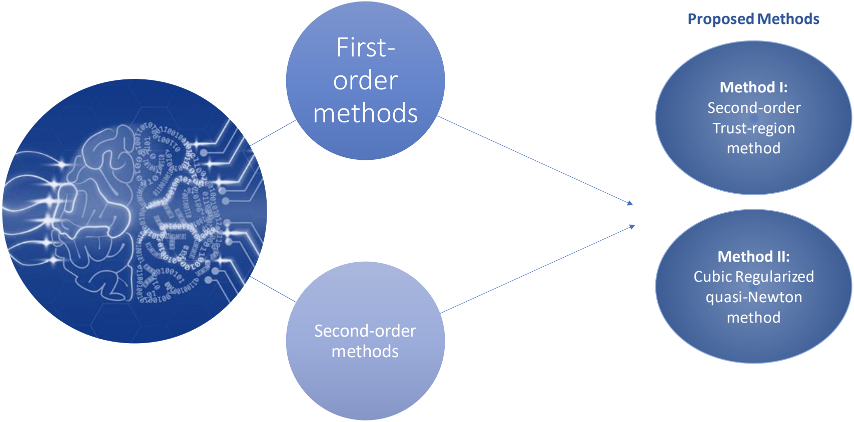
Optimization for machine learning: Tractible solutions to large-scale non-convex systems

Aditya Ranganath

Machine learning Postdoctoral Staff,
Center for Applied Scientific Computing (CASC),
Lawrence Livermore National Laboratory,
7000 East Ave., Livermore, CA - 94550

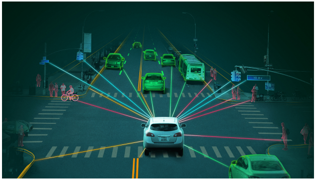
Outline

Deep Learning as Optimization Problems

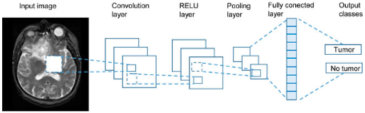


Deep learning and Machine learning

Chatbots



Autonomous driving



Brain tumor classification

AI music generation

- Deep learning and machine learning become ubiquitous in applications.
- Optimization plays a vital role in deep learning.

Optimization for non-convex functions

Problem formulation

Problem: We define the problem as

$$\min_{\Theta} f(\Theta)$$

where $\Theta \in \mathbb{R}^n$ and $f : \mathbb{R}^n \to \mathbb{R}$ is a non-linear, non-convex smooth function.

Optimization for non-convex functions

Problem formulation

Problem: We define the problem as

$$\min_{\Theta} f(\Theta)$$

where $\Theta \in \mathbb{R}^n$ and $f : \mathbb{R}^n \to \mathbb{R}$ is a non-linear, non-convex smooth function.

Goal: Find the minima to the problem.

Mathematical definition

The deep learning problem can be redefined as

$$\underset{\Theta \in \mathbb{R}^d}{\text{minimize}}\ f(\Theta) \equiv \frac{1}{n}\sum_{j=1}^{n} f_j(\Theta_j),$$

where

Mathematical definition

The deep learning problem can be redefined as

$$\underset{\Theta \in \mathbb{R}^d}{\text{minimize}} \; f(\Theta) \equiv \frac{1}{n} \sum_{j=1}^{n} f_j(\Theta_j),$$

where

- $\Theta \in \mathbb{R}^d$ is a vector of size $\approx 4 \times 10^5$,
- f_j depends on the j^{th} observation in $\{\mathbf{x}_j, \mathbf{y}_j\}_{j=1}^n$.

Optimization for non-convex functions

The Rosenbrock function:

$$\underset{\Theta_1,\Theta_2\in\mathbb{R}}{\text{minimize}}\ f(\Theta_1,\Theta_2) \equiv (1-\Theta_1)^2 + (\Theta_2-\Theta_1^2)^2,$$

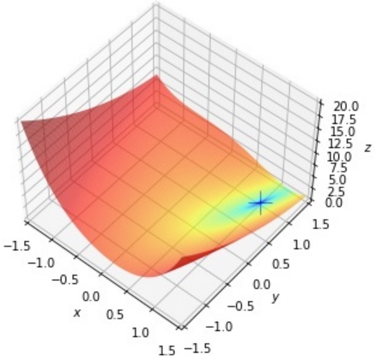
Minima: $\Theta_1^* = 1, \Theta_2^* = 1$.

Optimization for non-convex functions

The Rosenbrock function:

$$\underset{\Theta_1,\Theta_2\in\mathbb{R}}{\text{minimize}}\ f(\Theta_1,\Theta_2) \equiv (1-\Theta_1)^2 + (\Theta_2-\Theta_1^2)^2,$$

Minima: $\Theta_1^* = 1, \Theta_2^* = 1.$

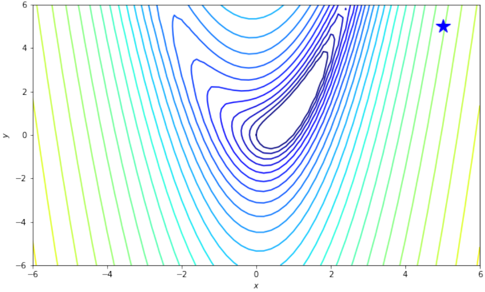


First-order optimization methods

Gradient-based (steepest) descent method:

$$\Theta_{k+1} = \Theta_k - \eta \nabla_\Theta f\left(\Theta_k\right),$$

where $\nabla_\Theta f(\Theta_k) \in \mathbb{R}^n$ is the gradient and $\eta \in \mathbb{R}$ is the step length.

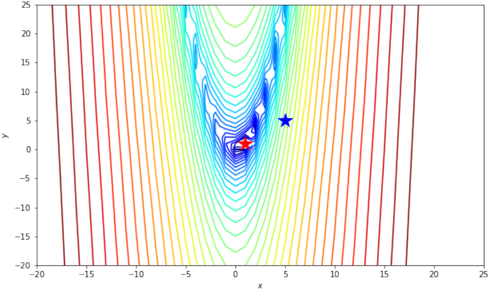


Second-order optimization methods

Newton's method:

$$\Theta_{k+1} = \Theta_k - [H(\Theta_k)]^{-1} \nabla_{\Theta} f(\Theta_k),$$

where $H(\Theta_k) \in \mathbb{R}^{n \times n}$ is the Hessian.



Observation and motivation

Gradient based methods

Benefits:

- Calculate gradient $\approx \mathcal{O}(n)$
- Storing gradient $\approx \mathcal{O}(n)$

Drawbacks:

- No curvature
- Saddle points
- Linear convergence

Newton's method

Benefits:

- Avoids saddle points
- Quadratic convergence

Drawbacks:

- Hessian storage $\approx \mathcal{O}(n^2)$
- Hessian invert $\approx \mathcal{O}(n^3)$
- Non-invertible Hessian

Observation and motivation

Gradient based methods

Benefits:

- Calculate gradient $\approx \mathcal{O}(n)$
- Storing gradient $\approx \mathcal{O}(n)$

Drawbacks:

- No curvature
- Saddle points
- Linear convergence

Newton's method

Benefits:

- Avoids saddle points
- Quadratic convergence

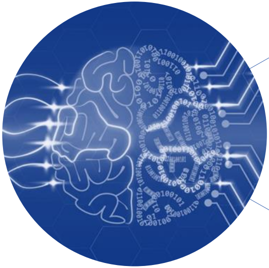
Drawbacks:

- Hessian storage $\approx \mathcal{O}(n^2)$
- Hessian invert $\approx \mathcal{O}(n^3)$
- Non-invertible Hessian

Motivation: Find a suitable compromise between first- and second-order methods.

Outline

Deep Learning as Optimization Problems



First-order methods

- Cheap to store
- Fast to compute

Second-order methods

- Quadratic convergence
- Avoids saddle points

Proposed Methods

Method I: Second-order Trust-region method

Method II: Cubic Regularized quasi-Newton method

I. Trust-region methods

Trust region subproblem:

$$\underset{\mathbf{p}\in\mathbb{R}^n}{\text{minimize}}\ \mathcal{Q}_k(\mathbf{p}) \equiv f(\Theta_k) + \mathbf{g}_k^\top \mathbf{p} + \frac{1}{2}\mathbf{p}^\top \mathbf{B}_k \mathbf{p}$$

$$\text{subject to} \|\mathbf{p}\|_2 \leq \Delta_k,$$

I. Trust-region methods

Trust region subproblem:

$$\begin{aligned}
&\underset{\mathbf{p}\in\mathbb{R}^n}{\text{minimize}}\ \mathcal{Q}_k(\mathbf{p}) \equiv f(\Theta_k) + \mathbf{g}_k^\top \mathbf{p} + \frac{1}{2}\mathbf{p}^\top \mathbf{B}_k \mathbf{p} \\
&\text{subject to} \|\mathbf{p}\|_2 \leq \Delta_k,
\end{aligned}$$

where

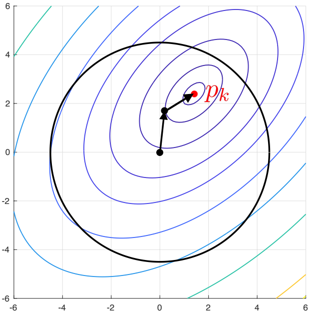
- $\mathbf{p} \in \mathbb{R}^n$ is the step,
- $\Delta_k \in \mathbb{R}^+$ is the trust-region radius,
- $\mathbf{g}_k = \nabla f(\Theta_k)$ is the gradient at Θ_k,
- $\mathbf{B}_k$ is the Hessian or approximation.

Trust region subproblem:

$$\underset{\mathbf{p}\in\mathbb{R}^n}{\text{minimize}}\ \mathcal{Q}_k(\mathbf{p}) \equiv f(\Theta_k) + \mathbf{g}_k^\top \mathbf{p} + \frac{1}{2}\mathbf{p}^\top \mathbf{B}_k \mathbf{p}$$
$$\text{subject to} \|\mathbf{p}\|_2 \leq \Delta_k,$$

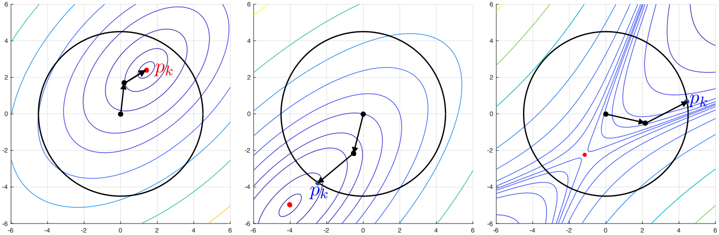
where

- $\mathbf{p} \in \mathbb{R}^n$ is the step,
- $\Delta_k \in \mathbb{R}^+$ is the trust-region radius,
- $\mathbf{g}_k = \nabla f(\Theta_k)$ is the gradient at Θ_k,
- $\mathbf{B}_k$ is the Hessian or approximation.



I. Solving the subproblem

Conjugate-Gradient (CG) method



- Convex $\mathcal{Q}$: p_k arrives at unconstrained minimizer.
- Convex $\mathcal{Q}$: p_k is defined where CG crosses the boundary.
- Non-convex $\mathcal{Q}$: terminates on the boundary along last CG iterate p_k.

Challenge: Computing the matrix-vector product in CG.
Proposed approach: Pearlmutter's technique

I. Hessian-vector products

Martens et al.[1]:

$$\mathbf{H}_k\mathbf{d} = \lim_{\epsilon\to 0} \frac{1}{\epsilon}\Big(\nabla f(\Theta + \epsilon\mathbf{d}) - \nabla f(\Theta)\Big).$$

Proposed approach: Pearlmutter's technique[2]:

$$\mathbf{H}_k\mathbf{d} = \lim_{r\to 0} \frac{\nabla f(\Theta + r\mathbf{d}) - \nabla f(\Theta)}{r} = \frac{\partial}{\partial r}\nabla f(\Theta + r\mathbf{d})\Big|_{r=0}.$$

Advantages:

- Uses true Hessian information,
- Cheap to store,
- Cheap to compute.

[1] J. Martens et al. "Deep learning via hessian-free optimization.". In: *ICML*. vol. 27. 2010, pp. 735–742.

[2] B. A Pearlmutter. "Fast exact multiplication by the Hessian". In: *Neural computation* 6.1 (1994), pp. 147–160.

I. Summary

1. Trust-region methods define a sequence of quadratric subproblems with a trust-region constraint.

I. Summary

1. Trust-region methods define a sequence of quadratric subproblems with a trust-region constraint.
2. Solve the trust-region subproblem using conjugate-gradient methods.

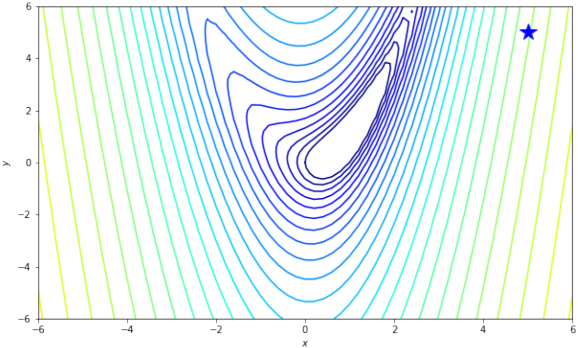
I. Summary

1. Trust-region methods define a sequence of quadratric subproblems with a trust-region constraint.
2. Solve the trust-region subproblem using conjugate-gradient methods.
3. Use Pearlmutter's fast-exact Hessian products.

I. Summary

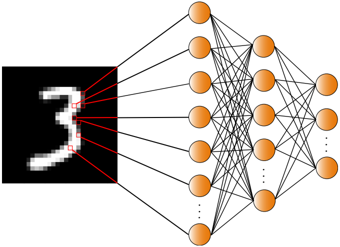
1. Trust-region methods define a sequence of quadratric subproblems with a trust-region constraint.
2. Solve the trust-region subproblem using conjugate-gradient methods.
3. Use Pearlmutter's fast-exact Hessian products.

I. Performance on Rosenbrock function



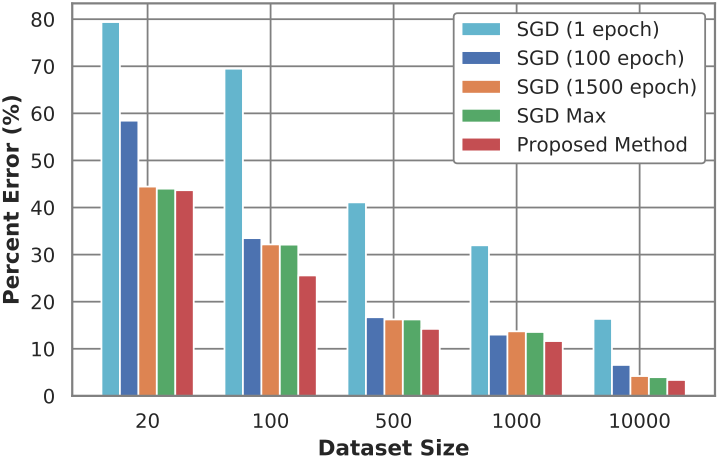
Evolution of iterates for the Rosenbrock function.

Deep learning architecture (Multi-Layer Perceptron):

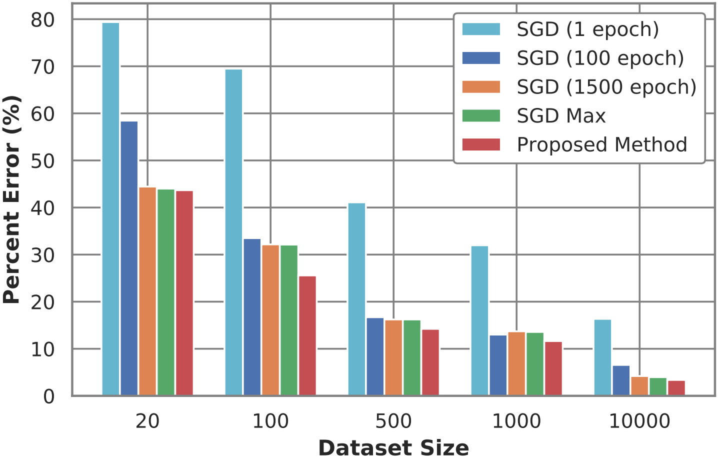


True label $\mathbf{y}$:	0	0	0	1	0	0	0	0	0	0
Classes:	0	1	2	3	4	5	6	7	8	9
Prediction $\hat{\mathbf{y}}$:	.01	.01	.01	.72	.01	.01	.01	.01	0.2	.01

I. Results and observations



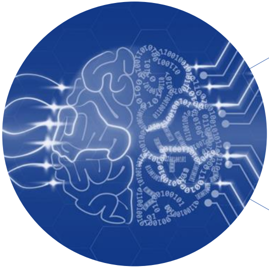
I. Results and observations



Observation: The method can be time-demanding due to the number of computations.

Outline

Deep Learning as Optimization Problems



First-order methods

- Cheap to store
- Fast to compute

Second-order methods

- Quadratic convergence
- Avoids saddle points

Proposed Methods

Method I: Second-order Trust-region method

Method II: Cubic Regularized quasi-Newton method

II. Adaptive regularization using cubics

Trust-region subproblem:

$$\begin{aligned} \underset{\mathbf{p}\in\mathbb{R}^n}{\text{minimize}} \quad & \mathcal{Q}_k(\mathbf{p}) \equiv f(\Theta_k) + \mathbf{g}_k^\top \mathbf{p} + \frac{1}{2}\mathbf{p}^\top \mathbf{B}_k \mathbf{p} \\ \text{subject to } & \|\mathbf{p}\|_2 \leq \Delta_k. \end{aligned}$$

II. Adaptive regularization using cubics

Trust-region subproblem:

$$\begin{aligned} &\underset{\mathbf{p}\in\mathbb{R}^n}{\text{minimize}} \quad \mathcal{Q}_k(\mathbf{p}) \equiv f(\Theta_k) + \mathbf{g}_k^\top \mathbf{p} + \frac{1}{2}\mathbf{p}^\top \mathbf{B}_k \mathbf{p} \\ &\text{subject to } \|\mathbf{p}\|_2 \leq \Delta_k. \end{aligned}$$

Adaptive Regularized Cubics (ARCs) subproblem

$$\underset{\mathbf{s}\in\mathbb{R}^n}{\text{minimize}} \; \mathcal{M}_k(\mathbf{s}) \equiv f(\Theta_k) + \mathbf{g}_k^\top \mathbf{s} + \frac{1}{2}\mathbf{s}^\top \mathbf{B}_k \mathbf{s} + \frac{\sigma_k}{3}\phi(\mathbf{s})^3.$$

II. Adaptive regularization using cubics

Trust-region subproblem:

$$\begin{aligned} &\underset{\mathbf{p}\in\mathbb{R}^n}{\text{minimize}} \quad \mathcal{Q}_k(\mathbf{p}) \equiv f(\Theta_k) + \mathbf{g}_k^\top \mathbf{p} + \frac{1}{2}\mathbf{p}^\top \mathbf{B}_k \mathbf{p} \\ &\text{subject to } \|\mathbf{p}\|_2 \leq \Delta_k. \end{aligned}$$

Adaptive Regularized Cubics (ARCs) subproblem

$$\underset{\mathbf{s}\in\mathbb{R}^n}{\text{minimize}} \ \mathcal{M}_k(\mathbf{s}) \equiv f(\Theta_k) + \mathbf{g}_k^\top \mathbf{s} + \frac{1}{2}\mathbf{s}^\top \mathbf{B}_k \mathbf{s} + \frac{\sigma_k}{3}\phi(\mathbf{s})^3.$$

Observation: Particular form of $\mathbf{B_k}$ allows for closed form solution.

II. Limited-memory symmetric-rank-1 updates

Limited-memory Symmetric-Rank-1 updates (L-SR1):

$$\mathbf{B}_{k+1} = \mathbf{B}_k + \frac{(\mathbf{y}_k - \mathbf{B}_k\mathbf{s}_k)(\mathbf{y}_k - \mathbf{B}_k\mathbf{s}_k)^\top}{(\mathbf{y}_k - \mathbf{B}_k\mathbf{s}_k)^\top \mathbf{s}_k},$$

where

- $\mathbf{B}_k$ is the Hessian approximation,
- $\mathbf{y}_k = \nabla f(\Theta_{k+1}) - \nabla f(\Theta_k)$,
- $\mathbf{s}_k = \Theta_{k+1} - \Theta_k$.

II. Limited-memory symmetric-rank-1 updates

Limited-memory Symmetric-Rank-1 updates (L-SR1):

$$\mathbf{B}_{k+1} = \mathbf{B}_k + \underbrace{\frac{(\mathbf{y}_k - \mathbf{B}_k\mathbf{s}_k)(\mathbf{y}_k - \mathbf{B}_k\mathbf{s}_k)^\top}{(\mathbf{y}_k - \mathbf{B}_k\mathbf{s}_k)^\top\mathbf{s}_k}}_{\text{Rank 1 (outer-product) update}},$$

where

- $\mathbf{B}_k$ is the Hessian approximation,
- $\mathbf{y}_k = \nabla f(\Theta_{k+1}) - \nabla f(\Theta_k)$,
- $\mathbf{s}_k = \Theta_{k+1} - \Theta_k$.

II. L-SR1 compact representation

Recursively,

$$\mathbf{B}_{k+1} = \mathbf{B}_0 + \underbrace{\sum_{j=0}^{m} \frac{(\mathbf{y}_j - \mathbf{B}_j\mathbf{s}_j)(\mathbf{y}_j - \mathbf{B}_j\mathbf{s}_j)^\top}{(\mathbf{y}_j - \mathbf{B}_j\mathbf{s}_j)^\top \mathbf{s}_j}}_{\text{rank } m \text{ update}}.$$

II. L-SR1 compact representation

Recursively,

$$\mathbf{B}_{k+1} = \mathbf{B}_0 + \underbrace{\sum_{j=0}^{m} \frac{(\mathbf{y}_j - \mathbf{B}_j\mathbf{s}_j)(\mathbf{y}_j - \mathbf{B}_j\mathbf{s}_j)^\top}{(\mathbf{y}_j - \mathbf{B}_j\mathbf{s}_j)^\top\mathbf{s}_j}}_{\text{rank } m \text{ update}}.$$

Compact representation of $\mathbf{B}_{k+1}$:

$$\mathbf{B}_{k+1} \; = \; \mathbf{B}_0 + \begin{bmatrix} \\ \boldsymbol{\Psi}_k \\ \\ \end{bmatrix} \begin{bmatrix} \mathbf{M}_k \end{bmatrix} \begin{bmatrix} \quad \boldsymbol{\Psi}_k^\top \quad \end{bmatrix},$$

where $\boldsymbol{\Psi}_k \in \mathbb{R}^{n\times m}$, $\mathbf{M} \in \mathbb{R}^{m\times m}$ and $\mathbf{B}_0 = \gamma\mathbf{I}$.
Note: $\mathbf{m} \ll \mathbf{n}$

QR decomposition of $\boldsymbol{\Psi}_k$:

$$\mathbf{B}_{k+1} = \gamma\mathbf{I} + \begin{bmatrix} \\ \boldsymbol{\Psi}_k \\ \\ \end{bmatrix} \begin{bmatrix} \mathbf{M}_k \end{bmatrix} \begin{bmatrix} \qquad \boldsymbol{\Psi}_k^\top \qquad \end{bmatrix},$$

II. QR decomposition

QR decomposition of $\boldsymbol{\Psi}_k$:

$$\mathbf{B}_{k+1} = \gamma \mathbf{I} + \begin{bmatrix} \mathbf{Q}_k \mathbf{R}_k \end{bmatrix} \begin{bmatrix} \mathbf{M}_k \end{bmatrix} \begin{bmatrix} \mathbf{R}_k^\top \mathbf{Q}_k^\top \end{bmatrix},$$

II. Eigendecomposition

Eigendecomposition of $\mathbf{R}_k \mathbf{M}_k \mathbf{R}_k^\top$:

$$\mathbf{B}_{k+1} = \gamma \mathbf{I} + \begin{bmatrix} \mathbf{Q}_k \mathbf{P}_k \\ \\ \end{bmatrix} \begin{bmatrix} \boldsymbol{\Lambda}_k \end{bmatrix} \begin{bmatrix} \mathbf{P}_k^\top \mathbf{Q}_k^\top \end{bmatrix},$$

Computing the eigenvectors of $\boldsymbol{\Psi}_k \mathbf{M}_k \boldsymbol{\Psi}_k^\top$:

$$\mathbf{B}_{k+1} = \gamma \mathbf{I} + \begin{bmatrix} \mathbf{U}_\parallel \end{bmatrix} \begin{bmatrix} \boldsymbol{\Lambda}_k \end{bmatrix} \begin{bmatrix} \mathbf{U}_\parallel^\top \end{bmatrix},$$

II. Orthonormal space

Computing the eigenvectors of $\mathbf{B}_{k+1}$:

$$\mathbf{B}_{k+1} = \gamma\mathbf{I} + \begin{bmatrix} \mathbf{U}_{\|} & \mathbf{U}_{\perp} \end{bmatrix} \begin{bmatrix} (\mathbf{\Lambda}_k)_{\|} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{U}_{\|}^{\top} \\ \mathbf{U}_{\perp}^{\top} \end{bmatrix}$$

II. Orthonormal space

Computing the eigenvectors of $\mathbf{B}_{k+1}$:

$$
\begin{aligned}
\mathbf{B}_{k+1} &= \gamma \mathbf{I} + \begin{bmatrix} \mathbf{U}_{\parallel} & \mathbf{U}_{\perp} \end{bmatrix} \begin{bmatrix} (\mathbf{\Lambda}_k)_{\parallel} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{U}_{\parallel}^{\top} \\ \mathbf{U}_{\perp}^{\top} \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{U}_{\parallel} & \mathbf{U}_{\perp} \end{bmatrix} \begin{bmatrix} (\mathbf{\Lambda}_k)_{\parallel} + \gamma \mathbf{I} & 0 \\ 0 & \gamma \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{U}_{\parallel}^{\top} \\ \mathbf{U}_{\perp}^{\top} \end{bmatrix}
\end{aligned}
$$

II. Orthonormal space

Computing the eigenvectors of $\mathbf{B}_{k+1}$:

$$\begin{aligned}
\mathbf{B}_{k+1} &= \gamma \mathbf{I} + \begin{bmatrix} \mathbf{U}_{\|} & \mathbf{U}_{\perp} \end{bmatrix} \begin{bmatrix} (\boldsymbol{\Lambda}_k)_{\|} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{U}_{\|}^\top \\ \mathbf{U}_{\perp}^\top \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{U}_{\|} & \mathbf{U}_{\perp} \end{bmatrix} \begin{bmatrix} (\boldsymbol{\Lambda}_k)_{\|} + \gamma \mathbf{I} & 0 \\ 0 & \gamma \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{U}_{\|}^\top \\ \mathbf{U}_{\perp}^\top \end{bmatrix} \\
&= \mathbf{U} \boldsymbol{\Lambda} \mathbf{U}^\top
\end{aligned}$$

II. Subproblem transformation

Recall the ARCs subproblem:

$$\underset{\mathbf{s}\in\mathbb{R}^n}{\text{minimize}}\ \mathcal{M}_k(\mathbf{s}) \equiv f(\Theta_k) + \mathbf{g}_k^\top \mathbf{s}_k + \frac{1}{2}\mathbf{s}_k^\top \mathbf{B}_k \mathbf{s}_k + \frac{\sigma_k}{3}\|\mathbf{s}_k\|_{\mathbf{U}}^3.$$

II. Subproblem transformation

Recall the ARCs subproblem:

$$\underset{\mathbf{s}\in\mathbb{R}^n}{\text{minimize}}\ \mathcal{M}_k(\mathbf{s}) \equiv f(\Theta_k) + \mathbf{g}_k^\top \mathbf{s}_k + \frac{1}{2}\mathbf{s}_k^\top \mathbf{B}_k \mathbf{s}_k + \frac{\sigma_k}{3}\|\mathbf{s}_k\|_{\mathbf{U}}^3.$$

Applying the change of variables $\|\mathbf{s}\|_{\mathbf{U}} \stackrel{\text{def}}{=} \|\mathbf{U}^\top \mathbf{s}\|_3 \stackrel{\text{def}}{=} \|\bar{\mathbf{s}}\|_3$:

$$\underset{\bar{\mathbf{s}}\in\mathbb{R}^n}{\text{minimize}}\ \bar{\mathcal{M}}_k(\bar{\mathbf{s}}) \equiv \bar{\mathbf{g}}_k^\top \bar{\mathbf{s}}_k + \frac{1}{2}\bar{\mathbf{s}}_k^\top \Lambda \bar{\mathbf{s}}_k + \frac{\sigma_k}{3}\|\bar{\mathbf{s}}_k\|_3^3.$$

II. Subproblem transformation

Recall the ARCs subproblem:

$$\underset{\mathbf{s}\in\mathbb{R}^n}{\text{minimize}}\ \mathcal{M}_k(\mathbf{s}) \equiv f(\Theta_k) + \mathbf{g}_k^\top \mathbf{s}_k + \frac{1}{2}\mathbf{s}_k^\top \mathbf{B}_k \mathbf{s}_k + \frac{\sigma_k}{3}\|\mathbf{s}_k\|_{\mathbf{U}}^3.$$

Applying the change of variables $\|\mathbf{s}\|_{\mathbf{U}} \overset{\text{def}}{=} \|\mathbf{U}^\top \mathbf{s}\|_3 \overset{\text{def}}{=} \|\bar{\mathbf{s}}\|_3$:

$$\underset{\bar{\mathbf{s}}\in\mathbb{R}^n}{\text{minimize}}\ \bar{\mathcal{M}}_k(\bar{\mathbf{s}}) \equiv \bar{\mathbf{g}}_k^\top \bar{\mathbf{s}}_k + \frac{1}{2}\bar{\mathbf{s}}_k^\top \Lambda \bar{\mathbf{s}}_k + \frac{\sigma_k}{3}\|\bar{\mathbf{s}}_k\|_3^3.$$

ARCs subproblem decomposition:

$$\underset{\bar{\mathbf{s}}\in\mathbb{R}^n}{\text{minimize}}\ \bar{\mathcal{M}}(\bar{\mathbf{s}}) \equiv \sum_{i=1}^{n} \underset{\bar{s}_i}{\text{minimize}} \left(\bar{g}_i \bar{s}_i + \frac{\lambda_i}{2}\bar{s}_i^{\,2} + \frac{\sigma}{3}|\bar{s}_i|^3 \right)$$

II. Subproblem transformation

Recall the ARCs subproblem:

$$\underset{\mathbf{s}\in\mathbb{R}^n}{\text{minimize}}\ \mathcal{M}_k(\mathbf{s}) \equiv f(\Theta_k) + \mathbf{g}_k^\top \mathbf{s}_k + \frac{1}{2}\mathbf{s}_k^\top \mathbf{B}_k \mathbf{s}_k + \frac{\sigma_k}{3}\|\mathbf{s}_k\|_{\mathbf{U}}^3.$$

Applying the change of variables $\|\mathbf{s}\|_{\mathbf{U}} \overset{\text{def}}{=} \|\mathbf{U}^\top \mathbf{s}\|_3 \overset{\text{def}}{=} \|\bar{\mathbf{s}}\|_3$:

$$\underset{\bar{\mathbf{s}}\in\mathbb{R}^n}{\text{minimize}}\ \bar{\mathcal{M}}_k(\bar{\mathbf{s}}) \equiv \bar{\mathbf{g}}_k^\top \bar{\mathbf{s}}_k + \frac{1}{2}\bar{\mathbf{s}}_k^\top \Lambda \bar{\mathbf{s}}_k + \frac{\sigma_k}{3}\|\bar{\mathbf{s}}_k\|_3^3.$$

ARCs subproblem decomposition:

$$\underset{\bar{\mathbf{s}}\in\mathbb{R}^n}{\text{minimize}}\ \bar{\mathcal{M}}(\bar{\mathbf{s}}) \equiv \sum_{i=1}^{n} \underset{\bar{s}_i}{\text{minimize}} \left(\bar{g}_i \bar{s}_i + \frac{\lambda_i}{2}\bar{s}_i^{\,2} + \frac{\sigma}{3}|\bar{s}_i|^3 \right)$$

Observation: **This has a closed form solution!**

II. Solution to the CR subproblem

Exact solution in $\bar{\mathbf{s}}$:

$$\bar{\mathbf{s}}^* = -\mathbf{C}\bar{\mathbf{g}},$$

where $\mathbf{C} = \mathrm{diag}(c_1, c_2 \ldots c_n)$ and

$$c_i = \frac{2}{\lambda_i + \sqrt{\lambda_i^2 + 4\sigma|\bar{g}_i|}}.$$

II. Solution to the CR subproblem

Exact solution in $\bar{\mathbf{s}}$:

$$\bar{\mathbf{s}}^* = -\mathbf{C}\bar{\mathbf{g}},$$

where $\mathbf{C} = \text{diag}(c_1, c_2 \ldots c_n)$ and

$$c_i = \frac{2}{\lambda_i + \sqrt{\lambda_i^2 + 4\sigma|\bar{g}_i|}}.$$

Exact solution in $\mathbf{s}$:

$$\mathbf{s}^* = \mathbf{U}\bar{\mathbf{s}}^*. \tag{1}$$

II. Convergence analysis

Assumptions:

A1. The loss function $f(\Theta)$ is continuously differentiable, i.e., $f \in C^1(\mathbb{R}^n)$.

II. Convergence analysis

Assumptions:
A1. The loss function $f(\Theta)$ is continuously differentiable, i.e., $f \in C^1(\mathbb{R}^n)$.
A2. The loss function $f(\Theta)$ is bounded below.

II. Convergence analysis

Assumptions:

A1. The loss function $f(\Theta)$ is continuously differentiable, i.e., $f \in C^1(\mathbb{R}^n)$.

A2. The loss function $f(\Theta)$ is bounded below.

A3. If $\{\Theta_{t_i}\}$ and $\{\Theta_{l_i}\}$ are subsequences of $\{\Theta_k\}$, then $\|\mathbf{g}_{t_i} - \mathbf{g}_{l_i}\| \to 0$ whenever $\|\Theta_{t_i} - \Theta_{l_i}\| \to 0$ as $i \to \infty$.

II. Convergence analysis

Assumptions:

A1. The loss function $f(\Theta)$ is continuously differentiable, i.e., $f \in C^1(\mathbb{R}^n)$.

A2. The loss function $f(\Theta)$ is bounded below.

A3. If $\{\Theta_{t_i}\}$ and $\{\Theta_{l_i}\}$ are subsequences of $\{\Theta_k\}$, then $\|\mathbf{g}_{t_i} - \mathbf{g}_{l_i}\| \to 0$ whenever $\|\Theta_{t_i} - \Theta_{l_i}\| \to 0$ as $i \to \infty$.

Lemma

The SR1 matrix $\mathbf{B}_{k+1}$ satsifies $\|\mathbf{B}_{k+1}\|_F \leq \kappa_B$ for all $k \geq 1$ for some $\kappa_B > 0$.

II. Convergence analysis

Assumptions:

A1. The loss function $f(\Theta)$ is continuously differentiable, i.e., $f \in C^1(\mathbb{R}^n)$.

A2. The loss function $f(\Theta)$ is bounded below.

A3. If $\{\Theta_{t_i}\}$ and $\{\Theta_{l_i}\}$ are subsequences of $\{\Theta_k\}$, then $\|\mathbf{g}_{t_i} - \mathbf{g}_{l_i}\| \to 0$ whenever $\|\Theta_{t_i} - \Theta_{l_i}\| \to 0$ as $i \to \infty$.

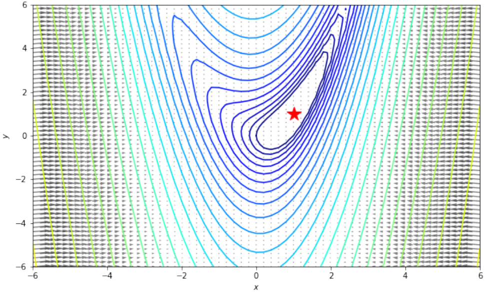
Lemma

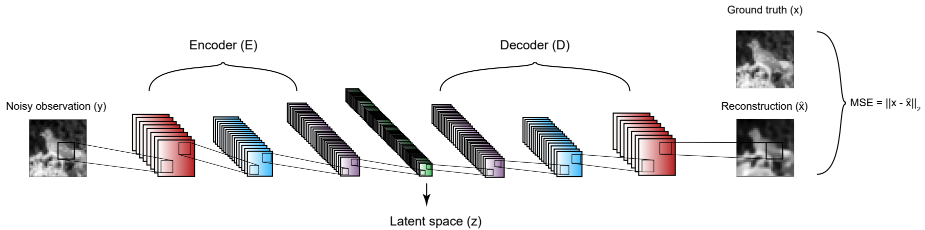
The SR1 matrix $\mathbf{B}_{k+1}$ *satsifies* $\|\mathbf{B}_{k+1}\|_F \leq \kappa_B \quad$ *for all* $k \geq 1$ *for some* $\kappa_B > 0$.

Theorem

*Under Assumptions **A1**, **A2**, and **A3**, if Lemma 1 holds, then* $\lim\limits_{k\to\infty} \|\mathbf{g}_k\| = 0$.

II. Evolution on the Rosenbrock function

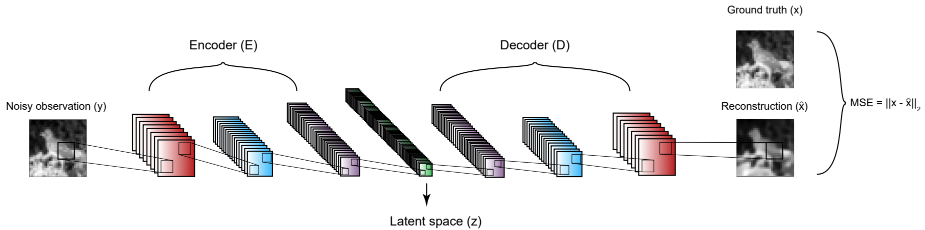




Autoencoder operation:

- Encoder: Downsamples image to latent space z.

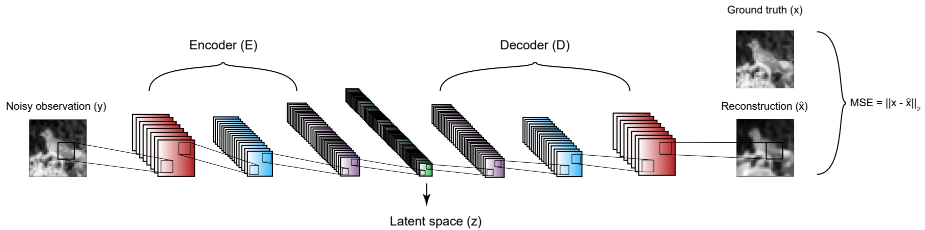
II. Experiment - Image reconstruction



Autoencoder operation:

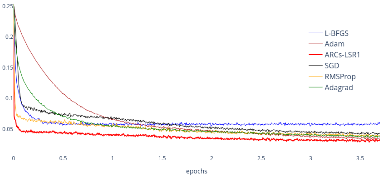
- Encoder: Downsamples image to latent space z.
- Decoder: Upsamples from z to image space.

II. Experiment - Image reconstruction

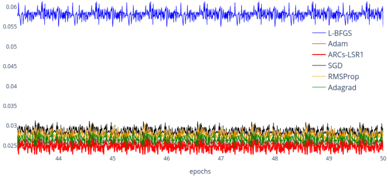


Autoencoder operation:

- Encoder: Downsamples image to latent space z.
- Decoder: Upsamples from z to image space.
- Loss function: Mean-square error between Reconstruction and Ground truth.



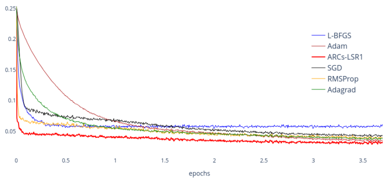
a.



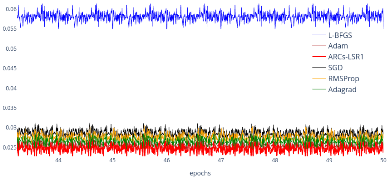
b.

Table: Results on MNIST dataset. Fig. a. Initial training response. Fig. b. Final training response

II. Experiment - Image reconstruction results



a.

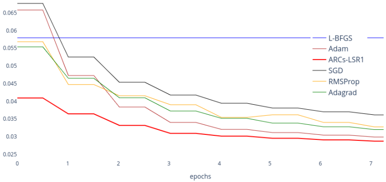


b.

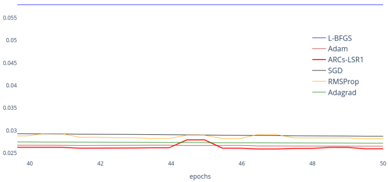
Table: Results on MNIST dataset. Fig. a. Initial training response. Fig. b. Final training response

Proposed approach minimizes the loss function in the fewest number of steps.

II. Experiment - Image reconstruction results

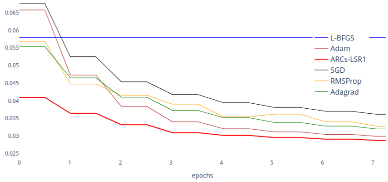


a.

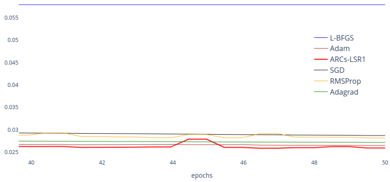


b.

Table: Results on MNIST dataset. Fig. a. Initial testing response Fig. b. Final testing response



a.

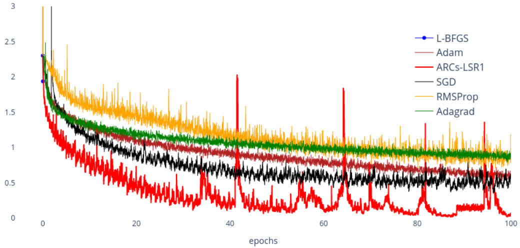


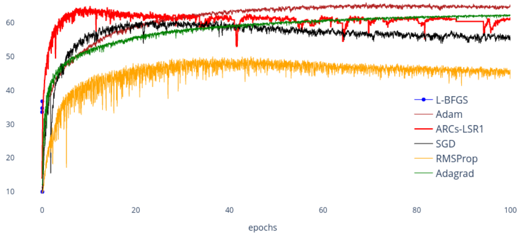
b.

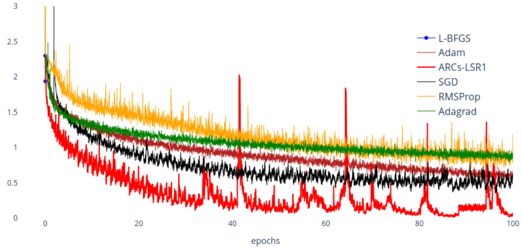
Table: Results on MNIST dataset. Fig. a. Initial testing response Fig. b. Final testing response

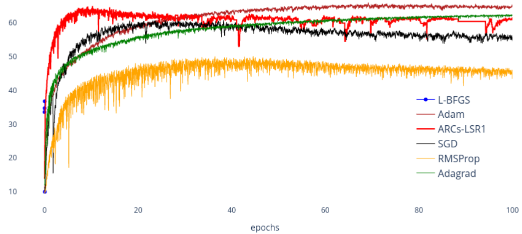
Proposed approach generalizes over the test dataset better in comparison.

II. Experiment - Classification results CIFAR10



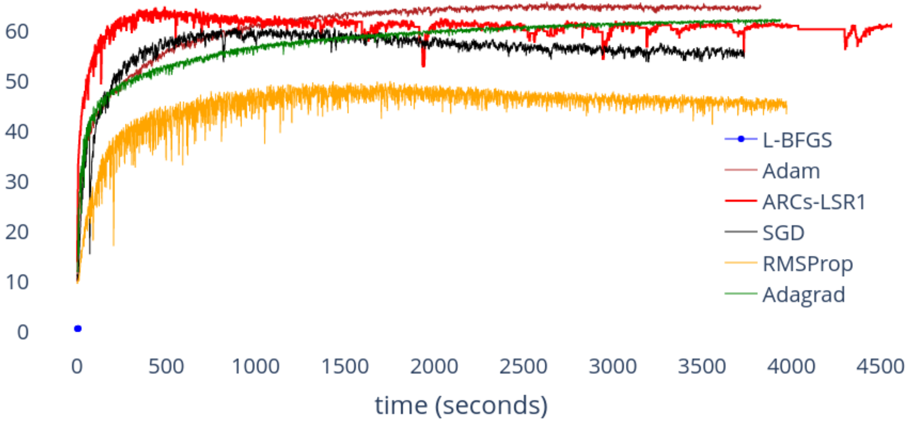




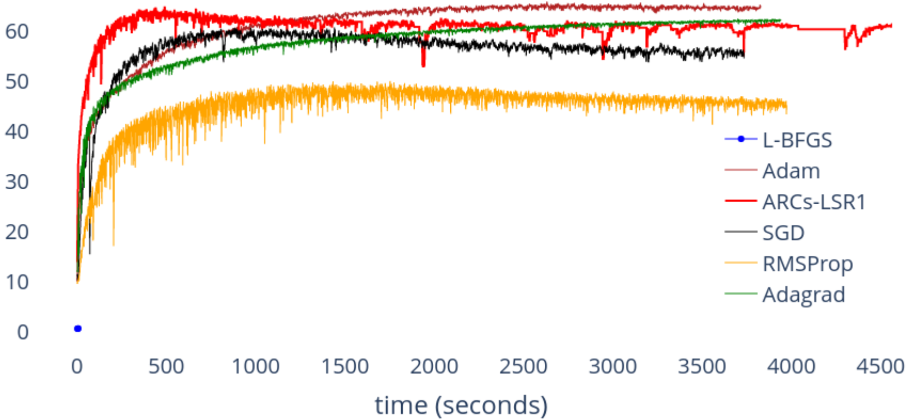


The proposed approach performs better than most existing state-of-the-art method.

II. Experiment - Timing results



II. Experiment - Timing results



Observation: Method converges fast.

Conclusion and Remarks

- Deep learning is an important tool in data science.

Conclusion and Remarks

- Deep learning is an important tool in data science.
- Optimization techniques play crucial roles in improving prediction models.

Conclusion and Remarks

- Deep learning is an important tool in data science.
- Optimization techniques play crucial roles in improving prediction models.
- Existing state-of-the-art methods use and retain only first-order information.

Conclusion and Remarks

- Deep learning is an important tool in data science.
- Optimization techniques play crucial roles in improving prediction models.
- Existing state-of-the-art methods use and retain only first-order information.
- We developed two methods that compromise first- and second-order methods.

Conclusion and Remarks

- Deep learning is an important tool in data science.
- Optimization techniques play crucial roles in improving prediction models.
- Existing state-of-the-art methods use and retain only first-order information.
- We developed two methods that compromise first- and second-order methods.
- The method uses an adaptive regularized cubics approach with a L-SR1 quasi-Newton approximation.

Conclusion and Remarks

- Deep learning is an important tool in data science.
- Optimization techniques play crucial roles in improving prediction models.
- Existing state-of-the-art methods use and retain only first-order information.
- We developed two methods that compromise first- and second-order methods.
- The method uses an adaptive regularized cubics approach with a L-SR1 quasi-Newton approximation.
- Numerical experiments demonstrate improvement over existing state-of-the-art methods.

THANK YOU