Linear Algebra and its Applications 431 (2009) 760–771

Contents lists available at ScienceDirect

Linear Algebra and its Applications

journal homepage: www.elsevier.com/locate/laa

Random projections for the nonnegative least-squares problem

Christos Boutsidis *, Petros Drineas

Computer Science Department, Rensselaer Polytechnic Institute, Troy, NY 12180, United States

ARTICLE INFO

Article history:
Received 8 December 2008
Accepted 5 March 2009
Available online 25 April 2009

Submitted by M. Tsatsomeros

Keywords:
Nonnegative least-squares
Sampling
Hadamard transform
Randomized algorithm
Random projections

ABSTRACT

Constrained least-squares regression problems, such as the Nonnegative Least Squares (NNLS) problem, where the variables are restricted to take only nonnegative values, often arise in applications. Motivated by the recent development of the fast Johnson–Lindestrauss transform, we present a fast random projection type approximation algorithm for the NNLS problem. Our algorithm employs a randomized Hadamard transform to construct a much smaller NNLS problem and solves this smaller problem using a standard NNLS solver. We prove that our approach finds a nonnegative solution vector that, with high probability, is close to the optimum nonnegative solution in a relative error approximation sense. We experimentally evaluate our approach on a large collection of term-document data and verify that it does offer considerable speedups without a significant loss in accuracy. Our analysis is based on a novel random projection type result that might be of independent interest. In particular, given a tall and thin matrix $\Phi \in \mathbb{R}^{n\times d}$ ($n \gg d$) and a vector $y \in \mathbb{R}^d$, we prove that the Euclidean length of Φy can be estimated very accurately by the Euclidean length of $\tilde{\Phi} y$, where $\tilde{\Phi}$ consists of a small subset of (appropriately rescaled) rows of Φ.

* Corresponding author.
E-mail addresses: boutsc@cs.rpi.edu (C. Boutsidis), drinep@cs.rpi.edu (P. Drineas).

0024-3795/$ - see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.laa.2009.03.026

1. Introduction

The Nonnegative Least Squares (NNLS) problem is a constrained least-squares regression problem where the variables are allowed to take only nonnegative values. More specifically, the NNLS problem is defined as follows:

Definition 1 (*Nonnegative Least Squares (NNLS)*). Given a matrix $A \in \mathbb{R}^{n\times d}$ and a target vector $b \in \mathbb{R}^n$, find a nonnegative vector $x_{opt} \in \mathbb{R}^d$ such that

$$x_{opt} = \arg \min_{x\in\mathbb{R}^d,\, x\geqslant 0} \|Ax - b\|_2^2 . \tag{1}$$

NNLS is a quadratic optimization problem with linear inequality constraints. As such, it is a convex optimization problem and thus it is solvable (up to arbitrary accuracy) in polynomial time [4]. In words, NNLS seeks to find the best nonnegative vector x_{opt} in order to approximately express b as a strictly nonnegative linear combination of the columns of A, i.e., $b \approx Ax_{opt}$.

The motivation for NNLS problems in data mining and machine learning stems from the fact that given least-squares regression problems on nonnegative data such as images, text, etc., it is natural to seek *nonnegative* solution vectors. (Examples of data applications are described in [6].) NNLS is also useful in the computation of the Nonnegative Matrix Factorization [16], which has received considerable attention in the past few years. Finally, NNLS is the core optimization problem and the computational bottleneck in designing a class of Support Vector Machines [22]. Since modern datasets are often massive, there is continuous need for faster, more efficient algorithms for NNLS.

In this paper we discuss the applicability of random projection algorithms for solving constrained regression problems, and in particular NNLS problems. Our goal is to provide fast approximation algorithms as alternatives to the existing exact, albeit expensive, NNLS methods. We focus on input matrices A that are tall and thin, i.e., $n \gg d$, and we present, analyze, and experimentally evaluate a random projection type algorithm for the nonnegative least-squares problem. Our algorithm utilizes a novel random projection type result which might be of independent interest. We argue that the proposed algorithm (described in detail in Section 3), provides relative error approximation guarantees for the NNLS problem. Our work is motivated by recent progress in the design of fast randomized approximation algorithms for unconstrained ℓ_p regression problems [10,7].

The following theorem is the main quality-of-approximation result for our randomized NNLS algorithm.

Theorem 1. *Let $\epsilon \in (0, 1]$. Let $A \in \mathbb{R}^{n\times d}$ and $b \in \mathbb{R}^n$ be the inputs of the NNLS problem with $n \gg d$. If the input parameter r of the* RandomizedNNLS *algorithm of Section 3 satisfies*

$$\frac{r}{\log r} \geqslant \frac{342 c_o^2 (d+1)\log(n)}{\epsilon^2}, \tag{2}$$

(for a sufficiently large constant c_o)[1] *then the* RandomizedNNLS *algorithm returns a nonnegative vector $\tilde{x}_{opt}$ such that*

$$\|A\tilde{x}_{opt} - b\|_2^2 \leq (1+\epsilon) \min_{x\in\mathbb{R}^d,\, x\geqslant 0} \|Ax - b\|_2^2 \tag{3}$$

holds with probability at least 0.5.[2] *The running time of the* RandomizedNNLS *algorithm is*

$$O(nd\log(r)) + T_{NNLS}(r, d) . \tag{4}$$

The latter term corresponds to the time required to exactly solve an NNLS problem on an input matrix of dimensions $r \times d$.

[1] c_o is an unspecified constant in [19].

[2] Note that a small number of repetitions of the algorithm suffices to boost its success probability.

One should compare the running time of our method to $T_{NNLS}(n, d)$, which corresponds to the time required to solve the NNLS problem exactly. We experimentally evaluate our approach on 3000 NNLS problems constructed from a large and sparse term-document data collection. On average (see Section 4.1), the proposed algorithm achieves a three-fold speedup when compared to a state-of-the-art NNLS solver [15] with a small (approx. 10%) loss in accuracy; a two-fold speedup is achieved with a 4% loss in accuracy. Computational savings are more pronounced for NNLS problems with denser input matrices A and vectors b (see Section 4.2).

The remainder of the paper is organized as follows. Section 2 reviews basic linear algebraic definitions and discusses related work. In Section 3 we present our randomized algorithm for approximating the NNLS problem, discuss its running time, and give the proof of Theorem 1. Finally, in Section 4 we provide an experimental evaluation of our method.

2. Background and related work

Let $[n]$ denote the set $\{1, 2, \ldots, n\}$. For any matrix $A \in \mathbb{R}^{n\times d}$ with $n \geqslant d$ let $A_{(i)}, i \in [n]$ denote the i-th row of A as a row vector, and let $A^{(j)}, j \in [d]$ denote the j-th column of A as a column vector. The Singular Value Decomposition (SVD) of A can be written as

$$A = U_A \Sigma_A V_A^T. \tag{5}$$

Assuming that A has full rank, $U_A \in \mathbb{R}^{n\times d}$ and $V_A \in \mathbb{R}^{d\times d}$ are orthonormal matrices, while Σ_A is a $d \times d$ diagonal matrix. Finally, $\|A\|_F^2 = \sum_{i=1}^{n} \sum_{j=1}^{d} A_{ij}^2$ denotes the square of the Frobenius norm of A and $\|A\|_2 = \sup_{x\in\mathbb{R}^d,\, x\neq 0} \|Ax\|_2 / \|x\|_2$ denotes the spectral norm of A.

The (non-normalized) $n \times n$ matrix of the Hadamard–Walsh transform H_n is defined recursively as follows:

$$H_n = \begin{bmatrix} H_{n/2} & H_{n/2} \\ H_{n/2} & -H_{n/2} \end{bmatrix}, \quad \text{with } H_2 = \begin{bmatrix} +1 & +1 \\ +1 & -1 \end{bmatrix}.$$

The $n \times n$ normalized matrix of the Hadamard–Walsh transform is equal to $\frac{1}{\sqrt{n}} H_n$; hereafter, we will denote this normalized matrix by H_n (n is a power of 2). For simplicity, throughout this paper we will assume that n is a power of two; padding A and b with all-zero rows suffices to remove the assumption. Finally, all logarithms are base two.

2.1. Random projection algorithms for unconstrained ℓ_p problems

The unconstrained least-squares regression problem (ℓ_2) takes as input a matrix $A \in \mathbb{R}^{n\times d}$ and a vector $b \in \mathbb{R}^n$, and returns as output a vector $x_{opt} \in \mathbb{R}^d$ that minimizes the distance $\|Ax - b\|_2^2$. Assuming $n \gg d$, various algorithms solve the problem exactly in $O(nd^2)$ time [14]. Drineas et al. [9,10] and Sarlos [21] give randomized algorithms to approximate the solution to such problems. The basic idea of these algorithms is to select a subset of rows from A and a subset of elements from b, and solve the induced problem exactly. The fundamental algorithmic challenge is how to form the induced problem. It turns out that sampling rows of A and elements of b with probabilities that are proportional to the ℓ_2 norms of the rows of the matrix of the left singular vectors of A suffices [9]. This approach is not computationally efficient, since computing these probabilities takes $O(nd^2)$ time. However, by leveraging the Fast Johnson–Lindenstrauss Transform of [2], one can design an $o(nd^2)$ algorithm for this problem [10]. The algorithm of this paper applies the same preconditioning step as the main Algorithm of [10]. The analysis though is very different from the analysis of [10] and is based on a novel random projection type result that is presented in Section 3. The difficulty of applying the analysis of [10] here is the fact that the solution of an NNLS problem cannot be written in a closed form. Finally, it should be noted that similar ideas are discussed in [7], where the authors present sampling-based approximation algorithms for the ℓ_p regression problem for $p = [1, \infty)$. The preconditioning step of the algorithm of this paper is different from the preconditioning step of the algorithm of [7].

2.2. Algorithms for the NNLS problem

We briefly review NNLS algorithms following the extensive review in [6]. Recall that the NNLS Problem is a quadratic optimization problem. Hence, all quadratic programming algorithms may be used to solve it. Methods for solving NNLS problems can be divided into three general categories: (i) active set methods, (ii) iterative methods, and (iii) other methods. The approach of Lawson and Hanson in [18] seems to be the first technique to solve NNLS problems. It is a typical example of an active set method and is implemented as the function *lsqnonneg* in Matlab. Immediate followups to this work include the technique of Bro and Jong [5] which is suitable for problems with multiple right hand sides, as well as the combinatorial NNLS approach of Dax [8]. The Projective Quasi-Newton NNLS algorithm of [15] is an example from the second category. It is an iterative approach based on the Newton iteration and the efficient approximation of the Hessian matrix. Numerical experiments in [15] indicate that it is a very fast alternative to the aforementioned active set methods. The sequential coordinate-wise approach of [12] is another example of an iterative NNLS method. Finally, interior point methods are suitable for NNLS computations [20]. A different approach appeared in [22]. It starts with a random nonnegative vector $x \in \mathbb{R}^d$ and updates it via elementwise multiplicative rules. Surveys on NNLS algorithms include [4,18,15].

3. A random projection type algorithm for the NNLS problem

This section describes our main algorithm for the NNLS problem. Our algorithm employs a randomized Hadamard transform to construct a much smaller NNLS problem and solves this smaller problem exactly using a standard NNLS solver. The approximation accuracy of our algorithm is a function of the size of the small NNLS problem.

3.1. The RandomizedNNLS algorithm

Algorithm RandomizedNNLS takes as inputs an $n \times d$ matrix A ($n \gg d$), an n-dimensional target vector b, and a positive integer $r < n$. It outputs a nonnegative d-dimensional vector $\tilde{x}_{opt}$ that approximately solves the original NNLS problem. Our algorithm starts by premultiplying the matrix A and the right hand side vector b with a random $n \times n$ diagonal matrix D, whose diagonal entries are set to $+1$ or -1 with equal probability. It then multiplies the resulting matrix DA and the vector Db with a small submatrix of the $n \times n$ normalized Hadamard-Walsh matrix H_n (see section 2). This submatrix of H_n – denoted by $\widetilde{H}$ – is constructed as follows: for all $i \in [n]$, the ith row of H_n is included in $\widetilde{H}$ with probability r/n. Clearly, the expected number of rows of the matrix $\widetilde{H}$ is equal to r. Finally, our algorithm returns the nonnegative vector $\tilde{x}_{opt} \in R^d$ that satisfies

$$\tilde{x}_{opt} = \arg \min_{x \in \mathbb{R}^d, x \geqslant 0} \left\| \widetilde{H} D \left(Ax - b \right) \right\|_2^2 . \tag{6}$$

In Section 3.3 we will argue that, for any $\epsilon \in (0, 1]$, if we set

$$r \geqslant 684 c_o^2 (d+1) \log(n) \log(342 c^2 (d+1) \log(n)/\epsilon^2)/\epsilon^2, \tag{7}$$

then $\left\| A\tilde{x}_{opt} - b \right\|_2^2$ is at most $(1+\epsilon)$ worse than the true optimum $\left\| Ax_{opt} - b \right\|_2^2$. This is a sufficient (but not necessary) condition for r in order to satisfy the relative error guarantees of Eq. (3). Indeed, in the experiments of Section 4 we will argue that empirically a much smaller value of r, for example $r = d + 20$, suffices.

Algorithm 1. The RANDOMIZEDNNLS algorithm.

Inputs: $A \in \mathbb{R}^{n\times d}$, $b \in \mathbb{R}^n$, positive integer $r < n$.
Output: a nonnegative vector $\tilde{x}_{opt} \in \mathbb{R}^d$.

1. Let H_n be the $n \times n$ normalized Hadamard–Walsh matrix.
2. Let S be an $n \times n$ diagonal matrix such that for all $i \in [n]$,

$$S_{ii} = \begin{cases} \sqrt{n/r}, & \text{with probability } r/n \\ 0, & \text{otherwise} \end{cases}$$

3. Let $\widetilde{H}$ be the matrix consisting of the non-zero rows of SH_n. (Notice that $\widetilde{H}$ has – in expectation – r rows.)
4. Construct the $n \times n$ diagonal matrix D such that, for all $i \in [n]$, $D_{ii} = +1$ with probability $1/2$; otherwise $D_{ii} = -1$.
5. Solve

$$\tilde{x}_{opt} = \arg \min_{x\in\mathbb{R}^d, x\geqslant 0} \left\|\widetilde{H}DAx - \widetilde{H}Db\right\|_2^2,$$

using any standard NNLS solver and return the vector $\tilde{x}_{opt}$.

3.2. The proof of Theorem 1

3.2.1. A random projection type result

In this section we prove a random projection type result based on the so-called subspace sampling procedure [11] that might be of independent interest. In particular, given a matrix $\Phi \in \mathbb{R}^{n\times d}$ with $n \gg d$ (a.k.a., Φ is tall and thin), and *any* vector $y \in \mathbb{R}^d$, we argue that the ℓ_2 norm of Φy can be estimated very accurately by $\widetilde{\Phi} y$, where $\widetilde{\Phi}$ consists of small subset of (appropriately rescaled) rows of Φ.

More specifically, consider the SUBSPACESAMPLING algorithm described below. This algorithm selects a small subset of rows of Φ to construct $\widetilde{\Phi}$; notice that $\widetilde{\Phi}$ has – in expectation – at most r rows. Also notice that $\widetilde{\Phi}$ contains the i-th row of Φ (appropriately rescaled) if and only if S_{ii} is non-zero. Lemma 1 bounds the approximation error for our subspace sampling algorithm.

Algorithm 2. SUBSPACESAMPLING algorithm.

Input: $\Phi \in \mathbb{R}^{n\times d}$, integer $r < n$, set of probabilities $p_i \geqslant 0, i \in [n]$ s.t. $\sum_{i\in[n]} p_i = 1$.
Output: $\widetilde{\Phi} \in \mathbb{R}^{\tilde{r}\times d}$, with $\mathbf{E}(\tilde{r}) \leqslant r$.

1. Let S be the $n \times n$ diagonal matrix such that for all $i \in [n]$,

$$S_{ii} = \begin{cases} 1/\sqrt{\min\{1, rp_i\}}, & \text{with probability } \min\{1, rp_i\} \\ 0 & \text{, otherwise} \end{cases}$$

2. Let $\widetilde{\Phi}$ be the matrix consisting of the non-zero rows of $S\Phi$.
(Notice that $\widetilde{\Phi}$ has – in expectation – r rows.)

Lemma 1. *Let $\epsilon \in (0,1]$. Let Φ be an $n \times d$ matrix ($n \gg d$), U_Φ be the $n \times d$ matrix containing the left singular vectors of Φ, and $U_{\Phi(i)}$ denote the i-th row of U_Φ. Let $\widetilde{\Phi}$ be constructed using the* SubspaceSampling *algorithm with inputs Φ, r, and sampling probabilities p_i. If for all $i \in [n]$,*

$$p_i \geqslant \beta \left\| U_{\Phi(i)} \right\|_2^2 \Big/ d \tag{8}$$

for some $\beta \in (0,1]$, and the parameter r satisfies

$$\frac{r}{\log(r)} \geqslant \frac{9c_o^2 d}{\beta \epsilon^2}, \tag{9}$$

for a sufficiently large constant c_o, then with probability at least $2/3$ all d-dimensional vectors y satisfy,

$$\left| \|\Phi y\|_2^2 - \left\|\widetilde{\Phi} y\right\|_2^2 \right| \leqslant \epsilon \|\Phi y\|_2^2 . \tag{10}$$

Proof. Let $\Phi = U_\Phi \Sigma_\Phi V_\Phi^T$ be the SVD of Φ with $U_\Phi \in \mathbb{R}^{n\times d}$, $\Sigma_\Phi \in \mathbb{R}^{d\times d}$, and $V_\Phi \in \mathbb{R}^{d\times d}$. Let S be the $n \times n$ diagonal matrix constructed at the first step of algorithm SubspaceSampling, and let $U_\Phi^T S^T S U_\Phi = I + E$, where I is the $d \times d$ identity matrix, and E some $d \times d$ matrix. Then, using these two definitions, submultiplicativity, and the orthogonality and unitary invariance of U_Φ and V_Φ,

$$\begin{aligned}
\left| \|\Phi y\|_2^2 - \left\|\tilde{\Phi} y\right\|_2^2 \right| &= \left| y^T \Phi^T \Phi y - y^T \Phi^T S^T S \Phi y \right| \\
&= \left| y^T V_\Phi \Sigma_\Phi^2 V_\Phi^T y - y^T V_\Phi \Sigma_\Phi U_\Phi^T S^T S U_\Phi \Sigma_\Phi V_\Phi^T y \right| \\
&= \left| y^T V_\Phi \Sigma_\Phi^2 V_\Phi^T y - y^T V_\Phi \Sigma_\Phi (I+E) \Sigma_\Phi V_\Phi^T y \right| \\
&= \left| y^T V_\Phi \Sigma_\Phi E \Sigma_\Phi V_\Phi^T y \right| \\
&\leqslant \left\| y^T V_\Phi \Sigma_\Phi \right\|_2 \|E\|_2 \left\| \Sigma_\Phi V_\Phi^T y \right\|_2 \\
&= \|E\|_2 \left\| \Sigma_\Phi V_\Phi^T y \right\|_2^2 \\
&= \|E\|_2 \left\| U_\Phi \Sigma_\Phi V_\Phi^T y \right\|_2^2 \\
&= \|E\|_2 \|\Phi y\|_2^2 .
\end{aligned}$$

Using Theorem 7 of [11] (originally proven by Rudelson and Virshynin in [19]), we see that

$$\mathbf{E}\left(\|E\|_2\right) \leqslant c_o \sqrt{\frac{\log(r)}{\beta r}} \|U_\Phi\|_2 \|U_\Phi\|_F = c_o \sqrt{\frac{d\log(r)}{\beta r}} \tag{11}$$

for a sufficiently large constant c_o (c_o is not specified in [19]). Markov's inequality implies that

$$\|E\|_2 \leqslant 3c_o \sqrt{\frac{d\log(r)}{\beta r}}, \tag{12}$$

with probability at least 2/3. Finally, using Eq. (9) concludes the proof of the lemma. □

3.2.2. Another useful result

Results in [2] imply the following lemma.

Lemma 2. *Let U be an $n \times d$ orthogonal matrix ($n \geqslant 20$ and $n \geqslant d$). Then, for all $i \in [n]$,*

$$\left\| (H_n D U)_{(i)} \right\|_2^2 \leqslant \frac{4.2 d \log n}{n} \tag{13}$$

holds with probability at least 0.9.

3.2.3. The proof of Theorem 1

We are now ready to prove Theorem 1. We apply lemma 1 for $\Phi = [H_nDA \quad -H_nDb] \in R^{n\times(d+1)}$, the parameter r of Theorem 1, sampling probabilities $p_i = 1/n$, for all $i \in [n]$, $\beta = 1/(4.2\log n)$, and $\epsilon' = \epsilon/3 \in (0, 1/3]$, where $\epsilon \in (0, 1]$ is the parameter of Theorem 1. Let U_Φ be the $n \times (d+1)$ matrix of the left singular vectors of Φ. Note that U_Φ is exactly equal to $U_\Phi = H_nDU_{[A\ -b]}$, where $U_{[A\ -b]}$ is the $n \times (d+1)$ matrix of the left singular vectors of $[A\ -b]$. Lemma 2 for $U_{[A\ -b]}$ and our choice of β, guarantee that for all $i \in [n]$, with probability at least 0.9

$$1/n \geqslant \beta \left\| \left(H_nDU_{[A-b]}\right)_{(i)} \right\|_2^2 \Big/ d \Rightarrow 1/n \geqslant \beta \left\| \left(U_\Phi\right)_{(i)} \right\|_2^2 \Big/ d.$$

The latter inequality implies that assumption (8) of Lemma 1 holds. Also, our choice of r, which satisfies inequality (2) in Theorem 1, our choice of β, and our choice of ϵ, guarantee that assumption (9) of Lemma 1 is also true. Since all d-dimensional vectors y satisfy Eq. (10), we pick $y = \begin{bmatrix} x_{opt} \\ 1 \end{bmatrix}$ and $\tilde{y} = \begin{bmatrix} \tilde{x}_{opt} \\ 1 \end{bmatrix}$, thus getting that with probability at least 2/3:

$$\begin{aligned}(1-\epsilon')\left\|H_nDAx_{opt} - H_nDb\right\|_2^2 &\leqslant \left\|\widetilde{H}_nDAx_{opt} - \widetilde{H}_nDb\right\|_2^2 \\ &\leqslant (1+\epsilon')\left\|H_nDAx_{opt} - H_nDb\right\|_2^2, \end{aligned} \tag{14}$$

and

$$\begin{aligned}(1-\epsilon')\left\|H_nDA\tilde{x}_{opt} - H_nDb\right\|_2^2 &\leqslant \left\|\widetilde{H}_nDA\tilde{x}_{opt} - \widetilde{H}_nDb\right\|_2^2 \\ &\leqslant (1+\epsilon')\left\|H_nDA\tilde{x}_{opt} - H_nDb\right\|_2^2. \end{aligned} \tag{15}$$

Manipulating Eqs. (14) and (15) we get

$$\begin{aligned}\left\|H_nDA\tilde{x}_{opt} - H_nDb\right\|_2^2 &\leqslant \frac{1}{1-\epsilon'}\left\|\widetilde{H}_nDA\tilde{x}_{opt} - \widetilde{H}_nDb\right\|_2^2 \\ &\leqslant \frac{1}{1-\epsilon'}\left\|\widetilde{H}_nDAx_{opt} - \widetilde{H}_nDb\right\|_2^2 \\ &\leqslant \frac{1+\epsilon'}{1-\epsilon'}\left\|H_nDAx_{opt} - H_nDb\right\|_2^2 \\ &\leqslant \left(1+3\epsilon'\right)\left\|H_nDAx_{opt} - H_nDb\right\|_2^2 \\ &\leqslant (1+\epsilon)\left\|H_nDAx_{opt} - H_nDb\right\|_2^2. \end{aligned}$$

The second inequality follows since $\tilde{x}_{opt}$ is the optimal solution of the NNLS problem of Eq. (6), thus x_{opt} is a sub-optimal solution, and the fourth inequality follows since $(1+\epsilon')/(1-\epsilon') \leqslant 1+3\epsilon'$, for all $\epsilon' \in (0, 1/3]$. In the last inequality, we set $\epsilon' = \epsilon/3$. To conclude the proof, notice that H_nD is an orthonormal square matrix and can be dropped without changing a unitarily invariant norm. Finally, since Lemmas 1 and 2 fail with probability at most 1/3 and 1/10 respectively, the union bound implies that Theorem 1 fails with probability at most 0.5.

3.3. What is the minimal value of r?

To derive values of r for which the RandomizedNNLS algorithm satisfies the relative error guarantees of Theorem 1, we need to solve Eq. (2); this is hard since the solution depends on the Lambart W function. Thus, we identify a range of values of r that are sufficient for our purposes. Using the fact that for any $\alpha \geqslant 4$, and for any $\gamma \geqslant 2\alpha\log(\alpha)$,

$$\frac{\gamma}{\log(\gamma)} \geqslant \alpha$$

and by setting $\alpha = 342c_o^2(d+1)\log(n)/\epsilon^2$ in Eq. (2) (note that $342c_o^2(d+1)\log(n)/\epsilon^2 \geqslant 4$), it can be proved that every r such that

$$r \geqslant 684c_o^2(d+1)\log(n)\log(342c_o^2(d+1)\log(n)/\epsilon^2)/\epsilon^2, \tag{16}$$

satisfies the inequality 2 (c_o is the constant of Theorem 1).

3.4. Running time analysis

In this subsection we analyze the running time of our algorithm. Let r be the minimal value that satisfies equation (16). First, computing DA and Db takes $O(nd)$ time. Since $\widetilde{H}$ has in expectation r rows, Ailon and Liberty in [3] argue that the computation of $\widetilde{H}DA$ and $\widetilde{H}Db$ takes $O(nd\log r)$ time. For our choice of r, this is

$$T_{precond} = O(nd\log(d\log(n)/\epsilon^2)).$$

After this preconditioning step, we employ an NNLS solver on the smaller problem. The computational cost of the NNLS solver on the small problem was denoted as $T_{NNLS}(r,d)$ in Theorem 1. $T_{NNLS}(r,d)$ cannot be specified exactly since theoretical running times for exact NNLS solvers are unknown. In the sequel we comment on the computational costs of some well defined segments of some NNLS solvers.

The NNLS formulation of Definition 1 is a convex quadratic program, and is equivalent to

$$\min_{x\in\mathbb{R}^d, x\geqslant 0} x^TQx - 2q^Tx,$$

where $Q = A^TA \in \mathbb{R}^{d\times d}$ and $q = A^Tb \in \mathbb{R}^d$. Computing Q and q takes $O(nd^2)$ time, and then the time required to solve the above formulation of the NNLS problem is independent of n. Using this formulation, our algorithm would necessitate $T_{precond}$ time for the computation of $\widetilde{H}DA$ (the preconditioning step described above), and then $\widetilde{Q} = (\widetilde{H}DA)^T\widetilde{H}DA$ and $\tilde{q} = (\widetilde{H}DA)^Tb$ can be computed in $T_{MM} = O(rd^2)$ time; given our choice of r, this implies

$$T_{MM} = O(d^3\log(n)/\epsilon^2).$$

Overall, the standard approach would take $O(nd^2)$ time to compute Q, whereas our method would need only $T_{precond} + T_{MM}$ time for the construction of $\widetilde{Q}$. Note, for example, that when $n = O(d^2)$ and regarding ϵ as a constant, $\widetilde{Q}$ can be computed $O(d/\log(d))$ times faster than Q.

On the other hand, many standard implementations of NNLS solvers (and in particular those that are based on active set methods) work directly on the formulation of Definition 1. A typical cost of these implementations is of the order $O(nd^2)$ per iteration. Other approaches, for example the NNLS method of [15], proceed by computing matrix–vector products of the form Au, for an appropriate d-dimensional vector u, thus cost typically $O(nd)$ time per iteration. In these cases our algorithm needs again $T_{precond}$ preprocessing time, but costs only $O(rd^2)$ or $O(rd)$ time per iteration, respectively. Again, if given our choice of r, the computational savings *per iteration* are comparable with the $O(d/\log(d))$ speedup described above.

4. Experimental evaluation

In this section, we experimentally evaluate our RandomizedNNLS algorithm on (i) large, sparse matrices from a text-mining application, and (ii) random matrices with varying sparsity. We mainly focus on employing the state-of-the-art solver of [15] to solve the small NNLS problem.

4.1. The TechTC300 dataset

Our data come from the Open Directory Project (ODP) [1], a multilingual open content directory of WWW links that is constructed and maintained by a community of volunteer editors. ODP uses a hierarchical ontology scheme for organizing site listings. Listings on similar topics are grouped into categories, which can then include smaller subcategories. Gabrilovich and Markovitch constructed a

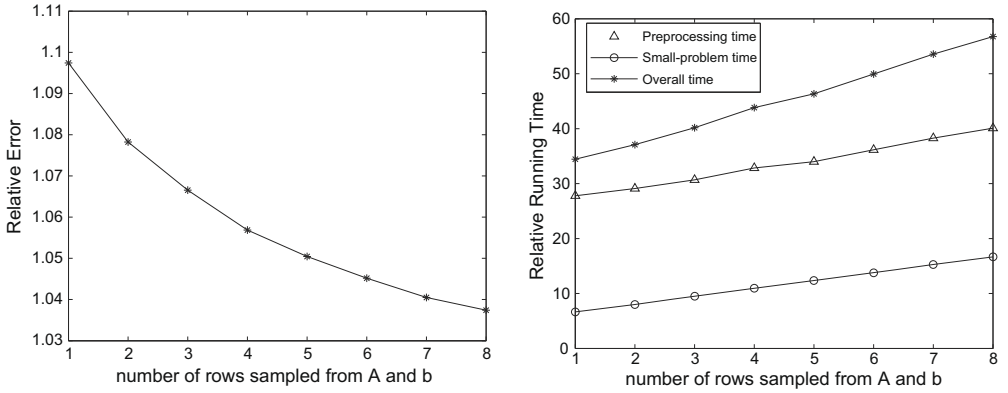

Fig. 1. Average results of the RandomizedNNLS algorithm compared to the algorithm of [15] on 3000 NNLS problems constructed from the TechTC300 dataset. *Relative Error* $:= \|A\tilde{x}_{opt} - b\|_2 \,/\, \|Ax_{opt} - b\|_2$, while *Overall Time* $:= 100T_{\tilde{x}_{opt}}/T_{x_{opt}}$. x_{opt} is computed with the method of [15] in $T_{x_{opt}}$ time, and $\tilde{x}_{opt}$ with the RandomizedNNLS algorithm (the last step employs the method of [15]) in $T_{\tilde{x}_{opt}}$ time. Points one through eight on the x-axis of the plots correspond to values of the parameter $r = d + i \cdot 50$ for $i = 1, \ldots, 8$, where d is the number of columns of A. On the right panel, *Preprocessing time* stands for the cost of the multiplication of A and b with $\widetilde{HD}$ (multiplied by 100 and divided by $T_{x_{opt}}$), and *Small-problem time* stands for the cost of solving the small problem using the algorithm of [15] (multiplied by 100 and divided by $T_{x_{opt}}$). For each point of the x-axis: *Overall time* = *Preprocessing time* + *Small-problem time*.

benchmark set of 300 term-document matrices from ODP, called TechTC300 (Technion Repository of Text Categorization Datasets [13]), which they made publicly available. Each term-document matrix of the TechTC300 dataset consists of a total of 150–400 documents from two different ODP categories, and a total of 15,000–35,000 terms. We chose this dataset because we believe that it does represent an important application area, namely text mining, and we do believe that the results from our experiments will be representative of the potential usefulness of our randomized NNLS algorithm in large, sparse, term-document NNLS problems.

We present average results from 3000 NNLS problems. More specifically, for each of the 300 matrices of the TechTC300 dataset, we randomly choose a column from the term-document matrix as the vector b, we assign the remaining columns of the same term-document matrix to the matrix A, and solve the resulting NNLS problem with inputs A and b. We repeat this process ten times for each term-document matrix of the TechTC300 dataset, and thus solve a total of 3000 problems. Whenever an NNLS routine is called, it is initialized with the all-zeros vector. We evaluate the accuracy and the running time of our algorithm when compared to two standard NNLS algorithms. The first one is described in [15],[3] and the second one is the active set method of [18], implemented as the built-in function *lsqnonneg* in Matlab. We would also like to emphasize that in [15] the authors compare their approach to other NNLS approaches and conclude that their algorithm is significantly faster. Note that the method of [15] operates on the quadratic programming formulation discussed in Section 3.1,[4] while *lsqnonneg* operates on the formulation of Definition 1. Finally, we implemented our RandomizedNNLS algorithm in Matlab. The platform used for the experiments was a 2.0 GHz Pentium IV with 1 GB RAM.

Our (average) results are shown in Fig. 1. We only focus on the algorithm of [15], which was significantly faster, running (on average) in five seconds, compared to more than one minute for the *lsqnonneg* function. We experimented with eight different values of the parameter r, which dictates the size of the small subproblem (see the RandomizedNNLS algorithm). More specifically, we set r to $d + i \cdot 50$, for $i = 1, \ldots, 8$, where d is the number of columns in the matrix A. Our results verify that (i) the RandomizedNNLS algorithm is very accurate, (ii) that it reduces the running time of the NNLS method of [15], and (iii) that there exists a natural tradeoff between the approximation accuracy and

[3] We would like to thank the authors of [15] for providing us with a Matlab implementation of their algorithm.

[4] The actual implementation involves computations of the form $t = Au$ and $s = A^T t$, avoiding the computation and storage of the matrix $A^T A$.

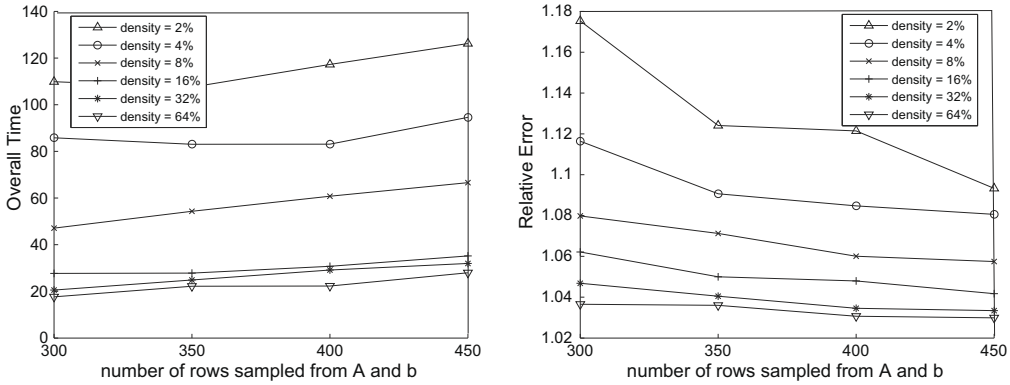

Fig. 2. Average results of the RandomizedNNLS algorithm compared to the algorithm of [15] on six sets of 100 NNLS problems with different density. *Relative Error* $:= \|A\tilde{x}_{opt} - b\|_2 \,/\, \|Ax_{opt} - b\|_2$, while *Overall Time* $:= 100 T_{\tilde{x}_{opt}} / T_{x_{opt}}$. x_{opt} is computed with the method of [15] in $T_{x_{opt}}$ time, and $\tilde{x}_{opt}$ with the RandomizedNNLS algorithm (the last step employs the method of [15]) in $T_{\tilde{x}_{opt}}$ time. For the six density parameters (2%, 4%, 8%, 16%, 32%, 64%), $T_{x_{opt}}$ was on average (0.26 s, 0.40 s, 0.94 s, 2.89 s, 6.27 s, 10.57 s), respectively.

the number of sampled rows. Notice, for example, that the running time of the state-of-the-art NNLS solver of [15] can be reduced from two to three times, while the residual error is from 4% up to 10% worse than the optimal residual error.

We briefly comment on the performance of RandomizedNNLS when compared to the *lsqnonneg* algorithm. As expected, the accuracy results are essentially identical with the method of [15], since both methods solve the NNLS problem exactly. Our speedup, however, was much more significant, ranging from 14-fold to 10-fold for $r = d + 50$ and $r = d + 400$, respectively (data not shown).

4.2. Sparse vs. dense NNLS problems

The astute reader might notice that we evaluated the performance of our algorithm in a rather adversarial setting. The TechTC300 data are quite sparse, hence existing NNLS methods would operate on sparse matrices. However, our preprocessing step in the RandomizedNNLS algorithm destroys the sparsity, and the induced subproblem becomes dense. Thus, we are essentially comparing the time required to solve a sparse, large NNLS problem to the time required to solve a dense, small NNLS problem. If the original problem were dense as well, we would expect more pronounced computational savings. In this section we experiment with random matrices of varying density in order to confirm this hypothesis.

First, it is worth noting that the sparsity of the input matrix A and/or the target vector b do not seem to affect the approximation accuracy of the RandomizedNNLS algorithm. This should not come as a surprise since our results in Theorem 1 do not make any assumptions on the inputs A and b. Indeed, our experiments in Fig. 2 confirm our expectations.

Prior to discussing our experiments on random matrices of varying density, it is worth noting that the NNLS solver of [15] has a running time that is a function of the number of non-zero entries of A. Indeed, the method of [15] is an iterative method where the computational bottleneck in the jth iteration involves computations of the form $A^T Au$, for a d-dimensional vector u. [15] implemented their algorithm by computing the two matrix–vector products Au and $A^T(Au)$ separately, thus never forming the matrix $A^T A$ and thus taking advantage of the sparsity of A. Indeed, NNLS problems with sparse coefficient matrices A are solved faster than NNLS problems with similar-size dense coefficient matrices A by using the method of [15].[5]

[5] The authors of [15] performed extensive numerical experiments to verify that observation; for example see the last row of Table 4 on page 14 in [15] and notice that the running time of their method increases as the density of A increases.

In order to measure how the speedup of our RandomizedNNLS algorithm improves as the matrix A and vector b become denser, we designed the following experiment. First, let the *density* of an NNLS problem denote the percentage of non-zero entries in A and b; for example, $density(A, b) = 10\%$ means that approximately $0.9(nd + n)$ entries in the $n \times d$ matrix A and the $n \times 1$ vector b are zero. We chose six density parameters (2%, 4%, 8%, 16%, 32%, and 64%) and generated 100 NNLS problems for each density parameter. More specifically, we first constructed ten $n \times (d + 1)$ random matrices with the target density (the non-zero entries are normally distributed in $[0, 1]$). Then, for each matrix, we randomly selected one column to form the vector b and assigned the remaining d columns to the matrix A. We repeated this selection process ten times for each of the ten $n \times (d + 1)$ matrices, thus forming a set of 100 NNLS problems with inputs A and b. We fixed the dimensions to $n = 10,000$ and $d = 300$ and we experimented with four values of $r = (d, d + 50, d + 100, d + 150)$. In Fig. 2 we present average results over the 100 NNLS problems for each choice of the density parameter. Notice that increasing the density of the inputs A and b, the computational gains increase as well. On top of that, our method becomes more accurate while the number of the zero entries in A and b become fewer. Notice for example, on the right plot of Fig. 2, when $r = 300$, the two extreme cases ($density = 2\%$ and $density = 64\%$) correspond to an 18% and a 4% loss in accuracy, respectively. Given these two observations as well as the actual times of $T_{x_{opt}}$ (see the caption of Fig. 2), we conclude that the random projection ideas empirically seem more promising for dense rather than sparse NNLS problems.

5. Conclusions

We presented a random projection algorithm for the Nonnegative Least Squares Problem. We experimentally evaluated our algorithm on a large, text-mining dataset, and verified that, as promised in our theoretical findings, practically it does give very accurate approximate solutions, while outperforming two standard NNLS methods in terms of computational efficiency. Future work includes the extension of our theoretical findings of Theorem 1 to NNLS problems with multiple right hand side vectors. An immediate application of this would be the computation of Nonnegative Matrix Factorizations based on Alternating Least Squares type approaches [16,17]. Finally, notice that, since our analysis is independent of the type of constraints on the vector x, our main algorithm can be employed to approximate a least-squares problem with any type of constraints on x.

Acknowledgements

We would like to thank Kristin P. Bennett and Michael W. Mahoney for useful discussions. The first author would like also thank the Institute of Pure and Applied Mathematics of the University of California at Los Angeles for its generous hospitality during the period September 2008–December 2008, when part of this work was done.

References

[1] Open direcory project. <http://www.dmoz.org/>.
[2] N. Ailon, B. Chazelle, Approximate nearest neighbors and the fast Johnson–Lindenstrauss transform, in: ACM Symposium on Theory of Computing, 2006, pp. 557–563.
[3] N. Ailon, E. Liberty, Fast dimension reduction using rademacher series on dual BCH codes, in: ACM-SIAM Symposium on Discrete Algorithms, 2008, pp. 1–9.
[4] A. Björck, Numerical Methods for Least Squares Problems, SIAM, 1996.
[5] R. Bro, S.D. Jong, A fast non-negativity-constrained least squares algorithm, J. Chem. 11 (5) (1997) 393–401.
[6] D. Chen, R. Plemmons, Nonnegativity constraints in numerical analysis, in: Symposium on the Birth of Numerical Analysis, Leuven Belgium, 2007.
[7] A. Dasgupta, P. Drineas, B. Harb, R. Kumar, M.W. Mahoney, Sampling algorithms and coresets for *lp* regression, in: ACM-SIAM Symposium on Discrete Algorithms, 2008, pp. 932–941.
[8] A. Dax, On computational aspects of bounded linear least squares problems, ACM Trans. Math. Softw. 17 (1) (1991) 64–73.
[9] P. Drineas, M. Mahoney, S. Muthukrishnan, Sampling algorithms for l_2 regression and applications, in: ACM-SIAM Symposium on Discrete Algorithms, 2006, pp. 1127–1136.
[10] P. Drineas, M. Mahoney, S. Muthukrishnan, T. Sarlos, Faster least squares approximation, Technical Report, 2007. Available from: <arXiv:0710.1435>.

[11] P. Drineas, M.W. Mahoney, S. Muthukrishnan, Relative-error cur matrix decompositions, SIAM J. Matrix Anal. Appl. 30 (2008) 844–881.
[12] V. Franc, V. Hlavac, M. Navara, Sequential coordinate-wise algorithm for the non-negative least squares problem, in: Computer Analysis of Images and Patterns, 2005, pp. 407–414.
[13] E. Gabrilovich, S. Markovitch, Text categorization with many redundant features: using aggressive feature selection to make SVMs competitive with C4.5, in: International Conference on Machine Learning, 2004, pp. 321–328.
[14] G. Golub, C.V. Loan, Matrix Computations, Johns Hopkins University Press, Baltimore, 1989.
[15] D. Kim, S. Sra, I.S. Dhillon, A new projected quasi-newton approach for solving nonnegative least squares problem, Technical Report CS-TR-06-54, The University of Texas at Austin, 2007.
[16] H. Kim, H. Park, Nonnegative matrix factorization based on alternating non-negativity-constrained least squares and the active set method, SIAM J. Matrix Anal. Appl. 30 (2) (2008) 713–730.
[17] J. Kim, H. Park, Toward faster nonnegative matrix factorization: a new algorithm and comparisons, Technical Report GT-CSE, 2008.
[18] C.L. Lawson, R.J. Hanson, Solving Least Squares Problems, Prentice-Hall, 1974.
[19] M. Rudelson, R. Vershynin, Sampling from large matrices: an approach through geometric functional analysis, J. ACM 54 (4) (2007).
[20] M.M.S. Bellavia, B. Morini, An interior point newton-like method for non-negative least squares problems with degenerate solution, Numer. Linear Algebra Appl. 13 (2006) 825–844.
[21] T. Sarlos, Improved approximation algorithms for large matrices via random projections, in: IEEE Symposium on Foundations of Computer Science, 2006, pp. 143–152.
[22] F. Sha, L.K. Saul, D.D. Lee, Multiplicative updates for nonnegative quadratic programming in support vector machines, Neurocomputing 71 (1–3) (2007) 363–373.