

DRSOM: A Dimension Reduced Second-Order Method and Preliminary Analyses

Chuwen Zhang[†], Dongdong Ge[†], Bo Jiang[†], and Yinyu Ye[‡]

[†]*School of Information Management and Engineering, Shanghai University of Finance and Economics*

[‡]*Department of Management Science and Engineering, Stanford University*

[†]*chuwzhang@gmail.com, ge.dongdong@shufe.edu.cn, isyebjiang@gmail.com, [‡]yinyu-ye@stanford.edu*

August 2, 2022

Abstract

We introduce a Dimension-Reduced Second-Order Method (DRSOM) for convex and nonconvex unconstrained optimization. Under a trust-region-like framework our method preserves the convergence of the second-order method while using *only* Hessian-vector products in two directions. Moreover, the computational overhead remains comparable to the first-order such as the gradient descent method. We show that the method has a complexity of $O(\epsilon^{-3/2})$ to satisfy the first-order and second-order conditions in the subspace. The applicability and performance of DRSOM are exhibited by various computational experiments in logistic regression, $\mathcal{L}_2 - \mathcal{L}_p$ minimization, sensor network localization, and neural network training. For neural networks, our preliminary implementation seems to gain computational advantages in terms of training accuracy and iteration complexity over state-of-the-art first-order methods including SGD and ADAM.

1 Introduction

In this paper, we consider the following unconstrained optimization problem

$$\min_{x \in \mathbb{R}^n} f(x), \quad (1)$$

where f is twice differentiable and has L -Lipschitz continuous gradient and M -Lipschitz Hessian. If f is nonconvex, we focus on finding a “stationary point” such that

$$\|\nabla f(x)\| \leq \epsilon \quad (2)$$

and x satisfies second-order necessary condition in certain subspace.

The gradient descent method (GD) generates an iterate x_{k+1} by using the negative gradient direction:

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k), \quad (3)$$

where α_k is the step-size. With L -Lipschitz condition, the GD (3) is able to find an s satisfying (2) in $O(\epsilon^{-2})$ iterations [20]. Polyak [23] considers the heavy ball method that includes a momentum (also called extrapolation) direction, $d_k = x_k - x_{k-1}$, and the gradient direction to update x_k :

$$x_{k+1} = x_k - \alpha_k^1 \nabla f(x_k) + \alpha_k^2 d_k, \quad (4)$$

where α_k^1, α_k^2 are step-sizes for gradient and *momentum* direction, respectively. This idea can also be found in the acceleration methods in a similar spirit. Specifically, for convex optimization, the accelerated gradient method reduces the iteration complexity to $O(\epsilon^{-1/2})$ [20]. This idea later becomes popular in stochastic optimization for adaptive and accelerated methods, such as SGD (with momentum) [26], Adam [17], and so forth. The Adam [17] also includes the momentum of second-moments of the gradient in a similar way. From a quite different perspective, the conjugate-gradient and parallel-tangent methods [16, 13, 18] can also be regarded as a way to utilize the extrapolation step.

If the second-order derivative information is further available, one can construct the iterates by sequential quadratic approximations to f :

$$m_k(d) = \min f(x_k) + \nabla f(x_k)^T d + \frac{1}{2} d^T \nabla^2 f(x_k) d \quad (5)$$

In the traditional trust-region method, the above problem is solved by restricting the direction d within an “adaptive” spherical trust region $\|d\| \leq \Delta_k$ where the approximation is acceptable when certain criterion is satisfied.

In Ye [33], an interior-spherical trust-region method is proposed to minimize a non-convex quadratic function subject to linear and bounded constraints, where the method has an iteration complexity $O(\epsilon^{-1})$ to compute a solution satisfying the ϵ -KKT condition. Moreover, the limit point of the algorithm that satisfies the second order necessary condition. In the seminal work of [21], Nesterov and Polyak [21] show that iteratively solving (5) with the cubic-regularization can also serve the purpose of minimizing certain general functions with the iteration complexity of $O(\epsilon^{-3/2})$. Since then, Cartis et al. [6, 7] consider an adaptive cubic-regularization (ARC) algorithm to allow weaker assumptions while preserving the $O(\epsilon^{-3/2})$ complexity bound. Coming back to the trust-region method, where the cubic regularization is absent, [18] establishes the same iteration bound with a fixed radius. Recently, Curtis et al. [12] extend this bound to more sophisticated trust-region updates.

Since the computation of a Hessian matrix and the global solution of a quadratic model are usually costly and less practical, various techniques have been found to bypass this difficulty. One typical strategy is to approximate the Hessian matrix by quasi-Newton methods [3], sampling techniques [27, 30, 8], and so forth. Another direction is to allow the inexact solutions of the cubic or trust-region model, for example, by achieving optimality in some subspace but possibly sub-optimality in the whole space. For the trust-region methods, we may choose from the negative gradient and the Newton direction such that the iterate x_k lies in the subspace:

$$x_k \in \text{span}\{\nabla f(x_k), [\nabla^2 f(x_k)]^{-1} \nabla f(x_k)\} \quad (6)$$

The idea of updating in subspace can be traced back to the dogleg, and two-dimensional subspace method [25, 4]. As the name indicates, this type of method only requires Hessian information in several directions. Except for the negative gradient and Newton direction, the most negative eigenvalue of the Hessian matrix may also be adopted as an alternative choice, see [1, 5]. It is also possible to explore optimality in higher dimensions, e.g., the Krylov subspace, using an iterative scheme like the Lanczos method. We refer the interesting readers to [11, 6, 12] for more details.

Other than solving a quadratic model, Carmon et al. [5] consider an accelerated gradient method (AGD) that incorporates the negative curvature to improve the $O(\epsilon^{-2})$ bound for first-order methods to $O(\epsilon^{-7/4})$. In contrast with the Lanczos-like iteration

that aims for in-depth exploration of the subspace, computing the most negative eigenvector of Hessian is a major concern in their framework. Furthermore, it only utilizes the Hessian-vector product $\nabla^2 f(x) \cdot v$, which can be efficiently computed at the expense of two calls of gradient oracle. In this light, the method in [5] lies in the family of Hessian-free methods (see, [19]).

It becomes quite natural for us to envelope the techniques mentioned above. Specifically, we introduce a *Dimension-Reduced Second-Order Method* (DRSOM):

$$x_{k+1} = x_k - \alpha_k^1 \nabla f(x_k) + \alpha_k^2 (x_k - x_{k-1}), \quad \text{and} \quad \alpha_k = \arg \min_{\alpha^T G_k \alpha \leq \delta} m_k^\alpha(\alpha)$$

where $\alpha_k = [\alpha_k^1, \alpha_k^2]$ is the stepsize, $m_k^\alpha(\alpha)$ is a 2-dimensional quadratic approximation to $f(x)$ in a subspace, and G_k is a 2×2 positive semidefinite matrix. Both $m_k^\alpha(\alpha)$ and G_k depend on $\{\nabla f(x_k), x_k - x_{k-1}\}$, and are specified in Section 2.2.

Unlike the heavy ball method and AGD, the DRSOM produces the stepsizes by solving a single 2-dimensional quadratic program with a spherical constraint. As we will show later, to update each iterate x_k , we only require two extra Hessian-vector products to construct the quadratic approximation, whose overhead remains comparable to the gradient descent method, due to no direct (iterative) Hessian evaluation is needed. This distinguishes our approach from traditional second-order methods using full-dimensional Hessian information or subspace strategies like the Lanczos method. We also note that the subproblem can be solved in log-polynomial time. Similar to the trust-region method [12], we show that DRSOM is equivalent to a quadratic-regularization method so that it can be interpreted as the “Radius-Free” approach. We also extend DRSOM to a mini-batch version that is commonly used to train neural networks.

Theoretically, we show that DRSOM has a $O(\epsilon^{-3/2})$ complexity to terminate at an iterate x_k satisfying the first-order condition (2) and the second-order condition in the certain subspace. It implies that the DRSOM has better iteration complexity than the first-order methods to provide a first-order solution with slightly more per-iteration cost.

We further justify the applicability and performance of DRSOM by providing comprehensive computational experiments. For logistic regression problem, the results show that the DRSOM is comparable to a popular first-order stochastic method, SAGA, and a second-order method LBFGS in terms of both training error and testing error. For the nonconvex $\mathcal{L}_2 - \mathcal{L}_p$ minimization problem, the results show that the DRSOM requires almost the same number of iterations as BFGS, and the number is larger than that of trust-region method with full Hessian information by a limited margin. For sensor network localization problem, we provide an example that the DRSOM manages to identify the ground truth, while the gradient descent method with a line search converges to a strict local optimum. Lastly, we develop a vanilla mini-batch DRSOM to train neural networks. We provide the results on two data-sets: Fashion-MNIST and CIFAR10. These preliminary results indicate that DRSOM consumes less iterations than standard the-state-of-the-art methods such as SGD and Adam, which demonstrates the potential of DRSOM for deep learning.

Our paper is organized as follows. In Section 2, we discuss the details of the DRSOM including its important building blocks. In Section 3, we give convergence analysis of DRSOM. Finally, the comprehensive numerical experiments are presented in Section 4.

2 The DRSOM

2.1 Notations, assumption and preliminary results

We first introduce a few notations and technical lemmas. We denote the standard Euclidean norm by $\|\cdot\|$. If $A \in \mathbb{R}^{n \times n}$, then $\|A\|$ is the induced \mathcal{L}_2 norm. If A is positive semidefinite, we use $A \succeq 0$. For compactness, we let $g_k = g(x_k) = \nabla f(x_k)$, $H_k = H(x_k) = \nabla^2 f(x_k)$. We denote $\mathcal{N}(\mu, \sigma^2)$ as the normal distribution with μ, σ^2 as mean and variance, respectively. We let $f_{\inf} = \inf f(x)$.

Throughout this paper, we assume f is bounded below and is twice differentiable with L -Lipschitz gradient and M -Lipschitz Hessian.

Assumption 1. Let $f : \mathbb{R}^n \mapsto \mathbb{R}$ and $f_{\inf} > -\infty$. Furthermore, it has L -Lipschitz continuous gradient and M -Lipschitz continuous Hessian such that for $\forall x, y \in \mathbb{R}^n$,

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\| \quad (7a)$$

$$\|\nabla^2 f(x) - \nabla^2 f(y)\| \leq M\|x - y\| \quad (7b)$$

We have the next useful lemma guaranteed by first- and second-order Lipschitz continuity. We will use these standard results repetitively.

Lemma 1 (Nesterov [20]). If $f : \mathbb{R}^n \mapsto \mathbb{R}$ satisfies the Assumption 1, then, for all $x, y \in \mathbb{R}^n$,

$$|f(y) - f(x) - \nabla f(x)^T(y - x)| \leq \frac{L}{2}\|y - x\|^2 \quad (8a)$$

$$\|\nabla f(y) - \nabla f(x) - \nabla^2 f(x)(y - x)\| \leq \frac{M}{2}\|y - x\|^2 \quad (8b)$$

$$\left| f(y) - f(x) - \nabla f(x)^T(y - x) - \frac{1}{2}(y - x)^T \nabla^2 f(x)(y - x) \right| \leq \frac{M}{6}\|y - x\|^3 \quad (8c)$$

2.2 Overview of algorithm

We now give a brief overview of our method. Recall the *Dimension-Reduced Second-Order Method* (DRSOM) mentioned above:

$$x_{k+1} = x_k - \alpha_k^1 g_k + \alpha_k^2 d_k, \quad \text{and} \quad \alpha_k = \arg \min m_k^\alpha(\alpha) \quad (9a)$$

where the 2-dimensional quadratic model $m_k^\alpha(\alpha)$ is defined as follows:

$$m_k^\alpha(\alpha) := f(x_k) + (c_k)^T \alpha + \frac{1}{2} \alpha^T Q_k \alpha \quad (10a)$$

$$Q_k = \begin{bmatrix} (g_k)^T H_k g_k & -(d_k)^T H_k g_k \\ -(d_k)^T H_k g_k & (d_k)^T H_k d_k \end{bmatrix} \in \mathcal{S}^2, c_k = \begin{bmatrix} -\|g_k\|^2 \\ +(g_k)^T d_k \end{bmatrix} \in \mathbb{R}^2. \quad (10b)$$

This idea, to our best knowledge, was first introduced in the lecture notes by Ye [31]. By further restricting a “trust-region” for α , we solve the following problem to construct the next iteration:

$$\min_{\alpha \in \mathbb{R}^2} m_k^\alpha(\alpha) \quad (11a)$$

$$\text{s.t. } \|\alpha\|_{G_k} \leq \Delta, G_k = \begin{bmatrix} (g_k)^T g_k & -(g_k)^T d_k \\ -(g_k)^T d_k & (d_k)^T d_k \end{bmatrix}, \quad (11b)$$

where $\|\alpha\|_{G_k}^2 = \alpha^T G_k \alpha$. To ensure global convergence, we adopt a trust-region framework that controls the progress made at each step and sequentially solves the two-dimensional quadratic model. Similar to the standard trust-region method, we introduce the reduction ratio for m_k^α at iterate x_k ,

$$\rho_k := \frac{f(x_k) - f(x_k + p_k)}{m_k^\alpha(0) - m_k^\alpha(\alpha_k)}, \quad (12)$$

such that if ρ_k is too small, our quadratic model is somehow inaccurate, and it prompts us to reduce the trust region radius, especially when $f(x_k) - f(x_k + p_k) < 0$; otherwise m_k^α is good enough, we increase the radius allowing larger search region.

We present the conceptual DRSOM in Algorithm 1, which derives from the standard trust-region method, see [11, 22].

Algorithm 1: A conceptual DRSOM algorithm

Data: Given $k_{\max}, \beta_1 < 1 < \beta_2, \zeta_1 < \zeta_2 \leq 1; \bar{\Delta} > 0, \Delta_0 \in (0, \bar{\Delta}),$ and $\eta \in [0, \zeta_1];$

1 **for** $k = 1, \dots, k_{\max}$ **do**

2 Solve (11a) for α_k , obtain $p_k = -\alpha_k^1 g_k + \alpha_k^2 d_k$, and ρ_k by (12);

3 **if** $\rho_k \leq \zeta_1$ **then**

4 Decrease $\Delta_{k+1} = \beta_1 \Delta_k$

5 **else**

6 **if** $\rho_k > \zeta_2$ and $\|d_{k+1}\| = \Delta_k$ **then**

7 $\Delta_{k+1} = \min \{\beta_2 \Delta_k, \bar{\Delta}\}$

8 **else**

9 $\Delta_{k+1} = \Delta_k$

10 **end**

11 **end**

12 **if** $\rho_k > \eta$ **then**

13 $x_{k+1} = x_k + d_{k+1}$

14 **else**

15 $x_{k+1} = x_k$

16 **end**

17 **end**

Now we have seen, at a glance, the fundamental elements of DRSOM presented in Algorithm 1. Before we get into technical details, we would like to offer an eager comparison with a few widely celebrated methods. First and foremost, the DRSOM can be seen as an “adaptive” heavy-ball method: the stepsizes for momentum and gradient are the global solution of the quadratic model. Similarly, in comparison to the parallel tangents (PARTAN), the stepsizes are computed simultaneously instead of two consecutive line searches, see [18]. If we focus on convex quadratic programming, the DRSOM coincides with the linear conjugate gradient method (CG), just like the case for PARTAN. The equivalence to CG is later given in Theorem 2.

Next, we depict by the following Lemma the connection between DRSOM and the “full-scale” trust-region method with model function $m_k(d)$, cf. (5).

Lemma 2 (Trust-region). *Consider the full-scale trust-region subproblem (TRS),*

$$\min_{d \in \mathbb{R}^n} m_k(d), \quad \text{s.t. } \|d\| \leq \Delta \quad (13)$$

If we let $d_{k+1} = -\alpha_k^1 g_k + \alpha_k^2 d_k$, then minimizing m_k^α (11a) solves the full-dimensional TRS (13).

Proof. Let $d := d_{k+1} = -\alpha_k^1 g_k + \alpha_k^2 d_k$, the original TRS:

$$m_k(d) = f(x_k) - \alpha_k^1 (g_k)^T g_k + \alpha_k^2 (g_k)^T d_k \quad (14a)$$

$$+ \frac{1}{2} \begin{bmatrix} \alpha_k^1 \\ \alpha_k^2 \end{bmatrix}^T \begin{bmatrix} -(g_k)^T \\ (d_k)^T \end{bmatrix} H_k \begin{bmatrix} -g_k & d_k \end{bmatrix} \begin{bmatrix} \alpha_k^1 \\ \alpha_k^2 \end{bmatrix} \quad (14b)$$

$$= f(x_k) + \alpha^T c_k + \frac{1}{2} \alpha^T Q_k \alpha, \quad (14c)$$

In view of (14c), it implies (14a) is equivalent to (10).

As for radius $\|d\| \leq \Delta$, we have:

$$\|d\|^2 = \begin{bmatrix} \alpha_k^1 \\ \alpha_k^2 \end{bmatrix}^T \begin{bmatrix} -(g_k)^T \\ (d_k)^T \end{bmatrix} \begin{bmatrix} -g_k & d_k \end{bmatrix} \begin{bmatrix} \alpha_k^1 \\ \alpha_k^2 \end{bmatrix} \quad (15a)$$

$$= \alpha^T \begin{bmatrix} (g_k)^T g_k & -(g_k)^T d_k \\ -(g_k)^T d_k & (d_k)^T d_k \end{bmatrix} \alpha \leq \Delta^2, \quad (15b)$$

which establishes the connection of trust-region radii for two problems. \square

This implies the DRSOM, as a trust-region variant, is actually a second-order method in a dimension-reduced format. We note that the DRSOM performs differently from other subspace strategies, such as the ones using classic Newton directions [4], and Lanczos basis of the Krylov space [6, 7], and many others. In essence, the DRSOM does not involve an eigenvalue problem nor orthogonalization of any kind; instead, it only requires two Hessian-vector products to construct the quadratic model. This notable distinction aligns DRSOM more closely with the first-order methods.

Let us formalize the subspace description by the following fact.

Lemma 3. *The subproblem (10) at an iterate x_k of DRSOM is equivalent to the following problem,*

$$\min_{d \in \mathbb{R}^n} f(x_k) + g_k^T d + \frac{1}{2} d^T \tilde{H}_k d, \quad \text{s.t. } \|d\| \leq \Delta, \quad (16)$$

where V_k is the orthonormal basis for $\mathcal{L}_k := \text{span}\{g_k, d_k\}$, and,

$$\tilde{H}_k = V_k V_k^T H V_k V_k^T \quad (17)$$

Proof. Let $R_k = [-g_k \ d_k]$, indeed, $\exists M_k \in \mathbb{R}^{2 \times 2}$ such that $R_k = V_k M_k$. If $d_k \neq \pm g_k$, M_k is a triangular matrix. For any $\alpha \in \mathbb{R}^2$, $\exists d \in \mathbb{R}^n$, $\|d\| \leq \Delta$ such that $V_k V_k^T d = R_k \alpha$. From the equivalence indicated in Lemma 2, we have,

$$m_k(d) = m_k^\alpha(\alpha) = f(x_k) + g_k^T V_k V_k^T d + \frac{1}{2} d^T V_k V_k^T H_k V_k V_k^T d \quad (18a)$$

$$= f(x_k) + g_k^T d + \frac{1}{2} d^T \tilde{H}_k d \quad (g_k \in \mathcal{L}_k), \quad (18b)$$

where (18b) follows from the fact that $g_k \in \mathcal{L}_k$ and $V_k V_k^T$ is a projection matrix. \square

By Lemma 3, our interpretation of the DRSOM is two-fold. First, one can obviously take it as the quadratic model in the subspace without eigenvalue procedures. The second view is more intriguing: if we place a truncated approximation \tilde{H}_k to the original Hessian H_k , then the explicit requirements $d \in \mathcal{L}_k$ can be dropped. As a result, the DRSOM may also be regarded as a cheap quasi-Newton method.

2.3 The Hessian-vector product and trust-region subproblem

The present section derives two subroutines that play a central role in DRSOM.

2.3.1 Computing the Hessian-vector Product

Note that at each iteration, we have to build a two-by-two Q_k in the quadratic model (11a). We further breakdown Q_k based on the fact that:

$$Q_k = \begin{bmatrix} -g_k^T \\ d_k^T \end{bmatrix} \begin{bmatrix} -H_k \cdot g_k & H_k \cdot d_k \end{bmatrix} \quad (19)$$

It remains to compute the right product of the gradient and extrapolation to the Hessian matrix, i.e., $H_k g_k, H_k d_k$, then the 2-D quadratic model can be constructed therein. This frees us from evaluating the full Hessian matrix, which can be costly to do for large problems. Since we only ask for two extra Hessian-vector products compared to the gradient method, the DRSOM is scalable to large-scale problems. In this part, we show this can be done in several ways.

Approximation by Difference A natural strategy following from second-order Lipschitz continuity (see Lemma 1) is to apply finite difference, see, for example, [22, 5]. By Taylor's theorem,

$$\|g(x_k + \epsilon \cdot v) - g_k - \epsilon H_k v\| \leq \frac{M}{2} \epsilon^2 \|v\|^2, \quad (20)$$

we set,

$$H_k \cdot v \approx \frac{1}{\epsilon} [g(x_k + \epsilon \cdot v) - g_k], \quad (21)$$

with an accuracy of $O(\epsilon)$, [22, p. 198]. Using the update formulae (20), we derive two extra gradients at $g(x_k + \epsilon g_k), g(x_k + \epsilon d_k)$, respectively. Finally, Q_k can be computed by further computing the inner products $v^T H_k v, v \in \{g_k, d_k\}$.

Analytic Form and Automatic Differentiation We also note such products can be calculated by further taking the gradients of the inner product, that is,

$$H_k g_k = \nabla(\frac{1}{2} g_k^T g_k), H_k d_k = \nabla(d_k^T g_k), \quad (22)$$

which enables us to emit both the explicit evaluation of Hessian matrix and matrix-vector products when constructing Q_k by capitalizing on modern automatic differentiation schemes.

2.3.2 Solving TRS

Similar to full-dimensional TRS, in our method, the subproblem (11a) can be solved efficiently. We introduce the following Lemma which is widely known

Lemma 4. *The vector α is the global solution to trust-region subproblem (11a) if the α is feasible and there exists Lagrange multiplier $\lambda \geq 0$ such that:*

$$(Q_k + \lambda G_k)\alpha + c_k = 0 \quad (23a)$$

$$Q_k + \lambda G_k \succeq 0 \quad (23b)$$

$$\lambda(\Delta - \|\alpha\|_{G_k}) = 0 \quad (23c)$$

We note that if Q_k is indefinite, by construction of Q_k and G_k :

$$Q_k + \lambda G_k = \begin{bmatrix} -g_k^T \\ d_k^T \end{bmatrix} (H_k + \lambda I) \begin{bmatrix} -g_k & d_k \end{bmatrix}, \quad (24)$$

there always exists λ that is sufficiently large for condition (23b).

If $Q_k \succeq 0$, we check if with unconstrained $\bar{\alpha} = -Q_k^{-1}c_k$, d_{k+1} lies in the interior of the ball; the Lagrange multiplier λ equals to zero if it holds. Otherwise, d_{k+1} must reach the border of the trust region with a positive λ ; the case is similar when Q_k is indefinite. Let $\lambda_{\min}(Q)$ be the leftmost eigenvalue of Q . Clearly, $\lambda \geq -\lambda_{\min}(Q_k)$ is a sufficient condition. This enables us to solve (23) by searching the correct value of λ using a root finding procedure. The choice is quite flexible, for example, see [22, 11, 18, 33] and so forth. We introduce the theorem in Luenberger and Ye [18], where the original proof is from Ye [32].

Theorem 1 (Luenberger and Ye [18], p. 282, Proposition 1). *An ϵ -global primal-dual optimizer (α^*, λ^*) satisfying (23) can be found in $O(\log \log(1/\epsilon))$ time.*

In view of eigenvalues of H_k in the subspace, one can also denote $\|d_{k+1}\|$ as function of λ . We have the following Lemma:

Lemma 5 (Ye [32]).

$$\alpha(\lambda) = - \sum_{j=1}^2 \frac{v_j^T c}{\lambda_j + \lambda} v_j, \quad (25)$$

where $(\lambda_1, v_1), (\lambda_2, v_2)$ are eigen-pairs of H_k in the subspace. If $\lambda \geq -\lambda_1$, $\|d_{k+1}(\lambda)\|$ is a decreasing function in λ .

2.4 Extensions

In this section, we propose two extensions to the original trust-region DRSOM. The first one is the *Radius-Free* DRSOM based on optimality conditions (23). In view of Lemma 5, we know $\|d_{k+1}\|$ is a decreasing function of λ_k . Since the radius for $\|d_{k+1}\|$ is implicitly defined by λ_k , (10) is equivalent to a quadratically regularized model. To this end, we introduce a *Radius-Free* strategy to adjust λ_k along the iterates instead of defining the radius Δ_k . This idea is an analog to cubic regularization, see [21].

Next, we introduce a mini-batch version to work on machine learning problems. The DRSOM in this case, Mini-batch DRSOM, is almost the same except that the quadratic model is computed per batch.

2.4.1 Radius-Free DRSOM

We consider the following problem:

$$\psi_\alpha(\lambda) := \min_{\alpha \in \mathbb{R}^2} f(x_k) + (c_k)^T \alpha + \frac{1}{2} \alpha^T Q_k \alpha + \lambda \|\alpha\|_{G_k}^2 \quad (26)$$

as an analog to trust-region subproblem (11a). At each iteration k , we choose an appropriate λ_k to solve $\psi_\alpha(\lambda_k)$ to permit a reasonable amount of decrease. Specifically, we can also compute an expected ratio of decrease ρ_k by (12), if it is too small, consider increase λ to find solutions with smaller radius. To this end, λ_k is dynamically adjusted by ρ_k just like the conceptual trust-region algorithm Algorithm 1.

Algorithm 2: Radius-Free DRSOM

Data: Given $k_{\max}, \lambda_k > \underline{\lambda}$, and $\eta \in [0, \zeta_1]$;

- 1 **for** $k = 1, \dots, k_{\max}$ **do**
- 2 Solve (26) for α_k , obtain p_k , and ρ_k by (12) ;
- 3 Adjust value of λ_{k+1} ;
- 4 **if** $\rho_k > \eta$ **then**
- 5 $x_{k+1} = x_k + d_{k+1}$
- 6 **else**
- 7 $x_{k+1} = x_k$
- 8 **end**
- 9 **end**

In Algorithm 2, the second-order condition in (23) asks for $Q_k + \lambda_k G_k \succeq 0$, and thus simply increase or decrease λ_k by a multiplier could be invalid. Let $\lambda_1 \leq \lambda_2$ be the eigenvalues of H_k in the subspace, we consider a simple adaptive rule to put λ_k in a desired interval $[\underline{\lambda}_k, \bar{\lambda}_k]$:

$$\underline{\lambda}_k = \max\{0, -\lambda_1\}, \quad \bar{\lambda}_k = \max\{\lambda_k, \lambda_2\} + \lambda_M \quad (27a)$$

$$\gamma_{k+1} = \begin{cases} \beta_2 \gamma_k, & \rho_k \leq \zeta_1 \\ \max\{\underline{\gamma}, \min\{\sqrt{\gamma_k}, \beta_1 \gamma_k\}\}, & \rho_k > \zeta_2 \end{cases} \quad (27b)$$

$$\lambda_k = \gamma_k \cdot \bar{\lambda}_k + \max\{1 - \gamma_k, 0\} \cdot \underline{\lambda}_k \quad (27c)$$

where $\gamma_k > 0$, λ_M is a big number to bound λ_k from above, and $\underline{\gamma}$ is the minimum level for γ_k . Furthermore, we also let $\beta_1 < 1 < \beta_2$ in the same spirit of Algorithm 1. Specifically, we increase γ_k if the model reduction is not accurate according to ρ_k .

In the above procedure (27), we adjust λ_k by γ_k instead, which is expected to be unaffected by the eigenvalues. If γ_k approaches to 0, then λ_k is close to $\underline{\lambda}_k$ which implies \tilde{H}_k is almost positive semi-definite. Otherwise, γ_k induces a large λ_k to give a small trust-region radius.

2.4.2 A mini-batch version of DRSOM

For machine learning problems where the data size can be enormous, we may easily extend the DRSOM by adopting the mini-batch strategy. The difference between the mini-batch version to its original counterpart is that Q_k is computed in a batch-wise way. Formally, we introduce a Mini-batch DRSOM in Algorithm 3, just like before, we may use the radius Δ_k or implicitly the λ_k instead.

Algorithm 3: Mini-batch DRSOM

Data: $X, i_{\max}, k_{\max}, \Delta_0$
Output: θ^*

```

1 for  $i = 1 : i_{\max}$  do
2   Randomly permute data;
3   for  $k = 1 : K$  do
4     Select a batch  $X_k$  of size  $N/K$ ;
5     Compute  $Q_k, c_k$  and solve (11a), obtain  $d_{k+1}$ , and  $\rho_k$  by (12) ;
6     Adjust  $\Delta_k$  (or  $\lambda_k$ ) by  $\rho_k$ ;;
7     if  $\rho_k > \eta$  then
8        $x_{k+1} = x_k + d_{k+1}$ 
9     else
10       $x_{k+1} = x_k$ 
11    end
12  end
13 end

```

3 Convergence Analysis

In this section, we provide a concise convergence analysis of DRSOM.

3.1 Finite convergence for convex quadratic programming

We first show DRSOM has finite convergence for convex quadratic programming.

$$\min f(x) = \frac{1}{2}x^T Ax + a^T x, \quad (28)$$

where $A \succeq 0$. For this case, we do not have to place a trust-region radius for DRSOM, i.e., Δ_k is sufficiently large, $\lambda_k = 0$ for all k . We have the following theorem.

Theorem 2. *If we apply DRSOM to (28) with no radius restriction, i.e., Δ is sufficiently large, then the DRSOM generates the same iterates of conjugate gradient method, if they start at the same point x_0 .*

Proof. To show its equivalence to the conjugate gradient method, we only have to prove the iterate x_k by DRSOM minimizes $f(x)$ in the subspace such that:

$$x_k \in \mathcal{L}_k = x_0 + \text{span}\{d_1, \dots, d_k\}.$$

Since there is no radius, the solution that minimizes m_k strictly corresponds to the optimizer of $f(x)$ in the subspace $x_k + \text{span}\{g_k, d_k\}$. In other words, the iterate of DRSOM can also be retrieved by simply choosing the stepsizes such that $x_{k+1} = \arg \min f(x)$ and $x_{k+1} = x_k - \alpha_k^1 g_k + \alpha_k^2 d_k$. From such a perspective, we show that the x_k is equivalent to \tilde{x}_k by the conjugate gradient method.

Note \tilde{x}_k minimizes $f(x)$ over the subspace below:

$$x_0 + \text{span}\{\tilde{d}_1, \dots, \tilde{d}_k\},$$

where $\tilde{d}_1, \dots, \tilde{d}_k$ are conjugate directions for CG. By construction, we see $x_1 = x_0 + \alpha_0^1 g_0$ and $d_1 = \alpha_0^1 g_0$, so that $x_1 = \tilde{x}_1$. Assume it holds for k , we know for CG:

$$\tilde{g}_k \in \text{span}\{\tilde{d}_k, \tilde{d}_{k+1}\},$$

and since $g_k = \tilde{g}_k$, $d_k = \tilde{d}_k$, we have

$$\text{span}\{\tilde{d}_k, \tilde{d}_{k+1}\} = \text{span}\{d_k, g_k\}, \quad (29)$$

Now we know from the next minimizer $x_{k+1} \in x_k + \text{span}\{d_k, g_k\}$:

$$x_{k+1} \in \tilde{x}_k + \text{span}\{\tilde{d}_k, \tilde{d}_{k+1}\},$$

and since \tilde{x}_{k+1} minimizes f over both $x_0 + \text{span}\{d_1, \dots, d_k, \tilde{d}_{k+1}\}$ and $x_k + \text{span}\{\tilde{d}_{k+1}\}$, we have that $x_{k+1} = \tilde{x}_{k+1}$ as the desired result. \square

For convex QP, other implications for CG can also be established for DRSOM, we refer the readers to any standard textbook, see, for example, [18, 20].

3.2 Complexity analysis for global convergence

Our next concern is the global convergence of the DRSOM. We show that, it has a $O(\epsilon^{-3/2})$ complexity to the first-order stationary point and second-order point in the subspace. We first introduce an assumption used in the analysis.

Assumption 2. *The Hessian matrix \tilde{H}_k along subspace \mathcal{L}_k satisfies:*

$$\|(H_k - \tilde{H}_k)d_{k+1}\| \leq C\|d_{k+1}\|^2 \quad (30)$$

The above assumption is standard in cubic regularized Newton method, see Cartis et al. [6]. In view of Lemma 3, it places a regularity condition such that H_k agrees with \tilde{H}_k along the directions d_{k+1} form by past iterates $\{x_k\}$. We also note that it is possible to allow looser restrictions for second-order methods, for example, see [30]. However, it is beyond the current concern of this paper.

Let us first inspect the properties of the reduction by the quadratic model.

Lemma 6 (Model reduction). *At iteration k , if $\lambda_k > 0$ in (10), it gives the following amount of decrease:*

$$m_k(d_{k+1}) - m_k(0) \leq -\frac{1}{2}\lambda_k\Delta_k^2 \quad (31)$$

Proof. In view of Lemma 3, we write the optimal condition:

$$(\tilde{H}_k + \lambda_k I)d_{k+1} + g_k = 0, \quad \tilde{H}_k + \lambda_k I \succeq 0, \quad (32a)$$

by ignoring the complementary slackness. It follows that:

$$m_k(d_{k+1}) - m_k(0) = g_k^T d_{k+1} + \frac{1}{2}d_{k+1}^T \tilde{H}_k d_{k+1} \quad (33a)$$

$$= -\frac{1}{2}d_{k+1}^T (\tilde{H}_k + \lambda_k I)d_{k+1} - \frac{1}{2}\lambda_k\|d_{k+1}\|^2 \leq -\frac{1}{2}\lambda_k\Delta_k^2 \quad (33b)$$

The proof is complete. \square

With second-order Lipschitz continuity (Lemma 1), take $\beta = \frac{M}{2}$, $\Delta = \frac{\sqrt{\epsilon}}{\beta}$, we immediately have:

$$f(x_{k+1}) \leq f(x_k) + (g_k)^T d_{k+1} + (d_{k+1})^T H_k(d_{k+1}) + \frac{\beta}{3}\|d_{k+1}\|^3 \quad (34a)$$

$$\leq f(x_k) - \frac{1}{2}\lambda_k\Delta^2 + \frac{1}{3}\beta\Delta^3 = f_k - \frac{\lambda_k\epsilon}{2\beta^2} + \frac{\epsilon^{3/2}}{3\beta^2} \quad (34b)$$

The above analysis based on a *fixed* trust-region radius is due to Luenberger and Ye [18]. We now have the following results for the complexity of DRSOM.

Theorem 3. Let Lipschitz condition Lemma 1 and Hessian regularity Assumption 2 hold. Let $\beta = \frac{M}{2}, \Delta_k = \frac{\sqrt{\epsilon}}{\beta}$, the DRSOM terminates in $O(6\beta^2(f(x_0) - f_{\inf})\epsilon^{-3/2})$ iterations. Furthermore, the iterate x_k satisfies the first-order condition, and the Hessian is positive semi-definite in the subspace \mathcal{L}_k :

$$\|g_k\| \leq \epsilon, \tilde{H}_k + \sqrt{\epsilon}I \succeq 0 \quad (35)$$

Proof. Assume that $\lambda_k \geq \sqrt{\epsilon}$, we have,

$$f(x_{k+1}) \leq f(x_k) - \frac{1}{6\beta^2}\epsilon^{3/2}, \quad (36)$$

By summing over k , we have,

$$f_{\inf} \leq f(x_k) \leq f(x_0) - \frac{k}{6\beta^2}\epsilon^{3/2}, \quad (37)$$

it amounts to $k \leq O(6\beta^2(f(x_0) - f_{\inf})\epsilon^{-3/2})$ iterations. We now inspect the norm of gradient. By second-order Lipschitz continuity:

$$\|g_{k+1}\| \leq \|g_{k+1} - g_k - \tilde{H}_k d_{k+1}\| + \|(g_k + \tilde{H}_k d_{k+1})\| \quad (38a)$$

$$\leq \left\| \int_0^1 [H(x_k + \tau d_{k+1}) - \tilde{H}_k] d_{k+1} d\tau \right\| + \mu_k \|d_{k+1}\| \quad (38b)$$

$$\leq \left\| \int_0^1 [H(x_k + \tau d_{k+1}) - H_k] d_{k+1} d\tau \right\| + \|(H_k - \tilde{H}_k)d_{k+1}\| + \mu_k \|d_{k+1}\| \quad (38c)$$

$$\leq \frac{1}{2} M \|d_{k+1}\|^2 + \mu_k \|d_{k+1}\| + \|(H_k - \tilde{H}_k)d_{k+1}\| \quad (38d)$$

In view of Assumption 2, we immediately have,

$$\|g_{k+1}\| \leq \left(\frac{1}{2} M + C \right) \|d_{k+1}\|^2 + \mu_k \|d_{k+1}\| \quad (39a)$$

$$\leq (\beta + C) \frac{\epsilon}{\beta^2} + \frac{\epsilon}{\beta} \quad (39b)$$

$$\leq \frac{2\epsilon}{\beta} + C \frac{\epsilon}{\beta^2} \quad (39c)$$

Finally, as long as $\lambda_k \leq \sqrt{\epsilon}$ is satisfied, again with the second-order optimality condition (32):

$$\tilde{H}_k + \sqrt{\epsilon}I \succeq 0, \quad (40)$$

which indicates the positive semi-definiteness of Hessian H_k in the subspace \mathcal{L}_k . We have the desired results. \square

As a byproduct the above theorem, we also have the following results.

Lemma 7. If $\lambda_k \leq \sqrt{\epsilon}$, and $\beta = \frac{M}{2}, \Delta = \frac{\sqrt{\epsilon}}{\beta}$, we have,

$$\frac{|g_{k+1}^T d_{k+1}|}{\|d_{k+1}\|} \leq \frac{\epsilon}{\beta} \quad (41a)$$

$$\frac{|g_{k+1}^T g_k|}{\|g_k\|} \leq \frac{\epsilon}{\beta} \quad (41b)$$

Proof. We note, since $d_{k+1}^T \tilde{H}_k d_{k+1} = d_{k+1}^T H_k d_{k+1}$,

$$g_{k+1}^T d_{k+1} = (g_{k+1} - g_k)^T d_{k+1} + g_k^T d_{k+1} \quad (42a)$$

$$= \int_0^1 d_{k+1}^T H(x_k + \tau d_{k+1}) d_{k+1} d\tau - d_{k+1}^T (\tilde{H}_k + \lambda_k I) d_{k+1} \quad (42b)$$

$$= \int_0^1 d_{k+1}^T (H(x_k + \tau d_{k+1}) - H(x_k)) d_{k+1} d\tau - \lambda_k \|d_{k+1}\|^2 \quad (42c)$$

we have, from second-order Lipschitz continuity:

$$-\frac{1}{2} M \|d_{k+1}\|^3 - \lambda_k \|d_{k+1}\|^2 \leq g_{k+1}^T d_{k+1} \leq \frac{1}{2} M \|d_{k+1}\|^3 - \lambda_k \|d_{k+1}\|^2 \quad (43a)$$

Since $\lambda_k \leq \sqrt{\epsilon}$,

$$-\frac{2\epsilon^{3/2}}{\beta^2} \leq g_{k+1}^T d_{k+1} \leq \frac{\epsilon^{3/2}}{\beta^2}, \quad (44)$$

such that $\frac{|g_{k+1}^T d_{k+1}|}{\|d_{k+1}\|} \leq O(\epsilon/\beta)$. Furthermore, since (41a) can be rewritten as,

$$\|V_k^T g_{k+1}\| \leq \frac{\epsilon}{\beta}.$$

By construction, it is possible to let the first column of V_k along direction of gradient g_k . It is thus to say,

$$\frac{|g_{k+1}^T g_k|}{\|g_k\|} \leq \|V_k^T g_{k+1}\| \leq \frac{\epsilon}{\beta} \quad (45)$$

So we complete the proof. \square

We have shown that the DRSOM has a complexity of $O(\epsilon^{-3/2})$ to satisfy the first-order condition. By the fact that DRSOM reaches over the usual $O(\epsilon^{-2})$ complexity for first-order methods, paired with its low inner overhead from the Hessian-vector products and cheap two-by-two trust-region subproblems, the DRSOM has a strong advantage over the first-order methods. Compared to the second-order method, it does not require a Hessian approximation, eigenvector direction or Krylov basis, which makes it suitable for large-scale problems. We note that the second-order analysis is weaker than the recent results for trust-region and cubic regularization of the full dimensional second-order methods, see [21, 6, 7, 12, 24]. In theory, the DRSOM only guarantees a second-order stationary point in the subspace spanned by the negative gradient and the momentum. Surprisingly enough, we find this low-dimensional subspace optimality already results in far better solutions in a few types of nonconvex problems than those from the first-order methods.

4 Numerical experiments

In this section, we provide detailed experiments of DRSOM on convex and nonconvex optimization problems. To demonstrate the efficacy of DRSOM, we implement the algorithm using the Julia programming language for convenient comparisons to first- and second-order methods. Most of the experiments, except the neural networks, are handled by the Julia version on a desktop of Mac OS with a 3.2 GHz 6-Core Intel Core i7 processor. The competing algorithms, including the (accelerated) gradient descent

method, LBFGS, and Newton trust-region method, are computed via a third-party package `Optim.jl`¹, including a set of line search algorithms in `LineSearches.jl`².

We also implement a version in PyTorch that enables experiments in neural network training. For this part, the DRSOM runs on a Ubuntu desktop with Intel Xeon CPU E5-2698 v4 processor and 1 NVIDIA Tesla V100. We provide a comparison of SGD and Adam. Note the SGD and Adam optimizer used in our experiments is provided by the official implementation of PyTorch³. The computational results can be divided into two parts accordingly.

- We first run DRSOM on a multinomial logistic regression model as an example of convex problems. Next, we provide results on two nonconvex problems: the $\mathcal{L}_2 - \mathcal{L}_p$ minimization and the sensor network localization.
- The rest of the experiments focus on training neural networks.

4.1 Logistic Regression

We consider a multinomial logistic regression model for the MNIST dataset. The training set contains 60,000 pictures for handwritten digits; the test set has 10,000 pictures. We present the classification performance of DRSOM for 10 and 40 epochs in comparison to a popular stochastic first-order method, SAGA, and a second-order method, LBFGS. Specifically, we run DRSOM and LBFGS in full-batch. Then, we collect the zero-one classification loss of training (training error) and test data (test error) in Table 1. These results show that DRSOM is comparable to SAGA and LBFGS

Epoch	Method	Training error	Testing error
10	SAGA	0.0699	0.0779
10	LBFGS	0.1245	0.1175
10	DRSOM	0.1149	0.1076
40	SAGA	0.0690	0.0759
40	LBFGS	0.0750	0.0783
40	DRSOM	0.0760	0.0819

Table 1: Performance of DRSOM on MNIST classification compared to other algorithms

4.2 $\mathcal{L}_2 - \mathcal{L}_p$ Minimization

We next test the performance of DRSOM for nonconvex $\mathcal{L}_2 - \mathcal{L}_p$ minimization. Recall $\mathcal{L}_2 - \mathcal{L}_p$ minimization problem (Ge et al. [14], Chen et al. [10], Chen [9]):

$$\phi^* = \min_{x \in \mathbb{R}^m} \phi(x) = \frac{1}{2} \|Ax - b\|_2^2 + \lambda \|x\|_p^p, \quad (46)$$

¹For details, see <https://github.com/JuliaNLSSolvers/Optim.jl>

²For details, see <https://github.com/JuliaNLSSolvers/LineSearches.jl>

³For details, see <https://pytorch.org/docs/stable/generated/torch.optim.Adam.html>

where $A \in \mathbb{R}^{n \times m}$, $b \in \mathbb{R}^n$, $0 < p < 1$. To overcome nonsmoothness of $\|\cdot\|_p$, we apply the smoothing strategy mentioned in [9]:

$$f(x) = \|Ax - b\|_2^2 + \lambda \sum_{i=1}^n s(x_i, \varepsilon)^p, \quad (47)$$

where $s(x_i, \varepsilon)$ is a smoothed approximation of $|x_i|$ and ε is a small pre-defined constant $\varepsilon > 0$:

$$s(x, \varepsilon) = \begin{cases} |x| & \text{if } |x| > \varepsilon \\ \frac{x^2}{2\varepsilon} + \frac{\varepsilon}{2} & \text{if } |x| \leq \varepsilon \end{cases} \quad (48)$$

We randomly generate datasets with different sizes n, m based on the following procedure. The elements of matrix A are generated by $A_{ij} \sim \mathcal{N}(0, 1)$ with 15% sparsity of 15%. To construct the true sparse vector $v \in \mathbb{R}^m$, we let for all i :

$$v_i \sim \begin{cases} 0 & \text{with probability } p = 0.5 \\ \mathcal{N}(0, \frac{1}{n}) & \text{otherwise} \end{cases}$$

Then we let $b = Av + \delta$ where δ is the noise generated as $\delta_i \sim \mathcal{N}(0, 1), \forall i$. The parameter λ is chosen as $\frac{1}{5}\|A^T b\|_\infty$. We generate instances for (n, m) from $(100, 10)$ to $(1000, 100)$. We set $p = 0.5$ and the smoothing parameter $\varepsilon = 1e^{-1}$.

Next, we test the performance of DRSOM and competing algorithms, including a first-order representative AGD, and two second-order methods, including LBFGS and the Newton trust-region method (Newton-TR). The AGD is facilitated with the Zhang-Hager line-search algorithm (see [34]). We report the iteration number needed to reach a first-order stationary point at a precision of $1e^{-6}$, precisely,

$$|\nabla f(x_k)| \leq \epsilon := 1e^{-6}$$

The iterations needed for a set of methods are reported in the Table 2. These results

n	m	DRSOM	AGD	LBFGS	Newton TR
100	10	18	43	14	6
100	20	31	72	23	7
100	100	47	136	42	10
200	10	21	27	15	5
200	20	23	45	21	6
200	100	40	131	39	9
1000	10	13	16	9	4
1000	20	16	23	13	5
1000	100	19	32	16	5

Table 2: Performance of DRSOM on (46) compared to other algorithms: iterations needed for precision $\epsilon = 1e^{-6}$

show that the DRSOM is fairly close to the full-dimensional second-order methods, especially the original Newton trust-region method; it is far better than AGD in most test cases.

4.3 Sensor Network Localization

We next visit another nonconvex optimization problem, namely, the Sensor Network Localization (SNL). The SNL problem is to find coordinates of ad hoc wireless sensors given pairwise distances in the network. Fruitful research has been found for this problem, among which the approach based on Semidefinite Programming Relaxation (SDR) has witnessed great success; see, for example, Biswas and Ye [2], Wang et al. [28], and many others. We here use the notations in [28].

Let n sensors be points in \mathbb{R}^d , besides, assume another set of m *known* points (usually referred to as anchors) whose exact positions are a_1, \dots, a_m . Let d_{ij} be the distance between sensor i and j , and \bar{d}_{ik} be the distance from the sensor i to anchor point k . We can then define the set of distances as edges in the network:

$$N_x = \{(i, j) : \|x_i - x_j\| = d_{ij} \leq r_d\}, N_a = \{(i, k) : \|x_i - a_k\| = \bar{d}_{ik} \leq r_d\}, \quad (49)$$

where r_d is a fixed parameter known as the *radio range*. The SNL problem considers the following quadratic constrained quadratic programming (QCQP) feasibility problem,

$$\|x_i - x_j\|^2 = d_{ij}^2, \forall (i, j) \in N_x \quad (50a)$$

$$\|x_i - a_k\|^2 = \bar{d}_{ik}^2, \forall (i, k) \in N_a \quad (50b)$$

Since the problem is nonconvex, the SDR approaches the above problem by a two-stage strategy. In the first step, we use semidefinite programming to solve a lifted convex relaxation:

$$\begin{aligned} & \min \quad 0 \bullet Z \\ & \text{s.t. } Z_{[1:2,1:2]} = I, \\ & \quad (0; e_i - e_j) (0; e_i - e_j)^T \bullet Z = d_{ij}^2 \quad \forall (i, j) \in N_x, \\ & \quad (-a_k; e_i) (-a_k; e_i)^T \bullet Z = \bar{d}_{ik}^2 \quad \forall (i, k) \in N_a \\ & \quad Z \succeq 0. \end{aligned} \quad (51)$$

We let I be the identity matrix of dimension 2, e_i be a n -vector of zeros except for a one at i -th entry. Z is the positive semidefinite matrix, such that,

$$Z = \begin{bmatrix} I & X \\ X^T & Y \end{bmatrix}, \quad (52)$$

which is equivalent to state $Y \succeq X^T X$. If $\text{rank}(Y) = 2$, the SDR solves the original problem; otherwise (Y, X) provides initial solution and needs further refinement. For example, at the second stage, we can solve the following nonlinear least-square problem (NLS):

$$\min_X \sum_{(i < j, j) \in N_x} (\|x_i - x_j\|^2 - d_{ij}^2)^2 + \sum_{(k, j) \in N_a} (\|a_k - x_j\|^2 - \bar{d}_{kj}^2)^2. \quad (53)$$

We may find local solutions by a gradient descent method (GD); see, for example, [2]. Alternatively, we apply the DRSOM to the sensor network localization problem. We here provide a randomly generated example with 80 points, 5 of which are anchors. We set the radio range to 0.5 and the random distance noise $n_f = 0.05$. We add a line search for GD that guarantees the strong Wolfe condition, see [22, p. 60]. We terminate at an iterate x_k if $\|g(x_k)\| \leq 1e^{-6}$.

Our results basically show that: if we initialize the NLS problem (53) for SNL by the SDR (51), then DRSOM and GD are comparable. However, if we do not have the SDR solution at hand, the DRSOM may usually provide better solutions than GD, which shows the benefit of the second-order optimality condition.

Figure 1 illustrates the realization results of GD and DRSOM with the SDR initialization. In this case, both algorithms are able to guarantee convergence to the ground truth.

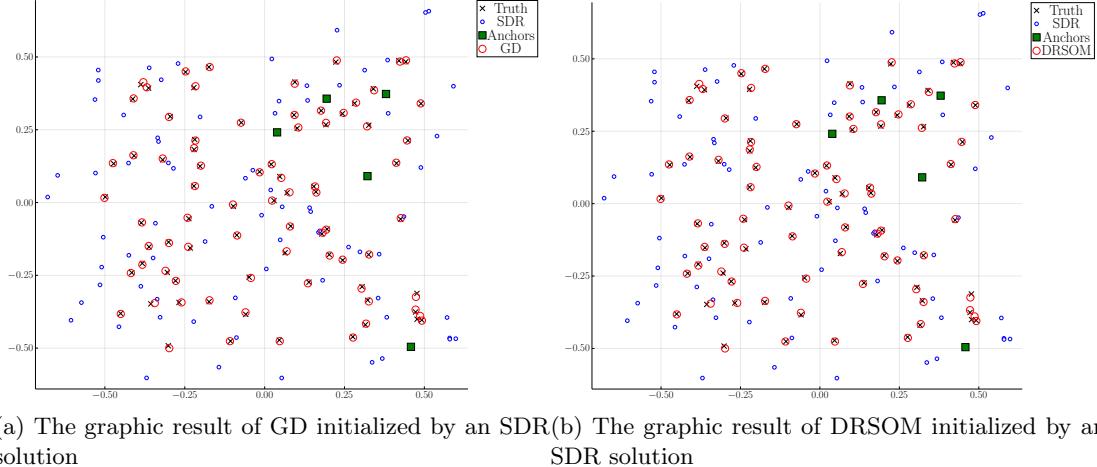


Figure 1: Comparison of localization by GD and DRSOM with SDR initialization. The rectangles and crosses represent the anchors and true locations, respectively. The blue circles are solutions by SDR, and the red circles are final solutions of GD/DRSOM.

As a comparison, Figure 2 depicts the case without solving SDR first. We use the same parameter settings as for the previous case. The GD and DRSOM are initialized by $X_i := 0, i = 1, \dots, n$. In this particular case, the GD fails to recover true positions; to our experience, it converges to a strict local minimum x^* such that $H(x^*) \succ 0$. However, the DRSOM can sometimes provide accurate solutions even without the SDR initialization. In this example, we rigorously provide a case where the DRSOM may

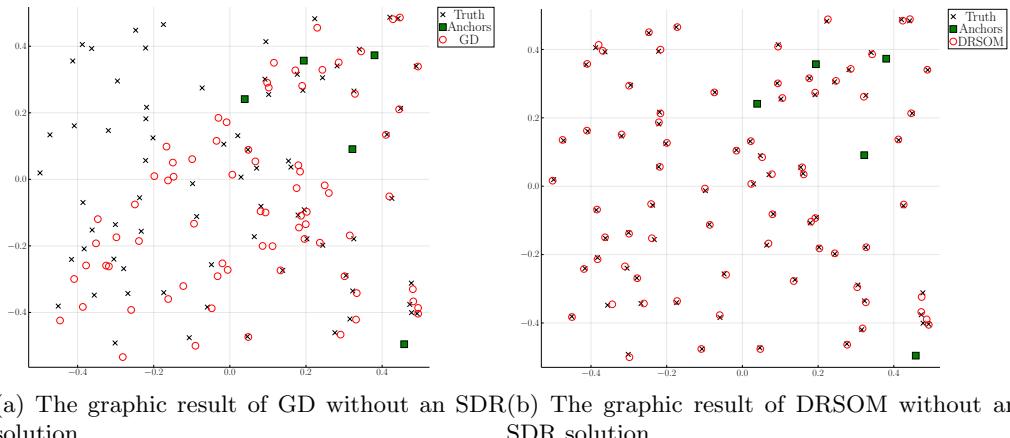


Figure 2: Comparison of localization by GD and DRSOM without SDR initialization. The meaning of each symbol is the same as Figure 1.

result in better local results (in this case, the global one) than the first-order methods; despite that, we only have optimal subspace guarantees in theory.

4.4 Neural Networks

In this section, we implement a vanilla Mini-Batch DRSOM to train neural networks. Our implementation is straightforward: for each mini-batch, we calculate the required gradients and Hessian-vector products, then the computation proceeds just like the “full-batch” version. Finally, we test our Mini-Batch DRSOM optimizer and compare the performance with the SGD and Adam optimizer.

4.4.1 Fashion-MNIST

We train a Neural Network model for classification on Fashion-MNIST, consisting of a training set of 60,000 examples and a test set of 10,000 examples [29]. The dataset is constructed from images with one of 10 labels, including **T-shirt/top**, **bag**, **dress** and so on⁴. In our test, we adopt a neural network model with two convolutional and two fully-connected layers⁵, which has about 16.84 million parameters. For SGD, we test on different momentum coefficients $\mu \in \{0.85, 0.9, 0.95, 0.99\}$ named after SGD- μ . Also, the Adam and SGD are set with the learning rate at $1e^{-3}$. All optimizers are tested with a batch size of 128.

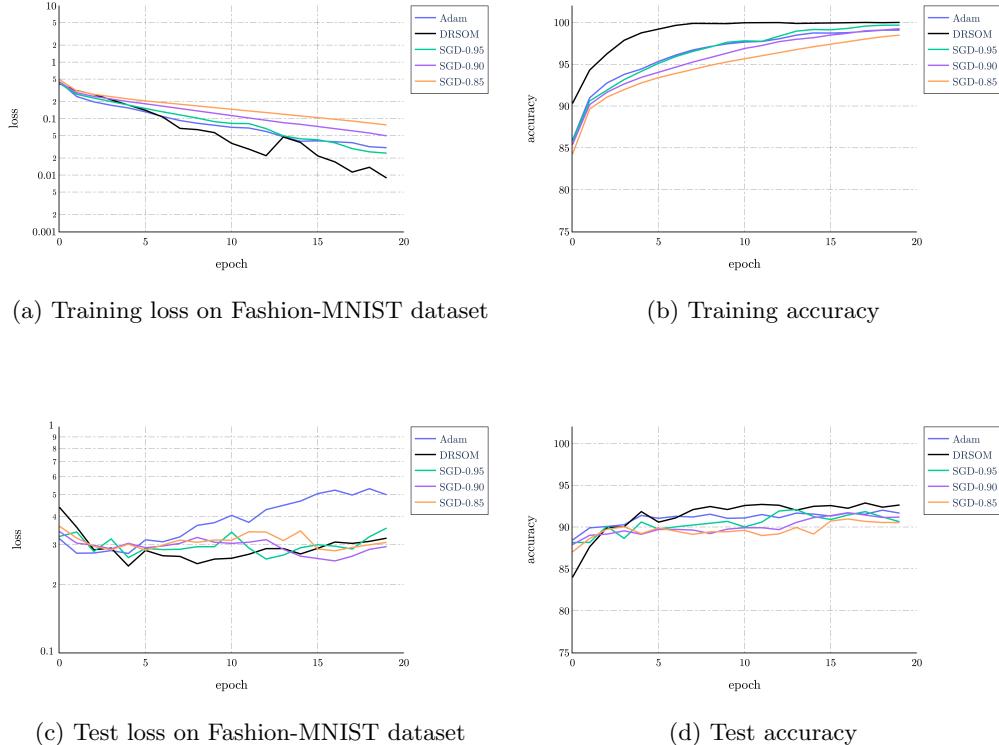


Figure 3: Training and test results of Adam, SGD, and DRSOM for a Neural network on Fashion-MNIST dataset

⁴For details, see <https://github.com/zalandoresearch/fashion-mnist>

⁵For details, see <https://github.com/ashmeet13/FashionMNIST-CNN/blob/master/Fashion.py>

From Figure 3, we see that in 20 epochs, the Adam, SGD ($\text{SGD}-\mu$), and DRSOM are both able to reach over 90% accuracy on the hold-out test set. The DRSOM, even with a naive Mini-batch strategy, is the best in terms of loss and accuracy in the training set. It also has the best test accuracy compared to the competitors. In our preliminary experiments, the DRSOM is two times slower than Adam per-iteration in running this case.

4.4.2 CIFAR10

To further take a look in *deep* neural networks, we also train a ResNet18 model (He et al. [15]) for CIFAR10. In our preliminary experiments, the vanilla DRSOM is five times slower than Adam in running time at the same iteration number; thus, we only run DRSOM in 50 epochs. For Adam, we collect the results in 100 epochs. All optimizers are tested with a batch size of 128. For Adam, we provide a learning rate scheduler to decay the learning rate by a factor of 2 in every k epochs started at an initial rate $1e^{-3}$; we name these variants by Adam- k where $k \in \{30, 40\}$.

To enable fair comparison, for DRSOM, we increase the lower bound of $\underline{\gamma}$ by the rule, $\underline{\gamma} := \underline{\gamma} \cdot \sigma, \sigma \in \{100, 1000\}$, in every 10 epochs corresponding to the update policy (27). We apply the strategy to mimic a mechanism of reducing the learning rate. Using the same fashion, we call them DRSOM- σ . We report the results in Figure 4. As the

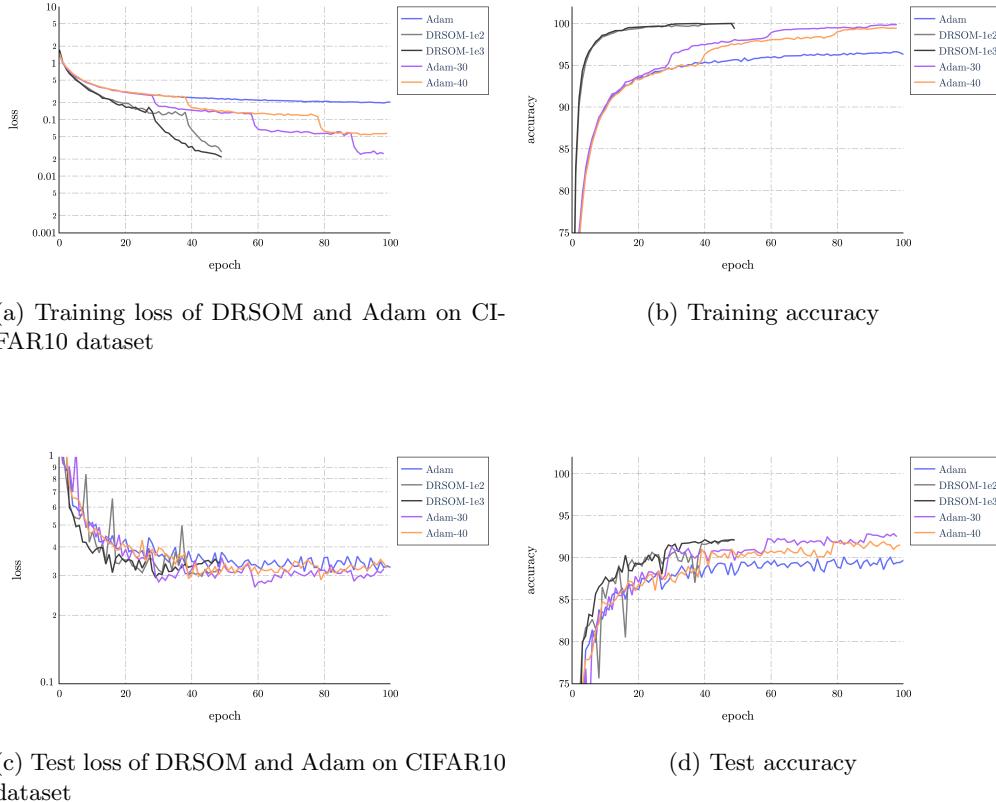


Figure 4: Training and test results of Adam and DRSOM for ResNet18 on CIFAR10

results show, the DRSOM has a sharp rate of increase in the beginning; with little tuning efforts, DRSOM- $1e^3$ (in 50 epochs) has competitive results to Adam in 100

epochs. These preliminary results, to our belief, motivate future research and better implementation of the DRSOM.

5 Discussion

In this paper, we introduce a Dimension-Reduced Second-order Method (DRSOM) that utilizes the sequential quadratic approximation in a low-dimensional subspace. Our method differs from existing second-order methods that require full-dimensional exact steps or iterative inexact steps. Notably, we solve easy-to-construct low-dimensional trust-region subproblems — a two-dimensional quadratic model — and require only the gradient, momentum, and two extra Hessian-vector products that cost almost the same as first-order methods.

We provide a concise analysis to show that our method reduces to the conjugate gradient method for convex quadratic programming. Then, we prove that the DRSOM converges to $\|g_k\| \leq \epsilon$ in $O(\epsilon^{-3/2})$ iterations under a common assumption in nonlinear optimization literature. Furthermore, at the same computational complexity, the iterate x_k has a Hessian that is $\sqrt{\epsilon}$ -positive semidefinite in the subspace, that is, $\tilde{H}_k + \sqrt{\epsilon}I \succeq 0$.

Computationally, we provide examples in convex and nonconvex optimization, and a preliminary implementation in deep learning. For logistic regression, $\mathcal{L}_2 - \mathcal{L}_p$ minimization, sensor network localization, the DRSOM outperforms the first-order methods. The low iteration complexity of DRSOM is exhibited by the similar performance to that of full-dimensional second-order methods in test cases of $\mathcal{L}_2 - \mathcal{L}_p$ minimization. As a spotlight of the DRSOM, we show that second-order optimality in the subspace sometimes implies better solutions on a specific example in sensor network localization. At last, we provide results on a mini-batch version to tackle neural network tasks. For problems like Fashion-MNIST classification, the DRSOM outperforms first-order stochastic methods, including the SGD and Adam. For more complex models like the Resnet18, the DRSOM performance for the same epoch count still shows a substantial superiority. However, the weakness of mini-batch DRSOM in running time could be worse. The reason is that the extra function and gradient evaluations need for DRSOM become time-consuming. Addressing these issues opens the door for future research on DRSOM with enhancements tailored for deep learning.

Acknowledgement

The authors would like to thank Tianyi Lin for the insightful comments that significantly improve the presentation of this paper.

References

- [1] Naman Agarwal, Zeyuan Allen-Zhu, Brian Bullins, Elad Hazan, and Tengyu Ma. Finding approximate local minima faster than gradient descent. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1195–1199, 2017.
- [2] Pratik Biswas and Yinyu Ye. Semidefinite programming for ad hoc wireless sensor network localization. In *Third International Symposium on Information Processing*

- in Sensor Networks, IPSN 2004*, pages 46–54, 2004. ISBN 1-58113-846-6. doi: 10.1145/984622.984630.
- [3] Raghu Bollapragada, Jorge Nocedal, Dheevatsa Mudigere, Hao-Jun Shi, and Ping Tak Peter Tang. A progressive batching L-BFGS method for machine learning. In *International Conference on Machine Learning*, pages 620–629. PMLR, 2018.
 - [4] Richard H. Byrd, Robert B. Schnabel, and Gerald A. Shultz. Approximate solution of the trust region problem by minimization over two-dimensional subspaces. *Mathematical Programming*, 40(1):247–263, January 1988. ISSN 1436-4646. doi: 10.1007/BF01580735. URL <https://doi.org/10.1007/BF01580735>.
 - [5] Yair Carmon, John C. Duchi, Oliver Hinder, and Aaron Sidford. Accelerated Methods for NonConvex Optimization. *SIAM Journal on Optimization*, 28(2):1751–1772, January 2018. ISSN 1052-6234, 1095-7189. doi: 10.1137/17M1114296. URL <https://pubs.siam.org/doi/10.1137/17M1114296>.
 - [6] Coralia Cartis, Nicholas I. M. Gould, and Philippe L. Toint. Adaptive cubic regularisation methods for unconstrained optimization. Part II: worst-case function-and derivative-evaluation complexity. *Mathematical Programming*, 130(2):295–319, December 2011. ISSN 0025-5610, 1436-4646. doi: 10.1007/s10107-009-0337-y. URL <http://link.springer.com/10.1007/s10107-009-0337-y>.
 - [7] Coralia Cartis, Nicholas I. M. Gould, and Philippe L. Toint. Adaptive cubic regularisation methods for unconstrained optimization. Part I: motivation, convergence and numerical results. *Mathematical Programming*, 127(2):245–295, April 2011. ISSN 0025-5610, 1436-4646. doi: 10.1007/s10107-009-0286-5. URL <http://link.springer.com/10.1007/s10107-009-0286-5>.
 - [8] Xi Chen, Bo Jiang, Tianyi Lin, and Shuzhong Zhang. Accelerating adaptive cubic regularization of Newton’s method via random sampling. *J. Mach. Learn. Res*, 2022.
 - [9] Xiaojun Chen. Smoothing methods for nonsmooth, nonconvex minimization. *Mathematical Programming*, 134(1):71–99, August 2012. ISSN 1436-4646. doi: 10.1007/s10107-012-0569-0. URL <https://doi.org/10.1007/s10107-012-0569-0>.
 - [10] Xiaojun Chen, Dongdong Ge, Zizhuo Wang, and Yinyu Ye. Complexity of unconstrained $L_2 - L_p$ minimization. *Mathematical Programming*, 143(1-2):371–383, February 2014. ISSN 0025-5610, 1436-4646. doi: 10.1007/s10107-012-0613-0. URL <http://link.springer.com/10.1007/s10107-012-0613-0>.
 - [11] Andrew R. Conn, Nicholas IM Gould, and Philippe L. Toint. *Trust region methods*. SIAM, 2000.
 - [12] Frank E. Curtis, Daniel P. Robinson, and Mohammadreza Samadi. A trust region algorithm with a worst-case iteration complexity of $\mathcal{O}(\epsilon^{-3/2})$ for nonconvex optimization. *Mathematical Programming*, 162(1):1–32, March 2017. ISSN 1436-4646. doi: 10.1007/s10107-016-1026-2. URL <https://doi.org/10.1007/s10107-016-1026-2>.
 - [13] Y. H. Dai and Y. Yuan. A Nonlinear Conjugate Gradient Method with a Strong Global Convergence Property. *SIAM Journal on Optimization*, 10(1):177–182,

- January 1999. ISSN 1052-6234. doi: 10.1137/S1052623497318992. URL <https://epubs.siam.org/doi/abs/10.1137/S1052623497318992>. Publisher: Society for Industrial and Applied Mathematics.
- [14] Dongdong Ge, Xiaoye Jiang, and Yinyu Ye. A note on the complexity of L_p minimization. *Mathematical Programming*, 129(2):285–299, 2011. doi: 10.1007/s10107-011-0470-2.
 - [15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
 - [16] Magnus R. Hestenes and Eduard Stiefel. Methods of conjugate gradients for solving. *Journal of research of the National Bureau of Standards*, 49(6):409, 1952.
 - [17] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization, January 2017. URL <http://arxiv.org/abs/1412.6980>. arXiv:1412.6980 [cs].
 - [18] David G. Luenberger and Yinyu Ye. *Linear and Nonlinear Programming*, volume 228 of *International Series in Operations Research & Management Science*. Springer International Publishing, Cham, 2021. ISBN 978-3-030-85449-2 978-3-030-85450-8. doi: 10.1007/978-3-030-85450-8. URL <https://link.springer.com/10.1007/978-3-030-85450-8>.
 - [19] James Martens. Deep learning via hessian-free optimization. In *ICML*, volume 27, pages 735–742, 2010.
 - [20] Yurii Nesterov. *Lectures on convex optimization*, volume 137. Springer, 2018.
 - [21] Yurii Nesterov and B.T. Polyak. Cubic regularization of Newton method and its global performance. *Mathematical Programming*, 108(1):177–205, August 2006. ISSN 1436-4646. doi: 10.1007/s10107-006-0706-8. URL <https://doi.org/10.1007/s10107-006-0706-8>.
 - [22] Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.
 - [23] B. T. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, January 1964. ISSN 0041-5553. doi: 10.1016/0041-5553(64)90137-5. URL <https://www.sciencedirect.com/science/article/pii/0041555364901375>.
 - [24] Clément W. Royer and Stephen J. Wright. Complexity analysis of second-order line-search algorithms for smooth nonconvex optimization. *SIAM Journal on Optimization*, 28(2):1448–1477, 2018. Publisher: SIAM.
 - [25] Gerald A. Shultz, Robert B. Schnabel, and Richard H. Byrd. A Family of Trust-Region-Based Algorithms for Unconstrained Minimization with Strong Global Convergence Properties. *SIAM Journal on Numerical Analysis*, 22(1):47–67, February 1985. ISSN 0036-1429. doi: 10.1137/0722003. URL <https://epubs.siam.org/doi/abs/10.1137/0722003>. Publisher: Society for Industrial and Applied Mathematics.

- [26] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147. PMLR, 2013.
- [27] Nilesh Tripuraneni, Mitchell Stern, Chi Jin, Jeffrey Regier, and Michael I Jordan. Stochastic Cubic Regularization for Fast Nonconvex Optimization. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL <https://proceedings.neurips.cc/paper/2018/hash/db1915052d15f7815c8b88e879465a1e-Abstract.html>.
- [28] Zizhuo Wang, Song Zheng, Yinyu Ye, and Stephen Boyd. Further relaxations of the semidefinite programming approach to sensor network localization. *SIAM Journal on Optimization*, 19(2):655–673, 2008. Publisher: SIAM.
- [29] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms, August 2017. arXiv: cs.LG/1708.07747.
- [30] Peng Xu, Fred Roosta, and Michael W. Mahoney. Newton-type methods for non-convex optimization under inexact Hessian information. *Mathematical Programming*, 184(1-2):35–70, November 2020. ISSN 0025-5610, 1436-4646. doi: 10.1007/s10107-019-01405-z. URL <http://link.springer.com/10.1007/s10107-019-01405-z>.
- [31] Yinyu Ye. Second Order Optimization Algorithms I. URL <https://web.stanford.edu/class/msande311/lecture12.pdf>.
- [32] Yinyu Ye. A New Complexity Result on Minimization of a Quadratic Function with a Sphere Constraint. In *Recent Advances in Global Optimization*, volume 176, pages 19–31. Princeton University Press, 1991. ISBN 978-1-4008-6252-8. doi: 10.1515/9781400862528.19. URL <https://www.degruyter.com/document/doi/10.1515/9781400862528.19/html>.
- [33] Yinyu Ye. On the complexity of approximating a KKT point of quadratic programming. *Mathematical Programming*, 80(2):195–211, January 1998. ISSN 1436-4646. doi: 10.1007/BF01581726. URL <https://doi.org/10.1007/BF01581726>.
- [34] Hongchao Zhang and William W. Hager. A nonmonotone line search technique and its application to unconstrained optimization. *SIAM journal on Optimization*, 14(4):1043–1056, 2004. Publisher: SIAM.