A Variation on the Randomized Kaczmarz Method for Solving Corrupted Overdetermined Systems

Jonas Carew and Nicholas Marshall
Department of Mathematics, Oregon State University, Corvallis, OR, USA

Abstract

A common method to solving overdetermined systems is the Kaczmarz method, and a variation on it, the randomized Kaczmarz method. Here, we show why the randomized Kaczmarz method works, and apply a variation on it to help deal with equation corruption. This variation allows the randomized Kaczmarz method to more consistently reach a result with near-minimal error.

The Randomized Kaczmarz Algorithm

For a linear system of equations, A is full rank $m \times n$ tall matrix

$$Ax = b$$

then for an arbitrary initial approximation x_0, we can compute the next iteration

$$x_{k+1} = x_k + \frac{b_{r(i)} - \langle a_{r(i)}, x_k \rangle}{\|a_{r(i)}\|_2^2} a_{r(i)}$$

where $r(i)$ is chosen from the set $\{1, 2, \ldots, m\}$ at random, with probability proportional to $\|a_{r(i)}\|_2^2$.

Theorem: Expected Convergence (Strohmer and Vershynin 2009)

Let $\kappa(A)$ be the scaled condition number

$$\kappa(A) = \|A\|_F \left\|A^{-1}\right\|_2$$

Then, the randomized Kaczmarz algorithm converges to the solution, x, in expectation, with average error

$$\mathbb{E}\,\|x_k - x\|_2^2 \leq (1 - \kappa(A)^{-2})^k \cdot \|x_0 - x\|_2^2$$

Proof of the Expected Convergence

We begin with the fact, that by the definition of $\|A\|_2$,

$$\sum_{j=1}^{m} |\langle z, a_j \rangle|^2 \geq \frac{\|z\|_2^2}{\|A^{-1}\|_2^2} \qquad \text{for all } z \in \mathbb{C}^n$$

and we can divide by the square of the Frobenius norm to get

$$\sum_{j=1}^{m} \frac{\|a_j\|_2^2}{\|A\|_F^2} \left| \left\langle z, \frac{a_j}{\|a_j\|_2} \right\rangle \right|^2 \geq \kappa(A)^{-2} \|z\|^2 \qquad \text{for all } z \in \mathbb{C}^n \tag{1}$$

we can then define a random variable Z as

$$Z = \frac{a_j}{\|a_j\|_2} \quad \text{with probability} \quad \frac{\|a_j\|_2^2}{\|A\|_F^2}, \quad j = 1, \ldots, m$$

which means (1) is equivalent to

$$\mathbb{E}|\langle z, Z \rangle|^2 \geq \kappa(A)^{-2} \|z\|_2^2 \quad \text{for all } z \in \mathbb{C}^n \tag{2}$$

Furthermore, we then define an orthogonal projection operator P as

$$Pz = z - \langle z - x, Z \rangle Z$$

and we notice that

$$\begin{aligned} P_{k+1} x_k &= x_k - \left\langle x_k - x, \frac{a_{r(i)}}{\|a_{r(i)}\|_2} \right\rangle \frac{a_{r(i)}}{\|a_{r(i)}\|_2} \\ &= x_k - \frac{\langle x_k, a_{r(i)} \rangle - \langle x, a_{r(i)} \rangle}{\|a_{r(i)}\|_2^2} a_{r(i)} \\ &= x_k + \frac{b_{r(i)} - \langle a_{r(i)}, x_k \rangle}{\|a_{r(i)}\|_2^2} a_{r(i)} = x_{k+1} \end{aligned}$$

this lets us see that $x_{k-1} - x_k$ is in the kernel of P_k. Since $x_{k-1} - x$ is in the solution space that P_k projects orthogonal to, we get orthogonality, and thus

$$\|x - x_k\|_2^2 + \|x_k - x_{k-1}\|_2^2 = \|x - x_{k-1}\|_2^2 \tag{3}$$

$$\implies \|x_k - x\|_2^2 = \|x_{k-1} - x\|_2^2 - \|x_{k-1} - x_k\|_2^2 \tag{4}$$

Our final tool for this proof will be that, by the definition of x_k,

$$\|x_{k-1} - x_k\|_2 = \langle x_{k-1} - x, Z_k \rangle$$

substituting this into (4), we get

$$\|x_k - x\|_2^2 \leq \left(1 - \left| \left\langle \frac{x_{k-1} - x}{\|x_{k-1} - x\|_2}, Z_k \right\rangle \right|^2 \right) \|x_{k-1} - x\|_2^2$$

taking the expectation on both sides conditional upon all previous choices of $Z_1, \ldots, Z_{k-1}$, we get

$$\mathbb{E}_{\{Z_1,\ldots,Z_{k-1}\}} \|x_k - x\|_2^2 \leq \left(1 - \mathbb{E}_{\{Z_1,\ldots,Z_{k-1}\}} \left| \left\langle \frac{x_{k-1} - x}{\|x_{k-1} - x\|_2}, Z_k \right\rangle \right|^2 \right) \|x_{k-1} - x\|_2^2$$

Using (2) and the independence of the realizations of $Z_1, \ldots, Z_k$, we get

$$\mathbb{E}_{\{Z_1,\ldots,Z_{k-1}\}} \|x_k - x\|_2^2 \leq (1 - \kappa(A)^{-2}) \|x_{k-1} - x\|_2^2$$

Taking the full expectation on both sides will give us

$$\mathbb{E}\,\|x_k - x\|_2^2 \leq (1 - \kappa(A)^{-2})\, \mathbb{E}\,\|x_{k-1} - x\|_2^2$$

The final step is to use induction to show that

$$\mathbb{E}\,\|x_k - x\|_2^2 \leq (1 - \kappa(A)^{-2})^k \cdot \|x_0 - x\|_2^2$$

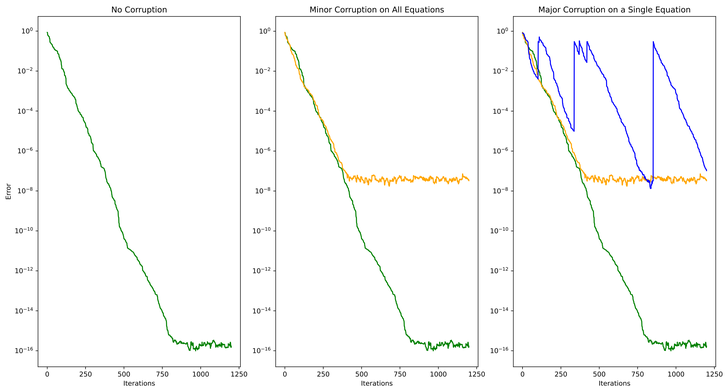
which completes the proof. □

The Problem of Corrupted Data

Corrupted data presents several issues when trying to solve it using the randomized Kaczmarz algorithm.

- The algorithm will never get too close to the solution, as it will randomly project onto corrupted equations.
- Pieces of bad data will repeatedly move the algorithm much further away from the solution.

Original and Corrupted Randomized Kaczmarz Algorithm



Minimizing the Effects of Corruption

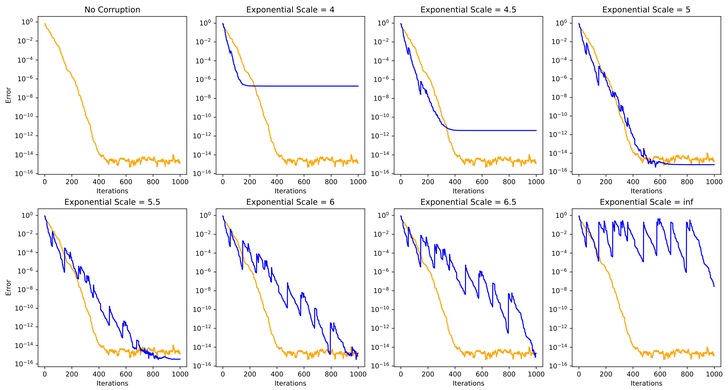
The first type of corruption, where all equations are modified, cannot be overcome without additional information. We have no clear reference for which equations are correct, so randomly projecting onto them in the standard way is the decision.
The second type of corruption presents a different challenge. As the corrupted equation gets chosen randomly, it will project out iterative solution far away from the actual solution. Because we randomly choose the equation, this will continue happening indefinitely. The method we created to overcome this, involves capping the norm of the shift of each iteration. To do this, we used this pseudocode, where `x` is our current iterating vector, and `shft` is the vector that is added to x_k to get x_{k+1} in the Kaczmarz algorithm.

```
cap = ((1 + (0.5 ^ scale)) ^ -k
if (norm(shft) > cap):
    shft *= cap/norm(shft)
x += shft
```

The `scale` is a tuning parameter, and depending on its value the results may vary.

Results of Limiting Variation

We have plotted various values of the `scale` against the plot with no (major) corruption.



Notice that when `scale` is infinity, the limit is never reached, and the algorithm acts as the original randomized Kaczmarz algorithm would on the corrupted dataset.

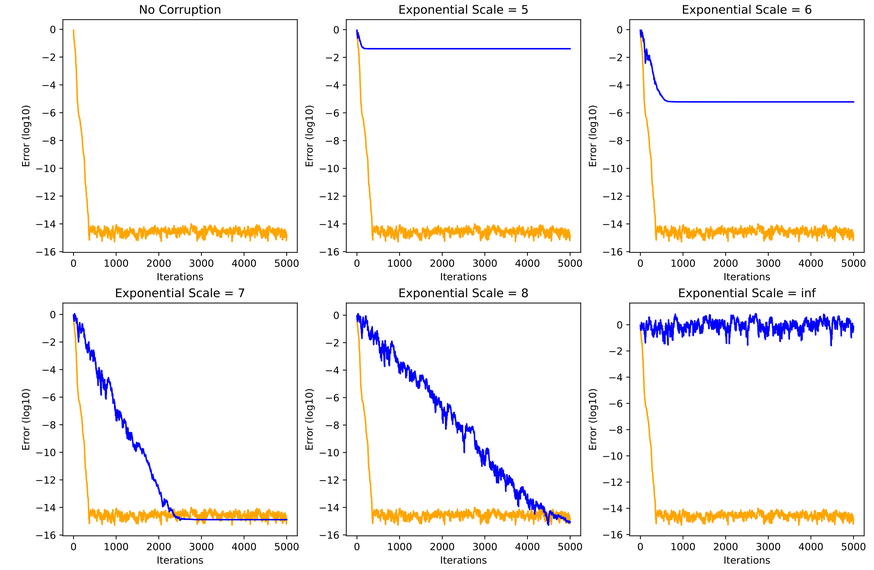
Particularly with this dataset, but also common with others, we found that for these single points of 'major' corruption we could get our modified randomized Kaczmarz to converge just as fast as the original algorithm would on uncorrupted data, however this does require tuning the `scale` parameter correctly. The values seen here are, however, fairly consistent across various trials done and 5-6 seems to be a very good value for `scale` to converge quickly with low 'major' corruption.

Higher Proportions of Corruption

The earlier tests were run when there was only 1 out of 100 equations with major corruption affecting them. Using this same algorithm with much higher proportions of corruption still gives good results, but it obviously becomes slower and less consistent.

Results of 25/100 Corrupted Equations

As the proportion of corrupted equations raises, we notice a trend shown in the graphs below. Lower values of `scale` no longer provide the fast convergence, and in fact never converge at all. In fact, even the value of `scale` $= 8$ even failed at one randomly chosen we tried.



Also note the much higher scaling on the x-axis, as it usually takes around 5000 iterations for a value of `scale` $= 8$ to converge.

Conclusions

This variation on the randomized Kaczmarz method is very powerful, as it allows us to overcome corruption on several fronts.

- With lower amounts of so-called 'major' corruption, we find that by choosing a `scale` value around 5-6 we can regularly get near-convergence at a pace close to that of the original Kaczmarz algorithm (acting on an uncorrupted dataset).
- With higher amounts of 'major' corruption, we can still get near-convergence, albeit much slower, by tuning the `scale` value higher.

This algorithm also gives us a level of consistency found in the higher `scale` values. As the value of the `scale` parameter increases, the time until we converge also increases, but the likelihood that it will converge and the error will not stagnate is also raised.

Further Research Potential

This problem has applications to many different fields, as slightly corrupted data is so common. Oftentimes though, we don't know how corrupted data will be. This is actually very closely related to known NP-Hard problem, called Maximum Feasible Subsystem, or MaxFS. It involves an infeasible system of the form $Ax \geq b$, and finding a feasible subsystem that contains the maximum number of inequalities.
An important algorithm in solving problems such as MaxFS is called RANSAC. RANSAC involves selecting a random subsample of points and fitting your desired model to them. You can the refine the model by using all the points that are designated as inliers. This is then repeated for a number of iterations until satisfaction, where the new result is taken if the model is better fit.

References

- Strohmer, Thomas, and Roman Vershynin. "A Randomized Kaczmarz Algorithm with Exponential Convergence." Journal of Fourier Analysis and Applications, vol. 15, no. 2, Apr. 2009, pp. 262–78, doi:10.1007/S00041-008-9030-4.
- Yu, Chanki, and Da Young Ju. "A Maximum Feasible Subsystem for Globally Optimal 3D Point Cloud Registration." Sensors (Basel, Switzerland) vol. 18,2 544. 10 Feb. 2018, doi:10.3390/s18020544.
- Jamie Haddock, Deanna Needell, Elizaveta Rebrova, and William Swartworth, "Quantile-based iterative methods for corrupted systems of linear equations", SIAM Journal on Matrix Analysis and Applications **43** (2022), no. 2, 605–637.