A Simple Algorithm for Nuclear Norm Regularized Problems

Martin Jaggi JAGGI@INF.ETHZ.CH
Marek Sulovský SMAREK@INF.ETHZ.CH
Institute of Theoretical Computer Science, ETH Zurich, CH-8092 Zurich, Switzerland

Abstract

Optimization problems with a nuclear norm regularization, such as e.g. low norm matrix factorizations, have seen many applications recently. We propose a new approximation algorithm building upon the recent sparse approximate SDP solver of (Hazan, 2008). The experimental efficiency of our method is demonstrated on large matrix completion problems such as the Netflix dataset. The algorithm comes with strong convergence guarantees, and can be interpreted as a first theoretically justified variant of Simon-Funk-type SVD heuristics. The method is free of tuning parameters, and very easy to parallelize.

1. Introduction

This paper considers large scale convex optimization problems with a nuclear norm regularization, as for instance low norm matrix factorizations. Such formulations occur in many machine learning and compressed sensing applications such as dimensionality reduction, matrix classification, multi-task learning and matrix completion (Srebro et al., 2004; Candes & Tao, 2009). Matrix completion by using matrix factorizations of either *low rank* or *low norm* has gained a lot of attention in the area of recommender systems (Koren et al., 2009) with the recently ended *Netflix Prize* competition.

Our new method builds upon the recent first-order optimization scheme for semi-definite programs (SDP) of (Hazan, 2008) and has strong convergence guarantees.

We consider the following convex optimization problems over matrices:

$$\min_{X\in\mathbb{R}^{n\times m}} f(X) + \mu\|X\|_* \qquad (1)$$

Appearing in *Proceedings of the 27th International Conference on Machine Learning*, Haifa, Israel, 2010. Copyright 2010 by the author(s)/owner(s).

and the corresponding constrained variant

$$\min_{X\in\mathbb{R}^{n\times m},\, \|X\|_*\leq\frac{t}{2}} f(X) \qquad (2)$$

where $f(X)$ is any differentiable convex function (usually called the loss function), $\|.\|_*$ is the *nuclear norm* of a matrix, also known as the *trace norm* (sum of the singular values, or ℓ_1-norm of the spectrum). Here $\mu > 0$ and $t > 0$ respectively are given parameters, usually called the *regularization parameter*.

When choosing $f(X) := \|\mathcal{A}(X) - b\|_2^2$ for some linear map $\mathcal{A} : \mathbb{R}^{n\times m} \to \mathbb{R}^p$, the above formulation (1) is the matrix generalization of the problem $\min_{x\in\mathbb{R}^n} \|Ax - b\|_2^2 + \mu\|x\|_1$, which is the important *ℓ_1-regularized least squares problem*, also known as the basis pursuit de-noising problem in compressed sensing literature. The analogue vector variant of (2) is the *Lasso* problem (Tibshirani, 1996) which is $\min_{x\in\mathbb{R}^n} \left\{\|Ax - b\|_2^2 \;\middle|\; \|x\|_1 \leq t\right\}$.

Recently (Toh & Yun, 2009; Liu et al., 2009) and (Ji & Ye, 2009) independently proposed algorithms that obtain an ϵ-accurate solution to (1) in $O(1/\sqrt{\epsilon})$ steps, by improving the algorithm of (Cai et al., 2008). More recently also (Mazumder et al., 2009) and (Ma et al., 2009) proposed algorithms in this line of so called *singular value thresholding* methods, but cannot guarantee a convergence speed. Each step of all those algorithms requires the computation of the singular value decomposition (SVD) of a matrix of the same size as the solution matrix, which is expensive even with the currently available fast methods such as PROPACK. Both (Toh & Yun, 2009) and (Ji & Ye, 2009) show that the primal error of their algorithm is smaller than ϵ after $O(1/\sqrt{\epsilon})$ steps, using an analysis in the spirit of (Nesterov, 1983).

We present a much simpler algorithm to solve problems of the form (2), which does not need any SVD computations. We achieve this by transforming the problem to a convex optimization problem on positive semi-definite matrices, and then using the approximate SDP solver of Hazan (2008). Hazan's algorithm

can be interpreted as the generalization of the *coreset* approach to problems on symmetric matrices. The algorithm has a strong approximation guarantee, in the sense of obtaining ϵ-small primal-dual error (not only small primal error). With the resulting approximate solution X, our algorithm also gives a matrix factorization $X = UV^T$ of rank $O(1/\epsilon)$ (with the desired bounded nuclear norm). Compared to (Nesterov, 1983), a moderately increased number of steps is needed in total, $O(1/\epsilon)$, which represents the price for the very severe simplification in each individual step of our method on the one hand and the improved (low) rank on the other hand.

We demonstrate that our new algorithm on standard datasets improves the state of the art nuclear norm methods, and scales to large problems such as matrix factorizations on the Netflix dataset. Furthermore, the algorithm is easy to implement and parallelize, as it only uses the power method (or Lanczos steps) to approximate the largest eigenvalue of a matrix.

Our method can also be interpreted as a modified, theoretically justified variant of Simon Funk's popular SVD heuristic (Webb, 2006), making it suitable for low norm matrix factorization. To our knowledge this is the first guaranteed convergence result for this class of SVD-like gradient descent algorithms. Unlike most other comparable algorithms, our general method is free of tuning parameters (apart from the regularization parameter).

Notation. For arbitrary real matrices, the standard *inner product* is defined as $\langle A, B\rangle := Tr(A^T B)$, and the (squared) *Frobenius matrix norm* $\|A\|_{Fro}^2 := \langle A, A\rangle$ is the sum of all squared entries of the matrix. By $\mathbb{S}^{d\times d}$ we denote the set of *symmetric* $d \times d$ matrices. $A \in \mathbb{R}^{d\times d}$ is called *positive semi-definite* (PSD), written as $A \succeq 0$, iff $v^T Av \geq 0\ \forall v \in \mathbb{R}^d$.

2. Hazan's Algorithm

Our main ingredient is the following simple gradient-descent type algorithm of (Hazan, 2008), to obtain sparse solutions to any convex optimization problems of the form

$$\min_{Z\in\mathcal{S}} f(Z)\ , \tag{3}$$

where $\mathcal{S} := \left\{Z \in \mathbb{S}^{d\times d} \;\middle|\; Z \succeq 0,\ Tr(Z) = 1\right\}$ is the set of PSD matrices of unit trace. The set $\mathcal{S}$ is sometimes called *Spectrahedron* and is a generalization of the unit simplex to the space of symmetric matrices. The algorithm guarantees ε-small primal-dual error after at most $O\left(\frac{1}{\varepsilon}\right)$ iterations, where each iteration only involves the calculation of a single approximate eigenvector of a matrix $M \in \mathbb{S}^{d\times d}$. In practice for example Lanczos or the power method can be used.

Algorithm 1 Hazan's Algorithm

Input: Convex f with curvature constant C_f, target accuracy ε.
Initialize $Z^{(1)} := v_0 v_0^T$ for arbitrary unit vector v_0.
for $k = 1$ **to** $\left\lceil \frac{4C_f}{\varepsilon} \right\rceil$ **do**
  Compute $v_k := \textsc{ApproxEV}\left(-\nabla f(Z^{(k)}), \frac{C_f}{k^2}\right)$.
  Let $\alpha_k := \frac{1}{k}$.
  Set $Z^{(k+1)} := Z^{(k)} + \alpha_k \left(v_k v_k^T - Z^{(k)}\right)$.
end for

Here $\textsc{ApproxEV}(M, \varepsilon')$ is a sub-routine that delivers an approximate largest eigenvector to a matrix M with the desired accuracy ε', meaning a unit length vector v such that $v^T Mv \geq \lambda_{\max}(M) - \varepsilon'$. Note that as our convex function f takes a symmetric matrix Z as an argument, its gradient $\nabla f(Z)$ is a symmetric matrix.

The actual running time for a given convex function $f : \mathbb{S}^{d\times d} \to \mathbb{R}$ depends on its *curvature constant* C_f (also called the modulus of convexity) defined as

$$C_f := \sup_{\substack{Z,V\in\mathcal{S},\alpha\in\mathbb{R},\\ Z'=Z+\alpha(V-Z)}} \tfrac{1}{\alpha^2}\left(f(Z') - f(Z) + \langle Z' - Z, \nabla f(Z)\rangle\right),$$

which turns out to be small for many applications[1].

The algorithm can be seen as a matrix generalization of the sparse greedy approximation algorithm of (Clarkson, 2008) for vectors in the unit simplex, called the *coreset* method, which has seen many successful applications in a variety of areas ranging from clustering to support vector machine training, smallest enclosing ball/ellipsoid, boosting and others. Here *sparsity* just gets replaced by *low rank*. The same Algorithm 1 with a well-crafted function f can also be used to solve arbitrary SDPs in feasibility form.

3. Transformation to convex problems

3.1. Motivation: Formulating Matrix Factorizations as Convex Optimization Problems

Approximate matrix factorization refers to the setting of approximating a given matrix $Y \in \mathbb{R}^{n\times m}$ (typically given only partially) by a product $X = UV^T$, under an additional low rank or low norm constraint, such that some error function $f(X)$ is minimized. Most of the currently known gradient-descent-type algorithms for matrix factorization suffer from the following problem:

[1] An overview of values of C_f for several classes of functions f can be found in (Clarkson, 2008)

Even if the loss-function $f(X)$ is convex in X, the same function expressed as a function $f(UV^T)$ of both the factor variables U and V usually becomes a non-convex problem (consider for example $U, V \in \mathbb{R}^{1\times 1}$ together with the identity function $f(x) = x$). Therefore many of the popular methods such as for example (Rennie & Srebro, 2005; Lin, 2007) can get stuck in local minima and so are neither theoretically nor practically well justified, see also (DeCoste, 2006).

These shortcomings can be overcome as follows: One can equivalently transform any low-norm matrix factorization problem (which is usually *not* convex in its two factor variables) into an optimization problem over symmetric matrices: For any function f on $\mathbb{R}^{n\times m}$, the optimization problem

$$\begin{array}{ll} \min\limits_{\substack{U\in\mathbb{R}^{n\times r}\\ V\in\mathbb{R}^{m\times r}}} & f(UV^T) \\ s.t. & \|U\|_{Fro}^2 + \|V\|_{Fro}^2 = t \end{array} \tag{4}$$

is equivalent to

$$\begin{array}{ll} \min\limits_{\substack{Z\in\mathbb{S}^{(n+m)\times(n+m)}\\ rank(Z)\le r}} & \hat{f}(Z) \\ s.t. & Z \succeq 0,\ Tr(Z) = t. \end{array} \tag{5}$$

where "equivalent" means that for any feasible solution of each problem, there is a feasible solution of the other problem with the same objective value. Here $\hat{f}$ is the same function as f, just acting on the corresponding off-diagonal rectangle of the larger, symmetric matrices $Z \in \mathbb{S}^{(n+m)\times(n+m)}$. Formally, $\hat{f}(Z) = \hat{f}\left(\begin{pmatrix} Z_1 & Z_2 \\ Z_2^T & Z_3 \end{pmatrix}\right) := f(Z_2)$. The equivalence holds simply because every PSD matrix Z can be written as some product $Z = \begin{pmatrix} U \\ V \end{pmatrix}(U^T\ V^T) = \begin{pmatrix} UU^T & UV^T \\ VU^T & VV^T \end{pmatrix}$, for $U \in \mathbb{R}^{n\times r}$ and $V \in \mathbb{R}^{m\times r}$, when $r \ge rank(Z)$. On the other hand, of course any arbitrary valued matrices U, V give rise to a PSD matrix Z of this form. Furthermore $Tr(Z) = Tr(UU^T) + Tr(VV^T) = \|U\|_{Fro}^2 + \|V\|_{Fro}^2$ holds by definition.

The main advantage of this reformulation is that if the rank $r := n+m$ is not restricted, the new problem (5) is now a *convex* problem over a nice well-studied convex domain (the cone of PSD matrices of fixed trace), whereas the original formulation (4) is usually not convex in both arguments U and V.

3.2. Nuclear Norm Regularized Problems

In the same spirit, we obtain that any nuclear norm regularized problem of the form (2) is equivalent to the convex problem given by the following Corollary 2.

Lemma 1. *For any non-zero matrix $X \in \mathbb{R}^{n\times m}$ and $t \in \mathbb{R}$:*

$$\|X\|_* \le \frac{t}{2}$$

iff

$$\exists \ \text{symmetric matrices } A \in \mathbb{S}^{n\times n}, B \in \mathbb{S}^{m\times m}$$
$$s.t. \quad \begin{pmatrix} A & X \\ X^T & B \end{pmatrix} \succeq 0 \quad \text{and} \quad Tr(A) + Tr(B) = t\ .$$

Proof. This is a slight variation of the argument of (Fazel et al., 2001; Srebro et al., 2004).
$\boxed{\Rightarrow}$ From the characterization $\|X\|_* = \min_{UV^T=X} \frac{1}{2}(\|U\|_{Fro}^2 + \|V\|_{Fro}^2)$ we get that $\exists\ U, V,\ UV^T = X$ s.t. $\|U\|_{Fro}^2 + \|V\|_{Fro}^2 = Tr(UU^T) + Tr(VV^T) \le t$, or in other words we have found a matrix $\begin{pmatrix} UU^T & X \\ X^T & VV^T \end{pmatrix}$ of trace say $s \le t$. If $s < t$, we add $(t-s)$ to the top-left entry of A, i.e. we add to A the PSD matrix $e_1 e_1^T$ (which again gives a PSD matrix).
$\boxed{\Leftarrow}$ As the matrix is symmetric and PSD, it can be (Cholesky) factorized to $(U;V)(U;V)^T$ s.t. $UV^T = X$ and $t = Tr(UU^T) + Tr(VV^T) = \|U\|_{Fro}^2 + \|V\|_{Fro}^2$, therefore $\|X\|_* \le \frac{t}{2}$. □

Corollary 2. *Any nuclear norm regularized problem of the form (2) is equivalent to*

$$\min_{\substack{Z\in\mathbb{S}^{(n+m)\times(n+m)}\\ Z\succeq 0,\ Tr(Z)=t}} \hat{f}(Z)\ . \tag{6}$$

Note that both transformations in this section are equivalent formulations and not just relaxations. As already mentioned above, an explicit factorization of any feasible solution to (5) or (6) — if needed — can always be directly obtained since $Z \succeq 0$. Alternatively, algorithms for solving the transformed problem (5) or (6) can directly maintain the approximate solution Z in a factorized representation, as achieved for example by Hazan's algorithm.

3.3. Two Variants of Regularization

The two original problem formulations (1) and (2) are very closely related, and used interchangeably in many applications: If X^* is an optimal solution to trade-off variant (1), then the same solution is also optimal for (2) when using the value $\|X^*\|_*$ as the norm constraint. On the other hand (1) is just the Lagrangian version of (2), with μ being the Lagrange multiplyer belonging to the single constraint. This is the same change in formulation as when going from regularized least squares formulation (the vector analogue of (1)), to the Lasso problem corresponding to (2) and vice versa.

4. Solving Nuclear Norm Regularized Problems

By the equivalent reformulation of the previous section, as in Corollary 2, we can now solve both general nuclear norm regularized problems and low norm matrix factorizations by using Hazan's algorithm.

Algorithm 2 Nuclear Norm Regularized Solver

1. Consider the transformed problem for $\hat{f}$ given by Corollary 2.
2. Adjust the function $\hat{f}$ by re-scaling all matrix entries by $\frac{1}{t}$.
3. Run Algorithm 1 for $\hat{f}(Z)$.

The following theorem shows that Algorithm 2 runs in time *linear* in the number N_f of non-zero entries of the gradient ∇f. This makes it very attractive in particular for recommender systems applications and matrix completion, where ∇f is a sparse matrix (same sparsity pattern as the observed entries).

Theorem 3. *Algorithm 2 obtains an approximate solution of primal-dual error $\leq \varepsilon$ for problems of the form (2) after at most $\left\lceil \frac{4C_f}{\varepsilon} \right\rceil$ many steps (or in other words approximate eigenvector computations).*
In the k-th call of ApproxEV(), *it is sufficient to perform $O(k)$ iterations of Lanczos method. Then the overall running time is $O\left(\frac{N_f}{\varepsilon^2}\right)$, or equivalently $O\left(\frac{1}{\varepsilon^2}\right)$ many sparse matrix-vector multiplications.*

Proof. We use Corollary 2 and then rescale all matrix entries by $\frac{1}{t}$. Then the running time of follows from Theorem 2 of (Hazan, 2008). □

The fact that each iteration of our algorithm is computationally very cheap — consisting only of the computation of an approximate eigenvector — strongly contrasts the existing "singular value thresholding" methods, which *in each step* need to compute an entire SVD. Such a single incomplete SVD computation (first k singular vectors) amounts to the same computational cost as an entire run of our algorithm (for k steps). Furthermore, those existing methods which come with a theoretical guarantee, i.e. (Toh & Yun, 2009; Liu et al., 2009; Ji & Ye, 2009; Ma et al., 2009), in their analysis assume that all SVDs used during the algorithm are exact, which is not feasible in practice. By contrast, our analysis is rigorous even if the used eigenvectors are only ε'-approximate.

Another nice property of Hazan's method is that the returned solution is guaranteed to be simultaneously of low rank (k after k steps), and that by incrementally adding the rank-1 matrices $v_k v_k^T$, the algorithm automatically maintains a matrix factorization of the approximate solution.

Also, Hazan's algorithm is designed to automatically stay within the feasible region $\mathcal{S}$, where most of the existing approximate SDP-like methods do need a projection step to get back to the feasible region (as e.g. (Lin, 2007; Liu et al., 2009)), which makes both their theoretical analysis and implementation much more complicated.

4.1. The Structure of the Eigenvalue Problem

For the actual computation of the approximate largest eigenvector in ApproxEV $\left(-\nabla\hat{f}(Z^{(k)}), \frac{C_f}{k^2}\right)$, either Lanczos method or the power method (as in PageRank, see e.g. (Berkhin, 2005)) can be used. Both methods are known to scale well to very large problems and can be parallelized easily, as each iteration consists of just one matrix-vector multiplication. However, we have to be careful that we obtain the eigenvector for the *largest* eigenvalue which is not necessarily the principal one (largest in absolute value). In that case the spectrum can be shifted by adding an appropriate constant to the diagonal of the matrix. (Hazan, 2008) made use of the fact that Lanczos method, which is theoretically better understood, provably obtains the required approximation quality in a bounded number of steps if the matrix is PSD (Arora et al., 2005).

For arbitrary loss function f, the gradient $-\nabla\hat{f}(Z)$, which is the matrix whose largest eigenvector we have to compute in the algorithm, is always a symmetric matrix of the block form $\nabla\hat{f}(Z) = \begin{pmatrix} 0 & G \\ G^T & 0 \end{pmatrix}$ for $G = \nabla f(Z_2)$, when $Z = \begin{pmatrix} Z_1 & Z_2 \\ Z_2^T & Z_3 \end{pmatrix}$. In other words $\nabla\hat{f}(Z)$ is the adjacency matrix of a weighted *bipartite graph*. One vertex class corresponds to the n rows of the original matrix Z_2 (*users* in recommender systems), the other class corresponds to the m columns (*items* or *movies*). The spectrum of $\nabla\hat{f}$ is always *symmetric*: Whenever $\begin{pmatrix} v \\ w \end{pmatrix}$ is an eigenvector for some eigenvalue λ, then $\begin{pmatrix} v \\ -w \end{pmatrix}$ is an eigenvector for $-\lambda$.

Hence, we have exactly the same setting as in the established Hubs and Authorities (HITS) model (Kleinberg, 1999). The first part of any eigenvector is always an eigenvector of the hub matrix $G^T G$, and the second part is an eigenvector of the authority matrix GG^T.

Repeated squaring. In the special case that the matrix X is very rectangular ($n \ll m$ or $n \gg m$), one of the two matrices G^TG or GG^T is very small. Then it is known that one can obtain an exponential speed-up in the power method by repeatedly squaring the smaller one of the matrices. In other words we can perform $O(\log \frac{1}{\varepsilon})$ many *matrix-matrix* multiplications instead of $O(\frac{1}{\varepsilon})$ *matrix-vector* multiplications.

4.2. Application to Matrix Completion and Low Norm Matrix Factorizations

For matrix factorization problems as for example from recommender systems (Koren et al., 2009), our algorithm is particularly suitable as it retains the sparsity of the observations, and constructs the solution in a factorized way. In the setting of a partially observed matrix such as in the Netflix case, the loss function $f(X)$ only depends on the observed positions, which are very sparse, so $\nabla f(X)$ — which is all we need for our algorithm — is also sparse.

We again suppose that we want to approximate a partially given matrix Y (let P be the set of known entries of the matrix) by a product $X = UV^T$ such that some convex loss function $f(X)$ is minimized. By T we denote the unknown test entries of the matrix we want to predict. Our algorithm applies to any convex loss function on a low norm matrix factorization problem, and we will only mention two cases in particular:

Our algorithm directly applies to *Maximum Margin Matrix Factorization* (MMMF) (Srebro et al., 2004), whose original (soft margin) formulation is the trade-off formulation (1) with $f(X) := \sum_{ij\in P} |X_{ij} - y_{ij}|$ being the hinge or ℓ_1-loss. Because this is not differentiable, the authors recommend using the differentiable smoothed hinge loss instead.

When using the standard squared loss function $f(X) := \sum_{ij\in P}(X_{ij} - y_{ij})^2$, the problem is known as *Regularized Matrix Factorization* (Wu, 2007), and our algorithm directly applies. This loss function is widely used in practice, has a nice gradient structure, and is just the natural matrix generalization of the ℓ_2-loss (notice the analogous Lasso and regularized least squares formulation). The same function is known as the rooted mean squared error, which was the quality measure used in the Netflix competition. We write $RMSE_{train}$ and $RMSE_{test}$ for the rooted error on the training ratings P and test ratings T respectively.

Running time and memory. From Theorem 3 we obtain that the running time of our Algorithm 2 is linear in the size of the input: Each matrix-vector multiplication in Lanczos or the power method exactly costs $|P|$ (the number of observed positions of the matrix) operations, and we know that in total we need at most $O\left(\frac{1}{\varepsilon^2}\right)$ many such matrix-vector multiplications. The same holds for the memory requirement: There is no need to store the entire factorization of $X^{(k)}$ (meaning all the vectors v_k), but instead we only update and store the prediction values $X^{(k)}_{ij}$ for $ij \in P \cup T$ in each step. This, together with the known ratings y_{ij} determines the sparse gradient matrix $\nabla f(X^{(k)})$ during the algorithm. Therefore, the total memory requirement is only $|P \cup T|$ (the size of the output) plus the size $n + m$ of a single feature vector v_k.

The constant C_f in the running time of Hazan's algorithm.

Lemma 4. *For the squared error $f(X) = \frac{1}{2}\sum_{ij\in P}(X_{ij} - y_{ij})^2$, it holds that $C_{\hat{f}} \le 1$.*

Proof. It is known that the constant $C_{\hat{f}}$ is upper bounded by the largest eigenvalue of the Hessian $\nabla^2 \hat{f}(\vec{Z})$ (here we consider $\hat{f}$ as a function on vectors). On can directly compute that the diagonal entries of $\nabla^2 \hat{f}(\vec{Z})$ are 1 at the entries corresponding to P, and zero everywhere else, hence $C_{\hat{f}} \le 1$. □

4.3. Two Improved Variants of Algorithm 1

The optimum on the line segment. Instead of fixing the step width to $\alpha_k := \frac{1}{k}$ in Algorithm 1, the $\alpha_k \in [0, 1]$ of best improvement in the objective function f can be found by line search. (Hazan, 2008) has already proposed binary search to find better values for α_k. In many cases, however, we can even compute it analytically in a straightforward manner: Consider $f_\alpha := f\left(Z^{(k+1)}_{(\alpha)}\right) = f\left(Z^{(k)} + \alpha\left(v_k v_k^T - Z^{(k)}\right)\right)$ and compute

$$0 \doteq \frac{\partial}{\partial\alpha} f_\alpha = \left\langle \nabla f\left(Z^{(k+1)}_{(\alpha)}\right), v_k v_k^T - Z^{(k)} \right\rangle \qquad (7)$$

If this equation can be solved for α, then the optimal such α_k can directly be used as the step size, and the convergence guarantee of Theorem 3 still holds.

For the squared error $f(X) = \frac{1}{2}\sum_{ij\in P}(X_{ij} - y_{ij})^2$, when we write $\bar{v}$ for the approximate eigenvector v_k in step k, the optimality condition (7) is equivalent to

$$\alpha_k = \frac{\sum_{ij\in P}(X_{ij} - y_{ij})(X_{ij} - \bar{v}_i\bar{v}_j)}{\sum_{ij\in P}(X_{ij} - \bar{v}_i\bar{v}_j)^2} \qquad (8)$$

Immediate Feedback in the Power Method. As a second small improvement, we propose a heuristic to speed up the eigenvector computation in ApproxEV $\left(-\nabla f(Z^{(k)}), \varepsilon'\right)$: Instead of multiplying the current candidate vector v_k with the matrix

$\nabla f(Z^{(k)})$ in each power iteration, we multiply with $\frac{1}{2}\left(\nabla f(Z^{(k)}) + \nabla f\left(Z^{(k)} + \frac{1}{k} v_k v_k^T\right)\right)$, i.e. the average of the old and the new gradient. This means we immediately take into account the effect of the new feature vector v_k. This heuristic (which unfortunately does not fall into our current theoretical guarantee) is inspired by stochastic gradient descent as in Simon Funk's method, which we describe in the following:

4.4. Relation to Simon Funk's SVD Method

Interestingly, our proposed framework can also be seen as a theoretically justified variant of Simon Funk's (Webb, 2006) and related approximate SVD methods, which were used as a building block by most of the teams participating in the Netflix competition (including the winner team). Those methods have been further investigated by (Paterek, 2007; Takács et al., 2009) and also (Kurucz et al., 2007), which already proposed a heuristic using the HITS formulation. These approaches are algorithmically extremely similar to our method, although they are aimed at a slightly different optimization problem, and do *not* directly guarantee bounded nuclear norm. Very recently, (Salakhutdinov & Srebro, 2010) observed that Funk's algorithm can be seen as stochastic gradient descent to optimize (1) when the regularization term is replaced by a *weighted* variant of the nuclear norm.

Simon Funk's method considers the standard squared loss function $f(X) = \frac{1}{2}\sum_{ij\in S}(X_{ij} - y_{ij})^2$, and finds the new rank-1 estimate (or feature) v by iterating $v := v + \lambda(-\nabla\hat{f}(Z)v - Kv)$, or equivalently

$$v := \lambda\left(-\nabla\hat{f}(Z) + \left(\frac{1}{\lambda} - K\right)\mathbf{I}\right)v\ , \qquad (9)$$

a fixed number of times. Here λ is a small fixed constant called the learning rate. Additionally a decay rate $K > 0$ is used for regularization, i.e. to penalize the magnitude of the resulting feature v. Clearly this matrix multiplication formulation (9) is equivalent to a step of the power method applied within our framework[2], and for small enough learning rates λ the resulting feature vector will converge to the largest eigenvector of $-\nabla\hat{f}(Z)$.

However in Funk's method, the magnitude of each new feature strongly depends on the starting vector v_0, the number of iterations, the learning rate λ as well as the decay K, making the convergence very sensitive to these parameters. This might be one of the reasons that so far no results on the convergence speed could be obtained. Our method is free of these parameters, the k-th new feature vector is always a unit vector scaled by $\frac{1}{\sqrt{k}}$. Also, we keep the Frobenius norm $\|U\|_{Fro}^2 + \|V\|_{Fro}^2$ of the obtained factorization exactly fixed during the algorithm, whereas in Funk's method — which has a different optimization objective — this norm strictly increases with every newly added feature.

Our described framework therefore gives a solid theoretical foundation for a modified variant of the experimentally successful method (Webb, 2006) and its related variants such as (Kurucz et al., 2007; Paterek, 2007; Takács et al., 2009), with proved approximation quality and running time.

5. Experimental Results

We run our algorithm for the following standard datasets[3] for matrix completion problems, using the squared error function.

dataset	*#ratings*	n	m
MovieLens 100k	10^5	943	1682
MovieLens 1M	10^6	6040	3706
MovieLens 10M	10^7	69878	10677
Netflix	10^8	480189	17770

Any eigenvector method can be used as a black-box in our algorithm. To keep the experiments simple, we used the power method[4], and performed $0.2 \cdot k$ power iterations in step k. If not stated otherwise, the only optimization we used is the improvement by averaging the old and new gradient as explained in Section 4.3. All results were obtained by our (single-thread) implementation in Java 6 on a 2.4 GHz Intel C2D laptop.

Sensitivity. The generalization performance of our method is relatively stable under different choices of the regularization parameter, see Figure 1:

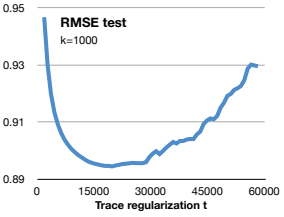

Figure 1. Sensitivity of the method on the choice of the regularization parameter t in (2), on MovieLens 1M.

[2] Another difference of our method to Simon Funk's lies in the stochastic gradient descent type of the later, i.e. "immediate feedback": During each matrix multiplication, it already takes the modified current feature v into account when calculating the loss $\hat{f}(Z)$, whereas our Algorithm 1 alters Z only after the eigenvector computation is finished.

[3] See www.grouplens.org and archive.ics.uci.edu/ml.

[4] We used the power method starting with the uniform unit vector. $\frac{1}{2}$ of the approximate eigenvalue corresponding to the previously obtained feature v_{k-1} was added to the matrix diagonal to ensure good convergence.

Movielens. Table 1 reports the running times of our algorithm on the three MovieLens datasets. Our algorithm gives an about 5.6 fold speed increase over (Toh & Yun, 2009), which is a very similar method to (Ji & Ye, 2009). (Toh & Yun, 2009) already improves the "singular value thresholding" methods (Cai et al., 2008) and (Ma et al., 2009). For MMMF, (Rennie & Srebro, 2005) report an optimization time of about 5 hours on the 1M dataset, but use the different smoothed hinge loss function so that the results cannot be directly compared. (Ma et al., 2009), (Srebro & Jaakkola, 2003) and (Ji & Ye, 2009) only obtained results on much smaller datasets.

Table 1. Running times t_{our} (in seconds) of our algorithm on the three MovieLens datasets compared to the reported timings t_{TY} of (Toh & Yun, 2009). The ratings $\{1, \dots, 5\}$ were used as-is and *not* normalized to any user and/or movie means. In accordance with (Toh & Yun, 2009), 50% of the ratings were used for training, the others were used as the test set. Here $NMAE$ is the mean absolute error, times $\frac{1}{5-1}$, over the total set of ratings. k is the number of iterations of our algorithm, $\#mm$ is the total number of sparse matrix-vector multiplications performed, and tr is the used trace parameter t in (2). They used Matlab/PROPACK on an Intel Xeon 3.20 GHz processor.

	$NMAE$	t_{TY}	t_{our}	k	$\#mm$	tr
100k	0.205	7.39	0.156	15	33	9975
1M	0.176	24.5	1.376	35	147	36060
10M	0.164	202	36.10	65	468	281942

In all the following experiments we have pre-normalized all training ratings to the simple average $\frac{\mu_i+\mu_j}{2}$ of the user and movie mean values, for the sake of being consistent with comparable literature.

For MovieLens 10M, we used partition r_b provided with the dataset (10 test ratings per user). The regularization parameter t was set to 48333. We obtained a $RMSE_{test}$ of 0.8617 after $k = 400$ steps, in a total running time of 52 minutes (16291 matrix multiplications). Our best $RMSE_{test}$ value was 0.8573, compared to 0.8543 obtained by (Lawrence & Urtasun, 2009) using their non-linear improvement of MMMF.

Algorithm Variants. Comparing the proposed algorithm variants from Section 4.3, Figure 2 demonstrates moderate improvements compared to our original Algorithm 2.

Netflix. Table 2 shows an about 13 fold speed increase of our method over the "Hard Impute" singular value thresholding algorithm of (Mazumder et al., 2009) on the Netflix dataset, where they used Matlab/PROPACK on an Intel Xeon 3 GHz processor.

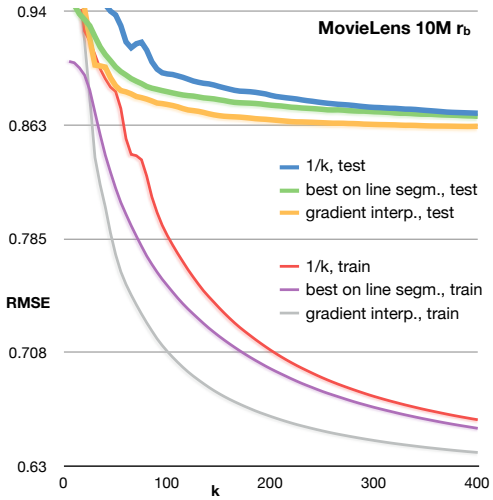

Figure 2. Improvements for the two algorithm variants described in Section 4.3, when running on MovieLens 10M.

Table 2. Running times t_{our} (in hours) of our algorithm on the Netflix dataset compared to the reported timings t_{M} of (Mazumder et al., 2009).

$RMSE_{test}$	t_{M}	t_{our}	k	$\#mm$	tr
0.986	3.3	0.144	20	50	99592
0.977	5.8	0.306	30	109	"
0.965	6.6	0.504	40	185	"
0.9478	n.a.	13.6	200	4165	"

Note that the primary goal of this experimental section is *not* to compete with the prediction quality of the best engineered recommender systems (which are usually ensemble methods). We just demonstrate that our method solves nuclear norm regularized problems of the form (2) on large sample datasets, obtaining strong performance improvements.

6. Conclusion

We have introduced a new method to solve arbitrary convex problems with a nuclear norm regularization, which is simple to implement and parallelize and scales very well. The method is parameter-free and comes with a convergence guarantee. This is, to our knowledge, the first guaranteed convergence speed result for the class of Simon-Funk-type algorithms.

Further interesting questions include whether a similar algorithm could be used if a strict low-rank constraint as in (4), (5) is simultaneously applied. This corresponds to fixing the sparsity of a solution in the coreset setting. Also, it remains to investigate if our algorithm can be applied to other matrix factorization problems such as (potentially only partially observed) kernel matrices as e.g. PSVM (Chang et al., 2007), PCA or [p]LSA, because our method could exploit the even simpler form of ∇f for symmetric matrices.

7. Acknowledgements

We would like to thank Arkadi Nemirovski and Elad Hazan for helpful discussions. M. Jaggi is supported by a Google Research Award and the Swiss National Science Foundation (SNF Project 20PA21-121957). M. Sulovský is supported by the Swiss National Science Foundation (SNF Project 200020-125027).

References

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

Berkhin, P. A survey on pagerank computing. *Internet Mathematics*, 2(1):73, 2005.

Cai, J-F, Candes, E, and Shen, Z. A singular value thresholding algorithm for matrix completion. *arXiv*, math.OC, 2008.

Candes, E and Tao, T. The power of convex relaxation: Near-optimal matrix completion. IEEE Trans. Inform. Theory (to appear). *arXiv*, cs.IT, 2009.

Chang, E, Zhu, K, Wang, H, Bai, H, Li, J, Qiu, Z, and Cui, H. PSVM: Parallelizing support vector machines on distributed computers. *NIPS*, 20:257–264. 2007.

Clarkson, K. Coresets, Sparse Greedy Approximation, and the Frank-Wolfe Algorithm. *SODA*, 2008.

DeCoste, D. Collaborative prediction using ensembles of maximum margin matrix factorizations. *ICML*, 2006.

Fazel, M, Hindi, H, and Boyd, S. A rank minimization heuristic with application to minimum order system approximation. *Proc. American Control Conference*, 6:4734–4739, 2001.

Hazan, E. Sparse approximate solutions to semidefinite programs. *LATIN*, pp. 306–316, 2008.

Ji, S and Ye, Y. An accelerated gradient method for trace norm minimization. *ICML*, 2009.

Kleinberg, J. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5), 1999.

Koren, Y, Bell, R, and Volinsky, C. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

Kurucz, M, Benczur, A, and Csalogany, K. Methods for large scale SVD with missing values. *SIGKDD*, 2007.

Lawrence, N and Urtasun, R. Non-linear matrix factorization with gaussian processes. *ICML*, 2009.

Lin, C-J. Projected gradient methods for nonnegative matrix factorization. *Neural Comput.*, 19(10):2756–2779, 2007.

Liu, Y-J, Sun, D, and Toh, K-C. An implementable proximal point algorithmic framework for nuclear norm minimization. *Optimization Online*, 2009.

Ma, S, Goldfarb, D, and Chen, L. Fixed point and bregman iterative methods for matrix rank minimization. *Mathematical Programming*, 2009.

Mazumder, R, Hastie, T, and Tibshirani, R. Spectral regularization algorithms for learning large incomplete matrices. *Submitted to JMLR*, 2009.

Nesterov, Y. A method of solving a convex programming problem with convergence rate O(1/sqr(k)). *Soviet Mathematics Doklady*, 27:372–376, 1983.

Paterek, A. Improving regularized singular value decomposition for collaborative filtering. *SIGKDD*, 2007.

Rennie, J and Srebro, N. Fast maximum margin matrix factorization for collaborative prediction. *ICML*, 2005.

Salakhutdinov, R and Srebro, N. Collaborative filtering in a non-uniform world: Learning with the weighted trace norm. *arXiv*, cs.LG, Feb 2010.

Srebro, N, Rennie, J, and Jaakkola, T. Maximum-Margin Matrix Factorization. *NIPS*, 17:1329–1336, 2004.

Srebro, N and Jaakkola, T. Weighted low-rank approximations. *ICML*, 2003.

Takács, G, Pilászy, I, Németh, B, and Tikk, D. Scalable collaborative filtering approaches for large recommender systems. *JMLR*, 10, 2009.

Tibshirani, R. Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society. Series B*, pp. 267–288, 1996.

Toh, K and Yun, S. An accelerated proximal gradient algorithm for nuclear norm regularized least squares problems. *Optimization Online*, 2009.

Webb, B. Netflix update: Try this at home. *Simon Funk's personal blog*, 2006. `http://sifter.org/~simon/journal/20061211.html`.

Wu, M. Collaborative filtering via ensembles of matrix factorizations. *SIGKDD*, 2007.