Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)

Accelerating Random Kaczmarz Algorithm Based on Clustering Information

Yujun Li and **Kaichun Mo** and **Haishan Ye**
Department of Computer Science and Engineering
Shanghai Jiao Tong University
{liyujun145,yhs12354123}@gmail.com, daerduomkch@sjtu.edu.cn

Abstract

Kaczmarz algorithm is an efficient iterative algorithm to solve overdetermined consistent system of linear equations. During each updating step, Kaczmarz chooses a hyperplane based on an individual equation and projects the current estimate for the exact solution onto that space to get a new estimate. Many vairants of Kaczmarz algorithms are proposed on how to choose better hyperplanes. Using the property of randomly sampled data in high-dimensional space, we propose an accelerated algorithm based on clustering information to improve block Kaczmarz and Kaczmarz via Johnson-Lindenstrauss lemma. Additionally, we theoretically demonstrate convergence improvement on block Kaczmarz algorithm.

1 Introduction

In many real applications, we will face with solving the overdetermined consistent system of linear equations of the form $Ax = b$, where $A \in \mathbb{R}^{n \times p}$ and $b \in \mathbb{R}^n$ are given data, and $x \in \mathbb{R}^p$ is unknown vector to be estimated. If A has a small size, we could directly solve the problem by computing pseudo-inverse, $x = A^{\dagger}b$. However, if A is of large size, either we cannot store it in the main memory or it is extremely time-consuming to compute the pseudo-inverse. Fortunately, in such cases, the Kaczmarz algorithm can be used to solve the problem, since we can update our current estimate to the exact solution x_* iteratively by only using a small fraction of entire data each time.

Recently, many variants of the classical Kaczmarz algorithm (Kaczmarz 1937) are proposed by researchers. Classical Kaczmarz algorithm performs each iterative step by selecting rows of A in a sequential order. Despite the power of it, theoretical guarantee for its rate of convergence is scarce. However, a randomized version of Kaczmarz algorithm (Strohmer and Vershynin 2009), denoted as RKA in this paper, yields provably exponential rate of convergence in expectation by sampling rows of A at random, with probability proportional to Euclidean norm. Recently, more accelerated variants of Kaczmarz algorithms are put forth by researchers. With the usage of well-known Johnson-Lindenstrauss lemma (Johnson and Lindenstrauss 1984), Eldar and Needell proposed RKA-JL algorithm (Eldar and Needell 2011), which selects the optimal update from a randomly chosen set of linear equations during each iteration. Empirical studies demonstrate an improved rate of convergence than former methods. Moreover, along another direction of researches on accelerating Kaczmarz algorithm, multiple rows of A are utilized at the same time to perform one single updating step. These RKA-Block methods (Elfving 1980; Needell and Tropp 2014; Briskman and Needell 2014) use subsets of constraints during iterations so that an expected linear rate of convergence can be provably achieved if utilizing randomization as well. However, geometric properties of blocks are important. Blocks with bad-condition number will interfere the rate of convergence.

Since Kaczmarz based algorithms are widely used nowadays (Natterer 1986; Sezan and Stark 1987; Dai, Soltanalian, and Pelckmans 2014; Thoppe, Borkar, and Manjunath 2014), many other branches of research on Kaczmarz are developed. In terms of inconsistent case, many useful methods (Needell 2010; Zouzias and Freris 2013) are proposed. When A is a large sparse matrix, there will appear a big fill-in, modified Kaczmarz via clustering Jaccard and Hamming distances can overcome this problem (Pomparău and Popa 2014; Pomparău 2013). However in this paper, we consider solving consistent system of linear equations with high-dimensional matrix A following Gaussian distribution.

One greedy idea highlighted by (Eldar and Needell 2011) emphasize the importance of choosing hyperplanes that give the furthest distance to the current estimate during updating steps. Many methods such as (Eldar and Needell 2011) are proposed to approximately utilize this idea by considering acceptable number of hyperplanes and selecting the furthest one. In this paper, we utilize clustering methods to gain more insight on where is the furthest hyperplane. Eventually, we incorporated clustering algorithms into one version of randomized Kaczmarz algorithm(RKA-JL) and block Kaczmarz algorithm(RKA-Block) in order to better approximate that optimal plane.

It is well-known that high-dimensioanl data points randomly sampled according to Gaussian distribution are much likely be orthogonal to each other. One can refer to (Blum, Hopcroft, and Kannan 2015) and Theorem 1 for details. Even though real data entities in high dimension are not ideally sampled from Gaussian distribution, they still tend to stretch along with different axes. In this paper, we cluster rows of A into different classes so that the center points of clusters tend to be orthogonal. After capturing the clustering information, we may first measure the distances between the centroids of clusters to the current estimate within a tolerable amount of time to gain knowledge about distances from the current estimate to all hyperplanes determined by rows of A. Then, we may further select one sample from the furthest cluster to approximate the unknown furthest hyperplane.

The paper offers the following contributions:

- After applying clustering method and utilizing clustering information, we improve RKA-JL and RKA-Block algorithms to speedup their convergences. The empirical experiments show the improvement clearly.
- Theoretically, we prove that clustering method could improve convergence of RKA-Block algorithm. In addition, we coarsely analyze the modified RKA-JL (RKA-Cluster-JL) in terms of runtime.

The remainder of the paper are organized as follows. In section 2, we give a short overview of two relevant algorithms proposed in recent years, which should be helpful in understanding our algorithms. Section 3 shows how to use clustering method to improve RKA-JL algorithm and RKA-Block algorithm. We prove theoretically that clustering method could improve convergence of RKA-Block algorithm. In section 4, we conduct some numerical experiments to show the improved performance of our algorithms. Section 5 concludes the paper briefly.[1]

2 Related Work

Consider the overdetermined consistent linear equation system $Ax = b$, where $A \in \mathbb{R}^{n \times p}$, $b \in \mathbb{R}^n$ and $x \in \mathbb{R}^p$. Denote x_0 the initial guess of x, $A_1, A_2, ..., A_n$ the rows of data matrix A and $b_1, b_2, ..., b_n$ the components in b. Besides, let x_* denotes the optimal value such that it satisfies $Ax_* = b$.

In classical Kaczmarz algorithm (Kaczmarz 1937), rows are iteratively picked in a sequential order. Denote x_k to be the estimate of x_* after k iterations. At the kth updating step, we first pick one hyperplane $A_i x = b_i$ and then x_k can be calculated by the former estimate x_{k-1} and the picked hyperplane as follows.

$$x_k = x_{k-1} + \frac{b_i - \langle A_i, x_{k-1} \rangle}{\|A_i\|_2^2} A_i \tag{1}$$

Our work mainly relies on RKA-JL algorithm (Eldar and Needell 2011) and RKA-Block algorithm (Needell and Tropp 2014). We will review these algorithms below.

2.1 RKA-JL

During each updating step, Kaczmarz algorithm chooses a hyperplane to project on. Thus, it is a combinatoric problem if we want to select out the optimal sequence of projecting hyperplanes and achieve the fastest convergence to the exact solution x_*. Unfortunately, such problem is too complicated to be solvable in reasonable amount of time. But, some greedy algorithm whose running time is affordable may be used to approximate the optimal sequence of hyperplanes and achieve a quite excellent convergence rate.

Before introducing a greedy algorithm proposed by (Eldar and Needell 2011), we have to note a key property shared by all Kaczmarz related algorithms. Recall that Kaczmarz algorithm iteratively update the current estimate to a new one by projecting on to one hyperplane. It is easy to observe that the Euclidean distance between the current estimate and the solution is monotonically decreased, which means that

$$\|x_{k+1} - x_*\|_2 \le \|x_k - x_*\|_2$$

This is obvious because

$$\begin{aligned} \|x_{k+1} - x_*\|_2^2 &= \|x_k - x_*\|_2^2 - \|x_{k+1} - x_k\|_2^2 \\ &\le \|x_k - x_*\|_2^2 \end{aligned}$$

Thus, if we can find a way to maximize $\|x_{k+1} - x_k\|_2$ at each iteration, we may achieve better convergence rate at last.

The exact greedy idea has been highlighted in (Eldar and Needell 2011) when they proposed RKA-JL algorithm. The idea is quite simple but reasonable. At the k-th updating step, we can choose the hyperplane $A_i x = b_i$ to project on where $A_i = \arg\max_{A_i} \|x_k - x_{k-1}\|_2$ and update the estimate as Eq(1). But, after thinking of the practical issue, we may find that it is unaffordable to sweep through all rows of A to pick the best one at each iteration. Thus, (Eldar and Needell 2011) proposed a procedure to approximate the best hyperplane before performing each updating step, which is the key component of RKA-JL algorithm.

Instead of sweeping through the whole data and comparing the update $\|x_{k+1} - x_k\|_2$, RKA-JL algorithm (Eldar and Needell 2011) selects p rows from A with probability $\|A_i\|_2^2 \,/\, \|A\|_F^2$, utilizes Johnson-Lindenstrauss lemma to approximate the distance $\|x_{k+1} - x_k\|_2 = \frac{|b_i - \langle A_i, x \rangle|}{\|A_i\|_2}$ and then choose the maximized one as the row to determine the projecting hyperplane. Besides, a testing step is launched to ensure that the chosen hyperplane will not be worse than that of classical.

In the initialization step, the multiplication of A and Φ costs $O(npd)$ time. And it takes $O(np)$ time to compute the sampling probability, which only needs to be computed once and coule be used in the following selection steps during each updating step. The selection step costs $O(pd)$ time, while the testing and updating step clearly only cost $O(p)$ time. Therefore each iteration costs $O(pd)$ time. The algorithm converge (in expectation) in $O(p)$ iterations, then the overall runtime is $O(npd + dp^2)$.

The whole process is given in Algorithm 1.

In fact, RKA-JL algorithm is only a weak approximation to the greedy idea mentioned above, since it only takes into consideration a small fraction of data each time. It is obvious that the greater value of d and row selection number will give greater improvements, but at the expense of greater computation cost at each iteration. There is a trade-off between improvement and computational expense. However, after utilizing clustering methods, more data can be used in

[1]Missing proof details can be found at the full paper http://arxiv.org/abs/1511.05362.

Algorithm 1 RKA-JL

```
1: Input: $A \in \mathbb{R}^{n\times p}$, $b \in \mathbb{R}^n$. Solve consistent linear equation system $Ax = b$, where $x \in \mathbb{R}^p$.
2: Initialization Step: Create a $d \times n$ Gaussian matrix $\Phi$ and set $\alpha_i = \Phi A_i$. Initialize $x_0$. Set $k = 0$.
3: Selection Step: Select $p$ rows so that each row $A_i$ is chosen with probability $\|A_i\|_2^2 / \|A\|_F^2$. For each row selected, calculate
       $$\gamma_i = \frac{|b_i - \langle \alpha_i, \Phi x_k\rangle|}{\|\alpha_i\|_2}$$
   and set $j = \arg\max_i \gamma_i$.
4: Test Step: Select the first row $A_l$ out of the $n$, explicitly calculate
       $$\hat{\gamma_j} = \frac{|b_j - \langle A_j, x_k\rangle|}{\|A_j\|_2} \quad \text{and} \quad \hat{\gamma_l} = \frac{|b_l - \langle A_l, x_k\rangle|}{\|A_l\|_2}$$
   If $\hat{\gamma_l} > \hat{\gamma_j}$, set $j = l$.
5: Update Step: Set
       $$x_{k+1} = x_k + \frac{b_j - \langle A_j, x_k\rangle}{\|A_j\|_2^2} A_j$$
6: Set $k = k + 1$. Go to step 3 until convergence.
```

selecting the best hyperplane while keeping the computational cost in an acceptable amount. We will illustrate our modified RKA-Cluster-JL algorithm in section 3.1.

2.2 RKA-Block

Another direction of researches related with Kaczmarz lies on the utilization of multiple rows of A to update at each updating step. In (Elfving 1980; Needell and Tropp 2014) block versions of Kaczmarz algorithm are proposed. Instead of just using one hyperplane at each updating step, block Kaczmarz uses multiple hyperplanes. To be specific, when updating x_k, we may project the old estimate x_{k-1} using A_τ which is a submatrix in A and its corresponding b_τ via $x_k = x_{k-1} + (A_\tau)^\dagger(b_\tau - A_\tau x_{k-1})$.

Algorithm 2 RKA-Block

```
1: Input: $A \in \mathbb{R}^{n\times p}$, $b \in \mathbb{R}^n$. Solve consistent linear equation system $Ax = b$, where $x \in \mathbb{R}^p$.
2: Initialization Step: Initialize $x_0$. Set $k = 0$.
3: Selection Step: Uniformly choose some $\tau$ rows to construct a matrix block $A_\tau$.
4: Update Step: Set
       $$x_{k+1} = x_k + (A_\tau)^\dagger(b_\tau - A_\tau x_k)$$
5: Set $k = k + 1$. Go to step 3 until convergence.
```

Block Kaczmarz algorithm selects several hyperplanes and project x_k onto the intersection of several hyperplanes. This procedure acts exactly the same as projecting the current estimate in hyperplanes iteratively until convergence. However, only one updating step is required to achieve that, while iteratively bouncing between these hyperplanes takes time. See Algorithm 2 for details.

While block Kaczmarz provably expected linear rate of convergence, it remains a problem on how to choose rows to construct blocks so that they are well-conditioned. We will show in section 3.2 that utilizing clustering information helps a lot. Besides, theoretical guarantee is given to demonstrate the acceleration of our modified RKA-Cluster-Block algorithm.

3 Methodology and Theoretical Analysis

In this section we will introduce our accelerated algorithms and give some theoretical analysis. Our observation is that high-dimensional Gaussian distributed data tends to stretch in nearly orthogonal clusters. See theorem below for Theoretical proof.

Theorem 1. *u and v are two vectors in $\mathbb{R}^d$. Suppose each entry of u and v are sampled from Gaussian distribution $\mathcal{N}(0, \sigma^2)$. Then one has the probability inequality*

$$P(\frac{|u^T v|}{\|u\|_2 \|v\|_2} \leq \epsilon) \geq \left(1 - \frac{1}{\epsilon^2(1-\delta)^4 d}\right)\left(1 - 6e^{-c\delta^2 d}\right)$$

where $\delta \in [0, 1]$ and c is a fixed constant.

Proof. (Sketch) Denote E_u the event that u holds inequality $\sigma(1-\delta)\sqrt{d} \leq \|u\|_2 \leq \sigma(1+\delta)\sqrt{d}$. Via Bayes' theorem we have $P(|u^T v| \leq \epsilon \|u\|_2 \|v\|_2) = P(|u^T v| \leq \epsilon \|u\|_2 \|v\|_2 | \overline{E_u} \cap \overline{E_v}) P(\overline{E_u} \cap \overline{E_v})$. We apply Chebyshev's inequality to bound the probability $P(|u^T v| \leq \epsilon \|u\|_2 \|v\|_2 | \overline{E_u} \cap \overline{E_v})$. Together with bound on the probability $P(\overline{E_u} \cap \overline{E_v})$, we finish the proof. We leave details to the full paper. □

Therefore, two randomly chosen items from a high-dimensional Gaussian distributed data have a high probability to be perpendicular. Even though real data entities in high dimension are not ideally sampled from Gaussian distribution, they still tend to stretch along with different axes (Blum, Hopcroft, and Kannan 2015). Thus, they can be grouped into different clusters where distances to each other are quite large. Moreover, if the data actually can be embedded in a small dimensional space, the number of clusters should be close to the rank of the space, and thus quite small. Thus, if we can measure the property of these clusters within affordable amount of time, we can quickly master some knowledge on the entire data set.

Base on this observation, we give two accelerating algorithms via taking advantage of clustering. The clustering algorithm needs not to be specific. Any clustering algorithm with runtime no more than $O(npd + dp^2)$ is acceptable. In practice, we use a k-means clustering method(King 2012). Clustering A's rows into k cluster will cost $O(knp)$. In experiments, $k = 4$. It can be absorbed in the order of RKA-JL's computing operation numbers. Therefore the runtime on clustering is acceptable.

3.1 RKA-Cluster-JL

To perform each projection, Kaczmarz select one hyperplane $A_i x = b_i$ to be projected on. We should notice that the hyperplane is uniquely determined by normal vector $A_i / \|A_i\|_2$. To see this, recall Eq(1). If we scale A_i and b_i by multiplying a constant c, it will not change the updating result. Besides, since all hyperplanes go through the point x_*, we can uniquely calculate out $b_i = A_i x_*$. Thus, it is enough to only consider the normal vectors of rows of A with unit length when choosing hyperplane to be projected on.

In this section, we will utilize clustering method to improve RKA-JL algorithm proposed by (Eldar and Needell 2011). We conduct a clustering algorithm on A to cluster the rows into k clusters, each of which has a representive vector or cluster center A_{c_j}, $j \in \{1, 2, ..., k\}$. At each iteration, we first choose the cluster representive vector A_{c_j} which gives the maximized update $\|x_{k+1} - x_k\|_2$. Then in the jth cluster we randomly choose p rows with probability proportional to $\|A_i\|_2^2 / \|A\|_F^2$. This procedure could help us on better approximating the furthest hyperplane since good planes are more likely to be in the furthest cluster. The whole process is detailedly stated in Algorithm 3.

The most difference between Algorithm 1 and Algorithm 3 is that Algorithm 3 chooses the best cluster representive point first, and then process as the same in Algorithm 1.

Ideally, if the furthest hyperplane to the current estimate lies in the furthest cluster, Algorithm 3 doesn't spend time to consider data points in other clusters at all, while Algorithm 1 still does that. If the total budget for sampled rows each time is fixed to s, Algorithm 3 spends $(s-c)$ of them searching in the right cluster, while Algorithm 1 only spends s/c, which drops the probability of it to find better hyperplane compared with Algorithm 3.

To theoretically analysize our algorithm, we propose the following proposition.

Proposition 2. *$A \in \mathbb{R}^{n\times p}$ is a row-normalized matrix, whose rows have unit length. Suppose the row vectors of A are uniformly distributed in the high-dimensional space. Cluster these row vectors by directions into $k = O(\log(p))$ clusters, each of which has $t = \frac{n}{k}$ rows. Among k clustering representive vectors, let A_c be the one maximizing the update $\|x_{k+1} - x_k\|_2^2$. Suppose the rows in the cth cluster have bigger updates than rows in other clusters. In RKA-JL, it set $d = O(\log(p))$ for Gaussian matrix $\Phi \in \mathbb{R}^{d\times p}$. Then the utility of the RKA-JL algorithm comparing kp rows to find a maximized one in $O((\log(p))^2 p)$ time is the same of the utility of RKA-Cluster-JL algorithm comparing $k + p$ rows in $O(\log(p)p)$ time.*

Since the updates are only determined by the directions of the rows, roughly speaking, the rows with similar directions in the same cluster will have similar updates. Therefore the rows in cluster c tends to have bigger updates than the rows in other cluster. This is essentially the key idea behind this algorithm.

3.2 RKA-Cluster-Block

In this subsection, we will apply clustering method to block Kaczmarz. We will show that by using the clustering information we can easily construct well-conditioned matrix blocks doing favor in the convergence analysis of block Kaczmarz.

Algorithm 3 RKA-Cluster-JL

```
1: Input: $A \in \mathbb{R}^{n\times p}$, $b \in \mathbb{R}^n$.
2: Output: Solve consistent linear equation system $Ax = b$, where $x \in \mathbb{R}^p$.
3: Initialization Step: Create a $d \times p$ Gaussian matrix $\Phi$ and set $\alpha_i = \Phi A_i$. Conduct a clustering algorithm in the rows of $A$, resulting in $c$ clusters with representive points $A_{\mathbf{c}_l}$, $l = \{1, 2, ..., c\}$. Initialize $x_0$. Set $k = 0$.
4: Selection Step: Calculate
       $$\hat{x_k} = \Phi x_k$$
   For each representive point, calculate
       $$r_l = \frac{|b_i - \langle A_{\mathbf{c}_l}, x_k\rangle|}{\|A_{\mathbf{c}_l}\|_2}$$
   and set $t = \arg\max_l r_l$.
   Select $p$ rows so that each row $A_i$ is chosen with probability $\|A_i\|_2^2 / \|A\|_F^2$ in the $t$th cluster. For each row selected, calculate
       $$\gamma_i = \frac{|b_i - \langle \alpha_i, \Phi x_k\rangle|}{\|\alpha_i\|_2}$$
   and set $j = \arg\max_i \gamma_i$.
5: Test Step: Select the first row $a_l$ out of the $n$, explicitly calculate
       $$\hat{\gamma_j} = \frac{|b_j - \langle a_j, x_k\rangle|}{\|a_j\|_2} \quad \text{and} \quad \hat{\gamma_l} = \frac{|b_l - \langle a_l, x_k\rangle|}{\|a_l\|_2}$$
   If $\hat{\gamma_l} > \hat{\gamma_j}$, set $j = l$.
6: Update Step: Set
       $$x_{k+1} = x_k + \frac{b_j - \langle a_j, x_k\rangle}{\|a_j\|_2^2} a_j$$
7: Set $k \leftarrow k + 1$. Go to step 4 until convergence.
```

Lemma 3. *(Needell and Tropp 2014). Suppose A is a matrix with full column rank that admits an (m, α, β) row paving T. Consider the least-squares problem*

$$\min \|Ax - b\|_2^2$$

Let x_* be the unique minimizer, and define $e := Ax_* - b$. Then for randomized block Kaczmarz method, one has

$$\mathbb{E}[\|x_j - x_*\|_2^2] \leq \left[1 - \frac{\sigma_{\min}^2(A)}{\beta m}\right] \|x_0 - x_*\|_2^2 + \frac{\beta}{\alpha}\frac{\|e\|_2^2}{\sigma_{\min}^2(A)}$$

where an (m, α, β) row paving is a partition $T = \{\tau_1, ..., \tau_m\}$ of row indices that satisfices

$$\alpha \leq \lambda_{\min}(A_{\tau_i} A_{\tau_i}^T) \quad \textit{and} \quad \lambda_{\max}(A_{\tau_i} A_{\tau_i}^T) \leq \beta$$

for each $\tau_i \in T$.

From the lemma above, we notice that the convergence rate of block Kaczmarz algorithm highly depends on the spectral norm β and condition number β/α of the block matrix. The smaller value of β and β/α will give us a faster convergence rate. See Algorithm 4 for our entire proposed algorithm.

Algorithm 4 RKA-Cluster-Block

```
1: Input: $A \in \mathbb{R}^{n \times p}$, $b \in \mathbb{R}^n$.
2: Output: Solve consistent linear equation system $Ax = b$, where $x \in \mathbb{R}^p$.
3: Initialization Step:
   Clustering: Conduct a clustering algorithm in the rows of $A$, resulting in $c$ clusters with representive vectors $A_{\mathbf{c}_l}, l = \{1, 2, ..., c\}$
   Partition: Randomly extract one row of each cluster and compose to a row submatrix $A_{\tau i}$ from $A$, where $i \in \{1, 2, ..., T\}$. And denote the corresponding values as $b_{\tau i}$.
   Setting: $k = 0$, $x_k = 0$, $N$ the number of iteration.
4: Selection Step: Uniformly Select $A_{\tau i}$ from $\{A_{\tau_1}, A_{\tau_2}, ..., A_{\tau_T}\}$.
5: Update Step: Set
       $x_{k+1} = x_k + (A_\tau)^\dagger (b_\tau - A_\tau x_k)$
6: Set $k = k + 1$. Go to step 4 until convergence.
```

Next, we will theoretically show that our algorithm have a better convergence rate under a mild assumption that data are sampled according to high-dimensional Gaussian distribution. The runtime analysis is similar to RKA-Cluster-JL.

To measure the orthogonality between matrix rows, we define the orthogonality value.

Definition 4 (Orthogonality Value)**.** *A is an $k \times p$ matrix. Let $\hat{A}$ be the matrix after normalizing rows of A. Then each row of $\hat{A}$ has unit length. Define Orthogonality Value*

$$ov(A) = \max_{i \neq j} |\langle \hat{A}_i, \hat{A}_j \rangle|$$

where $\hat{A}_i$ is the ith row of $\hat{A}$.

Clearly, the inequality $0 \le ov(A) \le 1$ holds for any matrix A. Take some examples to get a close look, $ov(I) = 0$, $ov(\text{ones}^2(5, 5)) = ov(\text{ones}(5, 5)/\sqrt{5}) = 1$. Then we could give a upper bound on spectral norm below.

Theorem 5. *$A \in \mathbb{R}^{k \times p}$, $A_i \in \mathbb{R}^p$ is the ith row of A, $\|A_i\|_2 = 1$ for all $i \in \{1, 2, ..., k\}$. Suppose the orthogonality value $ov(A) \le \epsilon$, then one has*

$$\left\|AA^T\right\|_2 \le 1 + k\epsilon.$$

Proof. (Sketch) Using the fact that $\|X\|_2^2 \le \|X\|_1 \|X\|_\infty$, we bound $\left\|AA^T\right\|_1$ and $\left\|AA^T\right\|_\infty$ respectively. We leave details to the full paper. □

²Matlab notation.

This theorem gives us an upper bound of spectral norm to the matrix with bound on orthogonality value. Lower orthogonality value corresponds to that A's rows are almost perpendicular. Thus, we can conclude that selecting rows from each cluster to construct block matrices can give us small spectral norm.

Theorem 6. *Let A be a $k \times p$ row-normalized matrix. Suppose $|\langle A_i, A_j \rangle| \ge \delta$ for all $i, j \in \{1, 2, ..., k\}$, then one has*

$$\left\|AA^T\right\|_2 \ge 1 + (k - 1)\delta.$$

While, this theorem gives us an lower bound to the matrix with relatively low orthogonality value. This corresponds to the case that if we choose rows from only one or two clusters since these selected rows have almost same direction. In such case, we proved that the spectral norms of block matrices are quite large. Thus, it is quite reasonable to say that choosing rows from each cluster should converge faster than choosing rows from one or two clusters.

Theorem 7. *$A \in \mathbb{R}^{k \times p}$, $A_i \in \mathbb{R}^p$ is the ith row of A, $\|A_i\|_2 = 1$ for all $i \in \{1, 2, ..., k\}$. Suppose the orthogonality value $ov(A) \ge \frac{1}{\epsilon}$, then one can bound the condition number of AA^T,*

$$cond(AA^T) \le \frac{1 + k\epsilon}{1 - \epsilon}.$$

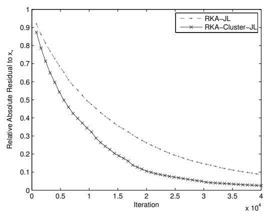
According to lemma 3, spectral norm and condition number have a huge impact to the convergence of block Kaczmarz. Based on the same assumption about low orthogonality value, the two theorems 5 and 7 give a theoretical analysis to the upper bounds of spectral norm and condition number. Also, the experiments show that block Kaczmarz with clustering is more robust in the noisy case.

It is necessary to notice that (Needell and Tropp 2014) proposed two algorithms to select block matrices. One approach is an iterative algorithm repeatedly extracting a well-conditioned row-submatrix from A until the paving is complete. This approach is based on the column subset selection method proposed by (Tropp 2009). The another approach is a random algorithm partitioning the rows of A in a random manner. The iterative algorithm will give a set of well-conditioned blocks, but at a much expense of computation. The random algorithm is easy to implement and bears an upper bound $\beta \le 6 \log(1 + n)$ with high probability $1 - n^{-1}$. Our construction for the clustering matrix blocks is more similar to the random algorithm in that we also construct the matrix block in a random manner, after clustering.

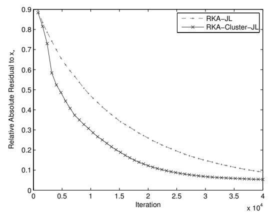
But to get the lower bound of α, the random algorithm needs a fast incoherence transform, which changes the original problem $\min \|Ax - b\|_2^2$ into $\left\|\tilde{A}x - \tilde{b}\right\|_2^2$, where $\tilde{A} = SA$, $\tilde{b} = Sb$ and S is the fast incoherence transforming matrix. Therefore it brings more noise into the original problem. Without changing the original form of the least square problem, our clustering method will construct well-conditioned matrix block with proven upper bounds.

4 Experiment

In this section we empirically evaluate our methods in comparison with RKA-JL and RKA-Block algorithms. We mainly follow (Needell and Tropp 2014) to conduct our experiments. The experiments are run in a PC with WIN7 system, i5-3470 core 3.2GHz CPU and 8G RAM.



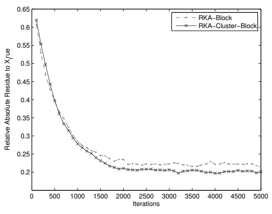
(a) Add noise $\mathcal{N}(0, 0.1)$



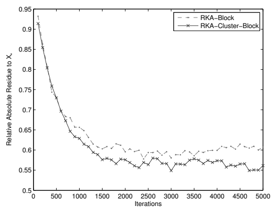
(b) Add noise $\mathcal{N}(0, 0.2)$

Figure 1: Convergence comparison between RKA-JL and RKA-Cluster-JL

First, we compare the proposed RKA-Cluster-JL with the original RKA-JL algorithm. We generate data that comprises of several clusters. Here, $n = 10000$ and $p = 1000$. Besides, since the real data is usually corrupted by white noise, we add Gaussian noise with mean 0 and standard deviation 0.1 or 0.2. Figure 1 below shows that RKA-Cluster-JL outperforms RKA-JL. To cluster the data, we use the K-means variant algorithm (King 2012).



(a) Add noise $\mathcal{N}(0, 0.1)$



(b) Add noise $\mathcal{N}(0, 0.2)$

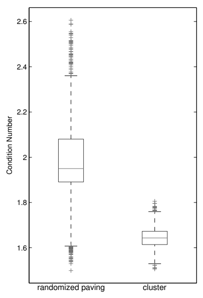
Figure 2: Convergence comparison between RKA-Block and RKA-Cluster-Block

Next, we compare the results produced by RKA-Block and RKA-Cluster-Block algorithms. We generate data that lies in four distinctive clusters. Here, $n = 10000$, $p = 1000$ and the block size is four. Then, as usual, white Gaussian noise with mean 0 and standard deviation 0.1 or 0.2 is added to the data to simulate the real world. From Figure 2 below, we can see that our algorithm performs better than RKA-Block algorithm.

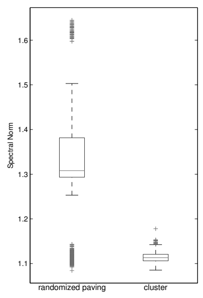
It is quite reasonable that our algorithm is better. Since our theoretical analysis tells us that smaller condition number or spectral norms of the block matrix, better performance the algorithm will have. We collect the condition numbers and spectral norm when these iterative algorithm running, and draw the box plots on Figure 3. It shows that our algorithm gives both smaller condition number and spectral norm.

5 Conclusion

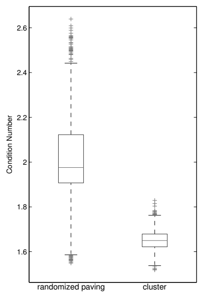
In this paper, we propose an acceleration approach to improving the RKA-JL algorithm. Our approach is based on a simple yet effective idea for clustering data points. Moreover, we have extended the clustering idea to construct well-conditioned matrix blocks, which shows improvement over the block Kaczmarz. When data points follow a Gaussian distribution, we have conducted theoretical and empirical analysis of our approaches.



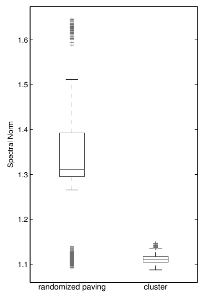
(a) $\mathcal{N}(0, 0.1)$



(b) $\mathcal{N}(0, 0.1)$



(c) $\mathcal{N}(0, 0.2)$



(d) $\mathcal{N}(0, 0.2)$

Figure 3: Comparison between RKA-Block and RKA-Cluster-Block in Condition number and spectral norm

References

Blum, A.; Hopcroft, J. E.; and Kannan, R. 2015. *Foundations of Data Science*.

Briskman, J., and Needell, D. 2014. Block kaczmarz method with inequalities. *Journal of Mathematical Imaging and Vision* 1–12.

Byrne, C. 2009. Bounds on the largest singular value of a matrix and the convergence of simultaneous and block-iterative algorithms for sparse linear systems. *International Transactions in Operational Research* 16(4):465–479.

Dai, L.; Soltanalian, M.; and Pelckmans, K. 2014. On the randomized kaczmarz algorithm. *arXiv preprint arXiv:1402.2863*.

Eldar, Y. C., and Needell, D. 2011. Acceleration of randomized kaczmarz method via the johnson–lindenstrauss lemma. *Numerical Algorithms* 58(2):163–177.

Elfving, T. 1980. Block-iterative methods for consistent and inconsistent linear equations. *Numerische Mathematik* 35(1):1–12.

Hong, Y., and Pan, C.-T. 1992. A lower bound for the smallest singular value. *Linear Algebra and Its Applications* 172:27–32.

Johnson, W. B., and Lindenstrauss, J. 1984. Extensions of lipschitz mappings into a hilbert space. *Contemporary mathematics* 26(189-206):1–1.

Kaczmarz, S. 1937. Angenäherte auflösung von systemen linearer gleichungen. *Bulletin International de lAcademie Polonaise des Sciences et des Lettres* 35:355–357.

King, A. 2012. Online k-means clustering of nonstationary data. *Prediction Project Report*.

Natterer, F. 1986. *The mathematics of computerized tomography*, volume 32. Siam.

Needell, D., and Tropp, J. A. 2014. Paved with good intentions: Analysis of a randomized block kaczmarz method. *Linear Algebra and its Applications* 441:199–221.

Needell, D. 2010. Randomized kaczmarz solver for noisy linear systems. *BIT Numerical Mathematics* 50(2):395–403.

Pomparău, I., and Popa, C. 2014. Supplementary projections for the acceleration of kaczmarz algorithm. *Applied Mathematics and Computation* 232:104–116.

Pomparău, I. 2013. On the acceleration of kaczmarz projection algorithm. *PAMM* 13(1):419–420.

Sezan, M. I., and Stark, H. 1987. Incorporation of a priori moment information into signal recovery and synthesis problems. *Journal of mathematical analysis and applications* 122(1):172–186.

Strohmer, T., and Vershynin, R. 2009. A randomized kaczmarz algorithm with exponential convergence. *Journal of Fourier Analysis and Applications* 15(2):262–278.

Thoppe, G.; Borkar, V.; and Manjunath, D. 2014. A stochastic kaczmarz algorithm for network tomography. *Automatica* 50(3):910–914.

Tropp, J. A. 2009. Column subset selection, matrix factorization, and eigenvalue optimization. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, 978–986. Society for Industrial and Applied Mathematics.

Zouzias, A., and Freris, N. M. 2013. Randomized extended kaczmarz for solving least squares. *SIAM Journal on Matrix Analysis and Applications* 34(2):773–793.