Scalable Second-Order Optimization Algorithms for Minimizing Low-rank Functions

Edward Tansley TANSLEY@MATHS.OX.AC.UK
Coralia Cartis CARTIS@MATHS.OX.AC.UK
Mathematical Institute, University of Oxford

Abstract

We present a random-subspace variant of cubic regularization algorithm that chooses the size of the subspace adaptively, based on the rank of the projected second derivative matrix. Iteratively, our variant only requires access to (small-dimensional) projections of first- and second-order problem derivatives and calculates a reduced step inexpensively. The ensuing method maintains the optimal global rate of convergence of (full-dimensional) cubic regularization, while showing improved scalability both theoretically and numerically, particularly when applied to low-rank functions. When applied to the latter, our algorithm naturally adapts the subspace size to the true rank of the function, without knowing it a priori.

1. Introduction

Second-order optimization algorithms for the unconstrained optimization problem

$$\min_{x\in\mathbb{R}^d} f(x),$$

where $f : \mathbb{R}^d \to \mathbb{R}$ is a sufficiently smooth, bounded-below function, use gradient and curvature information to determine iterates and so often experience faster convergence than first-order algorithms that only rely on gradient information. However, for high-dimensional problems, the computational complexity of these methods can be a barrier to their use in practice. We are concerned with the task of scaling up second-order optimization algorithms so that they are a practical option for high-dimensional problems.

A second-order algorithm designed to cope with high-dimensional problems is the R-ARC algorithm [14, 15], a random subspace variant of the Adaptive Regularization using Cubics (ARC) algorithm [3]. Subject to certain conditions on the random subspaces, R-ARC can attain the same convergence rate to an ϵ-approximate first-order minimizer as ARC. These conditions imply that R-ARC is particularly effective for functions with Hessians of rank bounded by some r (significantly) lower than the function dimension d. A class of functions with this property are *low-rank functions* [16], which have been frequently studied in the context of machine learning.

ARC and R-ARC ARC [3, 11] is an iterative algorithm that at iteration k, determines the step s_k by (approximately) solving the following local model[1]:

$$\arg\min_{s\in\mathbb{R}^d} m_k(s) = f(x_k) + \langle \nabla f(x_k),\ s\rangle + \frac{1}{2}\langle s,\ \nabla^2 f(x_k)s\rangle + \frac{\sigma_k}{3}\|s\|_2^3$$

1. Here $\sigma_k/3$ replaces $L/6$ in the well-known bound $f(x_k+s) \le f(x_k) + \langle \nabla f(x_k),\ s\rangle + \frac{1}{2}\langle s,\ \nabla^2 f(x_k)s\rangle + \frac{\sigma_k}{3}\|s\|_2^3$ (where L is the Lipschitz constant of $\nabla^2 f$).

© E. Tansley & C. Cartis.

where ∇f and $\nabla^2 f$ denote the gradient and Hessian of f, x_k is the current iterate and σ_k is the regularization parameter. Assuming Lipschitz continuity of the Hessian on the iterates' path, ARC requires at most $\mathcal{O}(\epsilon^{-3/2})$ iterations to attain an ϵ-approximate first-order minimizer; this convergence rate is optimal over a large class of second-order methods [6].

A random subspace variant of ARC, R-ARC was introduced in [14, 15]. At each iteration k, a random sketching matrix $S_k \in \mathbb{R}^{l \times d}$ is drawn from a distribution $\mathcal{S}$ and the search space for $s_k \in \mathbb{R}^d$ is restricted so the l-dimensional subspace $\mathrm{span}(S_k^\top)$. These papers prove that assuming certain embedding conditions on $\mathcal{S}$, the optimal $\mathcal{O}(\epsilon^{-3/2})$ iteration complexity can be attained by ARC, despite only accessing projected (first- and second-order) problem information at each iteration. We now state an informal version of this result, restricted to Gaussian matrices.

Theorem 1 (Informal, [14, 15]) *Suppose that $\mathcal{S}$ is the distribution of (scaled) $l \times d$ Gaussian matrices with $l = \mathcal{O}(r+1)$, where $r \leq d$ is an upper bound on the maximum rank of $\nabla^2 f(x_k)$ across all iterations, and that f has globally Lipschitz continuous second derivatives. Then R-ARC achieves the optimal $\mathcal{O}(\epsilon^{-3/2})$ rate of convergence, with high probability.*

The proof of this Theorem relies upon $\mathcal{S}$ being an oblivious subspace embedding (Definition 8) for matrices with rank $r+1$. Similar results can be established for matrix distributions other than Gaussian, with l possibly having a different dependency on r [4]. Theorem 1 can be applied to any suitable objective function f. However, the requirement that $l = \mathcal{O}(r+1)$ means that unless $r \ll d$, R-ARC is not guaranteed to be able to gain a significant dimensionality over ARC (by using only little problem information and computing an inexpensive reduced step), whilst maintaining the $\mathcal{O}(\epsilon^{-3/2})$ convergence rate.

In R-ARC, the sketch dimension l is fixed throughout the run of the algorithm. Theorem 1 requires l to be proportional to a bound r on the maximal Hessian rank at the iterates, but this may not be known a priori. This motivates us to develop a variant of R-ARC that can adapt the sketch/subspace size to local problem information.

Low-rank functions We now define a class of functions that particularly benefit from random subspace algorithms. These functions are also known as functions with *low effective dimensionality*, with *active subspaces* or *multi-ridge* functions [2, 5].

Definition 2 (Low-rank Functions [16]) *A function $f : \mathbb{R}^d \to \mathbb{R}$ is said to be of rank r, with $r \leq d$ if*

- *there exists a linear subspace $\mathcal{T}$ of dimension r such that for all $x_\top \in \mathcal{T} \subset \mathbb{R}^d$ and $x_\perp \in \mathcal{T}^\perp \subset \mathbb{R}^d$, we have $f(x_\top + x_\perp) = f(x_\top)$, where $\mathcal{T}^\perp$ is the orthogonal complement of $\mathcal{T}$;*
- *r is the smallest integer with this property*

We call $\mathcal{T}$ the effective subspace of f and $\mathcal{T}^\perp$ the constant subspace of f.

We state a lemma whose proof is included in Appendix A and which then helps us apply the results in [14, 15] to low-rank functions (note the requirement on a bound on the $\mathrm{rank}(\nabla^2 f(x_k))$ in Theorem 1).

Lemma 3 *If $f : \mathbb{R}^d \to \mathbb{R}$ is a low-rank function of rank r, and f is C^2, then for all $x \in \mathbb{R}^d$, $\nabla^2 f(x)$ has rank at most r.*

Overparameterized models in various applications are candidates for low-rank behaviour as we expect invariance to some reparameterization; provided such invariance is (approximately) linear. In deep neural networks, there are two sources of low-rank behaviour: the training loss as a function of parameters [7], and the trained net as a function of the input data [12]. Further, in hyperparameter optimization, low-rank behaviour is observed as network performance only depends upon a selection of hyperparameters [1].

Contributions We introduce R-ARC-D, a new variant of the R-ARC algorithm that can vary the size of the random susbspace between iterations. We detail an update scheme for the sketch size that adapts to the local Hessian rank of the iterates. This algorithm attains the optimal $\mathcal{O}(\epsilon^{-3/2})$ iteration complexity for an ϵ-approximate first-order minimizer, whilst maintaining a sketch dimension l_k that is $\mathcal{O}(r)$ for functions with Hessians with rank bounded by r, in particular low-rank functions, despite the algorithm not needing to know r a priori. Through numerical experiments on low-rank problems, we demonstrate the superior efficiency of this algorithm compared to the R-ARC and ARC algorithms on these problems.

2. Algorithm and Main Results

In this section, we present the R-ARC-D algorithm and conditions under which it can attain the optimal $\mathcal{O}(\epsilon^{-3/2})$ iteration complexity for finding an ϵ-approximate first-order local minimizer. R-ARC-D differs from the R-ARC algorithm presented by [14, 15] as it allows for the sketch size to be iteratively adjusted.

Algorithm 1: Random subspace cubic regularisation algorithm with variable sketching dimension (R-ARC-D)

Choose constants $\theta \in (0,1)$, $\kappa_T, \kappa_S \geq 0$. Initialize the algorithm by setting $x_0 \in \mathbb{R}^d$ and $k = 0$.
Draw a random matrix $S_k \in \mathbb{R}^{l_k \times d}$ from $\mathcal{S}$, and let

$$\hat{m}_k(\hat{s}) = \underbrace{f(x_k) + \langle \hat{\nabla} f(x_k),\ \hat{s}\rangle + \frac{1}{2}\langle \hat{s},\ \hat{\nabla}^2 f(x_k)\hat{s}\rangle}_{\hat{q}_k(\hat{s})} + \frac{\sigma_k}{3}\|S_k^\top \hat{s}\|_2^3, \tag{1}$$

where $\hat{\nabla} f(x_k) = S_k \nabla f(x_k)$ and $\hat{\nabla}^2 f(x_k) = S_k \nabla^2 f(x_k) S_k^\top$.

Compute $\hat{s}_k \in \mathbb{R}^{l_k}$ by approximately minimizing $\hat{m}$ such that

$$\hat{m}(\hat{s}_k) \leq \hat{m}(0); \quad \|\nabla \hat{m}(\hat{s}_k)\|_2 \leq \kappa_T \|S_k^\top \hat{s}\|_2^2; \quad \nabla^2 \hat{m}_k(\hat{s}_k) \succeq -\kappa_S \|S_k^\top \hat{s}\|_2.$$

Compute a trial step $s_k = S_k^T \hat{s}_k$.
Compute and check whether the decrease ratio satisfies:

$$\rho_k := \frac{f(x_k) - f(x_k + s_k)}{f(x_k) - \hat{q}_k(\hat{s})} \geq \theta.$$

If the decrease condition holds, set $x_{k+1} = x_k + s_k$ and $\sigma_{k+1} < \sigma_k$ [successful iteration].
Otherwise set $x_{k+1} = x_k$ and $\sigma_{k+1} > \sigma_k$ [unsuccessful iteration].
Increase the iteration count by setting $k = k + 1$ in both cases. Set $l_{k+1} \geq l_k$

R-ARC as presented in [14, 15] can be recovered from Algorithm 1 by fixing $l_k = l$ from some $l \geq 1$. Further setting $S_k = I_d$ for each iteration recovers the original ARC algorithm.

Evaluating sketched problem information Algorithm 1 only requires projected objective's gradients and Hessians (see $\hat{\nabla} f(x_k)$ and $\hat{\nabla}^2 f(x_k)$ in (1)). These can be calculated efficiently, without evaluation of full gradients and Hessians, using techniques such as using directional derivatives, block finite differences or automatic differentiation. For example, $\hat{\nabla} f(x_k)$ requires only l_k directional derivatives of f with respect to the rows of S_k.

2.1. An adaptive sketch size rule

We introduce a sketch size update rule that can be included in Algorithm 1. The motivation behind this update rule is that we seek to use local problem information to, in a sense, learn the rank of the function f (assuming that it has low-rank structure). To do this, we keep track of the observed ranks of the sketched Hessian. In Algorithm 1, we define the following for $k \geq 0$:

$$r_k := \text{rank}(\nabla^2 f(x_k)); \quad \hat{r}_k := \text{rank}(S_k \nabla^2 f(x_k) S_k^T); \quad \hat{R}_k := \max_{1 \leq j \leq k} \hat{r}_k.$$

Using these, we can give the following update rule:

$$l_{k+1} = \begin{cases} \max(C\hat{R}_k + 1, l_k) & \text{if } \hat{R}_k > \hat{R}_{k-1} \\ l_k & \text{otherwise.} \end{cases} \tag{$\star$}$$

where $C \geq 1$ is a user-defined constant. We make two remarks:

1. in the case that $C = 1$, we simply need to assess whether the sketched Hessian is singular, rather than know its rank. This is because $\hat{R}_k \leq l_k$ so $l_{k+1} > l_k$ only if $\hat{r}_k = l_k$.
2. for all k, we have $l_k \leq \max(Cr + 1, l_0)$ where r is the rank of f, and hence l_k remains $\mathcal{O}(r)$.

We now seek to show that if this update rule is followed, the sketch dimension l_k will increase in a manner that enables the same $\mathcal{O}(\epsilon^{-3/2})$ iteration complexity to drive the norm of the objective's gradient norm below ϵ as Theorem 1.

Lemma 4 *Letting $S_k \in \mathbb{R}^{l_k \times d}$ be a Gaussian matrix with $l_k \leq d$, we have*

$$\mathbb{P}(\hat{r}_k = \min(l_k, r_k)) = 1. \tag{2}$$

We apply this Lemma to prove the following Corollary.

Lemma 5 *Set $l_0 \geq 1$ and suppose that the update rule ($\star$) is applied to l_k. For all $k \geq 1$, If $l_k < Cr_k + 1$, then with probability 1, $\hat{R}_k > \hat{R}_{k-1}$.*

Proof By the update rule ($\star$), we have that $l_k < Cr_k + 1 \implies \hat{R}_{k-1} < r_k$. We also have that $l_k \geq \hat{R}_{k-1} + 1$. Therefore, we have that $l_k < Cr_k + 1 \implies \hat{R}_{k-1} < \min(l_k, r_k)$. Hence, by Lemma 4, we have that $\hat{r}_k = \min(l_k, r_k) > \hat{R}_{k-1}$ with probability 1. ■

Applying this Lemma allows us to prove the following convergence result.

Theorem 6 (R-ARC-D convergence result) *Suppose that $\mathcal{S}$ is the distribution of scaled Gaussian matrices and f is a low-rank function of rank r with Lipschitz-continuous second derivatives. Apply Algorithm 1 with the sketch update rule ($\star$) with $l_0 \geq 1, C = \lceil 4C_l(2+\log(16)) \rceil$ where C_l is defined in Lemma 9, then R-ARC achieves the optimal $\mathcal{O}(\epsilon^{-3/2})$ rate of convergence, with high probability.*

In terms of dimension-dependence, the $\mathcal{O}$ bound on the number of iterations in Theorem 6 is proportional to $\sqrt{d/r}$ as long as $r_k leq r$. Thus R-ARC-D benefits from the same optimal convergence rate result as R-ARC and ARC, up to a constant. Furthermore, when applied to low-rank functions, the algorithm is able to learn the function rank whilst solving local subproblems in smaller-dimensional subspaces.

3. Numerical Experiments

In these numerical experiments, we apply the R-ARC-D algorithm as described in Algorithm 1, using the l_k update rule ($\star$) with $C = 1$ for simplicity. The code we used is a modification of the ARC code used in [3]. We make a minor modification in that we only redraw S_k after successful iterations; this update step performs better empirically than redrawing after each iteration. The performance of R-ARC-D is compared with that of R-ARC and ARC. As a measure of budget, we use relative Hessians seen; if at iteration k, we draw a sketching matrix of size $l_k \times d$, we see $(l_k/d)^2$ relative Hessians. When calculating $\hat{\nabla} f(x_k)$ and $\hat{\nabla}^2 f(x_k)$, we calculate $\nabla f(x_k)$ and $\nabla^2 f(x_k)$, and then multiply by S_k; the computational efficiency could be much improved by applying techniques discussed in Section 2.

Augmented CUTEst problems To create low-rank functions to test on, we take CUTEst [9] problems of dimension (or rank) $r \approx 100$ and add dimensions and rotate to create low-rank problems of dimension $d = 1000$. Given a function $f : \mathbb{R}^r \to \mathbb{R}$, this can be achieved by sampling a random orthogonal matrix $Q \in \mathbb{R}^{d \times r}$ so that $Q^\top Q = I_r$ and define $g : \mathbb{R}^d \to \mathbb{R}$ by $g(x) = f(Q^\top x)$ to be a low-rank variant of f. To distinguish these problems from the standard CUTEst versions, we prefix the problem names with "l-". Problem details can be found in Table 1.

R-ARC-D update step In Figure 1, we plot a test to demonstrate how the adaptive sketch update rule ($\star$) works on an example problem, starting from $l_0 = 2$.

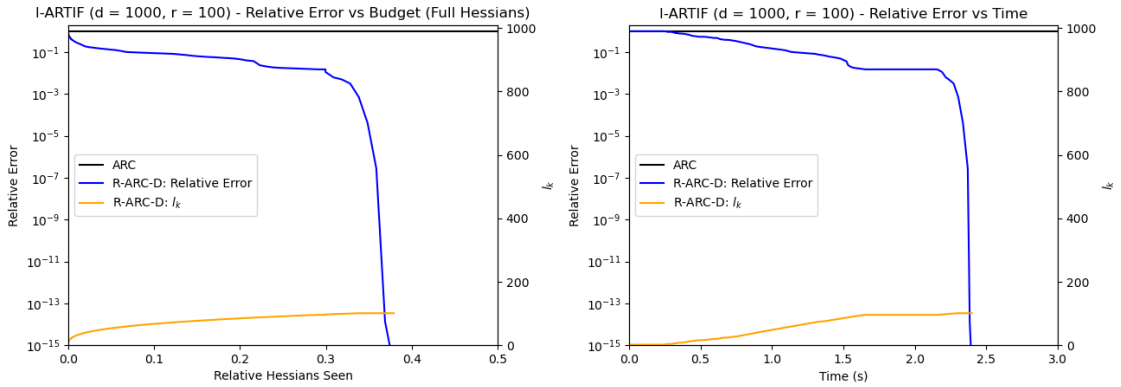

Figure 1: Example of R-ARC-D applied to the low-rank problem l-ARTIF

We see that the sketch dimension l_k increases to eventually reach the function rank. For l-ARTIF, we see that R-ARC-D significantly outperforms ARC, which fails to make a step in the budget and time taken for R-ARC-D to converge. We can also apply R-ARC-D to full-rank problems such as

ARTIF (with parameter N = 1000), which we plot in Figure 4, where R-ARC-D converges, but not faster than ARC.

Data Profiles Here we compare the performance between R-ARC-D and R-ARC through data profiles [10]. A description of the methodology can be found in Appendix B. The set of problems considers can be found in Table 1. For R-ARC-D, we again set $l_0 = 2$, whilst for R-ARC, we sketch at 1%, 5% and 7.5% of the original problem dimension. As the functions are of dimension $d = 1000$ with rank $r \approx 100$, this corresponds to $\approx 10 - 75\%$ of the function rank. The results are plotted in Figure 3, where we show results for tolerances $\tau = 1e-2$ (low-precision) and $\tau = 1e-5$ (high precision). We repeat each problem 5 times, with different Q matrices, treating each run as a separate problem in the plots.

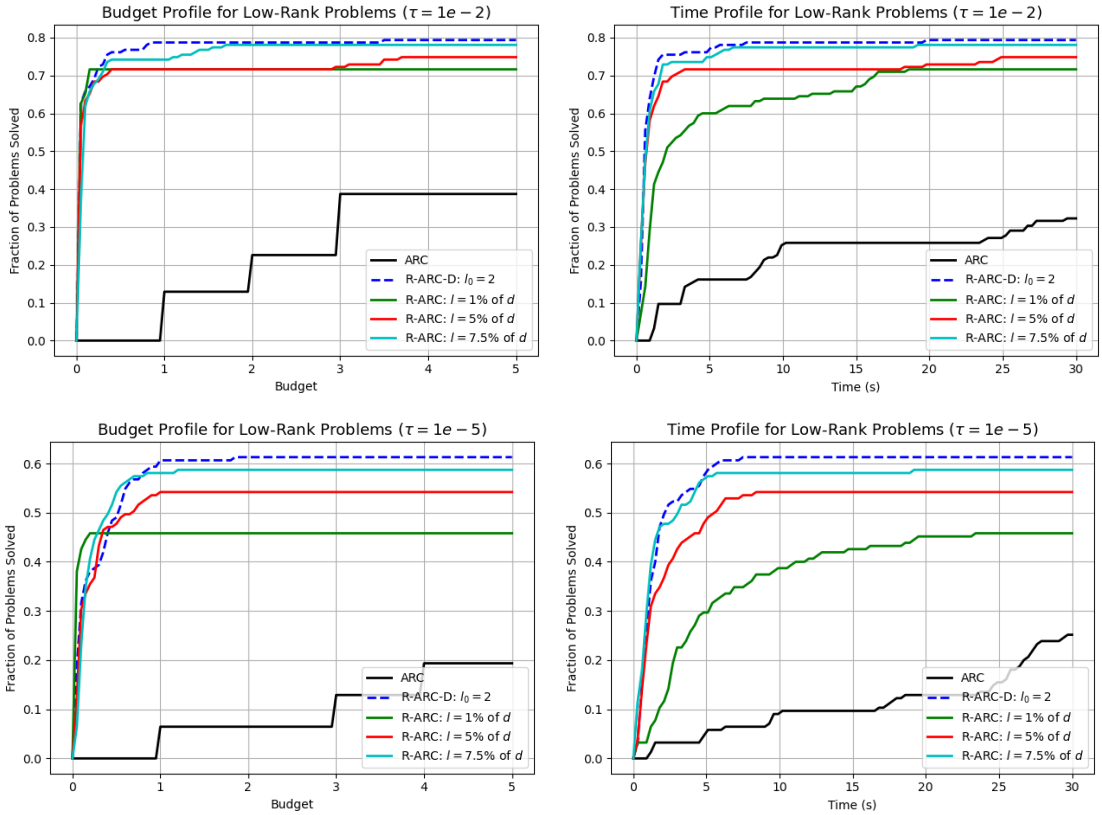

Figure 2: Data profiles of R-ARC-D compared to R-ARC and ARC

We see that R-ARC-D started at $l_0 = 2$ outperforms R-ARC regardless of the fixed sketch size used by R-ARC. This is more significant for the high-precision solutions ($\tau = 1e-5$) where R-ARC-D typically increases l_k until it reaches the function rank r. In Figure D, we plot individual problem performance for several of the problems considered here. In these plots, we see that whilst R-ARC with $l = 1\%$ of d occasionally performs best in terms of relative Hessians, it performs worst in terms of time due to it taking significantly more iterations. Thus overall, we found that R-ARC-D performs particularly well on low-rank problems, which we have demonstrated both numerically and theoretically.

Acknowledgments The first author's (ET) research was supported by the Oxford Centre for Doctoral Training in Mathematics of Random Systems (EPSRC EP/S023925/1), while the second author's (CC), by the Hong Kong Innovation and Technology Commission's Center for Intelligent Multi-dimensional Analysis (InnoHK Project CIMDA).

References

[1] James Bergstra and Yoshua Bengio. Random Search for Hyper-Parameter Optimization. *Journal of Machine Learning Research*, 13(10):281–305, 2012. ISSN 1533-7928.

[2] Coralia Cartis and Adilet Otemissov. A dimensionality reduction technique for unconstrained global optimization of functions with low effective dimensionality. *Information and Inference: A Journal of the IMA*, 11(1):167–201, March 2022. ISSN 2049-8772. doi: 10.1093/imaiai/iaab011. URL https://doi.org/10.1093/imaiai/iaab011.

[3] Coralia Cartis, Nicholas IM 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, 2011.

[4] Coralia Cartis, Jaroslav Fowkes, and Zhen Shao. Randomised subspace methods for non-convex optimization, with applications to nonlinear least-squares, November 2022. URL http://arxiv.org/abs/2211.09873. arXiv:2211.09873 [math].

[5] Coralia Cartis, Xinzhu Liang, Estelle Massart, and Adilet Otemissov. Learning the subspace of variation for global optimization of functions with low effective dimension, January 2024. URL http://arxiv.org/abs/2401.17825. arXiv:2401.17825 [math].

[6] Nicholas I.M. Gould Coralia Cartis and Philippe L. Toint. *Evaluation complexity of algorithms for nonconvex optimization: theory, computation, and perspectives*. Society for Industrial and Applied Mathematics, Philadelphia, 2022. ISBN 978-1-61197-699-1.

[7] Romain Cosson, Ali Jadbabaie, Anuran Makur, Amirhossein Reisizadeh, and Devavrat Shah. Gradient Descent for Low-Rank Functions, June 2022. URL http://arxiv.org/abs/2206.08257. arXiv:2206.08257 [cs, math].

[8] Elizabeth D Dolan and Jorge J Moré. Benchmarking optimization software with performance profiles. *Mathematical programming*, 91(2):201–213, 2002.

[9] Nicholas IM Gould, Dominique Orban, and Philippe L Toint. CUTEst: a constrained and unconstrained testing environment with safe threads for mathematical optimization. *Computational Optimization and Applications*, 60(3):545–557, 2015.

[10] Jorge J. Moré and Stefan M. Wild. Benchmarking Derivative-Free Optimization Algorithms. *SIAM Journal on Optimization*, 20(1):172–191, January 2009. ISSN 1052-6234. doi: 10.1137/080724083. URL https://epubs.siam.org/doi/10.1137/080724083. Publisher: Society for Industrial and Applied Mathematics.

[11] 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.

[12] Suzanna Parkinson, Greg Ongie, and Rebecca Willett. ReLU Neural Networks with Linear Layers are Biased Towards Single- and Multi-Index Models, June 2024. URL http://arxiv.org/abs/2305.15598. arXiv:2305.15598 [cs, stat].

[13] Tamás Sarlós. Improved approximation algorithms for large matrices via random projections. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 143–152, 2006. doi: 10.1109/FOCS.2006.37.

[14] Zhen Shao. *On Random Embeddings and Their Application to Optimisation*. PhD thesis, Mathematical Institute, University of Oxford, 2022.

[15] Zhen Shao and Coralia Cartis. Random-subspace adaptive cubic regularisation method for nonconvex optimisation. In *HOO-2022: Order up! The Benefits of Higher-Order Optimization in Machine Learning*, 2022.

[16] Ziyu Wang, Frank Hutter, Masrour Zoghi, David Matheson, and Nando De Feitas. Bayesian Optimization in a Billion Dimensions via Random Embeddings. *Journal of Artificial Intelligence Research*, 55:361–387, February 2016. ISSN 1076-9757. doi: 10.1613/jair.4806. URL https://jair.org/index.php/jair/article/view/10983.

[17] David P. Woodruff. Sketching as a tool for numerical linear algebra. *Found. Trends Theor. Comput. Sci.*, 10(1-2):1–157, 2014. ISSN 1551-305X. doi: 10.1561/0400000060. URL https://doi.org/10.1561/0400000060.

Appendix A. Additional results

Definition 7 (ϵ-subspace embedding [17]) *An ϵ-subspace embedding for a matrix $B \in \mathbb{R}^{d\times k}$ is a matrix $S \in \mathbb{R}^{l\times d}$ such that*

$$(1-\epsilon)\|y\|_2^2 \le \|Sy\|_2^2 \le (1+\epsilon)\|y\|_2^2 \quad \text{for all } y \in Y = \{y : y = Bz, z \in \mathbb{R}^k\}. \tag{3}$$

Definition 8 (Oblivious subspace embedding [13, 17]) *A distribution $\mathcal{S}$ on $S \in \mathbb{R}^{l\times d}$ is an (ϵ,δ)-oblivious subspace embedding for a given fixed/arbitrary matrix $B \in \mathbb{R}^{d\times k}$, we have that, with a high probability of at least $1-\delta$, a matrix S from the distribution is an ε-subspace embedding for B.*

Lemma 9 (Theorem 2.3 in [17]) *Let $\epsilon_S \in (0,1)$ and $S \in \mathbb{R}^{l\times d}$ be a scaled Gaussian matrix. Then for any fixed $d \times (d+1)$ matrix M with rank at most $r+1$, with probability $1-\delta_S$ we have that simultaneously for all $z \in \mathbb{R}^{d+1}$, $\|SMz\|_2^2 \ge (1-\epsilon_S)\|Mz\|_2^2$, where*

$$\delta_S = \exp -\frac{l(\epsilon_S)^2}{C_l} + r + 1 \tag{4}$$

and $C_l > 0$ is an absolute constant.

A.1. Proof of Lemma 3

In order to prove Lemma 3, we need the following result.

Lemma 10 ([5, 7]) *A function $f : \mathbb{R}^d \to \mathbb{R}$ has rank r, with $r \le d$ if and only if there exists a matrix $A \in \mathbb{R}^{r\times d}$ and a map $\sigma : \mathbb{R}^r \to \mathbb{R}$ such that $f(x) = \sigma(Ax)$ for all $x \in \mathbb{R}^d$.*

Proof [Proof of Lemma 3] Using Lemma 10, we can write

$$f(x) = \sigma(Ax) \tag{5}$$

where $A \in \mathbb{R}^{r\times d}$ and clearly, A has rank at most r. We have

$$\nabla f(x) = \frac{\partial \sigma(Ax)}{\partial x} = A^\top \cdot \frac{\partial \sigma(Ax)}{\partial (Ax)} = A^\top \nabla \sigma(Ax). \tag{6}$$

Furthermore, as f is C^2 by assumption, we have

$$\nabla^2 f(x) = \frac{\partial \nabla f(x)}{\partial x^T} = A^\top \cdot \frac{\partial \nabla \sigma(Ax)}{\partial x^T} = A^\top \cdot \frac{\partial \nabla \sigma(Ax)}{\partial (Ax)^T} \cdot A = A^\top [\nabla^2 \sigma(Ax)] A. \tag{7}$$

As $\mathrm{rank}(A) \le r$, we can conclude that $\nabla^2 f(x)$ has rank bounded above by r. ■

Appendix B. Data profile methodology

We measure algorithm performance using data profiles [10], which themselves are a variant of performance profiles [8]. We use *relative Hessians seen* as well as runtime for our data profiles. The relative Hessians seen at an iteration k is $(l_k/d)^2$ where l_k is the sketch dimension and d is the problem dimension. Using the same notation as in [4], for a given solver s, test problem $p \in \mathcal{P}$ and tolerance $\tau \in (0, 1)$, we determine the number of relative Hessians seen $N_p(s, \tau)$ required for a problem to be solved:

$$N_p(s, \tau) := \text{ \# of relative Hessians seen until } f(x_k) \leq f(x^*) + \tau(f(x_0) - f(x^*)).$$

We set $N_p(s, \tau) = \infty$ in the cases where the tolerance is not reached within the maximum number of iterations, which we take to be 2000.

To produce the data profiles, we plot

$$\pi^N_{s,\tau}(\alpha) := \frac{|\{p \in \mathcal{P} \ : \ N_p(s, \tau) \leq \alpha\}|}{|\mathcal{P}|} \text{ for } \alpha \in [0, 100],$$

namely, the fraction of problems solved after α relative Hessians seen. For runtime data profiles, we replace relative Hessians seen with the runtime in the above definitions.

Appendix C. Low-rank problems

#	Problem	d	r	$f(x_0)$	$f(x^*)$	Parameters
1	l-ARTIF	1000	100	1.8296×10^1	0	N = 100
2	l-ARWHEAD	1000	100	2.9700×10^2	0	N = 100
3	l-BDEXP	1000	100	2.6526×10^1	0	N = 100
4	l-BOX	1000	100	0	-1.1240×10^1	N = 100
5	l-BOXPOWER	1000	100	8.6625×10^2	0	N = 100
6	l-BROYDN7D	1000	100	3.5098×10^2	4.0123×10^1	N/2 = 50
7	l-CHARDIS1	1000	98	1.2817×10^1	0	NP1 = 50
8	l-COSINE	1000	100	8.6881×10^1	-9.9000×10^1	N = 100
9	l-CURLY10	1000	100	-6.2372×10^{-3}	-1.0032×10^4	N = 100
10	l-CURLY20	1000	100	-1.2965×10^{-2}	-1.0032×10^4	N = 100
11	l-DIXMAANA1	1000	90	8.5600×10^2	1	M = 30
12	l-DIXMAANF	1000	90	1.2253×10^3	1	M = 30
13	l-DIXMAANP	1000	90	2.1286×10^3	1	M = 30
14	l-ENGVAL1	1000	100	5.8410×10^3	0	N = 100
15	l-FMINSRF2	1000	121	2.5075×10^1	1	P = 11
16	l-FMINSURF	1000	121	3.0430×10^1	1	P = 11
17	l-NCB20	1000	110	2.0200×10^2	1.7974×10^2	N = 100
18	l-NCB20B	1000	100	2×10^2	1.9668×10^2	N = 100
19	l-NONCVXU2	1000	100	2.6397×10^6	2.3168×10^2	N = 100
20	l-NONCVXUN	1000	100	2.7270×10^6	2.3168×10^2	N = 100
21	l-NONDQUAR	1000	100	1.0600×10^2	0	N = 100
22	l-ODC	1000	100	0	-1.9802×10^{-2}	(NX, NY) = (10, 10)
23	l-OSCIGRNE	1000	100	3.0604×10^8	0	N = 100
24	l-PENALTY3	1000	100	9.8018×10^7	1×10^{-3}	N/2 = 50
25	l-POWER	1000	100	2.5503×10^7	0	N = 100
26	l-RAYBENDL	1000	126	9.8028×10^1	9.6242×10^1	NKNOTS = 64
27	l-SCHMVETT	1000	100	-2.8029×10^2	-2.9940×10^3	N = 100
28	l-SINEALI	1000	100	-8.4147×10^{-1}	-9.9010×10^3	N = 100
29	l-SINQUAD	1000	100	6.5610×10^{-1}	-3	N = 100
30	l-TOINTGSS	1000	100	8.9200×10^2	1.0102×10^1	N = 100
31	l-YATP2SQ	1000	120	9.1584×10^4	0	N = 10

Table 1: The 31 low-rank CUTEst test problems used in the data profiles.

Appendix D. Individual problem performance

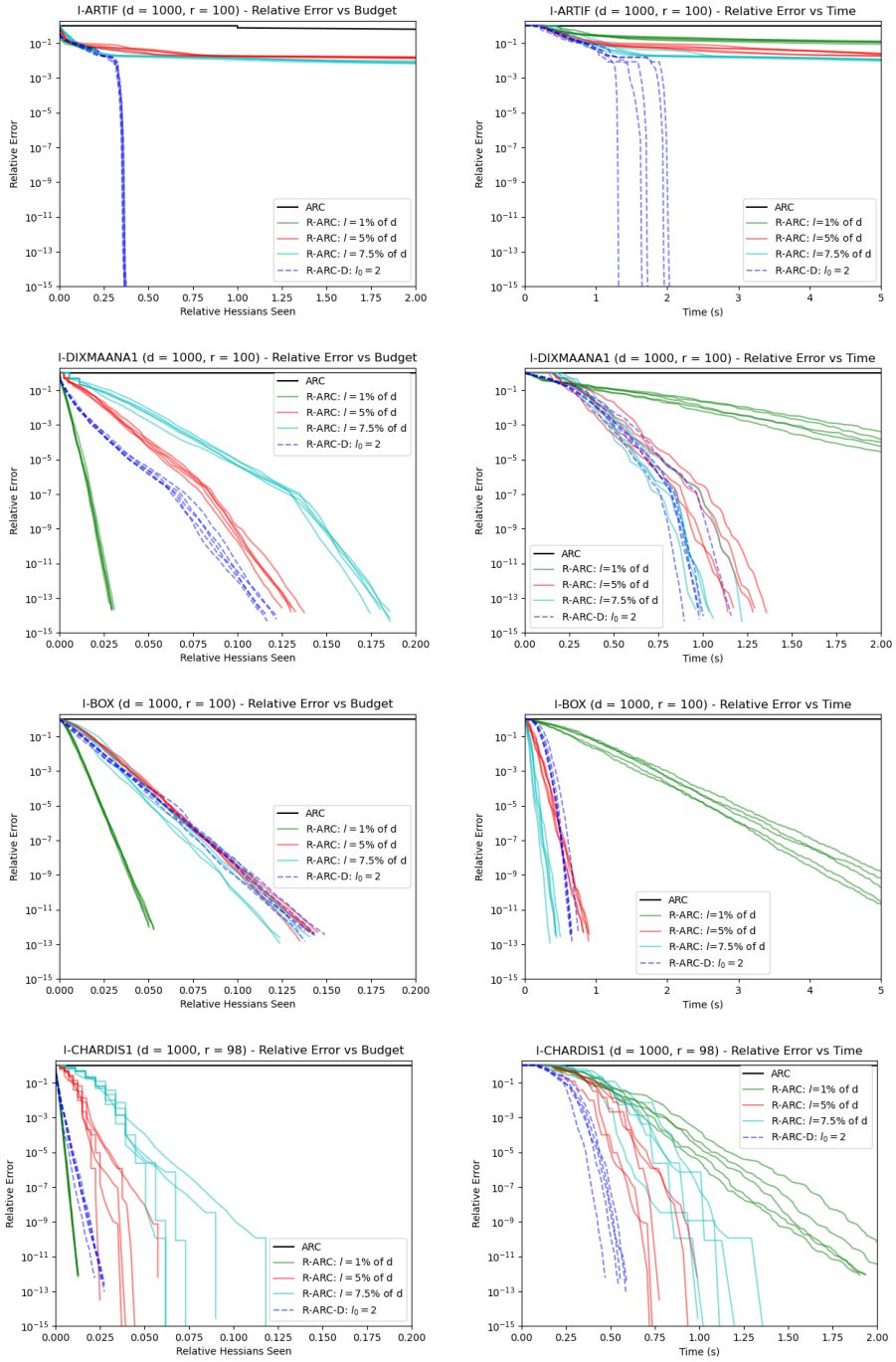

Figure 3: Comparison between R-ARC-D and R-ARC on low-rank problems from Table 1

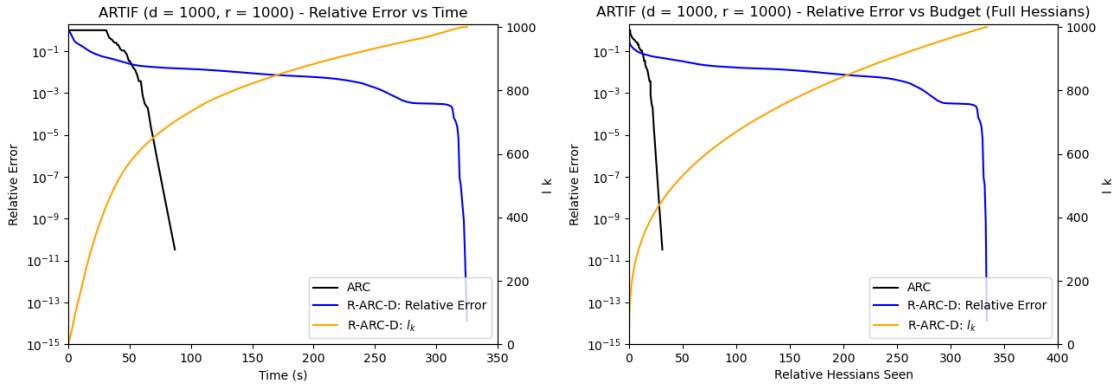

Figure 4: Example of R-ARC-D applied to the full-rank problem ARTIF (with parameter N = 1000), which has $r = d = 1000$.