimating the largest eigenvalue by the power and Lanczos algorithms with a random start. *SIAM journal on matrix analysis and applications*, 13:1094, 1992.
- Lee, K. and Bresler, Y. Admira: Atomic decomposition for minimum rank approximation. *Information Theory, IEEE Transactions on*, 56(9):4402–4416, 2010. ISSN 0018-9448.
- Mallat, S. and Zhang, Z. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, 41:3397–3415, 1993.
- Natarajan, B. Sparse approximate solutions to linear systems. *SIAM J. Computing*, 25(2):227–234, 1995.
- Needell, D. and Tropp, J.A. CoSaMP: Iterative signal recovery from incomplete and inaccurate samples. *Applied and Computational Harmonic Analysis*, 26(3):301–321, 2009. ISSN 1063-5203.
- Pati, YC, Rezaifar, R., and Krishnaprasad, PS. Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In *Signals, Systems and Computers, 1993. 1993 Conference Record of The Twenty-Seventh Asilomar Conference on*, pp. 40–44. IEEE, 2002. ISBN 0818641207.
- Recht, B. A simpler approach to matrix completion. *JMLR*, to appear.
- Recht, B., Fazel, M., and Parrilo, P.A. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *preprint*, 2007.
- Shalev-Shwartz, Shai and Shamir, Ohad. Collaborative filtering with the trace norm: Learning, bounding, and transducing. In *COLT*, 2011.
- Shalev-Shwartz, Shai, Zhang, Tong, and Srebro, Nathan. Trading accuracy for sparsity in optimization problems with sparsity constraints. *SIAM Journal on Optimization*, 20:2807–2832, 2010.
- Srebro, N., Alon, N., and Jaakkola, T. Generalization error bounds for collaborative prediction with low-rank matrices. *Advances In Neural Information Processing Systems*, 17, 2005.

A. Proofs

A.1. Proof of Theorem 1

To prove the theorem we need the following key lemma, which generalizes a result given in (Shalev-Shwartz et al., 2010).

Lemma 3 *Assume that f is β -smooth. Let I, \bar{I} be two subsets of $\mathcal{U} \times \mathcal{V}$. Let λ be a minimizer of $f(\lambda)$ over all vectors with support in I and let $\bar{\lambda}$ be a vector supported on \bar{I} . Assume that $f(\lambda) > f(\bar{\lambda})$, denote $s = \|\bar{\lambda}\|_1$, and let $\tau \in [0, 1]$. Let $(u, v) = \text{ApproxSV}(\nabla R(A(\lambda)), \epsilon)$. Then, there exists η such that*

$$f(\lambda) - f(\lambda + \eta \mathbf{e}^{u,v}) \geq \frac{(f(\lambda) - f(\bar{\lambda}))^2 (1 - \tau)^2}{2\beta s^2}.$$

Proof

Without loss of generality assume that $\bar{\lambda} \geq 0$ (if $\bar{\lambda}_{p,q} < 0$ for some (p, q) we can set $\bar{\lambda}_{-p,q} = -\bar{\lambda}_{p,q}$ and $\bar{\lambda}_{p,q} = 0$ without effecting the objective) and assume that $u^t \nabla(R(A(\lambda)))v \leq 0$ (if this does not hold, let $u = -u$). For any (p, q) , let $\nabla_{p,q} = p^t \nabla R(A(\lambda))q$ be the partial derivative of f w.r.t. coordinate (p, q) at λ and denote

$$Q_{p,q}(\eta) = f(\lambda) + \eta \nabla_{p,q} + \frac{\beta \eta^2}{2}.$$

Note that the definition of (u, v) and our assumption above implies that

$$-\nabla_{u,v} = |\nabla_{u,v}| \geq (1 - \tau) \max_{p,q} |\nabla_{p,q}|,$$

which gives

$$\nabla_{u,v} \leq (\tau - 1) \max_{p,q} |\nabla_{p,q}| = (1 - \tau) \min_{p,q} \nabla_{p,q}.$$

Therefore, for all $\eta \geq 0$ we have

$$Q_{u,v}(\eta) \leq f(\lambda) + (1 - \tau) \eta \min_{p,q} \nabla_{p,q} + \frac{\beta \eta^2}{2}.$$

In addition, the smoothness assumption tells us that for all η we have $f(\lambda + \eta \mathbf{e}^{u,v}) \leq Q_{u,v}(\eta)$. Thus, for any $\eta \geq 0$ we have

$$\min_a f(\lambda + a \mathbf{e}^{u,v}) \leq f(\lambda + \eta \mathbf{e}^{u,v}) \leq Q_{u,v}(\eta)$$

Combining the above we get

$$\min_a f(\lambda + a \mathbf{e}^{u,v}) \leq f(\lambda) + (1 - \tau) \eta \min_{(p,q) \in \bar{I} \setminus I} \nabla_{p,q} + \frac{\beta \eta^2}{2}.$$

Multiplying both sides by s and noting that

$$s \min_{(p,q) \in \bar{I} \setminus I} \nabla_{p,q} \leq \sum_{(p,q) \in \bar{I} \setminus I} \bar{\lambda}_{p,q} \nabla_{p,q}$$

we get that

$$\begin{aligned} & s \min_a f(\lambda + a \mathbf{e}^{u,v}) \\ & \leq s f(\lambda) + (1 - \tau) \eta \sum_{(p,q) \in \bar{I} \setminus I} \bar{\lambda}_{p,q} \nabla_{p,q} + s \frac{\beta \eta^2}{2}. \end{aligned}$$

Since λ is a minimizer of f over I we have that $\nabla_{p,q} = 0$ for $(p, q) \in I$. Combining this with the fact that λ is supported on I and $\bar{\lambda}$ is supported on \bar{I} we obtain that

$$\sum_{(p,q) \in \bar{I} \setminus I} \bar{\lambda}_{p,q} \nabla_{p,q} = \langle \bar{\lambda}, \nabla f(\lambda) \rangle = \langle \bar{\lambda} - \lambda, \nabla f(\lambda) \rangle.$$

From the convexity of f we know that $\langle \bar{\lambda} - \lambda, \nabla f(\lambda) \rangle \leq f(\bar{\lambda}) - f(\lambda)$. Combining all the above we obtain

$$s \min_a f(\lambda + a \mathbf{e}^{u,v}) \leq s f(\lambda) + (1 - \tau) \eta (f(\bar{\lambda}) - f(\lambda)) + s \frac{\beta \eta^2}{2}.$$

This holds for all $\eta \geq 0$ and in particular for $\eta = (f(\lambda) - f(\bar{\lambda}))(1 - \tau)/(s\beta)$ (which is positive). Thus,

$$s \min_a f(\lambda + a \mathbf{e}^{u,v}) \leq s f(\lambda) - \frac{(f(\lambda) - f(\bar{\lambda}))^2 (1 - \tau)^2}{2\beta s}.$$

Rearranging the above concludes our proof. \blacksquare

Equipped with the above lemma we are ready to prove Theorem 1.

Fix some \bar{A} and let $\bar{\lambda}$ be the vector of its singular values. Thus, $\|\bar{\lambda}\|_1 = \|\bar{A}\|_{\text{tr}}$ and $f(\bar{\lambda}) = R(\bar{A})$. For each iteration i , denote $\epsilon_i = f(\lambda^{(i)}) - f(\bar{\lambda})$, where $\lambda^{(i)}$ is the value of λ at the beginning of iteration i of GECCO, before we increase the rank to be i . Note that all the operations we perform in GECCO or one of its variants guarantee that the loss is monotonically non-increasing. Therefore, if $\epsilon_i \leq \epsilon$ we are done. In addition, whenever we increase the rank by 1, the definition of the update implies that $f(\lambda^{(i+1)}) \leq \min_{\eta} f(\lambda^{(i)} + \eta \mathbf{e}^{u,v})$, where $(u, v) = \text{ApproxSV}(R(A(\lambda^{(i)})), \tau)$. Lemma 3 implies that

$$\epsilon_i - \epsilon_{i+1} = f(\lambda^{(i)}) - f(\lambda^{(i+1)}) \geq \frac{\epsilon_i^2 (1 - \tau)^2}{2\beta \|\bar{A}\|_{\text{tr}}^2}. \quad (4)$$

Using Lemma B.2 from (Shalev-Shwartz et al., 2010), the above implies that for $i \geq 2\beta \|\bar{A}\|_{\text{tr}}^2 / (\epsilon(1 - \tau)^2)$ we have that $\epsilon_i \leq \epsilon$. We obtain that if $\|\bar{A}\|_{\text{tr}}^2 \leq \epsilon(r + 1)(1 - \tau)^2 / (2\beta)$ then $\epsilon_{r+1} \leq \epsilon$, which concludes the proof of Theorem 1. \blacksquare

A.2. Proof of Theorem 2

Let $\bar{\lambda}$ be the vector obtained from the SVD of \bar{A} , that is, $\bar{A} = A(\bar{\lambda})$ and $\|\bar{\lambda}\|_0 = \text{rank}(\bar{A})$. Note that

f is σ -strongly-convex over the support of $\bar{\lambda}$. Using Lemma 2.2 of (Shalev-Shwartz et al., 2010) we know that $\|\bar{\lambda}\|_1^2 \leq \frac{2\|\bar{\lambda}\|_0 f(0)}{\sigma}$. But, since $\|\bar{A}\|_{\text{tr}} = \|\bar{\lambda}\|_1$, $\text{rank}(\bar{A}) = \|\bar{\lambda}\|_0$, and $f(0) = R(0)$, we get

$$\|\bar{A}\|_{\text{tr}}^2 \leq \frac{2\text{rank}(\bar{A}) R(0)}{\sigma}.$$

The proof follows from the above using Theorem 1. ■