Faster Alternatives to the Randomized Extended Kaczmarz Algorithm

Bogdan Dumitrescu\textsuperscript{a,b}

May 28, 2014

Abstract

In this note we compare the randomized extended Kaczmarz (EK) algorithm and randomized coordinate descent (CD) for solving the full-rank overdetermined linear least-squares problem and prove that CD is always faster than EK. For the general least-squares problems, we show that running first CD to compute the residual and then standard Kaczmarz on the resulting consistent system is more efficient than EK.

\textsuperscript{a}Department of Automatic Control and Computers, University Politehnica of Bucharest, Spl. Independenței 313, Bucharest 060042, Romania. E-mail: bogdan.dumitrescu@acse.pub.ro

\textsuperscript{b}Department of Signal Processing, Tampere University of Technology, Finland. E-mail: bogdan.dumitrescu@tut.fi
1 Introduction

Given matrix $A \in \mathbb{R}^{m \times n}$ and vector $b \in \mathbb{R}^m$, the linear least-squares (LS) problem consists of finding $x \in \mathbb{R}^n$ such that $\|b - Ax\|_2$ is minimum. Unless explicitly stated, we consider the full-rank overdetermined problem, i.e. $m \geq n$ and $\text{rank}(A) = n$. Besides standard solutions based on orthogonal triangularization or the normal equations, significant recent interest was focused on randomized algorithms, showing clear benefits for certain categories of problems, especially for large dimensions and sparse matrices.

There are two main classes of randomized algorithms for the LS problem, both based on simple iterated projection operations. In coordinate descent (CD) [1], at iteration $k$, the current residual is projected onto a random column of the matrix $A$, in order to obtain the optimal LS update of a single element of the current approximation of the solution $x^{(k)}$. In the Kaczmarz algorithm [5], the solution $x^{(k)}$ is projected onto the hyperplane defined by a random row of the matrix $A$ and the respective element of $b$, thus obtaining the next approximation $x^{(k+1)}$. Unlike CD, randomized Kaczmarz converges only when the system $Ax = b$ is consistent. Otherwise, it hovers around the LS solution, within guaranteed bounds [2]. This behavior is natural, since at each iteration the approximated solution satisfies exactly an equation of the system $Ax = b$, which is not the case in general for the LS solution.

To fix this drawback, the extended Kaczmarz (EK) algorithm, in both randomized [6] and original deterministic [4] forms, simultaneously builds an approximation of the residual, such that a consistent system is asymptotically obtained, and applies Kaczmarz iterations for the current approximation of this system. Thus, the algorithm converges to the LS solution.

We show in this note that EK consists in fact of CD and Kaczmarz iterations, thus combining both classes of randomized algorithms. Since CD can find on its own the LS solution, we argue that EK can never be faster than CD, neither in terms of number of iterations, nor in terms of number of operations. So, we conclude that randomized CD should be preferred over EK in all situations, for overdetermined LS problems. We discuss also the general LS problem (not full rank) and show that EK can be safely replaced by CD followed by the usual Kaczmarz for faster convergence.

The notation resembles that from [6]. We denote by $A^{(i)}$ and $A^{(j)}$ the $i$-th row and $j$-th column of matrix $A$, respectively, both seen as column vectors. The scalar product of two vectors is denoted $\langle \cdot, \cdot \rangle$ and $[m] = \{1, \ldots, m\}$. The $i$-th unit vector is $e_i$. To distinguish between algorithms, we add the subscript EK, CD or K (the latter for standard Kaczmarz) to variables having the same meaning, but different values for the two algorithms. We denote $x_o$ the solution of the LS problem and $\mathcal{R}(A)$ the range of $A$.

2 Extended Kaczmarz vs coordinate descent

Algorithm 1 shows a slightly modified version of the randomized EK from [6]. Besides non-significant permutations of the steps and some different explanations, only step 6 is new here and does not affect the final outcome. Let us first discuss the structure
of the algorithm and explain its relation with CD. For further reference, we denote

$$r^{(k)} = b - Ax^{(k)}$$

(1)

the residual at iteration $k$.

The EK algorithm, as presented in [6], has two intertwined parts. In the first, a residual is built, converging to the optimal residual $b - Ax_o$ of the LS problem. At each iteration, a column $j_k$ is picked randomly as in step 3 of Algorithm 1, and the residual is projected onto the orthogonal complement of this column, thus obtaining a new residual (smaller in size than the previous because of the projection). However, this is exactly what CD does and that is why we denote this residual $r_{CD}^{(k)}$. Indeed, in CD, the residual is projected onto column $j_k$ in order to find the optimal update of the $j_k$-th element of the solution, as in step 4 (this projection maximizes the decrease of $\|b - Ax^{(k+1)}\|$ if only the $j_k$-th coordinate of $x^{(k)}$ is modified). After updating the solution as in step 6, the new residual from step 5 is indeed

$$r_{CD}^{(k+1)} = b - Ax_{CD}^{(k+1)} = b - Ax_{CD}^{(k)} - \mu A_{(j_k)} = r_{CD}^{(k)} - \mu A_{(j_k)}$$

and is orthogonal on column $j_k$, as one can easily check by plugging in the expression of the optimal update $\mu$:

$$\langle r_{CD}^{(k+1)} , A_{(j_k)} \rangle = \langle r_{CD}^{(k)} - \mu A_{(j_k)} , A_{(j_k)} \rangle = 0.$$

So, steps 3-5 of EK compute the CD residual. Only one more arithmetic operation per iteration is necessary to update the CD solution, as in step 6. So, Algorithm 1 without steps 7 and 8 is actually the randomized CD algorithm, which converges to

<table>
<thead>
<tr>
<th>Algorithm 1: Randomized Extended Kaczmarz</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Data:</strong> Matrix $A \in \mathbb{R}^{m \times n}$, vector $b \in \mathbb{R}^m$, stopping tolerance $\varepsilon$</td>
</tr>
<tr>
<td><strong>Result:</strong> Least-squares solution $x \in \mathbb{R}^n$ minimizing $|b - Ax|$</td>
</tr>
</tbody>
</table>

1. Initialize $r_{CD}^{(0)} = b$, $x_{CD}^{(0)} = 0$, $x_{EK}^{(0)} = 0.$
2. **for** $k = 0, 1, 2, \ldots$ **do**
   3. Pick $j_k \in [n]$ with probability $p_j = \|A(j)\|_2^2 / \|A\|_F^2$, $j \in [n]$
   4. Find optimal coordinate descent step: $\mu = \langle r_{CD}^{(k)} , A_{(j_k)} \rangle / \|A_{(j_k)}\|_2^2$
   5. Update CD residual: $r_{CD}^{(k+1)} = r_{CD}^{(k)} - \mu A_{(j_k)}$
   6. Update CD solution: $x_{CD}^{(k+1)} = x_{CD}^{(k)} + \mu e_{j_k}$
   7. Pick $i_k \in [m]$ with probability $q_i = \|A(i)\|_2^2 / \|A\|_F^2$, $i \in [m]$
   8. Update EK solution: $x_{EK}^{(k+1)} = x_{EK}^{(k)} + \frac{(b - r_{CD}^{(k)} - x_{CD}^{(k)}) - (x_{EK}^{(k)}) A(i_k)}{\|A(i_k)\|_2^2} A(i_k)$
   9. Check every $8 \min(m, n)$ iterations and terminate if it holds

   $$\frac{\|b - r_{CD}^{(k)} - Ax^{(k)}\|_2}{\|A\|_F \|x^{(k)}\|_2} \leq \varepsilon \quad \text{and} \quad \frac{\|A^T r_{CD}^{(k)}\|_2}{\|A\|_F \|x^{(k)}\|_2} \leq \varepsilon$$


the LS solution, i.e. $x^{(k)}_{CD} \rightarrow x_o$, $r^{(k)}_{CD} \rightarrow b - Ax_o$, see [1], or [3] for a more general treatment. The probabilities $p_j$ are also taken like in the randomized CD.

The second part of Algorithm 1, steps 7 and 8, implements a Kaczmarz iteration for the LS problem

$$Ax = b - r^{(k)}_{CD}. \quad (2)$$

Since CD converges to the LS solution, the above system becomes asymptotically consistent and hence the Kaczmarz iterations converge also to the true solution, as shown in [6]. The stopping criteria from step 9 are intentionally written for an approximation $x^{(k)}$ that can result from either EK or CD. The second criterion is meant for the CD iterations (the residual becomes orthogonal on $\mathcal{R}(A)$) and the first for the Kaczmarz iterations (a solution to (2) has been found).

**Proposition 1** In average (over the random draw of columns and rows), CD is faster than EK.

*Proof.* We give three arguments of different nature. They hold for a literal implementation of Algorithm 1, where the random columns $j_k$ are the same for EK and CD. If the algorithms are separately implemented, then the random columns are different and it may happen that EK is faster than CD, due to a more favorable draw of columns. However, CD is still faster on average, since it is faster for each common draw of columns.

*Argument 1* is purely technical and formally sufficient. We can safely assume that the EK and CD solution approximations have similar magnitudes, i.e. the values $\|x^{(k)}_{EK}\|$ and $\|x^{(k)}_{CD}\|$ do not make the stopping criteria significantly different for EK and CD, especially near convergence, the LS solution being unique. The second stopping criterion depends only on the CD residual, so EK cannot stop earlier than CD. The first stopping criterion is always met for CD, since $b - r^{(k)}_{CD} - Ax^{(k)}_{CD} = 0$ by definition (1). So, again, EK cannot stop earlier than CD. The number of operations per iteration is also in favor of CD. While CD has $O(m)$ operations, EK has an extra $O(n)$.

*Argument 2* is heuristic and is based on the general structure of the algorithms. CD computes an approximation of the optimal residual of the LS problem. EK computes an approximation of the LS solution of the system depending on the CD residual. So, EK builds an approximation based on the CD approximation. It is unlikely that EK converges faster than CD other than by accident.

*Argument 3* looks more closely at the residuals and substantiates the previous argument. Define

$$
\hat{r}^{(k)}_{EK} = b - r^{(k)}_{CD} - Ax^{(k)}_{EK} \quad (3)
$$

the residual of the system (2) that EK actually attempts to solve at iteration $k$. It results that

$$
\hat{r}^{(k)}_{EK} = r^{(k)}_{CD} + \hat{r}^{(k)}_{EK}. \quad (4)
$$

Using (1) for CD, it also follows from (3) that

$$
\hat{r}^{(k)}_{EK} = A(x^{(k)}_{CD} - x^{(k)}_{EK}).
$$
Since CD converges to the solution of the LS problem, its residual tends to become orthogonal to the range of $A$; hence
\[ \langle r^{(k)}_{CD}, \hat{r}^{(k)}_{EK} \rangle \to 0. \]

Hence one can infer from (4) that, at least asymptotically,
\[ \| r^{(k)}_{EK} \| > \| r^{(k)}_{CD} \|. \]

So, again, CD converges faster.

3 The general LS problem

Reminding that all the previous discussion was for full-rank overdetermined LS problems, let us look at the other cases. Consider first underdetermined systems, but still full-rank. In this case, the system $Ax = b$ is consistent and the standard Kaczmarz algorithm converges to the LS solution. There is no need of residual approximation, since the residual is zero, hence the CD part of EK is useless.

We pass now to the general LS problem, in which the matrix $A$ has arbitrary rank, and for which the deterministic EK algorithm [4] was originally intended. The LS solution is that with minimum norm $\| x \|_2$, among those minimizing the residual $\| b - Ax \|_2$. In this case, due to their specific projection operations, CD can minimize the residual, but not find a solution with minimum norm, while Kaczmarz can (almost) minimize the norm of the solution (if properly initialized with $x^{(0)} \in \mathcal{R}(A^T)$), but not that of the residual. EK combines their strengths to find the LS solution.

We argue that, however, there is better way to combine the two algorithms than intertwining them as in EK. We propose to run first CD for estimating the optimal residual $r_{CD}$ and only then Kaczmarz for finding the least norm solution of the consistent system
\[ Ax = b - r_{CD}. \]

We name CD+K this algorithm.

**Remark 1** In average, CD+K should be faster than EK.

We give below two arguments holding this assertion. Note that the cost per iteration is the same in CD+K and EK, since the operations in EK can be perfectly split between CD and Kaczmarz.

**Argument 1.** EK stops when the last of CD and Kaczmarz satisfy the corresponding stopping criteria from step 9 of Algorithm 1 (first criterion for Kaczmarz, second for CD). Instead, run in sequence, one of CD or Kaczmarz will stop earlier, so CD+K is faster.

**Argument 2.** Moreover, running Kaczmarz after CD has the advantage that it works from the start on the consistent system to be solved. In EK, the Kaczmarz iterations are made for the system (2), which is only an approximation of the final consistent system (5). So, it is natural to expect fewer Kaczmarz iterations in CD+K than in EK.
Remark 2 Running Kaczmarz after CD has a nice alternative interpretation. In this context, the Kaczmarz iteration has the form (see step 8 of Algorithm 1)

\[ x^{(k+1)}_K = x^{(k)}_K + \frac{\langle b - r_{CD}, e_{i_k} \rangle - \langle x^{(k)}_K, A^{(i_k)} \rangle}{\|A^{(i_k)}\|_2^2} A^{(i_k)} = x^{(k)}_K + \frac{\langle x_{CD} - x^{(k)}_K, A^{(i_k)} \rangle}{\|A^{(i_k)}\|_2^2} A^{(i_k)}, \]

where we have used the equality \( \langle b - r_{CD}, e_{i_k} \rangle = \langle Ax_{CD}, e_{i_k} \rangle = \langle x_{CD}, A^{(i_k)} \rangle \).

Denoting \( q^{(k)}_K = x_{CD} - x^{(k)}_K \), it results from (6) that

\[ q^{(k+1)}_K = q^{(k)}_K - \frac{\langle q^{(k)}_K, A^{(i_k)} \rangle}{\|A^{(i_k)}\|_2^2} A^{(i_k)}. \quad (7) \]

This is a projection operation on the orthogonal complement of the \( i_k \)-th row of \( A \), dual to the CD operation of projecting the residual on the orthogonal complement of column \( j_k \) (step 5 of Algorithm 1). Hence, \( q^{(k)}_K \) tends to the component of \( q^{(0)}_K \) that is orthogonal on \( \mathcal{R}(A^T) \). Initializing with \( q^{(0)}_K = x_{CD} \), which corresponds to the natural initialization \( x^{(0)}_K = 0 \), the iteration (7) converges to \( q_K \) satisfying \( x_K + q_K = x_{CD} \), with \( q_K \perp \mathcal{R}(A^T) \) and hence \( x_K \in \mathcal{R}(A^T) \). Since \( Aq_K = 0 \), the Kaczmarz solution satisfies the system (5). This means that \( x_K = x_o \), since the LS solution is the unique vector from \( \mathcal{R}(A^T) \) satisfying (5).

So, the iteration (7) starts with \( x_{CD} \), for which (5) already holds, and computes its projection onto the orthogonal complement of \( \mathcal{R}(A^T) \). Thus, it allows the computation of the projection of \( x_{CD} \) onto \( \mathcal{R}(A^T) \), which is the LS solution.

Of course, using (7) instead of (6) gives no computational advantage, but gives a dual view to the convergence of the Kaczmarz iterations.

\[ \Box \]

4 Numerical results

We have implemented Algorithm 1 in Matlab and report the performance of CD and EK only in terms of number of iterations, reminding however that at similar number of iterations CD is still faster. The algorithm has been run for a sufficiently high number of iterations, without any stopping criterion. For overdetermined LS problems, we have considered matrices belonging to two classes where EK was shown in [6] to have better performance than other algorithms: (i) dense well-conditioned matrices, generated with \texttt{randn}, and (ii) sparse random matrices with density 0.25, generated with \texttt{sprandn}.

We report the RMSE \( \sqrt{E(\|x^{(k)} - x_o\|^2)} \), obtained by averaging over 100 matrices from the same class. Figures 1 and 2 show the RMSE for dense matrices with same number of rows, \( n = 500 \), but different number of columns: \( m = 2000 \) and \( m = 10000 \), respectively. Figure 3 shows the RMSE for sparse matrices, \( n = 800 \), \( m = 2000 \). In all cases, the faster convergence of CD is clear. When the system is very overdetermined, CD has a jagged convergence, alternating many small advances with few large ones, but is still faster. For other matrix sizes, the results are similar.

To illustrate the behavior of CD+K, we have generated random matrices \( A \in \mathbb{R}^{m \times n} \), computed their SVD \( A = U \Sigma V^T \), kept only the \( r \) largest singular values
Figure 1: RMSE for dense matrices, $m = 2000$, $n = 500$.

Figure 2: RMSE for dense matrices, $m = 10000$, $n = 500$.

Figure 3: RMSE for sparse matrices, $m = 2000$, $n = 800$. 
in $\Sigma$ while setting the others to zero, and recomputed $A$ using the same above relation. For $m = 500$, $n = 2000$, $r = 400$, we show in Figure 4 the RMSE of the solution approximations in EK and in the Kaczmarz part of CD+K. The claim from Argument 2 of Remark 1 that the number of Kaczmarz iterations is smaller in CD+K than in EK, is thus substantiated.

5 Conclusions

The computational conclusion of all the facts presented in this note is the recommendation to replace EK with one of the following algorithms:

• CD, for full-rank overdetermined LS problems;
• Kaczmarz, for full-rank underdetermined problems;
• CD+K, for rank-deficient or unknown rank problems.

In particular, for the full-rank overdetermined problem, CD is always preferable to EK, due to the following reasons.

• In average, CD converges in fewer iterations than EK.
• CD has fewer operations per iteration.
• CD uses only the columns of the matrix $A$, while EK uses both columns and rows.

The conclusions apply equally to randomized EK [6] and the less efficient deterministic version [4].

Acknowledgment

The author is indebted to Liang Dai for stimulating discussions on the Kaczmarz algorithm and for pointing out [6].
References


