Medical Imaging

Prof. Dr. Tobias Knopp

23. November 2022

Institut für Biomedizinische Bildgebung

Iterative Reconstruction

Iterative Reconstruction

The SVD is a very advanced tool but what if

- The system matrix is sparse
- The system matrix is huge and does not fit into the main memory

In the second case, one usually has a formula for the matrix elements, which implies that the entire system matrix does not need to be setup in memory.

In both cases it is better to use *iterative solvers*.

Iterative Reconstruction

There are various iterative solvers, of which several can be grouped into the following two classes

- Krylov subspace methods
- Row- or column action methods

There are further classes, which we will, however not discuss at this point.

Iterative solver do not require element-wise access to the system matrix. Instead they require operations involving the system matrix. This can for instance be

- Matrix-vector multiplications with $\boldsymbol{A}$ or $\boldsymbol{A}^{\mathsf{H}}$, or
- Operations involving the matrix rows or columns.

Kaczmarz Method

- Also names *algebraic reconstruction technique* (ART) in the context of computed tomography.
- Fixed-point iteration, which converges to the solution $\boldsymbol{x}$ of the linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ if it exists.
- Let $l \geq 1$ be the iteration number and $\boldsymbol{x}^0 = \boldsymbol{0}$ be the start vector, then the Kaczmarz iteration is defined as

$$\boldsymbol{x}^{l+1} = \boldsymbol{x}^l + \frac{b_j - \boldsymbol{A}_{j,\cdot}\boldsymbol{x}^l}{\|\boldsymbol{A}_{j,\cdot}^{\mathsf{T}}\|_2^2} \boldsymbol{A}_{j,\cdot}^{\mathsf{H}}$$

- Row index j is usually chosen to sweep over all matrix rows so that one has two nested for loops and $j = l \bmod M$.

Kaczmarz Method – Derivation

Let us consider the j-th equation of the linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$. It can be expressed as

$$\boldsymbol{A}_{j,\cdot}\boldsymbol{x} = b_j.$$

Normalization of the vector $\boldsymbol{A}_{j,\cdot}$ yields

$$\begin{aligned} 0 &= \frac{b_j}{\|\boldsymbol{A}_{j,\cdot}^{\mathsf{T}}\|_2} - \frac{\boldsymbol{A}_{j,\cdot}}{\|\boldsymbol{A}_{j,\cdot}^{\mathsf{T}}\|_2}\boldsymbol{x} \\ &= d - \boldsymbol{n}^{\mathsf{H}}\boldsymbol{x}, \end{aligned}$$

where, $d = \frac{b_j}{\|\boldsymbol{A}_{j,\cdot}^{\mathsf{T}}\|_2}$ and $\boldsymbol{n} = \frac{\boldsymbol{A}_{j,\cdot}^{\mathsf{H}}}{\|\boldsymbol{A}_{j,\cdot}^{\mathsf{T}}\|_2}$ This is a hyperplane (e.g. line / plane in 2D and 3D) equation in Hessian normal form. Each vector within the hyperplane is orthogonal to the normal vector $\boldsymbol{n}$ and d is the distance to the origin. $\boldsymbol{n}$ points in direction of the hyperplane and for an orthogonal projection we just need to know the distance of a point to the plane.

Kaczmarz Method – Derivation

The distance of an arbitrary point $\tilde{\boldsymbol{x}}$ to the plane is

$$\mathrm{dist}(\tilde{\boldsymbol{x}}) = d - \boldsymbol{n}^{\mathsf{H}}\tilde{\boldsymbol{x}}$$

To project $\tilde{\boldsymbol{x}}$ onto the hyperplane, one thus has to add $\mathrm{dist}(\tilde{\boldsymbol{x}})\boldsymbol{n}$ to $\tilde{\boldsymbol{x}}$, i.e.

$$\begin{aligned}
\tilde{\boldsymbol{x}} + \mathrm{dist}(\tilde{\boldsymbol{x}})\boldsymbol{n} &= \tilde{\boldsymbol{x}} + (d - \boldsymbol{n}^{\mathsf{H}}\tilde{\boldsymbol{x}})\boldsymbol{n} \\
&= \tilde{\boldsymbol{x}} + \left(\frac{b_j}{\|\boldsymbol{A}_{j,\cdot}^{\mathsf{T}}\|_2} - \frac{\boldsymbol{A}_{j,\cdot}}{\|\boldsymbol{A}_{j,\cdot}^{\mathsf{T}}\|_2}\tilde{\boldsymbol{x}} \right) \frac{\boldsymbol{A}_{j,\cdot}^{\mathsf{H}}}{\|\boldsymbol{A}_{j,\cdot}^{\mathsf{T}}\|_2} \\
&= \tilde{\boldsymbol{x}} + \frac{b_j - \boldsymbol{A}_{j,\cdot}\tilde{\boldsymbol{x}}}{\|\boldsymbol{A}_{j,\cdot}^{\mathsf{T}}\|_2^2}\boldsymbol{A}_{j,\cdot}^{\mathsf{H}}
\end{aligned}$$

Converting this into an iteration process, we end up with the Kaczmarz iteration.

Kaczmarz Method – Geometric Interpretation

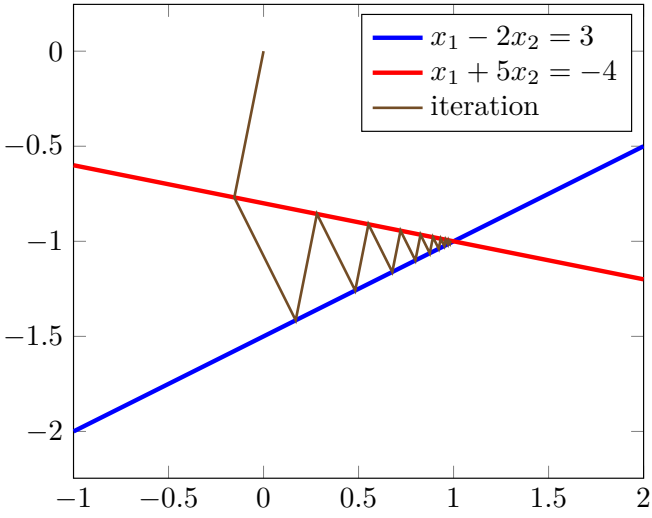
Consider 2×2 linear system

$$\begin{pmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$$

In this case each equation of the linear system describes a line in the $\mathbb{R}^2$. In the point where the lines intersect, the linear system has its solution.

The Kaczmarz iteration performs in each step an orthogonal projection on the hyperplane spanned by the j-th matrix row and the corresponding j-th element of the right hand side.

Kaczmarz Method – Geometric Interpretation



Kaczmarz Method – Convergence

Convergence speed (i.e. number of required iterations) depends on the similarity of successive matrix rows. If successive matrix rows are similar, more iterations are required.

Example: Convolution matrix

$$\begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

$\langle \boldsymbol{A}_{1,\cdot}, \boldsymbol{A}_{2,\cdot} \rangle_2 = 2$
but
$\langle \boldsymbol{A}_{1,\cdot}, \boldsymbol{A}_{4,\cdot} \rangle_2 = 0$
$\rightarrow$ not clever to run over matrix rows in order

Randomized Kaczmarz Method

There are two possible ways to improve the convergence speed of the Kaczmarz method

- If the structure of the system matrix is known, run through the matrix such that successive row indices have a small inner product
- Otherwise: Run in random order through the matrix rows

The second option is known as the *Randomized Kaczmarz* and can be shown to converge faster then the non-random Kaczmarz.

Regularized Kaczmarz

Kaczmarz method can be shown to solve

$$\|\boldsymbol{x}\|_2 \to \min \quad \text{subject to} \quad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$$

This problem is considered if the linear system is under-determined and has infinite solutions. In contrast, the problem

$$\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2 \to \min$$

is considered for over-determined linear systems. Furthermore, for ill-conditioned linear systems one actually wants to solve

$$\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 + \lambda\|\boldsymbol{x}\|_2^2 \to \min$$

How can this be done with Kaczmarz method?

Regularized Kaczmarz

Apply Kaczmarz algorithm to an extended system

$$\underbrace{\begin{pmatrix} \boldsymbol{A} & \lambda^{\frac{1}{2}}\boldsymbol{I} \end{pmatrix}}_{\tilde{\boldsymbol{A}} \in \mathbb{C}^{M \times (N+M)}} \underbrace{\begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{v} \end{pmatrix}}_{\tilde{\boldsymbol{x}} \in \mathbb{C}^{(N+M)}} = \boldsymbol{b}$$

Here, $\boldsymbol{v} \in \mathbb{C}^M$ in an auxiliary vector. Multiplying out yields

$$\begin{aligned} \boldsymbol{A}\boldsymbol{x} + \lambda^{\frac{1}{2}}\boldsymbol{v} &= \boldsymbol{b} \\ \Rightarrow \boldsymbol{v} &= -\lambda^{-\frac{1}{2}}(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}) \end{aligned}$$

Thus, the auxiliary vector will be the scaled residual after convergence.

What does the extended Kaczmarz calculate?

Regularized Kaczmarz

$$
\begin{aligned}
&\|\tilde{\boldsymbol{x}}\|_2 \to \min \quad \text{subject to} \quad \tilde{\boldsymbol{A}}\tilde{\boldsymbol{x}} = \boldsymbol{b} \\
\Leftrightarrow &\|\tilde{\boldsymbol{x}}\|_2^2 \to \min \quad \text{subject to} \quad \tilde{\boldsymbol{A}}\tilde{\boldsymbol{x}} = \boldsymbol{b} \\
\Leftrightarrow &\|\boldsymbol{x}\|_2^2 + \|\boldsymbol{v}\|_2^2 \to \min \quad \text{subject to} \quad \boldsymbol{v} = -\lambda^{-\frac{1}{2}}(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}) \\
\Leftrightarrow &\|\boldsymbol{x}\|_2^2 + \| - \lambda^{-\frac{1}{2}}(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b})\|_2^2 \to \min \\
\Leftrightarrow &\|\boldsymbol{x}\|_2^2 + \lambda^{-1}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 \to \min \\
\Leftrightarrow &\lambda\|\boldsymbol{x}\|_2^2 + \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 \to \min
\end{aligned}
$$

Thus, the extended Kaczmarz solves the Tikhonov regularized least squares problem.

Kaczmarz – Required operations

Kaczmarz method requires two elementary operations involving the system matrix $\boldsymbol{A}$

- An inner product $\alpha \leftarrow \boldsymbol{A}_{j,\cdot}\tilde{\boldsymbol{x}} = \sum_{n=1}^{N} \boldsymbol{A}_{j,n}\tilde{\boldsymbol{x}}_n$
- A vector update $\tilde{\boldsymbol{x}} \leftarrow \tilde{\boldsymbol{x}} + \alpha \boldsymbol{A}_{j,\cdot}^{\mathsf{H}}$
  i.e. $\tilde{\boldsymbol{x}}_n \leftarrow \tilde{\boldsymbol{x}}_n + \alpha\overline{\boldsymbol{A}_{j,n}}$ for $n = 1, \ldots, N$

Both are vector-vector operations that can be easily accelerated in case that the system matrix $\boldsymbol{A}$ is sparse. In that case only those indices are considered in the calculation for which $\boldsymbol{A}_{j,n} \neq 0$.

Kaczmarz – Required operations

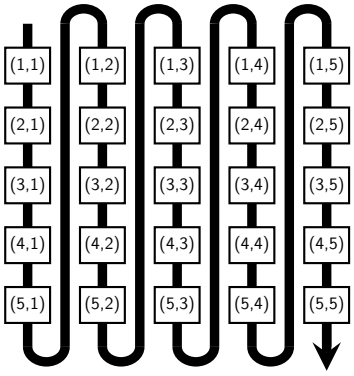
To implement the (non-regularized) Kaczmarz method in a generic fashion, it can be implemented as follows:

```
for i in rowIndexCycle
  j = rowindex[i]
  tau = dot_with_matrix_row(A, x, j)
  alpha = (b[j]-tau) / normA[j]
  kaczmarz_update!(A, x, j, alpha)
end
```

Here, `dot_with_matrix_row` and `kaczmarz_update!` are two functions that need to be implemented for each type of matrix, for instance it can be implemented for `Matrix{Float64}` and for `SparseMatrixCSC{Float64, Int64}`.

Kaczmarz – Required operations

In Julia dense matrices are stored in column major order, which means that the elements of the columns are stored next to each other in memory.



Performing row operations on such a data structure is very expensive since CPU caching cannot be utilized. To implement the Kaczmarz algorithm efficiently, one should thus first transpose the data and then use a transpose wrapper

```
julia> A = transpose(rand(3,3))
3×3 LinearAlgebra.Transpose{Float64,
                            Array{Float64,2}}:
 0.84271    0.342911  0.555876
 0.931374   0.886989  0.163034
 0.0734473  0.034807  0.296932
```

Conjugated Gradient

Conjugated Gradient

Conjugated Gradient (CG) is a popular Krylov subspace method that solves $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ for symmetric positive definite $\boldsymbol{A}$, i.e. $\boldsymbol{z}^{\mathsf{H}}\boldsymbol{A}\boldsymbol{z} > 0$ for any $\boldsymbol{z} \neq \boldsymbol{0}$.

For general $\boldsymbol{A}$ one can apply CG to the normal equation

$$\boldsymbol{A}^{\mathsf{H}}\boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}^{\mathsf{H}}\boldsymbol{b}$$

Regularization can also be added.

Conjugated Gradient

Algorithm 1 Conjugated Gradient Algorithm

$\boldsymbol{r}_0 \leftarrow \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0$

$\boldsymbol{v}_0 \leftarrow \boldsymbol{r}_0$

for $k = 0, \ldots, N-1$ **do**

  $\boldsymbol{z}_k \leftarrow \boldsymbol{A}\boldsymbol{v}_k$

  $\alpha_k \leftarrow \frac{\boldsymbol{r}_k^{\mathsf{H}} \boldsymbol{r}_k}{\boldsymbol{v}_k^{\mathsf{H}} \boldsymbol{z}_k}$

  $\boldsymbol{x}_{k+1} \leftarrow \boldsymbol{x}_k + \alpha_k \boldsymbol{v}_k$

  $\boldsymbol{r}_{k+1} \leftarrow \boldsymbol{r}_k - \alpha_k \boldsymbol{z}_k$

  $\beta_k \leftarrow \frac{\boldsymbol{r}_{k+1}^{\mathsf{H}} \boldsymbol{r}_{k+1}}{\boldsymbol{r}_k^{\mathsf{H}} \boldsymbol{r}_k}$

  $\boldsymbol{v}_{k+1} \leftarrow \boldsymbol{r}_{k+1} + \beta_k \boldsymbol{d}_k$

end for

Conjugated Gradient

Remarks:

- The CG algorithm converges in (less than) N iterations, often much faster.
- The convergence directly depends on the conditioning of the system matrix $\boldsymbol{A}$. The better it is conditioned, the faster is the convergence.
- In each iteration step 4. is the expensive one $\mathcal{O}(N^2)$.
  $\Rightarrow$ Total time complexity $\mathcal{O}(N^3)$.
- If $\boldsymbol{A}$ is not stored explicitly (Fourier transform, Radon transform), the CG algorithm allows for *matrix-free* calculation of the matrix-vector products.