Numerical optimization

Mines Nancy – Fall 2024

session 6 – conjugate gradient methods

Lecturer: Julien Flamant (CNRS, CRAN)

`julien.flamant@univ-lorraine.fr`

Office 425, FST 1er cycle

Course material:

`arche.univ-lorraine.fr/course/view.php?id=74098`

`github.com/jflamant/mines-nancy-fall24-optimization`

Setting

We consider the linear system of N equations

$$\mathbf{Q}\mathbf{x} = \mathbf{p} \tag{$\mathcal{L}$}$$

where $\mathbf{x} \in \mathbb{R}^N$, $\mathbf{Q} \in \mathbb{R}^{N\times N}$ and $\mathbf{p} \in \mathbb{R}^N$. We assume that $\mathbf{Q} \succ 0$.

The solution to $(\mathcal{L})$ is of course $\mathbf{x}^\star = \mathbf{Q}^{-1}\mathbf{p}$.

Comments

- for large N, the cost of matrix inversion makes the direct calculation of $\mathbf{x}^\star$ untractable;
- instead, we seek iterative methods to solve $(\mathcal{L})$;
- the specific structure of the problem calls for a tailored solution: this is known as the linear conjugate gradient method;
- problem $(\mathcal{L})$ is closely related to least squares problems.

main reference for this lecture: Nocedal and Wright (2006), Chapter 5.

Reminder: least squares normal equations

Consider the least squares problem

$$\min_{\mathbf{x}\in\mathbb{R}^N} \|\mathbf{y}-\mathbf{A}\mathbf{x}\|_2^2, \quad \mathbf{A}\in\mathbb{R}^{M\times N}, \mathbf{y}\in\mathbb{R}^M$$

where we assume that $\operatorname{rank}\mathbf{A} = N$.

Recall the normal equations for this optimization problem:

$$\nabla f(\mathbf{x}) = \mathbf{0} \Leftrightarrow \mathbf{A}^\top\mathbf{A}\mathbf{x} = \mathbf{A}^\top\mathbf{y}$$

Pose $\mathbf{Q} = \mathbf{A}^\top\mathbf{A}$ and $\mathbf{p} = \mathbf{A}^\top\mathbf{y}$, we obtain the linear system $(\mathcal{L})$. (recall that $\mathbf{Q} \succ 0$ since $\operatorname{rank}\mathbf{A} = N$). Hence the two problems are equivalent!

The linear conjugate gradient method is effective
for solving large-scale least squares problems

Outline

General idea: exploiting the conjugacy property

Conjugate gradient methods rely on the *conjugacy* property.

Let $\mathbf{Q} \in \mathbb{R}^{N \times N}$ a positive definite matrix.

A set of non-zero vectors $\{\mathbf{d}_0, \mathbf{d}_1, \ldots, \mathbf{d}_\ell\}$ is said to be *conjugate* with respect to $\mathbf{Q}$ if

$$\mathbf{d}_i^\top \mathbf{Q} \mathbf{d}_j = 0 \text{ for all } i \neq j.$$

Note that this implies that the $\mathbf{d}_i' s$ are linearly independent.

General idea: exploiting the conjugacy property

Conjugate gradient methods rely on the *conjugacy* property.

Let $\mathbf{Q} \in \mathbb{R}^{N\times N}$ a positive definite matrix.

A set of non-zero vectors $\{\mathbf{d}_0, \mathbf{d}_1, \dots, \mathbf{d}_\ell\}$ is said to be *conjugate* with respect to $\mathbf{Q}$ if

$$\mathbf{d}_i^\top \mathbf{Q} \mathbf{d}_j = 0 \text{ for all } i \neq j.$$

Note that this implies that the $\mathbf{d}_i' s$ are linearly independent.

Conjugate direction method

Let $\mathbf{x}^{(0)} \in \mathbb{R}^N$ a starting point and $\{\mathbf{d}_0, \mathbf{d}_1, \dots, \mathbf{d}_\ell\}$ a set of conjugate directions.

At iteration k, generate the new iterate as

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{d}_k$$

where α_k is the optimal step size for the quadratic function associated to the systems of equations $\mathbf{Q}\mathbf{x} = \mathbf{p}$.

Conjugate direction algorithm (I)

Computation of α_k Let $\phi(\alpha) = \frac{1}{2}\mathbf{t}(\alpha)^\top \mathbf{Q}\mathbf{t}(\alpha) - \mathbf{p}^\top \mathbf{t}(\alpha)$ where $\mathbf{t}(\alpha) = \mathbf{x}^{(k)} + \alpha \mathbf{d}_k$. Hence $\alpha_k = \arg\min_\alpha \phi(\alpha)$.

It is a quadratic convex function of α, so it suffices to cancel out the derivative to get α_k.

$$\begin{aligned}\phi'(\alpha) &= \mathbf{d}_k^\top \mathbf{Q}\left[\mathbf{x}^{(k)} + \alpha \mathbf{d}_k\right] - \mathbf{p}^\top \mathbf{d}_k \\ &= +\alpha \mathbf{d}_k^\top \mathbf{Q}\mathbf{d}_k + \mathbf{d}_k^\top\left[\mathbf{Q}\mathbf{x}^{(k)} - \mathbf{p}\right]\end{aligned}$$

Define $\mathbf{r}_k = \mathbf{Q}\mathbf{x}^{(k)} - \mathbf{p}$ the residual at iteration k. Then

$$\alpha_k = -\frac{\mathbf{d_k^\top r_k}}{\mathbf{d}_k^\top \mathbf{Q}\mathbf{d}_k}$$

Conjugate direction algorithm (II)

Theorem (Nocedal and Wright (2006))

The sequence $\{\mathbf{x}^{(k+1)}\}$ *generated by*

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{d}_k, \ \textit{where } \alpha_k = -\frac{\mathbf{d_k^\top r_k}}{\mathbf{d}_k^\top \mathbf{Q} \mathbf{d}_k}$$

where the $\{\mathbf{d}_k\}$ *are conjugate directions converges to the solution* $\mathbf{x}^\star$ *of the linear system in at most* N *steps.*

Proof Let us consider N conjugate directions. This implies that the directions are linearly independent and thus they must span $\mathbb{R}^N$. In particular, there exist $\sigma_0, \ldots, \sigma_{N-1} \in \mathbb{R}$ such that

$$\mathbf{x}^\star - \mathbf{x}^{(0)} = \sum_{i=0}^{N-1} \sigma_i \mathbf{d}_i$$

Proof (continued)

By using the conjugacy property, we can obtain the value of σ_k as

$$\mathbf{d}_k^\top \mathbf{Q}\left[\mathbf{x}^\star - \mathbf{x}^{(0)}\right] = \mathbf{d}_k^\top \mathbf{Q}\left[\sum_{i=0}^{N-1} \sigma_i \mathbf{d}_i\right] = \sigma_k \mathbf{d}_k^\top \mathbf{Q}\mathbf{d}_k$$

hence $\sigma_k = \frac{\mathbf{d}_k^\top \mathbf{Q}\left[\mathbf{x}^\star - \mathbf{x}^{(0)}\right]}{\mathbf{d}_k^\top \mathbf{Q}\mathbf{d}_k}$. Next we prove that $\alpha_k = \sigma_k$.

By the recursion formula, we have for $k > 0, \mathbf{x}^{(k)} = \mathbf{x}^{(0)} + \sum_{i=0}^{k-1} \alpha_i \mathbf{d}_i$. Then by conjugacy

$$\mathbf{d}_k^\top \mathbf{Q}\mathbf{x}^{(k)} = \mathbf{d}_k^\top \mathbf{Q}\mathbf{x}^{(0)} \Leftrightarrow \mathbf{d}_k^\top \mathbf{Q}\left[\mathbf{x}^{(k)} - \mathbf{x}^{(0)}\right] = 0$$

and as a result

$$\mathbf{d}_k^\top \mathbf{Q}\left[\mathbf{x}^\star - \mathbf{x}^{(0)}\right] = \mathbf{d}_k^\top \mathbf{Q}\left[\mathbf{x}^\star - \mathbf{x}^{(k)}\right] = \mathbf{d}_k^\top\left[\mathbf{p} - \mathbf{Q}\mathbf{x}^{(k)}\right] = -\mathbf{d}_k^\top \mathbf{r}_k$$

which permits to conclude that $\alpha_k = \sigma_k$.

Additional properties

Properties of the conjugate direction method

Let $\{\mathbf{x}^{(k)}\}$ be generated by the conjugate direction method. Then, for an iteration k, one has

- $\mathbf{r}_k = \mathbf{r}_{k-1} + \alpha_{k-1}\mathbf{Q}\mathbf{d}_{k-1}$
- $\mathbf{d}_i^\top \mathbf{r}_k = 0$ for $i = 0, 1, \ldots, k-1$
- $\mathbf{x}^{(k)}$ is the minimizer of $\phi(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top\mathbf{Q}\mathbf{x} - \mathbf{p}^\top\mathbf{x}$ over the set $\{\mathbf{x} \mid \mathbf{x}^{(0)} + \mathrm{span}\{\mathbf{d}_0, \mathbf{d}_1, \ldots, \mathbf{d}_{k-1}\}\}$

Comments

- See Nocedal and Wright (2006), p. 106 for a proof.
- The residual $\mathbf{r}_k$ only depends on the previous one and on the current iteration $(\alpha_{k-1}, \mathbf{d}_{k-1})$
- the residual $\mathbf{r}_k$ is orthogonal to the k previous conjugate directions

The linear conjugate gradient method

The conjugate directions method requires a set of conjugate directions $\{\mathbf{d}_k\}$, which can be determined in advance using, e.g.,

- eigenvalue decomposition of $\mathbf{Q}$;
- modification of the Gram-Schmidt process to produce a set of conjugate directions

$\rightarrow$ too costly for large scale applications

Solution: (linear) conjugate gradient (CG) method

- directions $\mathbf{d}_k$ are computed iteratively;
- each new direction $\mathbf{d}_k$ use only $\mathbf{d}_{k-1}$; it is automatically conjugated to all the previous directions $\mathbf{d}_0, \mathbf{d}_1, \ldots, \mathbf{d}_{k-1}$
- cheap computational cost and memory storage.

The linear conjugate gradient method

Idea: choose the direction $\mathbf{d}_k$ as a linear combination of $-\mathbf{r}_k$ and $\mathbf{d}_{k-1}$

$$\mathbf{d}_k = -\mathbf{r}_k + \beta_k \mathbf{d}_{k-1}$$

where β_k is set to impose conjugation between $\mathbf{d}_k$ and $\mathbf{d}_{k-1}$:

$$\beta_k = \frac{\mathbf{d}_{k-1}^\top \mathbf{Q} \mathbf{r}_k}{\mathbf{d}_{k-1}^\top \mathbf{Q} \mathbf{d}_{k-1}}$$

This gives us a preliminary version of CG.
Starting from $\mathbf{x}^{(0)} \in \mathbb{R}^N$ and $\mathbf{d}_0 = -\mathbf{r}_0$, iterate until convergence

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{d}_k, \text{ where } \alpha_k = -\frac{\mathbf{d}_\mathbf{k}^\top \mathbf{r}_\mathbf{k}}{\mathbf{d}_k^\top \mathbf{Q} \mathbf{d}_k}$$

$$\mathbf{d}_{k+1} = -\mathbf{r}_{k+1} + \beta_{k+1} \mathbf{d}_k, \text{ where } \beta_{k+1} = \frac{\mathbf{d}_k^\top \mathbf{Q} \mathbf{r}_{k+1}}{\mathbf{d}_k^\top \mathbf{Q} \mathbf{d}_k}$$

Results for CG: theorem

This first version is practical for *studying properties* of CG.

Krylov subspace: useful tool in numerical linear algebra

$$\mathcal{K}(\mathbf{r}; k) = \text{span}\{\mathbf{r}, \mathbf{Q}\mathbf{r}, \mathbf{Q}^2\mathbf{r}, \ldots, \mathbf{Q}^k\mathbf{r}\}$$

Theorem (Nocedal and Wright (2006))

Consider the k-th iteration of the CG method (not the final one). Then

$$\begin{aligned} \mathbf{r}_k^\top \mathbf{r}_i &= 0 \text{ for } i = 0, 1, \ldots, k-1 \\ \text{span}\{\mathbf{r}_0, \mathbf{r}_1, \ldots, \mathbf{r}_k\} &= \mathcal{K}(\mathbf{r}_0; k) \\ \text{span}\{\mathbf{d}_0, \mathbf{d}_1, \ldots, \mathbf{d}_k\} &= \mathcal{K}(\mathbf{r}_0; k) \\ \mathbf{d}_k^\top \mathbf{Q}\mathbf{d}_i &= 0 \text{ for } i = 0, 1, \ldots, k-1 \end{aligned}$$

and thus the sequence $\{\mathbf{x}^{(k)}\}$ *converges to* $\mathbf{x}^\star$ *in at most* N *iterations.*

proof: see Nocedal and Wright (2006), pp. 109-111.

Results for CG: some comments

Comments on the theorem

- residuals $\{\mathbf{r}_k\}$ are mutually orthogonal;
- Residuals $\mathbf{r}_k$ and search directions $\mathbf{d}_k$ all belong to the Krylov subspace of order k associated with $\mathbf{r}_0$;
- the search directions $\mathbf{d}_0, \mathbf{d}_1, \ldots, \mathbf{d}_{N-1}$ are conjugate wrt $\mathbf{Q}$
- by the theorem on conjugate directions (slide 6), this implies termination in at most N steps.
- these results all depend on the choice of $\mathbf{d}_0$! In fact, if one chooses $\mathbf{d}_0 \neq -\mathbf{r}_0$ (i.e., different from the steepest direction at $\mathbf{x}^{(0)}$) then the theorem does not longer holds.

Standard CG algorithm

Using the Theorems on slide 8 and 11, we can improve the computations of α_k and β_{k+1} in CG:

$$\alpha_k = -\frac{\mathbf{d}_k^\top \mathbf{r}_k}{\mathbf{d}_k^\top \mathbf{Q}\mathbf{d}_k} = -\frac{\left[-\mathbf{r}_k + \beta_k \mathbf{d}_{k-1}\right]^\top \mathbf{r}_k}{\mathbf{d}_k^\top \mathbf{Q}\mathbf{d}_k} = \frac{\|\mathbf{r}_k\|_2^2}{\mathbf{d}_k^\top \mathbf{Q}\mathbf{d}_k}$$

Moreover, observe that $\mathbf{r}_{k+1} - \mathbf{r}_k = \alpha_k \mathbf{Q}\mathbf{d}_k$ so that

$$\beta_{k+1} = \frac{\mathbf{d}_k^\top \mathbf{Q}\mathbf{r}_{k+1}}{\mathbf{d}_k^\top \mathbf{Q}\mathbf{d}_k} = \frac{\mathbf{d}_k^\top \mathbf{Q}\mathbf{d}_k}{\|\mathbf{r}_k\|_2^2} \frac{\left[\mathbf{r}_{k+1} - \mathbf{r}_k\right]^\top \mathbf{r}_{k+1}}{\mathbf{d}_k^\top \mathbf{Q}\mathbf{d}_k} = \frac{\|\mathbf{r}_{k+1}\|_2^2}{\|\mathbf{r}_k\|_2^2}$$

where we used the orthogonality of residuals.

Standard CG algorithm

For which problems?

- Solve "square" linear systems of the form $\mathbf{Q}\mathbf{x} = \mathbf{p}$ with $\mathbf{Q} \succ 0$
- For strictly convex quadratic problems $\min_{\mathbf{x}} \frac{1}{2}\mathbf{x}^\top \mathbf{Q}\mathbf{x} - \mathbf{p}^\top \mathbf{x}$ (or full column-rank least-squares problems)

Standard linear CG algorithm

```
1: input: $\mathbf{x}^{(0)} \in \mathbb{R}^N$
2: Set $\mathbf{r}_0 = \mathbf{Q}\mathbf{x}^{(0)} - \mathbf{p}$ and $\mathbf{d}_0 = -\mathbf{r}_0$.
3: $k = 0$
4: while $\|\mathbf{r}_k\|_2 \neq 0$ do
5:     $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{d}_k$, where $\alpha_k = \frac{\|\mathbf{r}_k\|_2^2}{\mathbf{d}_k^\top \mathbf{Q}\mathbf{d}_k}$
6:     $\mathbf{d}_{k+1} = -\mathbf{r}_{k+1} + \beta_{k+1}\mathbf{d}_k$, where $\beta_{k+1} = \frac{\|\mathbf{r}_{k+1}\|_2^2}{\|\mathbf{r}_k\|_2^2}$
7:     $k = k + 1$
8: end while
9: return $\mathbf{x}^{(k)}$
```

Numerical performance and conditioning

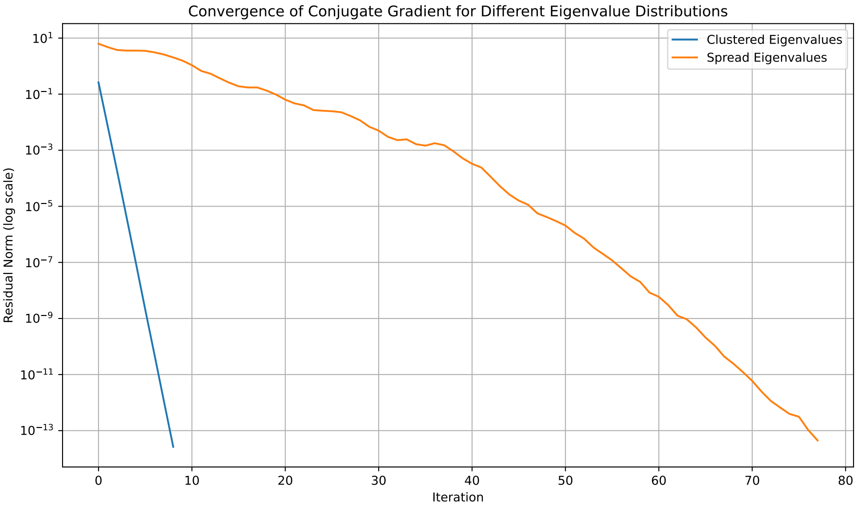
- convergence to $\mathbf{x}^\star$ is guaranteed after at most N iterations
- but it can be much faster!

  Trivial example Solve system $\mathbf{Q}\mathbf{x} = \mathbf{p}$ using CG when $\mathbf{Q} = \lambda \mathbf{I}_N$.
  first residual $\mathbf{r}_0 = \lambda \mathbf{x}^{(0)} - \mathbf{p}$ and direction $\mathbf{d}_0 = -\mathbf{r}_0$.
  first iteration:

$$\mathbf{x}^{(1)} = \mathbf{x}^{(0)} - \frac{\|\lambda \mathbf{x}^{(0)} - \mathbf{p}\|^2}{\lambda \|\lambda \mathbf{x}^{(0)} - \mathbf{p}\|^2}\|_2(\lambda \mathbf{x}^{(0)} - \mathbf{p}) = \mathbf{x}^{(0)} - \mathbf{x}^{(0)} + \lambda^{-1}\mathbf{p} = \mathbf{x}^\star$$

  Then $\mathbf{r}_{k+1} = \mathbf{0}$ and $\mathbf{d}_{k+1} = \mathbf{0}$. CG stops after exactly 1 iteration.
- convergence usually depends on the distribution of eigenvalues of $\mathbf{Q}$
- it can be shown that if $\mathbf{Q}$ has r distinct eigenvalues, then CG converges in at most r iterations.

Numerical experiment for $N = 100$



Summary for linear conjugate gradient

- exact convergence guarantees in at most N iterations

  compare with results of session 5 on descent methods
- state of the art for solving linear least squares in high-dimensions
- the closer the condition number of $\mathbf{Q}$ is to 1, the faster the convergence

  $\rightarrow$ use of preconditioners to improve the condition number of $\mathbf{Q}$
- implemented in many scientific libraries, e.g., `scipy.sparse.linalg.cg` (Python) or cgs (Matlab)

Outline

1 Linear conjugate gradient method

2 Nonlinear conjugate gradient method

Motivation

Linear conjugate gradient solves linear system $\mathbf{Qx} = \mathbf{p}$, or equivalently,

$$\min_{\mathbf{x}\in\mathbb{R}^N} f(\mathbf{x}) := \frac{1}{2}\mathbf{x}^\top \mathbf{Q}\mathbf{x} - \mathbf{p}^\top \mathbf{x}, \quad \mathbf{Q} \succ 0$$

Nonlinear conjugate gradient methods extend CG to non-quadratic objective functions f.

Several variants exists, two popular ones:

- the Fletcher-Reeves method
- the Polak-Ribière method

the key principle is the following:
replace the residual $\mathbf{r}_k$ (which is the gradient of quadratic f) by the evaluation of the gradient of the nonlinear objective f at iteration $\mathbf{x}^{(k)}$.

Fletcher-Reeves vs Polyak-Ribière

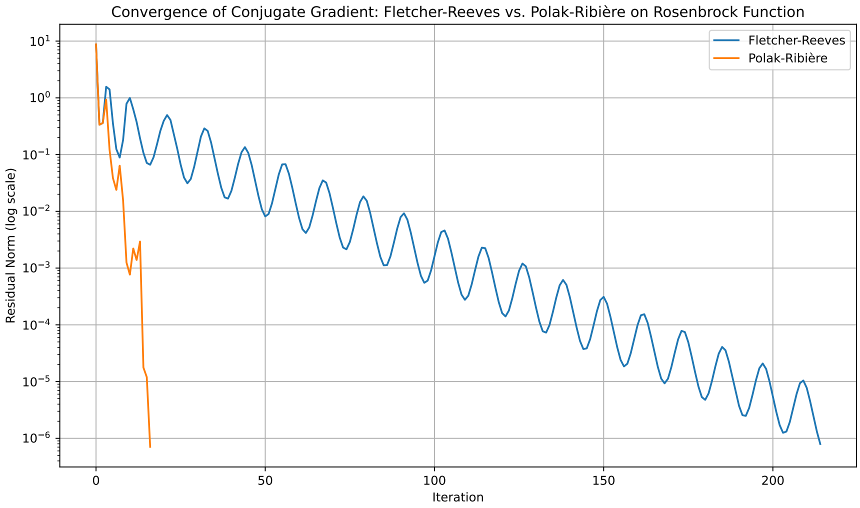
Fletcher-Reeves-CG algorithm

```
1: input: $\mathbf{x}^{(0)} \in \mathbb{R}^N$
2: Set $\nabla f_0 = \nabla f(\mathbf{x}^{(0)})$ and set $\mathbf{d}_0 = -\nabla f_0$.
3: $k = 0$
4: while $\|\nabla f_k\|_2 \neq 0$ do
5:     $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{d}_k$, where $\alpha_k$ is computed by line search
6:     $\mathbf{d}_{k+1} = -\nabla f_{k+1} + \beta^{\mathrm{FR}}_{k+1} \mathbf{d}_k$, where $\beta^{\mathrm{FR}}_{k+1} = \frac{\|\nabla f_{k+1}\|_2^2}{\|\nabla f_k\|_2^2}$
7:     $k = k + 1$
8: end while
9: return $\mathbf{x}^{(k)}$
```

- α_k must be chosen with care, to that $\mathbf{d}_{k+1}$ remains a descent direction. $\rightarrow$ use of strong Wolfe conditions
- Polak-Ribière: replace $\beta^{\mathrm{FR}}_{k+1}$ by

$$\beta^{\mathrm{PR}}_{k+1} = \frac{\nabla f_{k+1}^{\top} (\nabla f_{k+1} - \nabla f_k)}{\|\nabla f_k\|_2^2} \text{ or } \beta^{+}_{k+1} = \max(0, \beta^{\mathrm{PR}}_{k+1}) \text{ (more robust)}$$

Numerical experiment on Rosenbrock function



Summary

- PR is usually more efficient than FR; but this not a general rule and depends on the context
- many other variants exist: Hestenes-Stiefel, Dai–Yuan, etc.
- convergence analysis is still possible, but much more technical than for the linear conjugate gradient method: see Nocedal and Wright (2006), Chapter 5 for details.
- the Polak-Ribière method is implemented in Python: check out `scipy.optimize.fmin_cg`