Blendenpik: Randomized Least Squares

Project - Computational Linear Algebra, EPFL, Spring 22

Moritz Waldleben

May 2022

1 Introduction

Blendenpik [1] is a randomized algorithm for linear least-squares problems. The method constructs a random projection preconditioner for an iterative solver (LSQR) and converges in fewer steps than an unpreconditioned LSQR. Moreover, Blendenpik can beat the standard LAPACK implementation for very large matrices.

Instead of the LSQR algorithm used in the paper [1], we will use the minimum residual method (MINRES) with preconditioning. In the following report, we will discuss the main concepts of Blendenpik and illustrate some results with a custom MINRES implementation.

2 Motivation

Let us look at a large overdetermined system: $A \in \mathbb{R}^{m\times n}, b \in \mathbb{R}^m, m >> n$, and $\operatorname{rank}(A) = n$. The corresponding linear least-squares problem can be written as

$$\min_{x\in\mathbb{R}^n} \|A\mathbf{x} - \mathbf{b}\|_2 \tag{1}$$

One might think that the linear system has redundant information. We could thus try to solve a smaller system and sample $\mathcal{S}$ rows of the matrix to get an approximated solution.

$$x_{\mathcal{R}} = \arg\min_x \|A(\mathcal{S}, :)x - b(\mathcal{S})\|_2 \tag{2}$$

This reduction technique is rarely implemented, in practice, because of lacking useful bounds for the forward error (See [1] for a detailed explanation). An alternative way is to construct a preconditioner for the complete linear system 1 using randomly selected rows. A reduced QR factorization of the randomly sampled matrix $A(\mathcal{S}, :)$ is a suitable preconditioner. This property is further discussed in section 4. To implement Blendenpik using MINRES we would need to apply the iterative solver to the normal equation of the least-squares problem. The commented implementation of the complete algorithm is added in appendix A. Below an overview of the algorithm is shown.

Algorithm 1 Blendenpik overview (using MINRES)

```
1. Choose $\mathcal{S}$ of size $r$, a subset of rows of $A$
2. Calculate $A(\mathcal{S},:) = QR$
3. Calculate $A(\mathcal{S},:)A(\mathcal{S},:)^T$
4. MINRES to $AA^T$ with right and left preconditioner $R^{-T}, R^{-1}$
```

3 Coherence of a matrix

Now we will focus on how to randomly sample rows from a given matrix. To tackle this, we have to look at the coherence of a matrix A. Coherence tells us how much the solution of the system will depend on a single row of A. Formally coherence can be defined as

$$\mu(A) = \max_i \|U(i,:)\|_2^2, \tag{3}$$

U corresponds to an orthonormal matrix which spans the column space of A. We have as well that $\frac{n}{m} \leq \mu(A) \leq 1$.

To illustrate this definition, we will look at the coherence of a randomly generated orthogonal matrices of size $\mathbb{R}^{100\times 50}$.

```
% mean coherence of 1000 random matrices
rng(11);
coherences = [];
for i=1:1000
    U = orth(rand(1000, 50));
    coherence = max(sum(U.^2, 2));
    coherences = [coherences, coherence];
end
disp(mean(coherences))

>> 0.0720
```

The Matlab script above computes the average coherence number of randomly generated orthogonal matrices. As comparison, the minimal coherence for matrix for A is $\mu_{min}(A) = \frac{50}{1000} = 0.05$ from the definition in 3. The proof of the corresponding the bounds can be found in the B.

The result shows that the coherence number is nearly optimal for randomly generated matrices. On the other hand, we can expect this, as all elements of the matrix are created using a uniform distribution.

Let's now think of an extreme case where the coherence number is maximal. We could set all elements of a column of A to zero except one entry. The result is a coherence of $\mu(A) = 1$. More intuitively, this means that the row with the nonzero entry has to be sampled to capture the "essence or direction" of the column with one nonzero entry.

In contrast to a uniform sampled matrix discussed above, general matrices A will have high coherence numbers. But a good preconditioner obtained from random sampling should still resemble A and result in a low conditioner number for the iterative solver (see theorem 1). In summary, we have to control the coherence of A to obtain a solid preconditioner.

A technique to do so is preprocessing A with randomized row mixing. The rows of A are blended (therefore the name Blendenpik) and the obtained coherence of $A(\mathcal{S})$ drops. Here several strategies are possible. One of them is multiplying each row with $+1$ or -1. Then a discrete cosine transform (DCT) is applied to each column. In addition, the first row is multiplied by $\sqrt{2}$ to get an orthogonal transformation. The result is to reduce the coherence and a solid preconditioner can still be obtained.

(TODO: include equation for DCT)

4 Bound for the condition number

The obtained preconditioner R from the discussions in the previous chapter is just useful if we can bound the condition number for the preconditioned iterative solver. This ensures convergences in a few steps. In the case of the MINRES version of Blendenpik, we would replace A with AA^T and R with RR^T. This adaption will lead to similar bounds as the ones shown below.

The following gives us a desired bound and establishes a link between the condition and coherence number.

Theorem 1 *Let A be an $m \times n$ full rank matrix, and let $\mathcal{S}$ be a random sampling operator that samples $r \geq n$ rows from A uniformly. Let $\tau = C\sqrt{m\mu(A)\log(r)/r}$, where C is some constant defined in the proof. Assume that $\delta^{-1}\tau < 1$. With probability of at least $1-\delta$, the sampled matrix $\mathcal{S}A$ is full rank, and if $\mathcal{S}A = QR$ is a reduced QR factorization of $\mathcal{S}A$, we have*

$$\kappa\left(AR^{-1}\right) \leq \sqrt{\frac{1+\delta^{-1}\tau}{1-\delta^{-1}\tau}} \tag{4}$$

To prove the above we will state a 2 theorems which correspond to theorem 1 [2] and theorem 7 in [3] (less general and rephrased for our problem).

Theorem 2 *Suppose that l, m, and n are positive integers such that $m \geq l \geq n$. Suppose further that A is a full-rank $m \times n$ matrix, and that the SVD of A is*

$$A_{m\times n} = U_{m\times n}\Sigma_{n\times n}V^*_{n\times n}.$$

Suppose in addition that T is an $l \times m$ matrix such that the $l \times n$ matrix TU has full rank. Then, there exists an $n \times n$ matrix P, and an $l \times n$ matrix Q whose columns are orthonormal, such that

$$T_{l\times m}A_{m\times n} = Q_{l\times n}P_{n\times n}. \tag{5}$$

Furthermore, if P is any $n \times n$ matrix, and Q is any $l \times n$ matrix whose columns are orthonormal, such that P and Q satisfy , then the condition numbers of AP^{-1} and TU are equal.

Theorem 3 *Suppose $A \in \mathbb{R}^{m\times n}$, and $c \leq n$. Construct C with Algorithm 6, using the EXPECTED (c) algorithm. If the sampling probabilities $\{p_i\}_{i=1}^n$ used by the algorithm are of the form (44) or (45), then*

$$\mathbf{E}\left[\left\|AA^T - CC^T\right\|_2\right] \leq O(1)\sqrt{\frac{\log c}{\beta c}}\|A\|_F\|A\|_2.$$

To relate this theorem with everything introduced so far some clarifications are needed. First of all the mentioned EXPECTED (c) algorithm corresponds to (TODO: include algorithm) to a row sampling algorithm. Every column of a matrix A is sampled with probability cp_i and it holds that $\sum_{i=1}^n p_i = 1$. Hence in expectation c columns are sampled. After that a rescaling of all entries is done using a diagonal matrix D.

Secondly we have to clarify how the parameter $\beta \in (0, 1]$ is established. We have that

$$p_i \geq \beta\frac{\left\|A^{(i)}\right\|_2^2}{\|A\|_F^2} \tag{6}$$

We will now prove the theorem 1. First of all we substitute the general matrix A with the transpose U^T of an orthonormal matrix $U \in \mathbb{R}^{m\times n}$ where we know that $U^TU = I$. In this case we will sample c (TODO: renaming, clarify) rows in expectation. If we set the sample probability distribution for all rows equal we get ? meaning the diagonal elements of D are Secondly for $C = U^TSD$. If we apply a scaling of $D = \sqrt{\frac{m}{c}}$ (TODO: explain) then $C = \sqrt{\frac{m}{c}}U^TS$. Thus

$$\mathbf{E}\left[\left\|I_{n\times n} - \frac{m}{c}U(\mathcal{S},:)^TU(\mathcal{S},:)\right\|_2\right] \leq O(1)\sqrt{\frac{\log c}{\beta c}}\|U^T\|_F\|U^T\|_2 \tag{7}$$

Now we have to work on the right bound. We know that $|U^T\|_2 \leq \|U^T\|_F = \sqrt{m}$. We can further develop the inequality 6. As we have a bound on the coherence.

$$\frac{1}{\beta} \geq \frac{\left\|U^{T(i)}\right\|_2^2}{\|U^T\|_F^2 p_i} \tag{8}$$

TODO: Not sure how to continue here

Putting this together the finale equation becomes

$$\mathbf{E}\left[\left\|I_{n\times n} - U(\mathcal{S},:)^T U(\mathcal{S},:)\right\|_2\right] \leq C\sqrt{m\mu(A)\log(c)/c} \tag{9}$$

Where the right part of the inequality corresponds to the τ of theorem 1 which we wanted to prove.

From theorem 3 we know that the condition numbers $\kappa(AR^{-1})$ and $\kappa(U)$ are equal.

Now the we have to work on the left side of the inequality. We can a apply the Markov's inequality and get

$$\mathbb{P}(\left[\left\|I_{n\times n} - U(\mathcal{S},:)^T U(\mathcal{S},:)\right\|_2\right] \geq \tau^{-1}\delta) \leq \delta \tag{10}$$

With probability $\geq 1-\delta$ we thus have

$$\left\|I_{n\times n} - U(\mathcal{S},:)^T U(\mathcal{S},:)\right\|_2 \leq \tau^{-1}\delta \leq 1 \tag{11}$$

The bound 1 comes from the assumption in theorem 1.

Now we look at the eigenvalues of $U(\mathcal{S},:)^T U(\mathcal{S},:)$.... TODO: Rayleigh quotient argument

Which finally gives us the desired result.

5 Implementation

6 Numerical experiments

References

[1] Haim Avron and Petar Maymounkov. Blendenpik: Supercharging lapack's least-squares solver. *SIAM J. Scientific Computing*, 32:1217–1236, 01 2010.

[2] Petros Drineas, Michael Mahoney, and Senthilmurugan Muthukrishnan. Relative-error *cur* matrix decompositions. *SIAM Journal on Matrix Analysis and Applications*, 30:844–881, 05 2008.

[3] Vladimir Rokhlin and Mark Tygert. A fast randomized algorithm for overdetermined linear least-squares regression. *Proceedings of the National Academy of Sciences of the United States of America*, 105:13212–7, 09 2008.

A Matlab implementation Blendenpik

B Coherence bound proves