Computed Tomography: Algorithms, Insight, and Just Enough Theory

Chapter 13: Optimization Methods for Tomography

Cecilie Priller

TUM School of Computation, Information and Technology (CIT)
Department of Computer Science
Computational Imaging and Inverse Problems

July 12th, 2023

Outline

Optimization Problems in Computed Tomography

- Tomographic Reconstruction: Inverse problem Ax = b
  Solve the least-squares problem:

$$x_{LS} = \arg\min_x \frac{1}{2} \|b - Ax\|_2^2$$

- Cimmino's Method:

$$x^{k+1} = \frac{1}{m} \sum_{i=1}^{m} P_i(x^{(k)})$$

- SIRT:

$$x^{k+1} = x^k + D_1 A^T M_1 (b - Ax^{(k)})$$

- Regularization methods:
  - Tikhonov Regularization:

$$\min_{x} \left\{ \frac{1}{2}\|b - Ax\|_2^2 + \alpha \frac{1}{2}\|x\|_2^2 \right\}$$

  - Total Variation Regularization
    - anisotropic TV

$$TV_a(x) = \|(I_N \otimes D_M)x\|_1 + \|(D_N \otimes I_M)x\|_1$$

    - isotropic TV

$$TV_i(x) = \sum_{i=1}^{n} \sqrt{[(I_N \otimes D_M)x]_i^2 + [(D_N \otimes I_M)x]_i^2}$$

Goal:

- Discuss first-order optimization methods related to these problems

Optimization problem

General form:

$$\textit{minimise } g(x)$$
or
$$\textit{minimise } -g(x)$$

$$\text{subject to } C$$

with $g(x)$ as the *objective function* and a *set of constraints C*

- *C* empty -> unconstrained optimization problem
- *C* non-empty -> constrained optimization problem

Outline

Gradient of a multivariate function

Consider:

$$g : D \subseteq \mathbb{R}^n \to \mathbb{R}, \quad \boldsymbol{x} = (x_1, ..., x_n)^\top \mapsto g(x_1, ..., x_n)$$

$$\nabla g(\boldsymbol{x}) = \begin{pmatrix} \frac{\partial g}{\partial x_1} \\ \vdots \\ \frac{\partial g}{\partial x_n} \end{pmatrix}$$

gives us the direction of steepest ascent of g at point $\boldsymbol{x}$

Gradient of a multivariate function

Consider:

$$g : D \subseteq \mathbb{R}^n \to \mathbb{R}, \quad \boldsymbol{x} = (x_1, ..., x_n)^\top \mapsto g(x_1, ..., x_n)$$

$$\nabla g(\boldsymbol{x}) = \begin{pmatrix} \frac{\partial g}{\partial x_1} \\ \vdots \\ \frac{\partial g}{\partial x_n} \end{pmatrix}$$

gives us the direction of steepest ascent of g at point $\boldsymbol{x}$

We conclude:

$$-\nabla g(\boldsymbol{x})$$

gives us

Gradient of a multivariate function

Consider:

$$g : D \subseteq \mathbb{R}^n \to \mathbb{R}, \quad \boldsymbol{x} = (x_1, ..., x_n)^\top \mapsto g(x_1, ..., x_n)$$

$$\nabla g(\boldsymbol{x}) = \begin{pmatrix} \frac{\partial g}{\partial x_1} \\ \vdots \\ \frac{\partial g}{\partial x_n} \end{pmatrix}$$

gives us the direction of steepest ascent of g at point $\boldsymbol{x}$

We conclude:

$$-\nabla g(\boldsymbol{x})$$

gives us the direction of steepest descent of g at point $\boldsymbol{x}$

Minimising a function locally using the gradient

$$\text{minimise } g(x)$$

with $x \in \mathbb{R}^n$ and $g : \mathbb{R}^n \to \mathbb{R}$ continuously differentiable

- $\boldsymbol{x}^*$ global minimizer of $g \iff g(y) \geq g(\boldsymbol{x}^*) \quad \forall y \in \mathbb{R}^n$
- *first-order necessary condition:* $\nabla g(\boldsymbol{x}^*) = 0$ (stationary point)

Find a stationary point relative to an initial guess $\underline{\boldsymbol{x}^{(\mathbf{0})}}$

$$x^{(k+1)} = x^{(k)} - t_k \nabla g(x^{(k)}), \quad k = 0, 1, 2, \ldots$$

with $-\nabla g(x^{(k)})$ as the descent direction and t_k as a sufficiently small step size

Gradient method with a constant step size

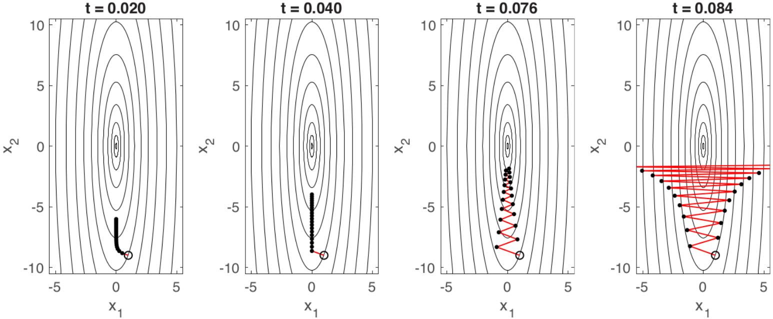


Figure 13.3. *Contour plots of g with the first* 20 *iterations of the gradient method using a constant step size. The initial guess is marked with a circle, and the subsequent iterations are marked with solid black dots connected by a red line that indicates the order of the iterates.*

[4]

Outline

Definition

A function g is ***Lipschitz continuous*** if and only if

$$\exists L \in \mathbb{R}. \quad \forall x, y \in \mathbb{R}^n. \quad \|g(x) - g(y)\| \leq L\|x - y\|$$

How is this helpful?

If the gradient of g is Lipschitz continuous with constant L:

$$g(y) \leq g(x) + \nabla g(x)^\top (y - x) + \frac{L}{2}\|y - x\|_2^2 \quad \forall x, y \in \mathbb{R}^n$$

Example: Least-squares problem

$g(x) = \frac{1}{2}\|b - Ax\|_2^2$

$$\boldsymbol{\nabla g(x) = A^\top (Ax - b)}$$

From this we derive that:

$$\begin{aligned} \boldsymbol{\|\nabla g(y) - \nabla g(x)\|_2} &= \|A^\top (Ay - b) - A^\top (Ax - b)\|_2 \\ &= \|A^\top A(y - x)\|_2 \\ &\boldsymbol{\leq \|A^\top A\|_2 \|y - x\|_2} \end{aligned}$$

We can deduce: ∇g is Lipschitz continuous with constant $L = \|A^\top A\|_2$

Boundedness of the Hessian Matrix

Let g be twice continuously differentiable, the *Hessian matrix of g* corresponds to

$$H(x) = \nabla^2 g(x) = \begin{pmatrix} \frac{\partial^2 g(x)}{\partial x_1^2} & \frac{\partial^2 g(x)}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 g(x)}{\partial x_1 \partial x_n} \\ \frac{\partial^2 g(x)}{\partial x_2 \partial x_1} & \frac{\partial^2 g(x)}{\partial x_2^2} & \cdots & \frac{\partial^2 g(x)}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 g(x)}{\partial x_n \partial x_1} & \frac{\partial^2 g(x)}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 g(x)}{\partial x_n^2} \end{pmatrix}$$

We can show that if $\|\nabla^2 g(x)\|_2 \leq L$ for all x then the gradient of g is Lipschitz continuous with constant L.

(using the Cauchy-Schwarz inequality and the fundamental theorem of calculus on $\|\nabla g(y) - \nabla g(x)\|_2$)

Converse also true

Power iteration

For $g(x) = \frac{1}{2}\|b - Ax\|_M^2$ with M positive definite

$$H = A^\top M A \quad \text{and} \quad L = \|A^\top M A\|_2$$

However H can be very large and difficult to work with! Since $\|H\|_2 = \lambda_{max}(H)$

- **power iteration** to approximate the largest eigenvalue.
- **algorithm:**

---

parameters: relative tolerance $\epsilon > 0$
$x^{(0)}$ =random vector
$x^{(0)} = x^{(0)}/\|x^{(0)}\|_2$
$\hat{\lambda}^{(0)} = 0$
for $k = 0, 1, 2, \ldots$ **do**
    $w = Hx^{(k)}$
    $\hat{\lambda}^{(k+1)} = \|w\|_2$
    $x^{(k+1)} = w/\hat{\lambda}^{(k+1)}$
    stop if $|\hat{\lambda}^{(k+1)} - \hat{\lambda}^{(k)}| \leq \hat{\lambda}^{(k)}\epsilon$
end for

Majorization

$\psi(y;x)$ is a majorization of g at x if

$$\psi(x;x) = g(x) \quad and \quad \psi(y;x) \geq g(y) \quad \forall y$$

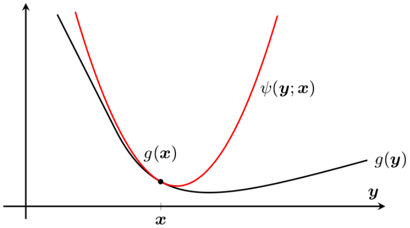


Figure 13.4. *The function $\psi(\boldsymbol{y};\boldsymbol{x})$ is a majorization of g at $\boldsymbol{x}$ since $\psi(\boldsymbol{x};\boldsymbol{x}) = g(\boldsymbol{x})$ and $\psi(\boldsymbol{y};\boldsymbol{x}) \geqslant g(\boldsymbol{x})$ for all $\boldsymbol{y}$.*

[4]

Minimizing the majorization

With an initial guess $x^{(0)}$:

$$x^{(k+1)} = \arg\min_{y} \left\{ \psi(y; x^{(k)}) \right\}, \quad k = 0, 1, 2, \ldots$$

Hence

$$g(x^{(k+1)}) \leq \psi(x^{(k+1)}; x^{(k)}) \leq \psi(x^{(k)}; x^{(k)}) = g(x^{(k)})$$

*Sufficient for descent property:

$$\psi(x^{(k+1)}; x^{(k)}) \leq \psi(x^{(k)}; x^{(k)})$$

Majorization minimization with Lipschitz continuous ∇g

Let $\nabla g(x)$ be Lipschitz continuous with constant L.

$$\begin{aligned}\boldsymbol{x^{(k+1)}} &= \arg\min_y \{g(x^{(k)}) + \nabla g(x^{(k)})^\top (y - x^{(k)}) + \frac{L}{2}\|y - x^{(k)}\|_2^2\} \\ &= \boldsymbol{x^{(k)} - \frac{1}{L}\nabla g(x^{(k)})}\end{aligned}$$

We can show that:

$$g(x^{(k+1)}) \leq g(x^{(k)}) - \frac{1}{2L}\|\nabla g(x^{(k)})\|_2^2$$

We can also show that:

$$\lim_{k\to\infty} \nabla g(x^{(k)}) = 0$$

We converge to a stationary point of g!

Example of a majorization minimization algorithm

General class of SIRT-like algorithms in the form:

$$x^{(k+1)} = x^{(k)} - \lambda_k D A^\top M (A x^{(k)} - b)$$

with $\lambda_k > 0$ as a scalar parameter and D and M positive definite and diagonal

- minimise objective function: $g(x) = \frac{1}{2}\|b - Ax\|_M^2$ via a scaled gradient method
- $\nabla g(x)$ is Lipschitz continuous with constant $L = 1$ under the assumption that $\|M^{1/2} A D^{1/2}\|_2 \leq 1$
  We construct the majorization:

$$g(y) \leq g(x) + \nabla g(x)^\top (y - x) + \frac{1}{2}(y - x)^\top D^{-1}(y - x)$$

- Minimise the right-hand side:

$$y = x - D \nabla g(x)$$

- Substitute $x^{(k)}$ for x and $x^{(k+1)}$ for y, we get the iteration:

$$x^{(k+1)} = x^{(k)} - \lambda_k D \nabla g(x^{(k)})$$

- We get a new upper bound:

$$g(x^{(k+1)}) \leq g(x^{(k)}) - \frac{2\lambda_k - \lambda_k^2}{2} \|\nabla g(x^{(k)})\|_D^2$$

(descent method for $\lambda_k \in (0, 2)$)

- The above analysis can be done for diagonal matrices:

$$D_{jj} = \left(\sum_{i=1}^{m} |A_{ij}|^{\alpha} \right)^{-1}, \qquad M_{ii} = \left(\sum_{j=1}^{n} |A_{ij}|^{2-\alpha} \right)^{-1}$$

$\alpha \in [0, 2]$ and $|A_{ij}|^0 = 1$ when $A_{ij} = 0$
$\rightarrow$ Cimmino method for $\alpha = 0$, SIRT method for $\alpha = 1$

Outline

Convexity

$g : \mathbb{R}^n \to \mathbb{R}$ is convex if and only if

$$\boldsymbol{g(\gamma x + (1-\gamma)y) \leq \gamma g(x) + (1-\gamma)g(y), \quad \gamma \in [0,1] \quad \forall x, y \in \mathbb{R}^n} \tag{1}$$

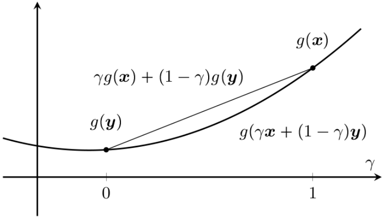


Figure 13.5. *A function g is convex if every line segment that joins two points on its graph (parametrized by $\gamma \in [0,1]$) does not lie below the graph at any point.*

[4]

Any stationary point of a convex function is a minimizer.

If g is continuously differentiable, then (1) is equivalent to

$$g(y) \geq g(x) + \nabla g(x)^\top (y - x), \quad \forall x, y \in \mathbb{R}^n$$

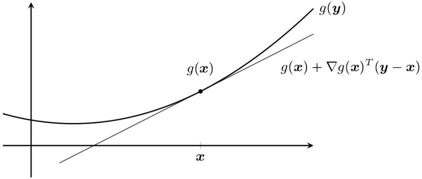


Figure 13.6. *The linearization of a continuously differentiable convex function yields a global linear lower bound.*

[4]

A stationary point is also a global minimizer.

Convergence of the gradient method

$g^* = \inf_x g(x)$ finite and attained at x^* with g being a differentiable convex function with Lipschitz continuous gradient.
Considering the gradient method with $t = \frac{\gamma}{L}$, $\gamma \in (0,2)$:

$$g(x^{(k)}) - g^* \leq \frac{2L\|x^{(0)} - x^*\|_2^2}{4 + \gamma(2-\gamma)k}$$

- objective suboptimality in $O(\frac{1}{k}) \longrightarrow g(x^{(k)})$ converges to g^* sublinearly
- sublinear convergence slower than linear convergence
- best upper bound for $\gamma = 1$, $t_k = \frac{1}{L}$
- $x^{(k)}$ with $g(x^{(k)}) - g(x^*) \leq \epsilon$ attained in at most $O(\frac{1}{\epsilon})$ iterations

Strong convexity

Function g is strongly convex with parameter $\mu > 0$ if

$$\tilde{g}(x) = g(x) - \frac{\mu}{2}\|x\|_2^2 \quad \text{is convex or}$$

$$g(\gamma x + (1-\gamma)y) \leq \gamma g(x) + (1-\gamma)g(y) - \frac{\gamma(1-\gamma)\mu}{2}\|x-y\|_2^2, \quad \forall \gamma \in [0,1], \forall x, y$$

If g is also continuously differentiable we can construct a convex quadratic global underestimator of g at any x:

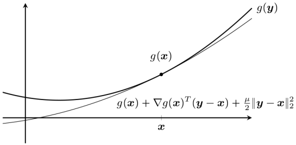


Figure 13.7. *A strongly convex function has a global convex* quadratic *lower bound at all points* $\boldsymbol{x}$.

[4]

Usefulness of strong convexity

- a stationary point x^* of g is a unique minimizer
- bounds that will be useful later:
  - upper bound on the objective suboptimality at x:

$$g(x) - g(x^*) \leq \frac{1}{2\mu}\|\nabla g(x)\|_2^2$$

  - upper bound on the distance from any x to the unique minimizer:

$$\|x^* - x\|_2 \leq \frac{2}{\mu}\|\nabla g(x)\|_2$$

- gradient method with the function being strongly convex with Lipschitz continuous gradient converges linearly to x^* (for $t_k = \frac{\gamma}{(L+\mu)}$ and $\gamma \in (0, 2]$)

$$\|x^{(k)} - x^*\|_2 \leq \left(1 - \frac{2\gamma\mu L}{(L+\mu)^2}\right)^{\frac{k}{2}} \|x^{(0)} - x^*\|_2$$

Convergence results

Using strong convexity we can deduce the following convergence results:

- Landweber's method

$$\|x^{(k)} - \overline{x}\|_2 \leq \left(1 - \frac{2}{1 + \mathsf{cond}(A)^2}\right)^k \|x^{(0)} - \overline{x}\|_2$$

- Cimmino's method

$$\|x^{(k)} - \overline{x}\|_2 \leq \left(1 - \frac{2}{1 + \mathsf{cond}(A^\top M A)}\right)^k \|x^{(0)} - \overline{x}\|_2$$

- Tikhonov regularization

$$\|x^{(k)} - x_{Tik}\|_2 \leq \left(1 - \frac{2}{2 + \|A\|_2^2/\alpha)}\right)^k \|x^{(0)} - x_{Tik}\|_2$$

Visualisation: Tikhonov regularization using the gradient method

Example 13.6:

$$\text{minimise} \frac{1}{2} \left\| \begin{pmatrix} A \\ \sqrt{\alpha} I \end{pmatrix} x - \begin{pmatrix} b \\ 0 \end{pmatrix} \right\|_2^2 \tag{2}$$

Reminder: For g convex, $t_k = \frac{\gamma}{L}$ and $\gamma \in (0, 2]$:

$$g(x^{(k)}) - g^* \leq \frac{2L\|x^{(0)} - x^*\|_2^2}{4 + \gamma(2-\gamma)k} \tag{3}$$

For g strongly convex, $t_k = \frac{\gamma}{(L+\mu)}$ and $\gamma \in (0, 2]$:

$$g(x^{(k)}) - g(x^*) \leq \frac{L}{2} \left(\frac{\frac{L}{\mu} - 1}{\frac{L}{\mu} + 1} \right)^{2k} \|x^{(0)} - x^*\|_2^2 \tag{4}$$

Asymptotic behaviour

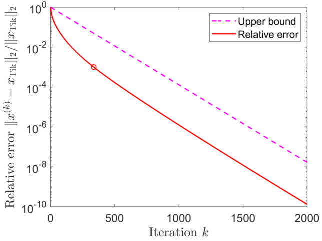


Figure 13.8. *Using the gradient method to solve the Tikhonov problem in Example 13.6. We show the relative error $\|\boldsymbol{x}^{(k)} - \boldsymbol{x}_{\mathrm{Tik}}\|_2/\|\boldsymbol{x}_{\mathrm{Tik}}\|_2$ versus the number of iterations k, together with the upper bound according to (13.37). A relative error of 10^{-3} is achieved after 336 iterations, as indicated by the circle.*

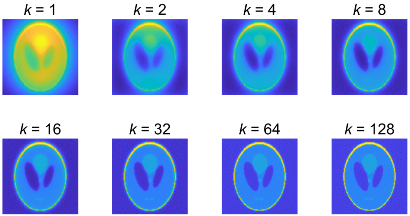


Figure 13.9. *Selected iterations from application of the gradient method to the Tikhonov problem in Example 13.6.*

[4]

For a strongly convex function (4) is asymptotically a better bound.

Nonasymptotic behaviour

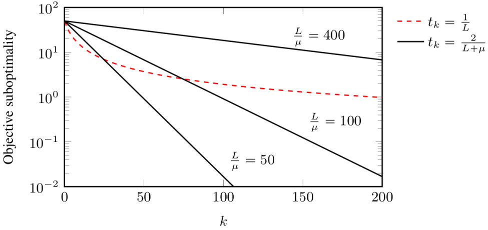


Figure 13.10. *Plot of the worst-case upper bounds* (13.22) *and* (13.28) *when* $L = 100$ *and* $\|\boldsymbol{x}^{(0)} - \boldsymbol{x}^\star\|_2 = 1$. *The latter bound depends on the ratio* L/μ.

[4]

Nonasymptotically, bound (4) is not necessarily better since it depends on $\frac{L}{\mu}$.

Outline

Exact line search

Reminder: Gradient method

$$x^{(k+1)} = x^{(k)} - t_k \nabla g(x^{(k)})$$

Goal: Find t_k such that the descent property holds $(g(x^{(k+1)}) < g(x^{(k)}))$

Possible solution: We define a step size that reduces our function optimally in the current iteration's search direction ("greedy algorithm")

$$t_k = \arg\min_{t>0} \{g(x^{(k)} - t \nabla g(x^{(k)}))\}$$

Problem: can be very inefficient (usage only in special scenarios)

Backtracking Line Search

- inexact line search method
- basis: **Armijo condition** for the gradient method with $\alpha \in]0, \frac{1}{2}[$

$$\boldsymbol{g(x^{(k)} - t\nabla g(x^{(k)})) \leq g(x^{(k)}) - \alpha t \|\nabla g(x^{(k)})\|_2^2}$$

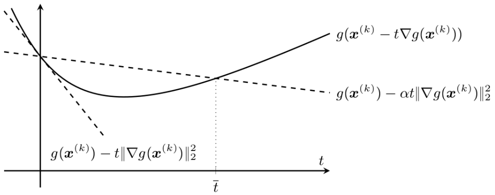


Figure 13.12. *Illustration of the Armijo condition* (13.38).

[4]

the reduction should be proportional to the step length and directional derivative

- **algorithm:**

---

parameters: $\alpha \in]0, \frac{1}{2}[$, $\beta \in (0, 1)$ and $t = t_0 > 0$
while $g(x^{(k)} - t\nabla g(x^{(k)})) > g(x^{(k)}) - \alpha t \|\nabla g(x^{(k)})\|_2^2$ **do**
  $t \longleftarrow t\beta$
end while

- **parameter α:** trade-off between the allowed step length and the required decrease, often smaller values chosen to allow bigger step sizes

- **parameter β:** trade-off between number of iterations of the algorithm and step length

BB Step Sizes

- adaptive step sizes without line search and no guaranteed descent
- **two rules:** Let Δy and Δs be

$$\boldsymbol{\Delta y} = \nabla g(x^{(k)}) - \nabla g(x^{(k-1)}) \text{ and } \boldsymbol{\Delta s} = x^{(k)} - x^{(k-1)}$$

  - BB1 step size

$$\boldsymbol{t_k^{BB1}} = \alpha_k^{-1}, \quad \alpha_k = \arg\min_{\alpha}\{\|\Delta y - \alpha\Delta s\|_2^2\} = \frac{\Delta s^\top \Delta y}{\|\Delta s\|_2^2}$$

  - BB2 step size

$$\boldsymbol{t_k^{BB2}} = \arg\min_{\beta}\{\|\beta\Delta y - \Delta s\|_2^2\} = \frac{\Delta s^\top \Delta y}{\|\Delta y\|_2^2}$$

- The first step size must be evaluated using a different method!
- Convergence is guaranteed under certain conditions (e.g. strongly convex and quadratic functions)

Associate plot with step size method

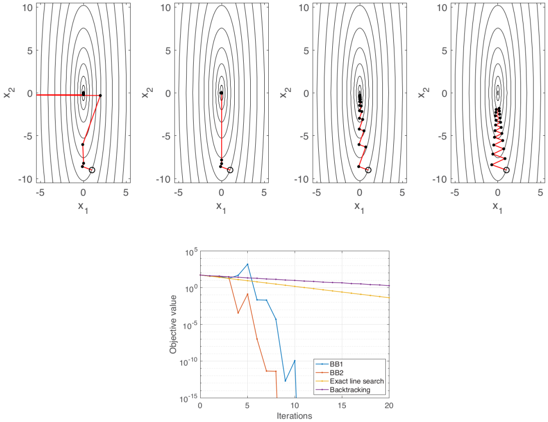
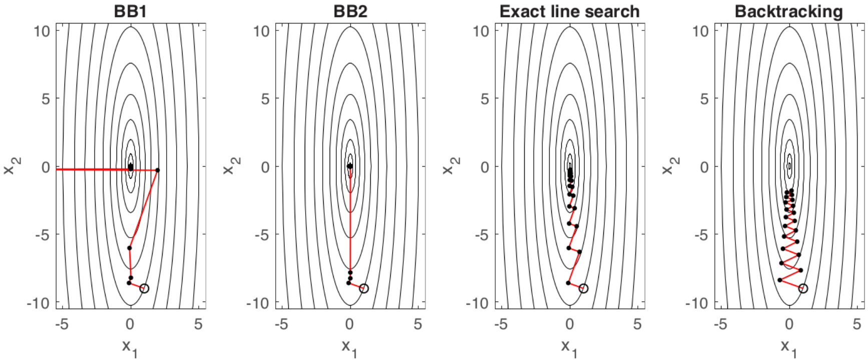


Figure 13.13. *Illustration of the gradient method applied to a quadratic function of two variables with the different step size rules. Each plot shows the initial point* $\boldsymbol{x}^{(0)}$ *(marked with a circle) and the first* 20 *iterations.*

[4]

Associate plot with step size method



Stopping Criteria

As seen the gradient method converges to a stationary point under some conditions.

But do we always have to keep iterating until we reach a stationary point?

Stopping Criteria

As seen the gradient method converges to a stationary point under some conditions.

But do we always have to keep iterating until we reach a stationary point?

No! Oftentimes it is sufficient to iterate until we reach a certain predefined termination condition.

Do you have any ideas for a simple stopping criterion?

Stopping Criteria

As seen the gradient method converges to a stationary point under some conditions.

But do we always have to keep iterating until we reach a stationary point?

No! Oftentimes it is sufficient to iterate until we reach a certain predefined termination condition.

Do you have any ideas for a simple stopping criterion?

For example:

$$\|\nabla g(x^{(k)})\|_2 \leq \epsilon$$

for a case-specific tolerance $\epsilon > 0$

Other options

$$\|x^{(k+1)} - x^{(k)}\|_2 \leq \epsilon \|x^{(k)}\|_2$$

which can be interpreted as the relative parameter change.

- Satisfied when $x^{(k+1)} \approx x^{(k)}$

Problem: scale invariance
Modification of variable: $x = Cy$ with $C \in \mathbb{R}^{n \times n}$ a nonsingular matrix

$$\tilde{g}(y) = g(Cy)$$

Then

$$\|\nabla g(x^{(k)})\|_2 \leq \epsilon$$

turns into

$$\|\nabla \tilde{g}(y^{(k)})\|_2 = \|C^\top \nabla g(x^{(k)})\|_2 \leq \epsilon$$

More options

- $g(x^{(k)}) - g(x^{(k+1)}) \leq \epsilon$
- limit the number of iterations

To be used with care!

Special case: strongly convex functions

Reminder:

$$g(x) - g(x^*) \leq \frac{1}{2\mu}\|\nabla g(x)\|_2^2 \qquad \|x^* - x\|_2 \leq \frac{2}{\mu}\|\nabla g(x)\|_2$$

- $\|\nabla g(x^{(k)})\|_2 \leq \sqrt{2\mu\epsilon_{obj}} \quad \Longrightarrow \quad g(x^{(k)}) - g(x^*) \leq \epsilon_{obj}$
- $\|\nabla g(x^{(k)})\|_2 \leq \frac{\mu\epsilon_{dist}}{2} \quad \Longrightarrow \quad \|x^{(k)} - x^*\|_2 \leq \epsilon_{dist}$

Outline

Constrained optimization problem

$$\text{minimise } \{g(x) + h(x)\}$$

with $g : \mathbb{R}^n \to \mathbb{R}$ convex and differentiable and $h : \mathbb{R}^n \to \mathbb{R} \cup \{-\infty, \infty\}$ closed convex

Let us look at the constrained optimization problem

$$\begin{gathered}\text{minimise } g(x) \\ \text{subject to } x \in C\end{gathered}$$

with $C \subseteq \mathbb{R}^n$ closed and convex

that we get for $h(x) = I_C(x)$

$$I_C(x) = \begin{cases} 0, & x \in C \\ \infty, & x \notin C \end{cases}$$

Proximal gradient methods

Proximal operator

$$prox_{th}(x) = \arg\min_y \left\{ th(y) + \frac{1}{2}\|y - x\|_2^2 \right\}$$

If g Lipschitz continuous with constant L

$$\psi(y; x) = g(x) + \nabla g(x)^\top (y - x) + \frac{L}{2}\|y - x\|_2^2 + h(y)$$

via majorization minimization:

$$\begin{aligned} x^{(k+1)} &= \arg\min_y \{\psi(y; x^{(k)})\} \\ &= \arg\min_y \{h(y) + \frac{L}{2}\|y - (x^{(k)} - (1/L)\nabla g(x^{(k)}))\|_2^2\} \\ &= prox_{th}(x^{(k)} - t\nabla g(x^{(k)})) \qquad \text{with } t = 1/L \end{aligned}$$

$x^{(0)} =$ initial vector

$t = 1/L$, where L is a Lipschitz constant associated with ∇g

for k=0,1,2,... **do**

$x^{(k+1)} = prox_{th}(x^{(k)} - t\nabla g(x^{(k)}))$

end for

algorithm:

$x^{(0)} =$ initial vector

$t = 1/L$, where L is a Lipschitz constant associated with ∇g

for k=0,1,2,... **do**

$$x^{(k+1)} = prox_{th}(x^{(k)} - t\nabla g(x^{(k)}))$$

end for

What would the running-time cost of this algorithm depend on?

algorithm:

$x^{(0)} =$ initial vector

$t = 1/L$, where L is a Lipschitz constant associated with ∇g

for k=0,1,2,... **do**

$$x^{(k+1)} = prox_{th}(x^{(k)} - t\nabla g(x^{(k)}))$$

end for

What would the running-time cost of this algorithm depend on?

The cost of the evaluation of the proximal operator

algorithm:

$x^{(0)} =$ initial vector
$t = 1/L$, where L is a Lipschitz constant associated with ∇g
for k=0,1,2,... **do**
$\quad x^{(k+1)} = prox_{th}(x^{(k)} - t\nabla g(x^{(k)}))$
end for

What would the running-time cost of this algorithm depend on?

The cost of the evaluation of the proximal operator

- in general same worst-case suboptimality bound as the gradient method **however**

  if g is strongly convex, the PG method with $t = 1/L$ converges linearly

Accelerated proximal gradient method (FISTA)

$x^{(0)} =$ initial vector
$y = x^{(0)}$
$t_0 = 1$
for k=0,1,2,... **do**
$\quad x^{(k+1)} = prox_{(1/L)h}(y - (1/L)\nabla g(y))$
$\quad t_{k+1} = \frac{1+\sqrt{1+4t_k^2}}{2}$
$\quad y = x^{(k+1)} + \frac{t_k - 1}{t_{k+1}}(x^{(k+1)} - x^{(k)})$
end for

Accelerated proximal gradient method (FISTA)

$x^{(0)} =$ initial vector
$y = x^{(0)}$
$t_0 = 1$
for k=0,1,2,... **do**
$\quad x^{(k+1)} = prox_{(1/L)h}(y - (1/L)\nabla g(y))$
$\quad t_{k+1} = \frac{1+\sqrt{1+4t_k^2}}{2}$
$\quad y = x^{(k+1)} + \frac{t_k - 1}{t_{k+1}}(x^{(k+1)} - x^{(k)})$
end for

- worst-case suboptimality $f(x^{(k)}) - f(x^*) = O(\frac{1}{k^2})$ (sublinear)
- not a descent method

Visualization

Problem:

$$\text{minimize } \{\tfrac{1}{2}\|b - Ax\|_2^2 + \tfrac{\gamma}{2}\|x\|_2^2\}$$
$$\text{subject to } x_i \geq 0, \quad i = 1, \ldots, n$$

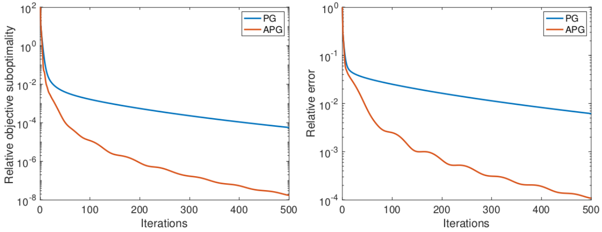


Figure 13.16. *Relative objective suboptimality* $(f(\boldsymbol{x}^{(k)}) - f(\boldsymbol{x}^\star))/|f(\boldsymbol{x}^\star)|$ *(left) and relative error* $\|\boldsymbol{x}^{(k)} - \boldsymbol{x}^\star\|_2/\|\boldsymbol{x}^\star\|_2$ *(right) for the PG and APG methods.*

[4]

Outline

Regularized least-squares problems

$$\text{minimise } \{\tfrac{1}{2}\|b - Ax\|_2^2 + \alpha J(x)\}$$

$$\text{subject to } x \in C$$

$x \in \mathbb{R}^n$, $\alpha > 0$ as the regularization parameter, C as a closed, convex set and a closed, convex regularization function $J(x)$

- bicriterion optimization function:
  - minimise the residual norm
  - minimise $J(x)$

Here our solution depends on α!

The trade-off curve

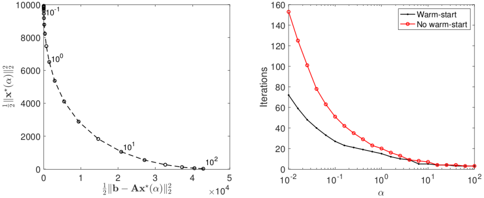


Figure 13.17. *Trade-off curve (left) and the number of iterations using a warm-start approach (right).*

[4]

- we solve the problem for different regularization parameters
- $x^*(\alpha)$ is a solution to the problem with parameter α

The trade-off curve

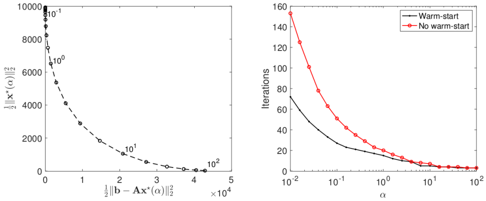


Figure 13.17. *Trade-off curve (left) and the number of iterations using a warm-start approach (right).*

[4]

- we solve the problem for different regularization parameters
- $x^*(\alpha)$ is a solution to the problem with parameter α
- warm start: if α close to α' often $x^*(\alpha)$ is close to $x^*(\alpha')$ too!
  How could we use this to speed things up?

The trade-off curve

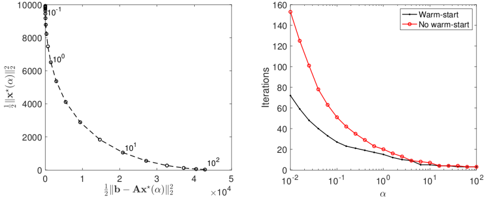


Figure 13.17. *Trade-off curve (left) and the number of iterations using a warm-start approach (right).*

[4]

- we solve the problem for different regularization parameters
- $x^*(\alpha)$ is a solution to the problem with parameter α
- warm start: if α close to α' often $x^*(\alpha)$ is close to $x^*(\alpha')$ too!
  How could we use this to speed things up?
  - use $x^*(\alpha)$ as the initial guess for solving $x^*(\alpha')$

TV regularisation

Goal: consider gradient method for nonsmooth functions

We will look at anisotropic TV of an $M \times N$ image with $x \in \mathbb{R}^{MN}$ as our representation of the image:

$$TV_a(x) = \|Dx\|_1, \qquad D = \begin{pmatrix} I_N \otimes D_M \\ D_N \otimes I_M \end{pmatrix}$$

We want to apply the gradient method to our regularization problem.

TV regularisation

Goal: consider gradient method for nonsmooth functions

We will look at anisotropic TV of an $M \times N$ image with $x \in \mathbb{R}^{MN}$ as our representation of the image:

$$TV_a(x) = \|Dx\|_1, \qquad D = \begin{pmatrix} I_N \otimes D_M \\ D_N \otimes I_M \end{pmatrix}$$

We want to apply the gradient method to our regularization problem.
What could be the problem?

TV regularisation

Goal: consider gradient method for nonsmooth functions

We will look at anisotropic TV of an $M \times N$ image with $x \in \mathbb{R}^{MN}$ as our representation of the image:

$$TV_a(x) = \|Dx\|_1, \qquad D = \begin{pmatrix} I_N \otimes D_M \\ D_N \otimes I_M \end{pmatrix}$$

We want to apply the gradient method to our regularization problem.
What could be the problem?

The function is convex but not differentiable

TV regularisation

Goal: consider gradient method for nonsmooth functions

We will look at anisotropic TV of an $M \times N$ image with $x \in \mathbb{R}^{MN}$ as our representation of the image:

$$TV_a(x) = \|Dx\|_1, \qquad D = \begin{pmatrix} I_N \otimes D_M \\ D_N \otimes I_M \end{pmatrix}$$

We want to apply the gradient method to our regularization problem.
What could be the problem?

The function is convex but not differentiable

Solution:
smoothing technique $\rightarrow$ approximate 1-norm by a continuously differentiable function

$$\|y\|_1 = \sum_{i=1}^{n} |y_i| \approx \sum_{i=1}^{n} \phi_\delta(y_i)$$

Possible approximating functions:

1. Lifting:

$$\phi_\delta(\tau) = \sqrt{\tau^2 + \delta^2} = \left\| \begin{pmatrix} \tau \\ \delta \end{pmatrix} \right\|_2$$

   with $\delta > 0$

2. Huber penalty function scaled by $\frac{1}{\delta}$:

$$\phi_\delta(\tau) = \sup_{|v| \le 1} \left\{ \tau v - \frac{\delta}{2} v^2 \right\} = \begin{cases} \frac{\tau^2}{2\delta}, & |\tau| \le \delta \\ |\tau| - \frac{\delta}{2}, & |\tau| > \delta \end{cases}$$

   - derivative is not differentiable!

3. Based on $|\tau| = \max(-\tau, \tau)$

$$\phi_\delta(\tau) = \delta \log(e^{\tau/\delta} + e^{-\tau/\delta})$$

with $\delta > 0$ controlling the quality of the approximation
Problem: the function can cause numerical overflow
Solution: equivalent approximation

$$\phi_\delta(\tau) = |\tau| + \delta \log(1 + e^{-2|\tau|/\delta})$$

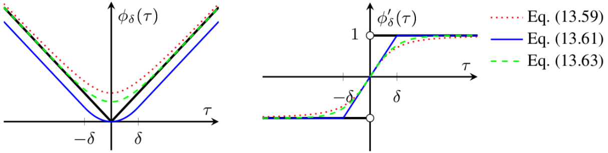


Figure 13.18. *Smooth approximations to the absolute value function.*

[4]

Comparison

(Very similar approaches can be done for the 2-norm of the isotropic TV.)

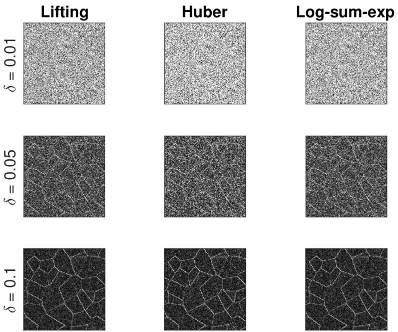


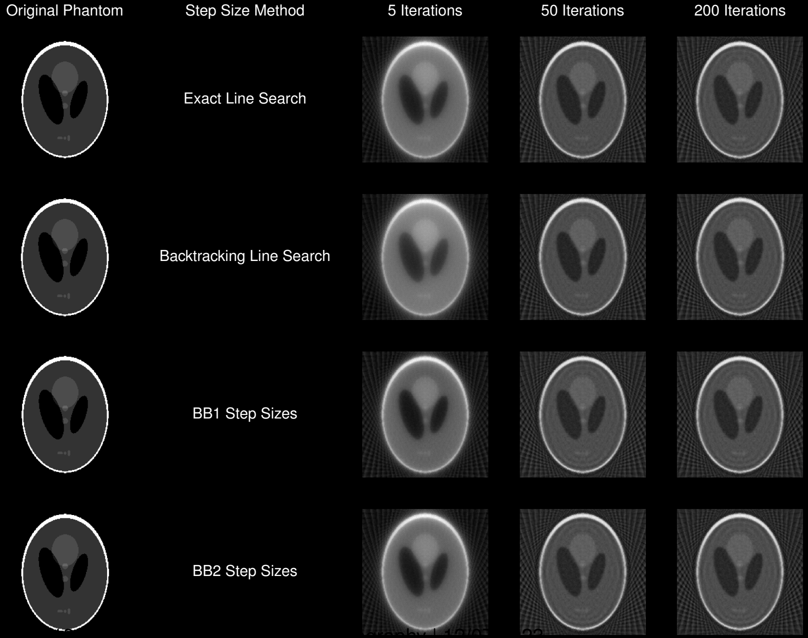
Figure 13.19. *Pixelwise magnitude of* $\nabla \mathrm{TV}_{\mathrm{i}}(\boldsymbol{x})$ *for different values of the smoothing parameter* δ *and where* $\boldsymbol{x}$ *represents the "grains" phantom from AIR Tools* II *with additive noise.*

[4]

Implementation

Gradient method with variable step size

```matlab
function [] = gradientmethod(step_size_method, maxiterations)
%function that uses the gradient method to approximate the least-squares solution of 1/2||Ax-b||²_2
%:param step_size_method: choice of step sizes methods between 'exact line search', 'backtracking line
%search', 'BB1 step sizes' and 'BB2 step sizes'
%:param maxiterations: maximum number of iterations

%set default step size method
if isempty(step_size_method)
   step_size_method = 'exact line search';
end

%test example created using AIR Tools II,
%generation of a 2-D parallel-beam tomography problem
N = 200; theta = 0:5:179; p = 2*N;
[A, b, x] = paralleltomo(N,theta,p);

%define starting point
x_0 = zeros(N*N, 1);

%define tolerance
epsilon = 1e-3;

%define provisory step size
t = 0.05;

%define current number of iterations
iteration = 0;

%define starting vector
currentX = x_0;
x_minus1 = zeros(N*N, 1);

%define objective function
f = @(x) (1/2) * (norm(A*x-b)^2);

grad_f = grad(currentX, A, b);

while and(norm(grad_f)>=epsilon, iteration<maxiterations)
    %define the correct step size depending on parameter 'step_size_method'
    switch step_size_method
        case 'exact line search'
            t = norm(grad(currentX, A, b))^2/norm(A*grad(currentX, A , b))^2;
        case 'backtracking line search'
            t = backtracking(currentX, f, A, b);
        case 'BB1 step sizes'
            if(iteration == 0)
               t = backtracking(currentX, f, A, b);
            else
               t = bb(currentX, x_minus1, A, b, 1);
            end
        case 'BB2 step sizes'
            if(iteration == 0)
               t = backtracking(currentX,f, A ,b);
            else
               t = bb(currentX, x_minus1, A, b, 2);
            end
        otherwise
    end

    %calculate new x
    newX = currentX- t*grad_f;

    if ~isfinite(newX)
        error('x is inf or NaN')
    end

    %update values
    iteration = iteration + 1;
    x_minus1 = currentX;
    currentX = newX;
    grad_f = grad(currentX, A, b);
end

%plot original phantom and result
x = reshape(x,N,N);
subplot(2, 2, 1);
imshow(x, []);
title('Original Phantom');

x_grad = reshape(currentX, [N, N]);
subplot(2, 2, 2);
imshow(x_grad, []);
title(sprintf('Approximated phantom in %d iterations', iteration));

%calculates gradient at x
function g = grad(x, A, b)
g = A'*(A*x-b);

%backtracking line search with alpha = 1e-2 and beta = 0.5
function t_k = backtracking(x, func, A, b)
t_k = 1;
alpha = 1e-2;
beta = 0.5;
while func(x-t_k*grad(x, A, b)) > func(x)-alpha * t_k * norm(grad(x, A ,b))^2
    t_k = t_k * beta;
end

%BB step sizes
%calculates the BB1 or BB2 step size depending on parameter 'version'
function t_k = bb(x, x_minus1, A, b, version)
delta_y = grad(x, A, b) - grad(x_minus1, A , b);
delta_s = x - x_minus1;
if(version == 1)
    alpha_k = (delta_s' * delta_y) / norm(delta_s)^2;
    t_k = 1 / alpha_k;
else
    t_k = (delta_s' * delta_y) / norm(delta_y)^2;
end
```


Original Phantom
Step Size Method
5 Iterations
50 Iterations
200 Iterations
Exact Line Search
Backtracking Line Search
BB1 Step Sizes
BB2 Step Sizes

References

[1] Prof. Dr. Hans-Joachim Bungartz. *Lecture notes in Modeling and Simulation*.

[2] *EPFL lecture notes: Chapter 4 Unconstrained optimization*. Apr. 2022.

[3] “Line Search Methods”. In: *Numerical Optimization*. New York, NY: Springer New York, 2006, pp. 30–65. ISBN: 978-0-387-40065-5. DOI: 10.1007/978-0-387-40065-5_3. URL: https://doi.org/10.1007/978-0-387-40065-5_3.

[4] Jakob Sauer Jørgensen William R. B. Lionheart Per Christian Hansen. “Chapter 13: Optimization Methods for Tomography”. In: *Computed Tomography: Algorithms, Insight, and Just Enough Theory*, pp. 275–315.