

Complexity analysis of second-order algorithms based on line search for smooth nonconvex optimization

Clément Royer - University of Wisconsin-Madison
Joint work with S. Wright

Beyond Convexity: Emerging Challenges in Data Science

Oaxaca, Mexico - October 26, 2017

We consider an unconstrained smooth problem:

$$\min_{x \in \mathbb{R}^n} f(x).$$

Assumptions on f

- f bounded from below.
- f twice continuously differentiable.
- f is not convex.

Second-order necessary point

x^* satisfies the **second-order necessary conditions** if

$$\nabla f(x^*) = 0, \quad \nabla^2 f(x^*) \succeq 0.$$

Basic paradigm

If x is not a second-order necessary point, $\exists d$ such that

- 1 $d^\top \nabla f(x) < 0$: **gradient-related direction**.
and/or
- 2 $d^\top \nabla^2 f(x) d < 0$: **negative curvature direction**
 \Rightarrow **specific to nonconvex problems**,

and f **decreases** along d .

Example: Nonconvex formulations of low-rank matrix problems

$$\min_{U \in \mathbb{R}^{n \times r}, V \in \mathbb{R}^{m \times r}} f(UV^T).$$

For common classes of problems:

- Second-order necessary points are **global minimizers** (or close).
- Saddle points have **negative curvature** (strict saddle property).

Other example: Fourth-order tensor orthogonal factorization.

$$\min_{\{u_i\} \subset \mathbb{S}^{n-1}} \sum_{i \neq j} T(u_i, u_i, u_j, u_j).$$

- Solutions **are** second-order necessary points (Ge et al, '15).

Example: Nonconvex formulations of low-rank matrix problems

$$\min_{U \in \mathbb{R}^{n \times r}, V \in \mathbb{R}^{m \times r}} f(UV^T).$$

For common classes of problems:

- Second-order necessary points are **global minimizers** (or close).
- Saddle points have **negative curvature** (strict saddle property).

Other example: Fourth-order tensor orthogonal factorization.

$$\min_{\{u_i\} \subset \mathbb{S}^{n-1}} \sum_{i \neq j} T(u_i, u_i, u_j, u_j).$$

- Solutions **are** second-order necessary points (Ge et al, '15).
- Interest: Second-order necessary points of **nonconvex problems**.
- Need: **Efficient algorithms**.

Principle

For a given method, two tolerances $\epsilon_g, \epsilon_H \in (0, 1)$:

- **Obj:** bound the **worst-case cost** of reaching x_k such that

$$\|\nabla f(x_k)\| \leq \epsilon_g, \quad \lambda_k = \lambda_{\min}(\nabla^2 f(x_k)) \geq -\epsilon_H.$$

- **Focus:** Bound dependencies on ϵ_g, ϵ_H .

Principle

For a given method, two tolerances $\epsilon_g, \epsilon_H \in (0, 1)$:

- **Obj:** bound the **worst-case cost** of reaching x_k such that

$$\|\nabla f(x_k)\| \leq \epsilon_g, \quad \lambda_k = \lambda_{\min}(\nabla^2 f(x_k)) \geq -\epsilon_H.$$

- **Focus:** Bound dependencies on ϵ_g, ϵ_H .

- Definition of **cost** ?
- **Best** rates ?

Nonlinear optimization literature

- Classical cost: Number of (expensive) iterations.
- Best methods: Newton-type frameworks.
1 iteration \Leftrightarrow 1 subproblem/Newton system solve.

Nonlinear optimization literature

- Classical cost: Number of (expensive) iterations.
- Best methods: Newton-type frameworks.
1 iteration \Leftrightarrow 1 subproblem/Newton system solve.

Some algorithms	Bounds
Classical trust region <i>Cartis, Gould, Toint '10</i>	$\mathcal{O}(\max\{\epsilon_g^{-2}\epsilon_H^{-1}, \epsilon_H^{-3}\})$
Cubic regularization <i>Nesterov, Polyak '06</i> <i>Cartis, Gould, Toint '10</i>	$\mathcal{O}(\max\{\epsilon_g^{-\frac{3}{2}}, \epsilon_H^{-3}\})$
TRACE trust region <i>Curtis, Robinson, Samadi '14</i>	

Existing complexity results (2)

Learning/Statistics community

- Specific setting $\epsilon_g = \epsilon, \epsilon_H = \mathcal{O}(\sqrt{\epsilon})$.
- Gradient-based \Rightarrow cheaper iterations.
- Cost measure: *Hessian-vector products/gradient evaluations*.

Existing complexity results (2)

Learning/Statistics community

- Specific setting $\epsilon_g = \epsilon, \epsilon_H = \mathcal{O}(\sqrt{\epsilon})$.
- Gradient-based \Rightarrow cheaper iterations.
- Cost measure: *Hessian-vector products/gradient evaluations*.

Some algorithms	Bounds
Gradient descent methods with random noise <i>Jin et al '17</i>	$\tilde{\mathcal{O}}(\epsilon^{-2})$
Accelerated gradient methods for nonconvex problems <i>Carmon et al '16, Allen-Zhu et al '16</i>	$\tilde{\mathcal{O}}(\epsilon^{-\frac{7}{4}})$

- $\tilde{\mathcal{O}}(\cdot)$: logarithmic factors.
- Results hold with **high probability**.

Illustrate all the possible complexities...

- In terms of iterations, evaluations, etc.
- For arbitrary ϵ_g, ϵ_H .
- Deterministic and high probability results.

Illustrate all the possible complexities...

- In terms of iterations, evaluations, etc.
- For arbitrary ϵ_g , ϵ_H .
- Deterministic and high probability results.

...in a single framework

- Based on [line search](#).
- Only require Hessian-vector products/gradient calls.
- [Best complexity guarantees](#).

- 1 Introduction
- 2 A second-order algorithm
- 3 Inexact variants
- 4 Discussion

- 1 Introduction
- 2 A second-order algorithm
- 3 Inexact variants
- 4 Discussion

Parameters: $x_0 \in \mathbb{R}^n$, $\theta \in (0, 1)$, $\eta > 0$, $\epsilon_g \in (0, 1)$, $\epsilon_H \in (0, 1)$.

For $k=0, 1, 2, \dots$

- 1 Compute a search direction d_k .
- 2 Perform a backtracking line search to compute $\alpha_k = \theta^{j_k}$ such that

$$f(x_k + \alpha_k d_k) < f(x_k) - \frac{\eta}{6} \alpha_k^3 \|d_k\|^3.$$

- 3 Set $x_{k+1} = x_k + \alpha_k d_k$.

Selecting the search direction d_k

Step 1: Use gradient related information

- Compute

$$g_k = \nabla f(x_k), \quad R_k = \frac{g_k^\top \nabla^2 f(x_k) g_k}{\|g_k\|^2}.$$

- If $R_k < -\epsilon_H$, set

$$d_k = \frac{R_k}{\|g_k\|} g_k.$$

- Elseif $R_k \in [-\epsilon_H, \epsilon_H]$ and $\|g_k\| > \epsilon_g$, set

$$d_k = -\frac{g_k}{\|g_k\|^{1/2}}.$$

- Otherwise perform [Step 2](#).

Selecting the search direction d_k (2)

Step 2: Use eigenvalue information

- Compute an eigenpair (v_k, λ_k) such that $\lambda_k = \lambda_{\min}(\nabla^2 f(x_k))$ and

$$\nabla^2 f(x_k) v_k = \lambda_k v_k, \quad v_k^\top g_k \leq 0, \quad \|v_k\| = 1.$$

- **Case $\lambda_k < -\epsilon_H$:** $d_k = -\lambda_k v_k$;
- **Case $\lambda_k > \epsilon_H$ - Newton step:**

$$d_k = d_k^n, \quad \nabla^2 f(x_k) d_k^n = -g_k;$$

- **Case $\lambda_k \in [-\epsilon_H, \epsilon_H]$ - regularized Newton step:**

$$d_k = d_k^r, \quad (\nabla^2 f(x_k) + 2\epsilon_H) d_k^r = -g_k.$$

- Five possible directions.
 - Two ways of scaling $-g_k$:
 - By its (negative) curvature;
 - By its norm;
 - Negative eigenvector;
 - Newton step;
 - Regularized Newton step.

- Five possible directions.
 - Two ways of scaling $-g_k$:
 - By its (negative) curvature;
 - By its norm;
 - Negative eigenvector;
 - Newton step;
 - Regularized Newton step.
- **One line-search technique with cubic decrease.**

- Five possible directions.
 - Two ways of scaling $-g_k$:
 - By its (negative) curvature;
 - By its norm;
 - Negative eigenvector;
 - Newton step;
 - Regularized Newton step.
- **One line-search technique with cubic decrease.**

What is the complexity of this method ?

Approximate solution

x_k is an (ϵ_g, ϵ_H) -**point** if

$$\min \{ \|\nabla f(x_k)\|, \|\nabla f(x_{k+1})\| \} \leq \epsilon_g, \quad \lambda_{\min}(\nabla^2 f(x_k)) \geq -\epsilon_H.$$

Approximate solution

x_k is an (ϵ_g, ϵ_H) -point if

$$\min \{ \|\nabla f(x_k)\|, \|\nabla f(x_{k+1})\| \} \leq \epsilon_g, \quad \lambda_{\min}(\nabla^2 f(x_k)) \geq -\epsilon_H.$$

On $\min \{ \|\nabla f(x_k)\|, \|\nabla f(x_{k+1})\| \}$

- Possible to add a stopping criterion to guarantee termination at a point at which $\|\nabla f(x_k)\| \leq \epsilon_g$.
- $\|g_{k+1}\|$ appears because of Newton-type steps.

Assumptions

- $\mathcal{L}_f(x_0) = \{x | f(x) \leq f(x_0)\}$ compact.
- f twice continuously differentiable on a open set containing $\mathcal{L}_f(x_0)$, with Lipschitz continuous Hessian.
- L_H : Lipschitz constant for $\nabla^2 f$.
- f_{low} : lower bound on $\{f(x_k)\}$.
- U_H : upper bound on $\|\nabla^2 f(x_k)\|$.

Key principle

Bound the **decrease** produced at every step while an (ϵ_g, ϵ_H) -point has not been reached.

- Same proof for every direction;
- If unit step is accepted by line search, guaranteed decrease;
- Otherwise, lower bound on accepted step size.

Key principle

Bound the **decrease** produced at every step while an (ϵ_g, ϵ_H) -point has not been reached.

- Same proof for every direction;
- If unit step is accepted by line search, guaranteed decrease;
- Otherwise, lower bound on accepted step size.

General decrease lemma

If at the k -th iteration, an (ϵ_g, ϵ_H) -point has not been reached, then

$$f(x_k) - f(x_{k+1}) \geq c \min \left\{ \frac{3}{\epsilon_g^2}, \epsilon_H^3, \epsilon_g^3 \epsilon_H^{-3} \right\},$$

where c depends on L_H, η, θ .

Iteration complexity bound

The method reaches an (ϵ_g, ϵ_H) -point in at most

$$C \max \left\{ \epsilon_g^{-\frac{3}{2}}, \epsilon_H^{-3}, \epsilon_g^{-3} \epsilon_H^3 \right\}$$

iterations, with $C = \mathcal{O}((f_0 - f_{\text{low}})L_H^3)$.

Specific rates:

- $\epsilon_g = \epsilon$, $\epsilon_H = \sqrt{\epsilon}$: $\mathcal{O}(\epsilon^{-\frac{3}{2}})$.
- $\epsilon_g = \epsilon_H = \epsilon$: $\mathcal{O}(\epsilon^{-3})$.
- Optimal bounds for second-order Newton-type methods (Cartis, Gould and Toint '17).

- 1 Introduction
- 2 A second-order algorithm
- 3 Inexact variants**
- 4 Discussion

- The method needs **Hessian information**:
 - Linear system solve;
 - Eigenvalue/Eigenvector computation.
- Also gives **optimal iteration complexity**.

Introducing inexactness

- The method needs **Hessian information**:
 - Linear system solve;
 - Eigenvalue/Eigenvector computation.
- Also gives **optimal iteration complexity**.

Inexactness

- Perform the matrix operations **inexactly**.
- Main cost unit: **matrix-vector product**/gradient evaluation.

Example of inexact subroutines

- Linear systems: Accelerated gradient, **CG**,...
- Eigenvalues: Approximate PCA, **Lanczos**,...

Conjugate gradient for linear systems

- We solve systems of the form $Hd = -g$, with $\epsilon_H I \preceq H \preceq UI$.

Conjugate gradient for linear systems

- We solve systems of the form $Hd = -g$, with $\epsilon_H I \preceq H \preceq UI$.

Conjugate Gradient (CG)

- We apply the conjugate gradient algorithm with stopping criterion:

$$\|Hd + g\| \leq \frac{\xi}{2} \min \{\|g\|, \epsilon_H \|d\|\}, \quad \xi \in (0, 1).$$

- If $\kappa = \frac{U}{\epsilon_H}$, the CG method will find such a vector in at most

$$\min \left\{ n, \frac{1}{2} \sqrt{\kappa} \log \left(4\kappa^{\frac{5}{2}} / \xi \right) \right\} = \min \{ n, \mathcal{O}(\sqrt{\kappa} \log(\kappa/\xi)) \}$$

iterations/**matrix-vector products**.

Lanczos for eigenvalue computation

- Lanczos method to compute a minimum eigenvector.
- Can fail if deterministic \Rightarrow **Random start**.

Lanczos for eigenvalue computation

- Lanczos method to compute a minimum eigenvector.
- Can fail if deterministic \Rightarrow **Random start**.

Lanczos iterations

Let $H \in \mathbb{R}^{n \times n}$ symmetric with $\|H\| \leq U_H$, $\epsilon > 0$, $\delta \in (0, 1)$.

With probability at least $1 - \delta$, the Lanczos procedure applied to $U_H I - H$ outputs a vector v such that

$$v^T H v \leq \lambda_{\min}(H) + \epsilon.$$

in at most

$$\min \left\{ n, \frac{\ln(n/\delta^2)}{2\sqrt{2}} \sqrt{\frac{U_H}{\epsilon}} \right\}$$

iterations/matrix-vector products.

Framework (inexact ?)

Parameters: $x_0 \in \mathbb{R}^n$, $\theta \in (0, 1)$, $\eta > 0$, $\epsilon_g \in (0, 1)$, $\epsilon_H \in (0, 1)$.

For $k=0, 1, 2, \dots$

- 1 Compute a search direction d_k .
- 2 Perform a backtracking line search to compute $\alpha_k = \theta^{j_k}$ such that

$$f(x_k + \alpha_k d_k) < f(x_k) - \frac{\eta}{6} \alpha_k^3 \|d_k\|^3.$$

- 3 Set $x_{k+1} = x_k + \alpha_k d_k$.

Step 1: Use gradient related information

- Compute

$$g_k = \nabla f(x_k), \quad R_k = \frac{g_k^\top \nabla^2 f(x_k) g_k}{\|g_k\|^2}.$$

- If $R_k < -\epsilon_H$, set

$$d_k = \frac{R_k}{\|g_k\|} g_k.$$

- Elseif $R_k \in [-\epsilon_H, \epsilon_H]$ and $\|g_k\| \geq \epsilon_g$, set

$$d_k = -\frac{g_k}{\|g_k\|^{\frac{1}{2}}}.$$

- Otherwise perform the **Inexact** Step 2.

Selecting the direction d_k - Inexact version (2)

Inexact Step 2: Use (inexact) eigenvalue information

- Using Lanczos, compute a pair (v_k^i, λ_k^i) such that **w. p. $1 - \delta$** ,

$$\lambda_k^i = [v_k^i]^\top \nabla^2 f(x_k) v_k^i \leq \lambda_k + \frac{\epsilon_H}{2}, \quad [v_k^i]^\top g_k \leq 0, \quad \|v_k^i\| = 1.$$

- Case $\lambda_k^i < -\frac{1}{2}\epsilon_H$:** $d_k = -\lambda_k^i v_k^i$;
- Case $\lambda_k^i > \frac{3}{2}\epsilon_H$:**
 - **Inexact Newton:** Use CG to obtain

$$d_k = d_k^{in}, \quad \|\nabla^2 f(x_k) d_k^{in} + g_k\| \leq \frac{\xi}{2} \min \{\|g_k\|, \epsilon_H \|d_k^{in}\|\};$$

- Case $\lambda_k^i \in [-\frac{1}{2}\epsilon_H, \frac{3}{2}\epsilon_H]$:**
 - **Inexact regularized Newton:** Use CG to obtain

$$d_k = d_k^{ir}, \quad \|\nabla^2 f(x_k) + 2\epsilon_H\| d_k^{ir} + g_k\| \leq \frac{\xi}{2} \min \{\|g_k\|, \epsilon_H \|d_k^{ir}\|\}.$$

Complexity analysis of the inexact method

- **Identical reasoning:** 5 steps, 1 proof.
- **Using Lanczos with random start**, the decrease only holds in probability.
- **With CG**, the inexact Newton and regularized Newton give slightly different formulas.

Decrease lemma

For any iteration k , if x_k is not an (ϵ_g, ϵ_H) -point,

$$f(x_k) - f(x_{k+1}) \geq \hat{c} \min \left\{ \epsilon_g^{\frac{3}{2}}, \epsilon_H^3, \epsilon_g^3 \epsilon_H^{-3} \right\},$$

with probability at least $1 - \delta$, and \hat{c} only depends on L_H, η, θ .

Iteration complexity

An (ϵ_g, ϵ_H) -point is reached in at most

$$\hat{K} := \frac{f_0 - f_{\text{low}}}{\hat{c}} \max \left\{ \epsilon_g^{-\frac{3}{2}}, \epsilon_H^{-3}, \epsilon_g^{-3} \epsilon_H^3 \right\},$$

iterations, **with probability at least $1 - \hat{K}\delta$** .

Cost complexity

The number of **Hessian-vector products or gradient evaluations** needed to reach an (ϵ_g, ϵ_H) -point is at most

$$\min \left\{ n, \mathcal{O} \left(U_H^{1/2} \epsilon_H^{-\frac{1}{2}} \log(\epsilon_H^{-1}/\xi) \right), \mathcal{O} \left(U_H^{1/2} \epsilon_H^{-\frac{1}{2}} \log(n/\delta^2) \right) \right\} \hat{K},$$

w. p. at least $1 - \hat{K}\delta$.

Simplified setting: $\epsilon_g = \epsilon, \epsilon_H = \sqrt{\epsilon}$.

An $(\epsilon, \sqrt{\epsilon})$ -point is reached in at most

- $\mathcal{O}(\epsilon^{-3/2})$ iterations,
- $\tilde{\mathcal{O}}(\epsilon^{-7/4})$ Hessian-vector products/gradient evaluations,

w. p. at least $1 - \mathcal{O}(\epsilon^{-3/2}\delta)$.

⇒ Matches best known results for accelerated gradient-based methods.

Simplified setting: $\epsilon_g = \epsilon, \epsilon_H = \sqrt{\epsilon}$.

An $(\epsilon, \sqrt{\epsilon})$ -point is reached in at most

- $\mathcal{O}(\epsilon^{-3/2})$ iterations,
- $\tilde{\mathcal{O}}(\epsilon^{-7/4})$ Hessian-vector products/gradient evaluations,

w. p. at least $1 - \mathcal{O}(\epsilon^{-3/2}\delta)$.

⇒ Matches best known results for accelerated gradient-based methods.

Setting $\delta = 0$ gives *in probability 1*:

- Iterations: $\mathcal{O}(\epsilon^{-3/2})$.
- Hessian-vector/gradients: $\mathcal{O}(\epsilon^{-7/4})$.

Our algorithm

- Has optimal/best known complexity guarantees.
- Can be implemented without Lipschitz constants (but uses line search).
- **Explicitly uses second-order information.**

Complexity analysis of second-order line-search algorithms for smooth nonconvex optimization, C. W. Royer and S. J. Wright, arXiv:1706.03131.

Our algorithm

- Has optimal/best known complexity guarantees.
- Can be implemented without Lipschitz constants (but uses line search).
- **Explicitly uses second-order information.**

Complexity analysis of second-order line-search algorithms for smooth nonconvex optimization, C. W. Royer and S. J. Wright, arXiv:1706.03131.

Two questions

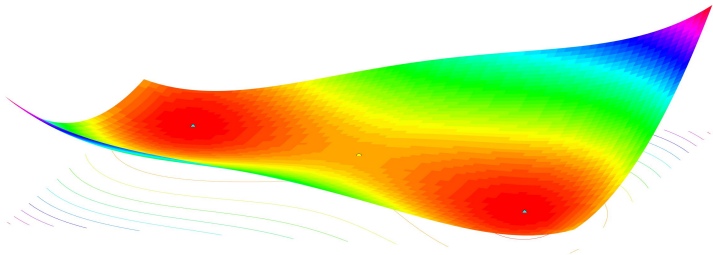
On problems such as those from data science...

- 1 Why second-order methods rather than gradient descent ?
- 2 Are second-order methods the best option to tackle nonconvexity ?

My questions

Why second-order methods rather than gradient descent ?

Complexity bounds are good, do not tell the whole story.

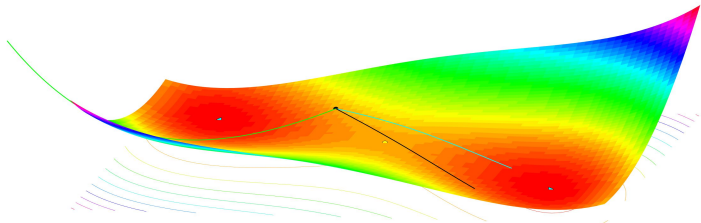


Why second-order methods rather than gradient descent ?

Complexity bounds are good, do not tell the whole story.

- **Newton steps**

Is there something hidden in $\min\{\|g_k\|, \|g_{k+1}\|\}$?



Why second-order methods rather than gradient descent ?

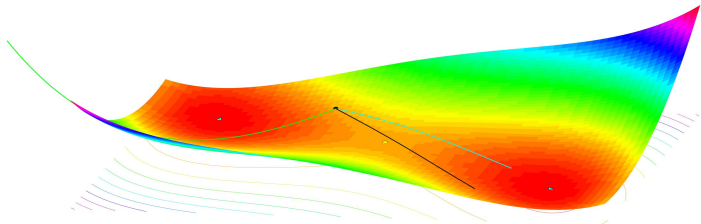
Complexity bounds are good, do not tell the whole story.

- **Newton steps**

Is there something hidden in $\min\{\|g_k\|, \|g_{k+1}\|\}$?

- **Negative curvature directions**

*Exploit **alternative paths** towards the minimum ?*



Why second-order methods rather than gradient descent ?

Complexity bounds are good, do not tell the whole story.

- **Newton steps**

Is there something hidden in $\min\{\|g_k\|, \|g_{k+1}\|\}$?

- **Negative curvature directions**

*Exploit **alternative paths** towards the minimum ?*

Why second order for nonconvex problems ?

Second order may not be enough for finding flat/good/global minima.

- **Beyond strict saddle point property**

*How does a second-order method behave with **singular Hessians** ?*

Why second-order methods rather than gradient descent ?

Complexity bounds are good, do not tell the whole story.

- **Newton steps**

Is there something hidden in $\min\{\|g_k\|, \|g_{k+1}\|\}$?

- **Negative curvature directions**

*Exploit **alternative paths** towards the minimum ?*

Why second order for nonconvex problems ?

Second order may not be enough for finding flat/good/global minima.

- **Beyond strict saddle point property**

*How does a second-order method behave with **singular Hessians** ?*

- **Beyond second order**

*Use **high-order** derivatives to deal with flat regions ?*