SAN: Stochastic Average Newton Algorithm for Minimizing Finite Sums

Jiabin Chen[1,5]   **Rui Yuan**[2,5]   **Guillaume Garrigos**[3]   **Robert M. Gower**[4]

Abstract

We present a principled approach for designing stochastic Newton methods for solving finite sum optimization problems. Our approach has two steps. First, we re-write the stationarity conditions as a system of nonlinear equations that associates each data point to a new row. Second, we apply a Subsampled Newton Raphson method to solve this system of nonlinear equations. Using our approach, we develop a new Stochastic Average Newton (SAN) method, which is incremental by design, in that it requires only a single data point per iteration. It is also cheap to implement when solving regularized generalized linear models, with a cost per iteration of the order of the number of the parameters. We show through numerical experiments that SAN requires no knowledge about the problem, neither parameter tuning, while remaining competitive as compared to classical variance reduced gradient methods (e.g. SAG and SVRG), incremental Newton and quasi-Newton methods (e.g. SNM, IQN).

1 Introduction

Consider the problem of minimizing a sum of terms

$$w^* \in \operatorname*{argmin}_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} f_i(w) \stackrel{\text{def}}{=} f(w), \qquad (1)$$

where f_i is a convex twice differentiable loss over a given i-th data point. When the number of *data points* n and *features d* are large, first order methods such as SGD (Stochastic Gradient Descent), SAG (Schmidt et al., 2017), SVRG (Johnson and Zhang, 2013) and ADAM (Kingma and Ba, 2015) are the methods of choice for solving (1) because of their low cost per iteration. The issue with first order methods is that they can require extensive parameter tuning and their convergence depends heavily on the condition number of the data. Consequently, to make a first order method work well requires careful tweaking and tuning from an expert, and a careful choice of the model itself. Indeed, neural networks have evolved in such a way that allows for SGD to converge, such as the introduction of batch norm (Ioffe and Szegedy, 2015) and the push for more over-parametrized networks which greatly speed-up the convergence of SGD (Ma et al., 2018; Vaswani et al., 2019; Gower et al., 2021). Thus the reliance on first order methods ultimately restricts the choice and development of alternative models.

There is now a concerted effort to develop efficient stochastic second order methods that can exploit the sum of terms structured in (1). The hope for second order methods for solving (1) is that they require less parameter tuning and converge for wider variety of models and datasets. In particular, here we set out to develop stochastic second order methods that achieve the following objective.

Objective 1. Develop a second order method for solving (1) that is *incremental, efficient*, scales well with the dimension d, and that requires no *knowledge from the problem*, neither *parameter tuning*.

Most stochastic second order methods are not incremental, and thus fall short of our first criteria. This is due to the fact that most of these methods are only guaranteed to work in a large mini-batch size regime, and not with a single sample. For instance, the subsampled Newton methods (Roosta-Khorasani and Mahoney, 2019; Bollapragada et al., 2018; Liu and Roosta, 2021; Erdogdu and Montanari, 2015; Kohler and Lucchi, 2017) require potentially large mini-batch

---

[1]Baidu Inc., École Polytechnique.   [2]Meta AI. LTCI, Télécom Paris, Institut Polytechnique de Paris. [3]Université de Paris and Sorbonne Université, CNRS, Laboratoire de Probabilités, Statistique et Modélisation. [4]CCM, Flatiron Institute. LTCI, Télécom Paris, Institut Polytechnique de Paris. [5]Equal contribution.

sizes in order to guarantee that the subsampled Newton direction closely matches the full Newton direction in high probability. Stochastic quasi-Newton methods (Byrd et al., 2011; Mokhtari and Ribeiro, 2015; Moritz et al., 2016; Gower et al., 2016) SDNA (Qu et al., 2016), and the Newton sketch (Pilanci and Wainwright, 2017) and Lissa (Agarwal et al., 2017), suffer from the same drawback: the need for large mini-batches or full gradient evaluation to work, which makes them all not incremental.

The two existing methods that we are aware of that are truly incremental are *IQN* (Incremental Quasi-Newton) (Mokhtari et al., 2018; Gao and Ribeiro, 2020) and *SNM* (Stochastic Newton Method) (Kovalev et al., 2019; Rodomanov and Kropotov, 2016). Both methods also enjoy a fast local convergence rate. Their only drawback is their computational and memory costs per iteration are at least $\mathcal{O}(d^2)$ (see Table 1 and Section C.4 for more details). This is prohibitive in a setting where the number of parameters for the model is large. Our goal is to develop a method that is not only incremental, but also has a cost per iteration of $\mathcal{O}(d)$, as is the case for first-order methods like SGD.

In this paper we develop two new Newton methods for solving (2) that effectively make use of second order information, are incremental, and are governed by a single global convergence theory. Our starting point for developing these methods is to re-write the stationarity conditions

$$\frac{1}{n}\sum_{i=1}^{n}\nabla f_i(w) = 0. \qquad (2)$$

At this point, we could apply Newton's method for solving nonlinear equations, otherwise known as the Newton Raphson method. However, this approach would ultimately require a full pass over the data at each iteration.

To avoid taking full passes over the data, we re-write (2) by introducing n auxiliary variables $\alpha_i \in \mathbb{R}^d$ and solving instead the nonlinear system given by

$$\frac{1}{n}\sum_{i=1}^{n}\alpha_i = 0, \qquad (3)$$

$$\alpha_i = \nabla f_i(w), \quad \forall i \in \{1, \ldots, n\}. \qquad (4)$$

Clearly (3–4) have the same solutions in w. The advantage of (3–4) is that each gradient lies on a separate row. Consequently, applying a *subsampled* Newton Raphson method, that is sampling a row and then applying Newton Raphson, to (3–4) will result in an incremental method. We refer to (3–4) as the *function splitting* formulation, since it splits the gradient across different rows.

To solve (3–4) efficiently, we propose SAN (*Stochastic Average Newton*) in Section 2.1. SAN is a subsampled Newton Raphson method that is based on a new variable metric extension of *SNR* (Sketch Newton Raphson Method) (Yuan et al., 2021) that we present in Section 4, which is itself a nonlinear extension of the Sketch-and-Project method for solving linear systems (Gower and Richtárik, 2015a). By using a different subsampling of the rows (3–4), we also derive SANA in Section 2.2, which is a variant of SAN that uses unbiased estimates of the gradient.

Note that the idea of applying a subsampled Newton method to a well-chosen system of optimality conditions is not new. Indeed, it was recently shown in (Yuan et al., 2021) that the SNM method (Kovalev et al., 2019) can be seen as the application of a subsampled Newton method to the equations

$$\frac{1}{n}\sum_{i=1}^{n}\nabla f_i(\alpha_i) = 0,$$

$$w = \alpha_i, \quad \text{for } i = 1, \ldots, n, \qquad (5)$$

which clearly are equivalent to (2). Consequently, the two methods SAN and SNM are both subsampled Newton Raphson methods applied to either a function or a *variable splitting* formulation of (2).

The contributions of our paper are the following:

- We propose combining the function splitting reformulation (3–4) with a subsampled Newton Raphson as a tool for designing stochastic Newton methods, all of which are *variance reducing* in the sense that they are incremental and they converge with a constant step size.

- We introduce SAN (Stochastic Average Newton method) by using this tool, which is incremental and parameter-free, in that, SAN works well with a step size $\gamma = 1$ independently of the underlying dataset or the objective function.

- By specializing to GLMs (Generalized Linear Models), we develop an efficient implementation of SAN that has the same cost per iteration as the first-order methods. We perform extensive numerical experiments and show that SAN is competitive as compared to SAG and SVRG.

- To provide a convergence theory of our methods, we extend the class of Sketched Newton Raphson (Yuan et al., 2021) methods to allow for a variable metric that includes SAN as a special case.

In Section 2, we show how to derive the SAN and SANA methods. We then present two different experimental settings comparing SAN to variance reduced gradients methods in Section 3. In Section 4, we study SAN/SANA as instantiations of a new *variable metric*

Sketch Newton Raphson method and present a convergence theory for this class of method.

The following will be assumed throughout the paper.

Assumption 1.1. For all $i \in \{1, \ldots, n\}$, the function $f_i : \mathbb{R}^d \longrightarrow \mathbb{R}$ is of class C^2 and verifies $\nabla^2 f_i(w) \succ 0$ for every $w \in \mathbb{R}^d$.

2 Function splitting methods

The advantage of the function splitting formulation given by (3) and (4) is that there is a separate row for each data point. We will now take advantage of this, and develop new incremental Newton methods based on subsampling the rows of (3–4).

The reformulation given in (3–4) is a large system of nonlinear equations. For brevity, let $p := (n+1)d$ and $x = [w ; \alpha_1 ; \cdots ; \alpha_n] \in \mathbb{R}^p$ be the stacking[1] of the w and α_i variables. Thus solving (3–4) is equivalent to solve $F(x) = 0$, where

$$F : \mathbb{R}^p \to \mathbb{R}^p \qquad (6)$$
$$x \mapsto \left[\tfrac{1}{n} \sum \alpha_i; \nabla f_1(w) - \alpha_1; \cdots ; \nabla f_n(w) - \alpha_n \right].$$

Solving nonlinear equations has long been one of the core problems in numerical analysis, with variants of the Newton Raphson method (Ortega and Rheinboldt, 2000) being one of the core techniques. From a given iterate $x^k \in \mathbb{R}^p$, the Newton Raphson method computes the next iterate x^{k+1} by linearizing F around x^k and solving the *Newton system*

$$\nabla F(x^k)^\top (x^{k+1} - x^k) = -F(x^k). \qquad (7)$$

Here $\nabla F(x) \in \mathbb{R}^{p \times p}$ denotes the Jacobian matrix of F at x, and it is assumed that (7) has a solution. The least norm solution of the Newton system is given by

$$x^{k+1} = x^k - \nabla F(x^k)^{\top \dagger} F(x^k), \qquad (8)$$

where † denotes the Moore-Penrose pseudoinverse.

This update can also be written as a projection step:

$$x^{k+1} = \operatorname{argmin} \|x - x^k\|^2$$
$$\text{s.t. } \nabla F(x^k)^\top (x - x^k) = -F(x^k). \qquad (9)$$

In our setting, (8) is prohibitively expensive because it requires access to all of the data at each step and the solution of a large $(n+1)d \times (n+1)d$ linear system. To bring down the cost of each iteration, and to have a resulting incremental method, at each iteration we will

---

[1]In this paper vectors are columns by default, and given $x_1, \ldots, x_n \in \mathbb{R}^q$ we note $[x_1; \ldots; x_n] \in \mathbb{R}^{qn}$ the (column) vector stacking the x_i's on top of each other.

subsample the rows of the Newton system before taking a projection step. Next, we present two methods based on subsampling. Later on Section 4, we generalize this subsampling approach to make use of *sketches* of the system.

2.1 SAN: the Stochastic Average Newton method

The SAN method is a subsampled Newton Raphson method that alternates between sampling equation (3) or sampling one of the equations in (4). After sampling, we then apply a step of Newton Raphson to the sampled equation.

To detail the SAN method, let $\pi \in (0, 1)$ be a fixed probability, and let $x^k = [w^k; \alpha_1^k; \ldots; \alpha_n^k] \in \mathbb{R}^p$ be a given k-th iterate. With probability π the SAN method samples equation (3) and focuses on finding a solution to this equation. Since (3) is a linear equation, it is equal to its own Newton equation. Furthermore, this linear equation (3) has n variables and only one equation, thus it has infinite solutions. We choose a single one of these infinite solution by using a projection step

$$\alpha_1^{k+1}, \ldots, \alpha_n^{k+1} = \operatorname*{argmin}_{\alpha_1, \ldots, \alpha_n \in \mathbb{R}^d} \sum_{i=1}^n \left\| \alpha_i - \alpha_i^k \right\|^2$$
$$\text{s.t. } \tfrac{1}{n} \sum_{i=1}^n \alpha_i = 0. \qquad (10)$$

The solution to this projection is given in line 4 in Algorithm 1 when $\gamma = 1$. We have added the step size $\gamma \in (0, 1]$ to act as relaxation.

Alternatively, with probability $(1 - \pi)$ the SAN method then samples the j-th equation in (4) uniformly among the n equations. To get the Newton system of $\nabla f_j(w) = \alpha_j$, we linearize around $w^k \in \mathbb{R}^d$ and $\alpha_j^k \in \mathbb{R}^d$ and set the linearization to zero giving

$$\nabla f_j(w^k) + \nabla^2 f_j(w^k)(w - w^k) = \alpha_j.$$

This linear equation has $2d$ unknowns, and thus also has infinite solutions. Again, we use a projection step to pick a unique solution as follows

$$\alpha_j^{k+1}, w^{k+1} = \operatorname*{argmin}_{\alpha_j, w \in \mathbb{R}^d} \left\| \alpha_j - \alpha_j^k \right\|^2 + \left\| w - w^k \right\|_{\nabla^2 f_j(w^k)}^2$$
$$\text{s.t. } \nabla f_j(w^k) + \nabla^2 f_j(w^k)(w - w^k) = \alpha_j. \qquad (11)$$

Here we have introduced a projection under a norm $\|w\|_{\nabla^2 f_j(w^k)} \overset{\text{def}}{=} \langle \nabla^2 f_j(w^k)w, w \rangle$ which is based on the Hessian matrix $\nabla^2 f_j(w^k)$. Performing a projection step with respect to the metric induced by the Hessian is often used in Newton type methods such as interior point methods (Renegar, 2001) and quasi-Newton

methods (Goldfarb, 1970). Moreover, we observed that this choice of metric resulted in a much faster algorithm (see Section C.5 for experiments that highlight this). The closed form solution to the above is given in lines 8-10 in Algorithm 1 when $\gamma = 1$ (see Lemma A.2 for the details).

We gather all these updates in Algorithm 1 and call the resulting method the *Stochastic Average Newton* method, or *SAN* for short.

Algorithm 1: SAN: Stochastic Average Newton

Input: $\{f_i\}_{i=1}^n$, step size $\gamma \in (0,1]$, probability $\pi \in (0,1)$, max iteration T

1   Initialize $\alpha_1^0, \cdots, \alpha_n^0, w^0 \in \mathbb{R}^d$
2   **for** $k = 1, \ldots, T$ **do**
3   **With probability π update:**
4       $\alpha_i^{k+1} = \alpha_i^k - \dfrac{\gamma}{n} \sum_{j=1}^n \alpha_j^k, \; \forall i \in \{1, \cdots, n\}$
5   **Otherwise with probability $(1-\pi)$:**
6   Sample uniformly $j \in \{1, \cdots, n\}$
7       $\mathbf{H}_k = \mathbf{I}_d + \nabla^2 f_j(w^k)$
8       $d^k = -\mathbf{H}_k^{-1}\left(\nabla f_j(w^k) - \alpha_j^k\right)$
9       $w^{k+1} = w^k + \gamma d^k$
10      $\alpha_j^{k+1} = \alpha_j^k - \gamma d^k$

Output: Last iterate w_{T+1}

The SAN method is incremental, since it can be applied with as little as one data point per iteration. SAN can also be implemented in such a way that the cost per iteration is $\mathcal{O}(d)$ in expectation. Indeed, the *averaging step* on line 4 contributes with a $\pi \times \mathcal{O}(nd)$ cost to the total cost in expectation, since all of the vectors $\alpha_i \in \mathbb{R}^d$ for $i = 1, \ldots, n$, are updated. But as we found through expensive testing in Section C.3, SAN converges quickly if π is of the order of $\mathcal{O}(1/n)$, reducing the cost in expectation to $\mathcal{O}(d)$. Further, the average of the α_i's can be efficiently implemented by maintaining and updating a variable $\overline{\alpha}^k = \frac{1}{n} \sum_{j=1}^n \alpha_j^k$. The main cost for SAN is in solving the linear system $(\mathbf{I}_d + \nabla^2 f_j(w^k))d = \alpha_j^k - \nabla f_j(w^k)$. Solving this system with a direct solver would cost $\mathcal{O}(d^3)$. Alternatively, the solution can be approximated using an iterative Krylov method for which each iteration costs $\mathcal{O}(d)$ by using backpropagation (Freund et al., 1992; Christianson, 1992) to compute Hessian-vector products. For regularized generalized linear models (GLMs), the total cost of this matrix inversion is only $\mathcal{O}(d)$ operations, as we show next.

Generalized Linear Models. Regularized GLMs are models for which we have

$$f_i(w) = \phi_i(a_i^\top w) + \lambda R(w), \tag{12}$$

where $\phi_i : \mathbb{R} \to \mathbb{R}$ is a loss function associated with the i-th data point $a_i \in \mathbb{R}^d$, $\lambda > 0$ is a regularization parameter and R is a regularizer that is twice differentiable and separable, i.e. $R(w) = \sum_{i=1}^d R_i(w_i)$ with $R_i : \mathbb{R} \to \mathbb{R}$. The inversion on line 8 of Algorithm 1 can be efficiently computed using the Woodbury identity because the Hessian $\nabla^2 f_j(w) = \phi_j''(a_j^\top w)(a_j a_j^\top) + \lambda \nabla^2 R(w)$ is a rank-one perturbation of a diagonal matrix, which costs $\mathcal{O}(d)$ to invert (see Lemma B.2 for an explicit formula).

Remark 2.1 (SAN vs. SNM for GLMs). SAN can be implemented efficiently for all GLMs with separable regularizers. This is not the case for SNM (Kovalev et al., 2019), which can only be implemented efficiently when the regularizer is the L2 norm. For other separable regularizers, the cost per iteration for SNM is to $\mathcal{O}(d^3)$ instead of $\mathcal{O}(d^2)$. See Section C.4 in the supp. material for details.

2.2   SANA: alternative with simultaneous projections

Here we present SANA, an alternative version of the SAN method. Instead of alternating between projecting onto linearizations of (3) and (4), the SANA method projects onto the intersection of (3) and the linearization of a subsampled equation (4). In other words, the next iterate $x^{k+1} = [w^{k+1}; \alpha_1^{k+1}; \ldots; \alpha_n^{k+1}]$ is defined as the unique solution of

$$\underset{w, \alpha_1, \ldots, \alpha_n \in \mathbb{R}^d}{\text{argmin}} \sum_{i=1}^n \left\| \alpha_i - \alpha_i^k \right\|^2 + \left\| w - w^k \right\|_{\nabla^2 f_j(w^k)}^2,$$
$$\text{s.t. } \nabla f_j(w^k) + \nabla^2 f_j(w^k)(w - w^k) = \alpha_j,$$
$$\tfrac{1}{n} \sum_{i=1}^n \alpha_i = 0. \tag{13}$$

The closed form solution of (13) corresponds to lines 4–8 in Algorithm 2 when the relaxation parameter is $\gamma = 1$ (see Lemma A.4 for a proof).

Algorithm 2: SANA

Input: $\{f_i\}_{i=1}^n$, step size $\gamma \in (0,1]$, max iteration T

1   Initialize $w^0, \alpha_1^0, \cdots, \alpha_n^0 \in \mathbb{R}^d$ s.t. $\sum_{i=1}^n \alpha_i^0 = 0$
2   **for** $k = 1, \ldots, T$ **do**
3   Sample uniformly $j \in \{1, \ldots, n\}$
4     $\mathbf{H}_k = (1 - \frac{1}{n})\mathbf{I}_d + \nabla^2 f_j(w^k)$
5     $d^k = -\mathbf{H}_k^{-1}(\nabla f_j(w^k) - \alpha_j^k)$
6     $w^{k+1} = w^k + \gamma d^k$
7     $\alpha_j^{k+1} = \alpha_j^k - \gamma(1 - \frac{1}{n})d^k$
8     $\alpha_i^{k+1} = \alpha_i^k + \frac{\gamma}{n}d^k, \quad$ for $i \neq j$

Output: Last iterate w_{T+1}

Computing one step of this method requires access to only one function f_j (through its gradient and Hessian evaluated at w^k). In terms of computational cost, each step requires inverting the $d \times d$ matrix $(1 - \frac{1}{n})I_d + \nabla^2 f_j(w^k)$. As with SAN, this cost reduces to $\mathcal{O}(d)$ in the context of generalized linear models. See Algorithm 4 in the appendix for the resulting implementation for GLMs. Yet even in the case of GLMs, the SANA method costs $\mathcal{O}(nd)$ per iteration because it updates all the α_i vectors at every iteration. Thus the SANA method has complexity which is $\mathcal{O}(n)$ times larger than SAN in expectation.

Both SAN and SANA can be interpreted as a stochastic relaxed Newton method that uses estimates of the gradient. Indeed, computing d^k in Algorithms 1 and 2 requires solving a relaxed Newton system

$$\left(\delta \mathbf{I}_d + \nabla^2 f_j(w^k)\right) d^k = \alpha_j^k - \nabla f_j(w^k), \qquad (14)$$

where $\delta = 1$ and $\delta = 1 - \frac{1}{n}$, respectively. The right hand side of this Newton system is a biased estimate of the gradient for SAN and an unbiased estimate for SANA. To see this, for simplicity, let $\gamma = 1$. Taking expectation conditioned on time k over the right-hand side of (14) gives

$$\mathbb{E}\left[\alpha_j^k - \nabla f_j(w^k)\right] = \frac{1}{n}\sum_{i=1}^{n} \alpha_i^k - \nabla f(w^k).$$

For SANA, because the averaging constraint is always enforced in (13), we have that $\frac{1}{n}\sum_{i=1}^{n} \alpha_i^k = 0$, thus the right hand side of (14) is always an unbiased estimate of the negative gradient. As for SAN, the averaging constraint is only enforced every so often with the update on line 4 in Algorithm 1. Thus for SAN, the right hand is a biased estimate of the negative gradient until the averaging constraint is enforced. In this sense, SAN and SANA are analogous to SAG (Schmidt et al., 2017) and SAGA (Defazio et al., 2014). We found in practice that this biased estimate of the gradient did not hurt the empirical performance of SAN, and thus we focus on experiments on SAN in Section 3.

3 Experiments for SAN applied for GLMs

Here we compare SAN in Algorithm 1 against two variance reduced gradient methods SAG (Schmidt et al., 2017) and SVRG (Johnson and Zhang, 2013) for solving regularized GLMs (12), where $\phi_i(t) = \log\left(1 + e^{-y_i t}\right)$ is the logistic loss, $y_i \in \{-1, 1\}$ is the i-th target value, and R is the regularizer.

We use eight datasets in our experiments taken from LibSVM (Chang and Lin, 2011),[2] with disparate prop-

erties (see details of the datasets in Table 2). We fixed an initial random seed, evaluated each method 10 times, and stopped when the gradient norm was below 10^{-6} or a maximum of 50 effective passes over data had been reached. In all of our experiments, we plot effective data passes[3] vs gradient norm, and plot the central tendency as a solid line and all other executions as a shaded region. Plots with function sub-optimality are also provided in Figure 3 Section C.2, and show much the same relative rankings amongst the methods as the gradient norm plots.

For all methods, we used the default step size. For instance, for SAG and SVRG we use the step size $\frac{1}{L_{\max}}$ where $L_{\max}$ is the largest smoothness constant of f_i, for $i = 1, \ldots, n$. This step size is significantly larger than what has been proven to work for SAG and SVRG.[4] Yet despite this, it is the default setting in sklearn's logistic regression solver (Pedregosa et al., 2011), and we also found that it worked well in practice. The other hyperparameter of SVRG is the inner loop size which is set to n throughout our experiments. As for SAN, we set the probability $\pi = \frac{1}{n+1}$ and step size $\gamma = 1$. More details of the experiments are in Section C.1. [5]

Logistic regression with L2 regularization. We consider L2-regularized logistic regression, i.e. the regularizer is $R(w) = \frac{\lambda}{2} \|w\|^2$ with $\lambda = 1/n$. From Figure 1, SAN outperforms SAG and SVRG in all eight datasets, except for `mushrooms`, `ijcnn1` and `covtype`, where SAN remains competitive with SAG or SVRG. Note as well that for reaching an approximate solution at early stage, SAN outperforms SAG and SVRG in all datasets, except for `covtype`. Furthermore, SAN often has a smaller variance compared to SAG and SVRG based on eye-balling the shaded error bars in Figure 1 which was produced by multiple executions.

Logistic regression with pseudo-Huber regularization. We also tested logistic regression with pseudo-Huber regularizer. The pseudo-Huber regularizer is defined as $R(w) = \lambda \sum_{i=1}^{d} R_i(w_i)$ with $R_i(w_i) = \delta^2 \left(\sqrt{1 + \left(\frac{w_i}{\delta}\right)^2} - 1\right)$ and is used to pro-

---

[2] All datasets can be found downloaded on https://

`www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`. Some of the datasets can be found originally in (Mohammad et al., 2012; Chang and Lin, 2001; Blackard and Dean, 1999; Wang et al.; Lewis et al., 2004; Dua and Graff, 2017).

[3] By effective data passes we mean the number of data access divided by n.

[4] SAG has been proven to converge with a step size of $1/16L_{\max}$ (Schmidt et al., 2017) and SVRG provably converges with a step size of $1/10L_{\max}$ and loop size of $m = 10L_{\max}/\mu$ where μ is the strong convexity parameter $f(w)$ (Johnson and Zhang, 2013).

[5] The code is available on https://github.com/nathansiae/Stochastic-Average-Newton.
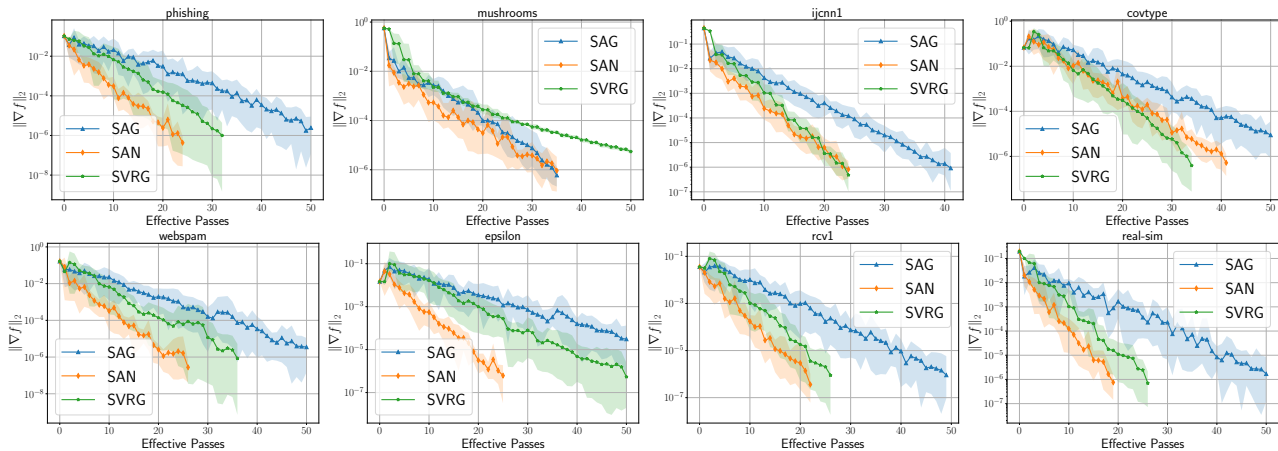
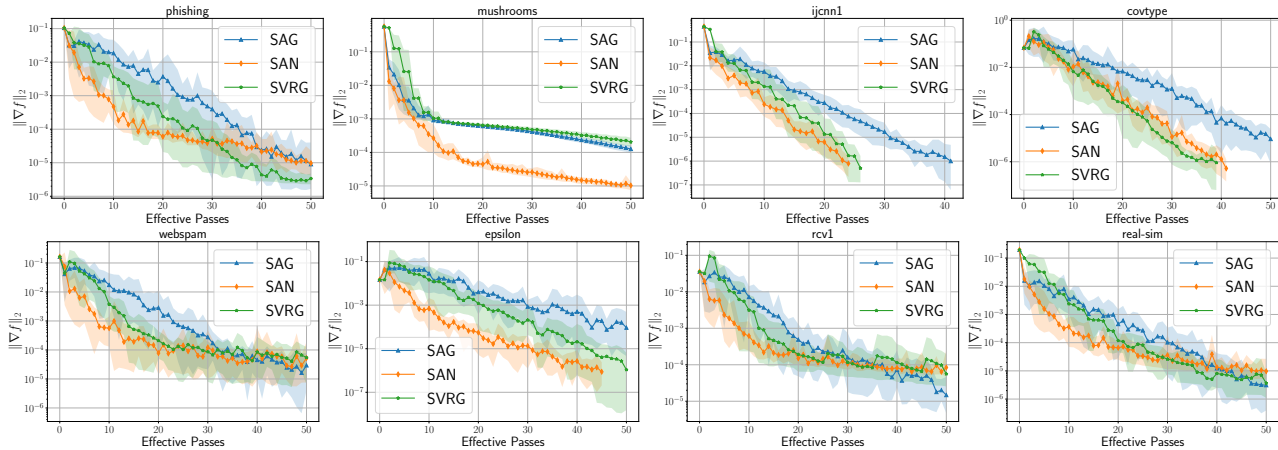Figure 1: Logistic regression with L2 regularization.

Figure 2: Logistic regression with pseudo-Huber regularization.

mote the sparsity of the solution (Fountoulakis and Gondzio, 2016). We set $\delta = 1$ and $\lambda = 1/n$. See Section C.1 for more properties and interpretations of the pseudo-Huber regularizer. From Figure 2, SAN is competitive with SAG and SVRG. Changing the regularizer from an L2 to pseudo-Huber has resulted in a slower convergence for all methods, except on the datasets `ijcnn1` and `covtype`. SVRG is notably slower when using the pseudo-Huber regularizer, while SAG is the least affected, and SAN is in between. Besides, SAN again outperforms SAG and SVRG in all datasets, except for `covtype`, for reaching an approximate solution at early stage and has a smaller variance compared to SAG and SVRG.

Overall, these tests confirm that SAN is efficient for a wide variety of datasets and problems. SAN is efficient in terms of effective passes and cost per iteration. It benefits from both using second order information yet still has the same cost $\mathcal{O}(d)$ as the stochastic first order methods. SNM and IQN, the only other methods that

fit our stated objective, have a $\mathcal{O}(d^2)$ cost per iteration for L2-regularized GLMs. SNM costs even more for other regularizers and IQN has a $\mathcal{O}(nd^2)$ memory cost. In Section C.4, we present experiments comparing SAN/SANA to the SNM and IQN algorithms.

Another advantage of SAN is that it requires no prior knowledge of the datasets nor tuning of the hyperparameters. To show this, we did a grid search over π and the step size γ of SAN, see Tables 3 and 4 in Section C.3, where we found that SAN performs equally well for a wide range of values of π and γ. Thus for simplicity we set $\pi = \frac{1}{n+1}$ and $\gamma = 1$ in the experiments. In contrast, SAG and SVRG require the computation of $L_{\max}$ and the performance is highly affected by the step size (see Table 5 and 6 in Section C.3).

However, the downside of SAN is that it stores nd scalars much like SAG/SAGA (Defazio et al., 2014). See Table 1 the comparison among different algorithms.

Table 1: Average cost of one iteration of various stochastic methods applied to GLMs.

	memory	memory access	data access	computational cost
SAN	$\mathcal{O}(nd)$	$\boldsymbol{\mathcal{O}(d)}$	$\boldsymbol{\mathcal{O}(1)}$	$\boldsymbol{\mathcal{O}(d)}^*$
SANA	$\mathcal{O}(nd)$	$\mathcal{O}(nd)$	$\boldsymbol{\mathcal{O}(1)}$	$\mathcal{O}(nd)$
SAG	$\mathcal{O}(nd)$	$\boldsymbol{\mathcal{O}(d)}$	$\boldsymbol{\mathcal{O}(1)}$	$\boldsymbol{\mathcal{O}(d)}$
SVRG	$\boldsymbol{\mathcal{O}(d)}$	$\boldsymbol{\mathcal{O}(d)}$	$\boldsymbol{\mathcal{O}(1)}$	$\boldsymbol{\mathcal{O}(d)}$
SNM	$\mathcal{O}(n+d^2)$	$\mathcal{O}(d^2)$	$\boldsymbol{\mathcal{O}(1)}$	$\mathcal{O}(d^2)^{**}$
IQN	$\mathcal{O}(nd^2)$	$\mathcal{O}(d^2)$	$\boldsymbol{\mathcal{O}(1)}$	$\mathcal{O}(d^2)$

*For SAN this $\mathcal{O}(d)$ computational cost is derived when $\pi = \mathcal{O}(1/n)$.

**For SNM this $\mathcal{O}(d^2)$ computational cost only holds for a L2 regularizer.

4 Sketched Newton Raphson with a variable metric

Though our main focus is in solving the function splitting reformulation (3) and (4), we find that our forthcoming theory holds for a large class of variable metric *Sketched Newton Raphson* methods (Yuan et al., 2021), of which SAN/SANA are special cases. All proofs are given in Section E.

In designing each method, we first reformulated our original problem (1) as a system of nonlinear equations $F(x) = 0$ for a given choice of a smooth map $F : \mathbb{R}^p \longrightarrow \mathbb{R}^m$ and where $p, m \in \mathbb{N}$ are appropriately chosen dimensions, e.g., (6) for SAN/SANA. We then proposed using a subsampled Newton-Raphson method for solving these nonlinear equations. Here we extend this subsampling to make use of any randomized *sketch* of the system. That is, consider a random *sketching* matrix $\mathbf{S}_k \in \mathbb{R}^{m \times \tau}$ sampled from some distribution, where $\tau \in \mathbb{N}$ is significantly smaller than p or m. We use this sketching matrix to compress the rows of the Newton system (7) at each iteration by left multiplying as follows

$$\mathbf{S}_k^\top F(x^k) + \mathbf{S}_k^\top \nabla F(x^k)^\top (x - x^k) = 0. \quad (15)$$

The resulting system has τ rows and is underdetermined. To pick a solution, we use the projection

$$x^{k+1} = \arg\min_{x \in \mathbb{R}^p} \left\| x - x^k \right\|^2 \quad (16)$$
$$\text{s.t. } \mathbf{S}_k^\top F(x^k) + \mathbf{S}_k^\top \nabla F(x^k)^\top (x - x^k) = 0.$$

The method in (16) is known as the SNR (*Sketched Newton Raphson*) method (Yuan et al., 2021). The SNR method affords a lot of flexibility by choosing different distributions for the sketching matrices. Yet it is not flexible enough to include SAN/SANA, since these require projections under a variable metric. To allow for projections under norms other than the L2 norm, we introduce a random positive-definite *metric* matrix $\mathbf{W}_k \in \mathbb{R}^{p \times p}$ that defines the norm under which we project. Introducing as well a damping parameter

$\gamma > 0$, as we did for SAN and SANA, we obtain the following method

$$\bar{x}^{k+1} = \operatorname{argmin} \|x - x^k\|_{\mathbf{W}_k}^2$$
$$\text{s.t. } \mathbf{S}_k^\top \nabla F(x^k)^\top (x - x^k) = -\mathbf{S}_k^\top F(x^k),$$
$$x^{k+1} = (1 - \gamma)x^k + \gamma \bar{x}^{k+1}. \quad (17)$$

We call this method the *Sketched Newton Raphson with Variable Metric* (SNRVM for short). The closed form expression[6] for the iterates (17) is given by

$$x^{k+1} = x^k - \gamma \mathbf{W}_k^{-1} \nabla F(x^k) \mathbf{S}_k \quad (18)$$
$$\cdot \left(\mathbf{S}_k^\top \nabla F(x^k)^\top \mathbf{W}_k^{-1} \nabla F(x^k) \mathbf{S}_k \right)^\dagger \mathbf{S}_k^\top F(x^k).$$

SAN/SANA are both instances of the SNRVM method by choosing $\mathbf{S}_k$ as a subsampling matrix and $\mathbf{W}_k$ depending on the stochastic Hessian matrices. In Section D we provide a detailed derivation of SAN/SANA as an instance of the SNRVM method.

We assume that at each iteration, the random matrices $(\mathbf{S}_k, \mathbf{W}_k)$ are sampled according to a proper finite distribution $\mathcal{D}_{x^k}$ defined in the following.

Assumption 4.1 (Proper finite distribution)**.** For every $x \in \mathbb{R}^p$, there exists $r \in \mathbb{N}$, probabilities $\pi_1, \ldots, \pi_r > 0$ with $\sum_{i=1}^r \pi_i = 1$, and matrices $\{\mathbf{S}_i(x), \mathbf{W}_i(x)\}_{i=1}^r$ s.t. for $i = 1, \ldots, r$, we have

$$\mathbb{P}_{(\mathbf{S},\mathbf{W})\sim\mathcal{D}_x}[(\mathbf{S}, \mathbf{W}) = (\mathbf{S}_i(x), \mathbf{W}_i(x))] = \pi_i.$$

The addition of a variable metric to SNR has proven to be very challenging in terms of establishing a convergence theory. The current convergence theory and proofs techniques for SNR in Yuan et al. (2021) all fail with the addition of a variable metric. This is not so surprising, considering the historic difficulty in developing theory for variable metric methods such as the quasi-Newton methods. Despite the immense practical success of quasi-Newton methods, a meaningful non-asymptotic convergence rate has eluded the optimization community for 70 years, with the first results

---

[6]See Lemma D.1 in the supp. material for a proof.

having only just appeared this year (Rodomanov and Nesterov, 2021a,b,c; Jin and Mokhtari, 2021).

We start by introducing a technical assumption which guarantees that (17) is well defined, which is always true for SAN and SANA.

Assumption 4.2. For every $x \in \mathbb{R}^p$, the matrices $\nabla F(x)^\top \nabla F(x)$ and $\mathbb{E}_{\mathcal{D}_x}[\mathbf{SS}^\top]$ are invertible, and every matrix $\mathbf{W} \sim \mathcal{D}_x$ is symmetric positive definite.

Proposition 4.3. Assumptions 4.1 and 4.2 are verified for SAN and SANA, under Assumption 1.1.

Let us now introduce the surrogate function

$$\hat{f}_k(x) \stackrel{\text{def}}{=} \frac{1}{2} \|F(x)\|^2_{(\nabla F(x^k)^\top \mathbf{W}_k^{-1} \nabla F(x^k))^\dagger}, \quad (19)$$

where $\mathbf{W}_k \sim \mathcal{D}_{x^k}$. This function is closely related to the SNRVM algorithm. Indeed, it is possible to show that x^{k+1} is obtained by minimizing a quadratic approximation of $\hat{f}_k$ along a random subspace (see Lemma E.2). More precisely, x^{k+1} is the solution of

$$\underset{x \in \mathbb{R}^p}{\text{argmin}} \ \hat{f}_k(x^k) + \langle \nabla \hat{f}_k(x^k), x - x^k \rangle + \frac{1}{2\gamma} \|x - x^k\|^2_{\mathbf{W}_k}$$

$$\text{s.t.} \ x \in x^k + \mathbf{Im}\left(\mathbf{W}_k^{-1} \nabla F(x^k)\mathbf{S}_k\right)$$

Our forthcoming Theorem 4.5 shows that $\hat{f}_k(x^k)$ converges linearly to zero in expectation. To achieve this, we need to make an assumption that controls the evolution of $\hat{f}_k$ along the iterations.

Assumption 4.4. There exists $L > 0$ such that, for every $k \in \mathbb{N}$ and every $x \in \mathbb{R}^p$: $\hat{f}_{k+1}(x) \leq \hat{f}_k(x^k) + \langle \nabla \hat{f}_k(x^k), x - x^k \rangle + \frac{L}{2} \|x - x^k\|^2_{\mathbf{W}_k}$.

We now state our core convergence result, which we prove in Section E.4 in the Appendix.

Theorem 4.5. Let Assumptions 4.1, 4.2 and 4.4 hold, and let $\gamma = 1/L$. Let $(\mathbf{S}, \mathbf{W}) \sim \mathcal{D}_x$ and let

$$\mathbf{H}(x) \stackrel{\text{def}}{=} \mathbb{E}\left[\mathbf{S}\left(\mathbf{S}^\top \nabla F(x)^\top \mathbf{W}^{-1} \nabla F(x)\mathbf{S}\right)^\dagger \mathbf{S}^\top\right],$$

$$\rho(x) \stackrel{\text{def}}{=} \min_{i=1,\dots,r} \lambda_{\min}^+\left(\mathbf{M}_i \mathbf{H}(x)\mathbf{M}_i^\top\right),$$

where $\mathbf{M}_i \stackrel{\text{def}}{=} \mathbf{W}_i(x^k)^{-\frac{1}{2}} \nabla F(x)$. Assume that there exists $\rho > 0$ such that $\inf_{k \in \mathbb{N}} \rho(x^k) \geq \rho$ almost surely. It follows that

$$\mathbb{E}\left[\hat{f}_k(x^k)\right] \leq (1 - \rho\gamma)^k \mathbb{E}\left[\hat{f}_0(x^0)\right] \text{ a.s.}$$

When the metric is constant along iterations and $F(x)$ is a linear function, or equivalently our original problem (1) is a quadratic, then the SNRVM method (18)

is known as the sketch-and-project method (Gower and Richtárik, 2015a). In Section E.5, we show that Theorem 4.5 when specialized to this case allows us to recover the well known convergence rates for solving linear systems using sketch-and-project.

The existence of a lower bound $\rho > 0$ in Theorem 4.5 can be guaranteed, provided that we can uniformly control the matrices $\mathbf{S}, \mathbf{W}$ and $\nabla F(x)$. Let us make this more precise:

Assumption 4.6. Assumption 4.2 holds, and $m = p$. We assume that there exists a set $\Omega \subset \mathbb{R}^p$ and constants $\mu_W, L_W, \bar{\mu}_S, L_S, \mu_{\nabla F}, L_{\nabla F}$ in $(0, +\infty)$ such that, for all $x \in \Omega$, for all $(\mathbf{S}, \mathbf{W}) \sim \mathcal{D}_x$,

$$\text{spec}(\mathbf{W}) \subset [\mu_W, L_W], \quad \sigma(\nabla F(x)) \subset [\mu_{\nabla F}, L_{\nabla F}]$$
$$\|\mathbf{SS}^\top\| \leq L_S, \quad \bar{\mu}_S \leq \lambda_{min}\left(\mathbb{E}\left[\mathbf{SS}^\top\right]\right).$$

where $\text{spec}(M)$ (resp. $\sigma(M)$) denote the set of eigenvalues (resp. singular values) of a square matrix M.

This assumption not only allows us to explicit the rates, but also to obtain rates in terms of $F(x^k)$ itself, instead of the surrogate $\hat{f}_k(x^k)$.

Theorem 4.7. Let Assumptions 4.1, 4.4, and 4.6 hold, and let $\gamma = 1/L$. Let $\{x^k\}_{k \in \mathbb{N}}$ be generated by the SNRVM algorithm, and suppose that $x^k \in \Omega$ almost surely. Then for all $k \in \mathbb{N}$ we have that

$$\frac{1}{2}\mathbb{E}\left[\|F(x^k)\|^2\right] \leq C(1 - \gamma\rho)^k \quad \text{almost surely,}$$

with $\rho = \frac{\mu_{\nabla F}^2}{L_{\nabla F}^2} \frac{\mu_W}{L_W} \frac{\bar{\mu}_S}{L_S}$, and $C = \mathbb{E}\left[\hat{f}_0(x^0)\right] \frac{L_{\nabla F}^2}{\mu_W}$.

Assumption 4.6 can be ensured for SAN and SANA under reasonable assumptions, like the regularity of the functions f_i, or the boundedness of the sequence of iterates, which we observe in practice. We summarize this in the next Theorem, which is our main convergence result for SAN and SANA.

Theorem 4.8. Let Assumptions 1.1 and 4.4 hold. Let $\{x^k\}_{k \in \mathbb{N}}$ be a sequence generated by SAN with $\pi = 1/(n+1)$, or by SANA, with $\gamma = 1/L$. Suppose that *one* of the following statements is true:

a) there exists $0 < \mu_f \leq L_f$ such that every function f_i is μ_f-strongly convex and has a L_f-Lipschitz continuous gradient;

b) $\{x^k\}_{k \in \mathbb{N}}$ is bounded almost surely.

Then there exists $C, \rho > 0$ such that:

$$\mathbb{E}\left[\|F(x^k)\|^2\right] \leq C(1 - \gamma\rho)^k \text{ a.s.}$$

If a) holds, we can take $\rho = \frac{\min\{1,\mu_f^3\}}{14n^3(2+L_f^2)^2\max\{1,L_f^3\}}$, and $C = 6n\mathbb{E}\left[\hat{f}_0(x^0)\right]\frac{\max\{1,L_f^2\}(2+L_f^2)}{\min\{1,\mu_f\}}$.

The resulting linear rate of convergence ρ in Theorem 4.8 depends on n and converges to 1 as n goes to infinity. This is not surprising, since it is also the case for variance reduced methods such as SAGA Defazio et al. (2014) and SVRG Johnson and Zhang (2013). We also note that our theoretical rate has a much worse dependence in μ_f, L_f and n than the rates of SVRG and SAGA, because of the presence of exponents greater than 1. This might suggest that our analysis is not tight: indeed we observed empirically that SAN performs as well as SVRG and SAGA, even in a regime where n is large and the problem is severely ill-conditioned (see Table 2 for more details).

5   Conclusion

We introduced the use of a subsampled Newton Raphson method applied to a specific function splitting problem as a tool for designing new incremental Newton methods. We showcase this by developing SAN, an average Newton method that is empirically highly competitive as compared to variance reduced gradient methods, and does not require parameter tuning. Further venues of investigation include:

- Improving our theoretical analysis, to obtain rates that better matches the ones of usual variance reduced methods, motivated by our numerical experiments.
- Leveraging SNRVM's structure to design a more efficient variant of SAN including mini-batching. This should be simple thanks to our function splitting point of view, as we would simply sample many rows at once in (5).
- Exploring different sketching techniques together with original and alternative splitting point of views to design methods that have yet to be discovered.

References

Mark Schmidt, Nicolas Le Roux, and Francis Bach. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 162(1):83–112, Mar 2017.

Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems 26*, pages 315–323. Curran Associates, Inc., 2013.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015*, 2015.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, Lille, France, 07–09 Jul 2015. PMLR.

Siyuan Ma, Raef Bassily, and Mikhail Belkin. The power of interpolation: Understanding the effectiveness of SGD in modern over-parametrized learning. In *ICML*, volume 80 of *JMLR Workshop and Conference Proceedings*, pages 3331–3340, 2018.

Sharan Vaswani, Francis Bach, and Mark W. Schmidt. Fast and faster convergence of SGD for over-parameterized models and an accelerated perceptron. In *AISTATS 2019*, pages 1195–1204, 2019.

Robert M. Gower, Othmane Sebbouh, and Nicolas Loizou. Sgd for structured nonconvex functions: Learning rates, minibatching and interpolation. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130, pages 1315–1323. PMLR, 2021.

Farbod Roosta-Khorasani and Michael W. Mahoney. Sub-sampled newton methods. *Math. Program.*, 174 (1-2):293–326, 2019.

Raghu Bollapragada, Richard H Byrd, and Jorge Nocedal. Exact and inexact subsampled Newton methods for optimization. *IMA Journal of Numerical Analysis*, 39(2):545–578, 04 2018.

Yang Liu and Fred Roosta. Convergence of newton-mr under inexact hessian information. *SIAM J. Optim.*, 31(1):59–90, 2021.

Murat A. Erdogdu and Andrea Montanari. Convergence rates of sub-sampled Newton methods. In *Advances in Neural Information Processing Systems 28*, pages 3052–3060. Curran Associates, Inc., 2015.

Jonas Moritz Kohler and Aurélien Lucchi. Sub-sampled cubic regularization for non-convex optimization. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 1895–1904, 2017.

Richard H Byrd, Gillian M Chin, Will Neveitt, and Jorge Nocedal. On the use of stochastic Hessian information in optimization methods for machine learning. *SIAM Journal on Optimization*, 21(3):977–995, 2011.

Aryan Mokhtari and Alejandro Ribeiro. Global convergence of online limited memory BFGS. *The Journal of Machine Learning Research*, 16:3151–3181, 2015.

Philipp Moritz, Robert Nishihara, and Michael I. Jordan. A linearly-convergent stochastic L-BFGS al-

gorithm. In *International Conference on Artificial Intelligence and Statistics*, volume 51, pages 249–258, 2016.

Robert M. Gower, Donald Goldfarb, and Peter Richtárik. Stochastic block BFGS: Squeezing more curvature out of data. *Proceedings of the 33rd International Conference on Machine Learning*, 2016.

Zheng Qu, Peter Richtárik, Martin Takáč, and Olivier Fercoq. SDNA: Stochastic dual Newton ascent for empirical risk minimization. In *Proceedings of the 33rd International Conference on Machine Learning*, 2016.

Mert Pilanci and Martin J. Wainwright. Newton sketch: A near linear-time optimization algorithm with linear-quadratic convergence. *SIAM Journal on Optimization*, 27(1):205–245, 2017.

Naman Agarwal, Brian Bullins, and Elad Hazan. Second-order stochastic optimization for machine learning in linear time. *Journal of Machine Learning Research*, 18(116):1–40, 2017.

Aryan Mokhtari, Mark Eisen, and Alejandro Ribeiro. Iqn: An incremental quasi-newton method with local superlinear convergence rate. *SIAM Journal on Optimization*, 28(2):1670–1698, 2018. doi: 10.1137/17M1122943.

Zhangyang Gao and Alejandro Ribeiro. Incremental greedy bfgs: An incremental quasi-newton method with explicit superlinear rate. 2020.

Dmitry Kovalev, Konstantin Mishchenko, and Peter Richtarik. Stochastic Newton and cubic Newton methods with simple local linear-quadratic rates. *arxiv:1912.01597*, 2019.

Anton Rodomanov and Dmitry Kropotov. A superlinearly-convergent proximal newton-type method for the optimization of finite sums. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 2597–2605. PMLR, 20–22 Jun 2016.

Rui Yuan, Alessandro Lazaric, and Robert M. Gower. Sketched newton-raphson, 2021.

Robert Mansel Gower and Peter Richtárik. Randomized iterative methods for linear systems. *SIAM Journal on Matrix Analysis and Applications*, 36(4): 1660–1690, 2015a.

J M Ortega and W C Rheinboldt. *Iterative Solution of Nonlinear Equations in Several Variables*. Society for Industrial and Applied Mathematics, 2000.

James Renegar. *A mathematical view of interior-point methods in convex optimization*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2001. ISBN 0-89871-502-4.

Donald Goldfarb. A family of variable-metric methods derived by variational means. *Mathematics of Computation*, 24(109):23–26, 1970.

Roland W. Freund, Gene H. Golub, and Noel M. Nachtigal. Iterative solution of linear systems. 1992. doi: 10.1017/S0962492900002245.

Bruce Christianson. Automatic Hessians by reverse accumulation. *IMA Journal of Numerical Analysis*, 12(2):135–150, 1992.

Aaron Defazio, Francis Bach, and Simon Lacoste-julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems 27*, pages 1646–1654. 2014.

Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2: 27:1–27:27, 2011.

Rami Mohammad, Fadi Thabtah, and T. Mccluskey. An assessment of features related to phishing websites using an automated technique. In *2012 International Conference for Internet Technology and Secured Transactions (ICITST 2012)*, pages 492–497. IEEE, 01 2012. ISBN 978-1-4673-5325-0.

Chih-Chung Chang and Chih-Jen Lin. Ijcnn 2001 challenge: generalization ability and text decoding. In *IJCNN'01. International Joint Conference on Neural Networks. Proceedings (Cat. No.01CH37222)*, volume 2, pages 1031–1036 vol.2, 2001. doi: 10.1109/IJCNN.2001.939502.

Jock Blackard and Denis Dean. Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables. *Computers and Electronics in Agriculture*, 24:131–151, 1999.

De Wang, Danesh Irani, and Calton Pu. Evolutionary study of web spam: Webb spam corpus 2011 versus webb spam corpus 2006. In *Proc. of 8th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom 2012)*, October 2012 .

D. D. Lewis, Y. Yang, T. G. Rose, and F. Li. Rcv1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5 (Apr):361–397, 2004.

Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python.

Journal of Machine Learning Research, 12:2825–2830, 2011.

Kimon Fountoulakis and Jacek Gondzio. A second-order method for strongly convex l1 -regularization problems. *Math. Program.*, 156(1-2):189–219, 2016.

Anton Rodomanov and Yurii Nesterov. Greedy quasi-newton methods with explicit superlinear convergence. *SIAM Journal on Optimization*, 31(1):785–811, 2021a. doi: 10.1137/20M1320651.

Anton Rodomanov and Yurii E. Nesterov. New results on superlinear convergence of classical quasi-newton methods. *J. Optim. Theory Appl.*, 188(3):744–769, 2021b. doi: 10.1007/s10957-020-01805-8.

Anton Rodomanov and Yurii Nesterov. Rates of superlinear convergence for classical quasi-newton methods. *Mathematical Programming*, Feb 2021c. ISSN 1436-4646. doi: 10.1007/s10107-021-01622-5.

Qiujiang Jin and Aryan Mokhtari. Non-asymptotic superlinear convergence of standard quasi-newton methods, 2021.

Adi Ben-Israel and A. Charnes. Contributions to the theory of generalized inverses. *Journal of the Society for Industrial and Applied Mathematics*, 11(3):667–699, 1963.

R. Penrose. A generalized inverse for matrices. *Mathematical Proceedings of the Cambridge Philosophical Society*, 51(3):406–413, 1955. ISSN 1469-8064, 0305-0041. doi: 10.1017/S0305004100030401.

Robert Gower, Dmitry Koralev, Felix Lieder, and Peter Richtarik. Rsn: Randomized subspace newton. In *Advances in Neural Information Processing Systems 32*, pages 614–623. Curran Associates, Inc., 2019.

Robert M. Gower and Peter Richtárik. Stochastic dual ascent for solving linear systems. *arXiv:1512.06890*, 2015b.

D. Leventhal and A. S. Lewis. Randomized methods for linear constraints: Convergence rates and conditioning. *Mathematics of Operations Research*, 35(3):641–654, 2010.

Appendix

Table of Contents

The Appendix is organized as follows: In Section A, we carefully derive the closed form updates of SAN and SANA presented in Algorithm 1 and 2. In Section B, we specialize SAN and SANA for the case of regularized generalized linear models and provide more detailed and efficient pseudo-codes for such case. In Sections C, we give further details on the numerical experiments and provide additional experiments for SANA to compare with SNM. In Section D and E, we provide the proofs for the claims and results in Section 4.

A   A closed form expression for SAN and SANA

In this section, we show that the updates of the SAN method given in Algorithm 1 are equivalent to the implicit formulation in (10)-(11). We then derive the closed form updates of the SANA method in Section A.2. Finally in Section A.3, we provide a useful lemma. It provides an alternatively way to directly deduce the closed form updates (10) and (11) of SAN.

A.1   Closed form expression for SAN

We start with the following technical lemma.

Lemma A.1. Let $j \in \{1, \ldots, n\}$. Let $\hat{w} \in \mathbb{R}^d$, and $\hat{\alpha}_1, \ldots, \hat{\alpha}_n \in \mathbb{R}^d$. Let $c_j \in \mathbb{R}^d$, and $\mathbf{H}_j \in \mathbb{R}^{d \times d}$ be a symmetric positive definite matrix. The optimization problem

$$\min_{w, \alpha_1, \ldots, \alpha_n \in \mathbb{R}^d} \quad \frac{1}{2} \sum_{i=1}^{n} \|\alpha_i - \hat{\alpha}_i\|^2 + \frac{1}{2} \|w - \hat{w}\|_{\mathbf{H}_j}^2$$

$$\text{subject to} \qquad \mathbf{H}_j(w - \hat{w}) - \alpha_j = c_j,$$

has a unique solution $(w, \alpha_1, \ldots, \alpha_n)$ given by

$$
\begin{aligned}
w &= \hat{w} + (\mathbf{I}_d + \mathbf{H}_j)^{-1}(c_j + \hat{\alpha}_j), \\
\alpha_j &= \hat{\alpha}_j - (\mathbf{I}_d + \mathbf{H}_j)^{-1}(c_j + \hat{\alpha}_j), \\
\alpha_i &= \hat{\alpha}_i \text{ for } i \neq j.
\end{aligned}
$$

Proof. Denoting $x = (w, \alpha_1, \ldots, \alpha_n)$, let us define

$$\Phi(x) \stackrel{\text{def}}{=} \frac{1}{2} \sum_{i=1}^{n} \|\alpha_i - \hat{\alpha}_i\|^2 + \frac{1}{2} \|w - \hat{w}\|_{\mathbf{H}_j}^2, \quad \text{and} \quad \Psi_j(x) \stackrel{\text{def}}{=} \mathbf{H}_j(w - \hat{w}) - \alpha_j - c_j. \quad (20)$$

The fact that $\mathbf{H}_j$ is positive definite implies that Φ is strongly convex. Moreover Ψ_j is affine, so we deduce that this problem has a unique solution. Moreover, this solution, let us call it $x = (w, \alpha_1, \ldots, \alpha_n)$, is characterized as

the unique vector in $\mathbb{R}^{(n+1)d}$ satisfying the following KKT conditions:

$$(\exists \beta_j \in \mathbb{R}^d) \quad \text{such that} \quad \begin{cases} \nabla \Phi(x) + \nabla \Psi_j(x) \beta_j = 0, \\ \Psi_j(x) = 0. \end{cases} \tag{21}$$

The derivatives in the above KKT conditions are given by

$$\nabla \Phi(x) = \begin{bmatrix} \mathbf{H}_j(w - \hat{w}) \\ \alpha_1 - \hat{\alpha}_1 \\ \vdots \\ \alpha_n - \hat{\alpha}_n \end{bmatrix} \quad \text{and} \quad \nabla \Psi_j(x) = \begin{bmatrix} \mathbf{H}_j \\ \mathbf{0}_d \\ \vdots \\ \mathbf{I}_d \\ \vdots \\ \mathbf{0}_d \end{bmatrix} \quad \leftarrow j+1 \tag{22}$$

Using the expression for these derivatives, we can rewrite the KKT conditions (21) as

$$(\exists \beta_j \in \mathbb{R}^d) \quad \text{such that} \quad \begin{cases} \mathbf{H}_j(w - \hat{w}) + \mathbf{H}_j \beta_j = 0, \\ \alpha_j - \hat{\alpha}_j - \beta_j = 0, \\ \alpha_i - \hat{\alpha}_i + 0 = 0, \text{ for all } i \neq j \\ \mathbf{H}_j(w - \hat{w}) - \alpha_j - c_j = 0. \end{cases} \tag{23}$$

We immediately see that $\alpha_i = \hat{\alpha}_i$ for $i \neq j$. Combining the second and fourth equations in (23), we obtain

$$\beta_j = \alpha_j - \hat{\alpha}_j = \mathbf{H}_j(w - \hat{w}) - c_j - \hat{\alpha}_j.$$

Multiplying this new equality by $\mathbf{H}_j$ allows us to rewrite the first equation in (23) as:

$$\mathbf{H}_j(w - \hat{w}) + \mathbf{H}_j \beta_j = 0 \Leftrightarrow \mathbf{H}_j(w - \hat{w}) + \mathbf{H}_j^2(w - \hat{w}) = \mathbf{H}_j(c_j + \hat{\alpha}_j).$$

Using the fact that $\mathbf{H}_j$ is invertible, the latter is equivalent to write:

$$w = \hat{w} + (\mathbf{I}_d + \mathbf{H}_j)^{-1}(c_j + \hat{\alpha}_j).$$

Moreover, since $\mathbf{H}_j(\mathbf{I}_d + \mathbf{H}_j)^{-1} = \mathbf{I}_d - (\mathbf{I}_d + \mathbf{H}_j)^{-1}$, we can also turn the fourth equation in (23) into

$$\alpha_j = \mathbf{H}_j(w - \hat{w}) - c_j = \left(\mathbf{I}_d - (\mathbf{I}_d + \mathbf{H}_j)^{-1}\right)(c_j + \hat{\alpha}_j) - c_j = \hat{\alpha}_j - (\mathbf{I}_d + \mathbf{H}_j)^{-1}(c_j + \hat{\alpha}_j).$$

This proves the claim. $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\square$

Lemma A.2. Let $\pi \in [0, 1]$ and $\gamma \in (0, 1]$ a step size. Algorithm 1 (SAN) is equivalent to the following algorithm:
With probability π, update according to

$$\begin{cases} \bar{x}^{k+1} = \operatorname{argmin} \|w - w^k\|^2 + \sum_{i=1}^{n} \|\alpha_i - \alpha_i^k\|^2 \quad \text{subject to} \quad \frac{1}{n} \sum_{i=1}^{n} \alpha_i = 0, \\ x^{k+1} = (1 - \gamma)x^k + \gamma \bar{x}^{k+1}, \end{cases} \tag{24}$$

Otherwise with probability $(1 - \pi)$, sample $j \sim \{1, \ldots, n\}$ uniformly and update according to

$$\begin{cases} \bar{x}^{k+1} = \operatorname{argmin} \|w - w^k\|_{\nabla^2 f_j(w^k)}^2 + \sum_{i=1}^{n} \|\alpha_i - \alpha_i^k\|^2 \\ \qquad\qquad \text{subject to} \quad \nabla^2 f_j(w^k)(w - w^k) - \alpha_j = -\nabla f_j(w^k), \\ x^{k+1} = (1 - \gamma)x^k + \gamma \bar{x}^{k+1}. \end{cases} \tag{25}$$

Proof. Suppose that we are in the case (which holds with probability π) given by (24). In the projection step, we see that w is not present in the constraint, which implies that $\bar{w}^{k+1} = w^k$, and therefore $w^{k+1} = w^k$. On the other hand, $(\bar{\alpha}_1, \ldots, \bar{\alpha}_n)$ is the projection of $(\alpha_1, \ldots, \alpha_n)$ onto a simple linear constraint, and can be computed in closed form as

$$(\forall i \in \{1, \ldots, n\}) \quad \bar{\alpha}_i^{k+1} = \alpha_i^k - \frac{1}{n} \sum_{i=1}^{n} \alpha_i^k.$$

Consequently

$$(\forall i \in \{1, \ldots, n\}) \quad \alpha_i^{k+1} = (1 - \gamma)\alpha_i^k + \gamma \left(\alpha_i^k - \frac{1}{n} \sum_{i=1}^{n} \alpha_i^k \right) = \alpha_i^k - \frac{\gamma}{n} \sum_{i=1}^{n} \alpha_i^k,$$

which gives us exactly the step 4 in Algorithm 1.

Let now j be in $\{1, \ldots, n\}$ sampled uniformly, and suppose that we are in the case given by (25). Using Lemma A.1 we can compute an explicit form for $\bar{x}^{k+1}$ given by

$$
\begin{aligned}
\bar{w}^{k+1} &= w^k + \left(\mathbf{I}_d + \nabla^2 f_j(w^k)\right)^{-1} (\alpha_j^k - \nabla f_j(w^k)), \\
\bar{\alpha}_j^{k+1} &= \alpha_j^k - \left(\mathbf{I}_d + \nabla^2 f_j(w^k)\right)^{-1} (\alpha_j^k - \nabla f_j(w^k)), \\
\bar{\alpha}_i^{k+1} &= \hat{\alpha}_i^k \text{ for all } i \neq j.
\end{aligned}
$$

Consequently, after applying the relaxation step we have

$$
\begin{aligned}
w^{k+1} &= w^k + \gamma \left(\mathbf{I}_d + \nabla^2 f_j(w^k)\right)^{-1} (\alpha_j^k - \nabla f_j(w^k)), \\
\alpha_j^{k+1} &= \alpha_j^k - \gamma \left(\mathbf{I}_d + \nabla^2 f_j(w^k)\right)^{-1} (\alpha_j^k - \nabla f_j(w^k)), \\
\bar{\alpha}_i^{k+1} &= \hat{\alpha}_i^k \text{ for all } i \neq j.
\end{aligned}
$$

which is exactly the steps 8-10 in Algorithm 1. $\qquad \square$

A.2 Closed form expression for SANA

Lemma A.3. Let $j \in \{1, \ldots, n\}$. Let $c_j \in \mathbb{R}^d$, $\hat{w} \in \mathbb{R}^d$, and let $\hat{\alpha}_1, \ldots, \hat{\alpha}_n \in \mathbb{R}^d$ be such that $\sum_{i=1}^{n} \hat{\alpha}_i = 0$. Let $\mathbf{H}_j \in \mathbb{R}^{d \times d}$ be a positive definite matrix. Then the optimization problem

$$
\begin{aligned}
\min_{w, \alpha_1, \ldots, \alpha_n \in \mathbb{R}^d} \quad & \frac{1}{2} \sum_{i=1}^{n} \|\alpha_i - \hat{\alpha}_i\|^2 + \frac{1}{2} \|w - \hat{w}\|_{\mathbf{H}_j}^2, \\
\text{subject to} \quad & \mathbf{H}_j(w - \hat{w}) - \alpha_j = c_j, \\
& \sum_{i=1}^{n} \alpha_i = 0,
\end{aligned}
\tag{26}
$$

has a unique solution $(w, \alpha_1, \ldots, \alpha_n)$ given by

$$
\begin{aligned}
d &= \left(\frac{n-1}{n}\mathbf{I}_d + \mathbf{H}_j\right)^{-1} (c_j + \hat{\alpha}_j), \\
w &= \hat{w} + d, \\
\alpha_j &= \hat{\alpha}_j - \frac{n-1}{n}d, \\
\alpha_i &= \hat{\alpha}_i + \frac{1}{n}d, \quad \text{for } i \neq j.
\end{aligned}
$$

Proof. Noting $x = (w, \alpha_1, \ldots, \alpha_n)$, let us define $\Phi(x)$ and $\Psi_j(x)$ as in (20), together with

$$\Psi_0(x) \stackrel{\text{def}}{=} \sum_{i=1}^{n} \alpha_i.$$

The fact that $\mathbf{H}_j$ is positive definite implies that we are minimizing a strongly convex function over a set of affine equations. We deduce that this problem has a unique solution. Moreover, this solution, let us call it $x = (w, \alpha_1, \ldots, \alpha_n)$, is characterized as the unique vector in $\mathbb{R}^{(n+1)d}$ satisfying the following KKT conditions:

$$(\exists \beta_0, \beta_j \in \mathbb{R}^d) \quad \text{such that} \quad \begin{cases} \nabla \Phi(x) + \nabla \Psi_0(x)\beta_0 + \nabla \Psi_j(x)\beta_j = 0, \\ \Psi_0(x) = 0, \\ \Psi_j(x) = 0. \end{cases}$$

Here, we can compute $\nabla \Phi(x)$ and $\nabla \Psi_j(x)$ as in (22), together with

$$\nabla \Psi_0(x) = \begin{bmatrix} \mathbf{0}_d \\ \mathbf{I}_d \\ \vdots \\ \mathbf{I}_d \end{bmatrix}.$$

Therefore, we can rewrite the KKT conditions as

$$(\exists \beta_0, \beta_j \in \mathbb{R}^d) \quad \text{such that} \quad \begin{cases} \mathbf{H}_j(w - \hat{w}) + 0 + \mathbf{H}_j\beta_j = 0, \\ \alpha_j - \hat{\alpha}_j + \beta_0 - \beta_j = 0, \\ \alpha_i - \hat{\alpha}_i + \beta_0 + 0 = 0, \text{ for all } i \neq j \\ \sum_{i=1}^n \alpha_i = 0 \\ \mathbf{H}_j(w - \hat{w}) - \alpha_j - c_j = 0. \end{cases} \quad (27)$$

The last equation in (27) can be rewritten as

$$\alpha_j = \mathbf{H}_j(w - \hat{w}) - c_j. \quad (28)$$

Summing the equations involving α_i for $i \neq j$, and using the fact that $\sum_{i=1}^n \alpha_i = \sum_{i=1}^n \hat{\alpha}_i = 0$, together with (28), we can deduce that

$$0 = \sum_{i \neq j}(\alpha_i - \hat{\alpha}_i + \beta_0) = -\alpha_j + \hat{\alpha}_j + (n-1)\beta_0 = -\mathbf{H}_j(w - \hat{w}) + c_j + \hat{\alpha}_j + (n-1)\beta_0.$$

In other words, we obtain that

$$\beta_0 = \frac{1}{n-1}\left(\mathbf{H}_j(w - \hat{w}) - (c_j + \hat{\alpha}_j)\right). \quad (29)$$

Injecting the above expression into the second equation of (27), and using again (28), gives

$$\begin{aligned} \beta_j &= \mathbf{H}_j(w - \hat{w}) - c_j - \hat{\alpha}_j + \frac{1}{n-1}\left(\mathbf{H}_j(w - \hat{w}) - (c_j + \hat{\alpha}_j)\right) \\ &= \frac{n}{n-1}\left(\mathbf{H}_j(w - \hat{w}) - (c_j + \hat{\alpha}_j)\right) \end{aligned}$$

Combining this expression of β_j with the first equation in (27) leads to

$$\begin{aligned} 0 &= \mathbf{H}_j(w - \hat{w}) + \frac{n}{n-1}\mathbf{H}_j\left(\mathbf{H}_j(w - \hat{w}) - (c_j + \hat{\alpha}_j)\right) \\ &= \mathbf{H}_j\left(\left(\mathbf{I}_d + \frac{n}{n-1}\mathbf{H}_j\right)(w - \hat{w}) - \frac{n}{n-1}(c_j + \hat{\alpha}_j)\right) \end{aligned}$$

Using the fact that $\mathbf{H}_j$ is positive definite, we obtain that:

$$\begin{aligned} w &= \hat{w} + \frac{n}{n-1}\left(\mathbf{I}_d + \frac{n}{n-1}\mathbf{H}_j\right)^{-1}(c_j + \hat{\alpha}_j) \\ &= \hat{w} + \left(\frac{n-1}{n}\mathbf{I}_d + \mathbf{H}_j\right)^{-1}(c_j + \hat{\alpha}_j), \\ &= \hat{w} + d, \end{aligned}$$

where d is defined as $\left(\frac{n-1}{n}\mathbf{I}_d + \mathbf{H}_j\right)^{-1}(c_j + \hat{\alpha}_j)$.

Going back now to (28) we can write

$$
\begin{aligned}
\alpha_j &= \mathbf{H}_j d - c_j = \mathbf{H}_j \left(\frac{n-1}{n}\mathbf{I}_d + \mathbf{H}_j\right)^{-1}(c_j + \hat{\alpha}_j) - c_j \\
&= \left(\mathbf{I}_d - \frac{n-1}{n}\left(\frac{n-1}{n}\mathbf{I}_d + \mathbf{H}_j\right)^{-1}\right)(c_j + \hat{\alpha}_j) - c_j \\
&= \hat{\alpha}_j - \frac{n-1}{n}\left(\frac{n-1}{n}\mathbf{I}_d + \mathbf{H}_j\right)^{-1}(c_j + \hat{\alpha}_j) \\
&= \hat{\alpha}_j - \frac{n-1}{n}d.
\end{aligned}
$$

It remains to compute α_i, for $i \neq j$. Start with the first equation of (27) and see that $w - \hat{w} + \beta_j = 0$. This implies that $\beta_j = -d$. We can therefore use the second equation of (27) to write that

$$
\beta_0 = \beta_j - (\alpha_j - \hat{\alpha}_j) = -d + \frac{n-1}{n}d = -\frac{1}{n}d.
$$

We can finally call the third equation of (27) and write that $\alpha_i = \hat{\alpha}_i - \beta_0 = \hat{\alpha}_i + \frac{1}{n}d$. $\qquad\square$

Lemma A.4. Let $\gamma \in (0, 1]$ be a step size. Algorithm 2 (SANA) is equivalent to the following algorithm: update the iterates according to

$$
\begin{cases}
\bar{x}^{k+1} = & \operatorname{argmin} \|w - w^k\|^2_{\nabla^2 f_j(w^k)} + \sum_{i=1}^{n}\|\alpha_i - \alpha_i^k\|^2 \\
& \text{subject to} \begin{cases} \frac{1}{n}\sum_{i=1}^{n}\alpha_i = 0, \\ \nabla^2 f_j(w^k)(w - w^k) - \alpha_j = -\nabla f_j(w^k), \end{cases} \\
x^{k+1} = & (1 - \gamma)x^k + \gamma\bar{x}^{k+1}.
\end{cases}
\tag{30}
$$

Proof. Consider the iterates defined by (30). Using Lemma A.3, we can compute an explicit form for $\bar{x}^{k+1}$:

$$
\begin{aligned}
d^k &= \left(\frac{n-1}{n}\mathbf{I}_d + \nabla^2 f_j(w^k)\right)^{-1}(\alpha_j^k - \nabla f_j(w^k)), \\
\bar{w}^{k+1} &= w^k + d^k, \\
\bar{\alpha}_j^{k+1} &= \alpha_j^k - \frac{n-1}{n}d^k, \\
\bar{\alpha}_i^{k+1} &= \alpha_i^k + \frac{1}{n}d^k, \quad \text{for } i \neq j.
\end{aligned}
$$

After applying the relaxation step $x^{k+1} = (1 - \gamma)x^k + \gamma\bar{x}^{k+1}$, we obtain exactly the steps 5-8 in Algorithm 2. $\quad\square$

A.3 Generic projection onto linear systems

Here we provide a useful lemma that can directly deduce the closed form updates of (10) and (11) of SAN. It will also be used later in the appendix.

Lemma A.5. Let $\mathbf{A} \in \mathbb{R}^{n \times d}$, $\mathbf{S} \in \mathbb{R}^{n \times \tau}$, $b \in \mathbf{Im}(\mathbf{A})$, and $\mathbf{H}$ be a symmetric positive definite matrix. The optimization problem

$$
\begin{aligned}
x^* &= \arg\min_{x \in \mathbb{R}^d} \quad \frac{1}{2}\|x\|^2_{\mathbf{H}}, \\
&\text{subject to } \mathbf{S}^\top \mathbf{A} x = \mathbf{S}^\top b,
\end{aligned}
$$

has a unique solution, called the weighted sketch-and-project optimal solution:

$$x^* = \mathbf{H}^{-1}\mathbf{A}^\top \mathbf{S}(\mathbf{S}^\top \mathbf{A}\mathbf{H}^{-1}\mathbf{A}^\top \mathbf{S})^\dagger \mathbf{S}^\top b. \tag{31}$$

Proof. First, note that this problem is strongly convex because $\mathbf{H}$ is supposed to be positive definite, and therefore admits a unique solution $x^* \in \mathbb{R}^d$. Second, since $\mathbf{H}$ is invertible, we can do the change of variables $y \overset{\text{def}}{=} \mathbf{H}^{1/2}x$. This allows us to write that $x^* = (\mathbf{H})^{-1/2} y^*$ where y^* is the unique solution of

$$\arg\min_{y\in\mathbb{R}^d} \frac{1}{2} \|y\|^2 ,$$

$$\text{subject to } \mathbf{S}^\top \mathbf{A}\mathbf{H}^{-1/2}y = \mathbf{S}^\top b.$$

The unique solution to the above problem is the minimal-norm solution of the linear system $\mathbf{S}^\top \mathbf{A} (\mathbf{H})^{-1/2} y = \mathbf{S}^\top b$, which can be simply expressed by using the pseudo-inverse (Ben-Israel and Charnes, 1963, Definition 1) :

$$y^* = \left(\mathbf{S}^\top \mathbf{A}\mathbf{H}^{-1/2}\right)^\dagger \mathbf{S}^\top b.$$

Using the relation $\mathbf{M}^\dagger = \mathbf{M}^\top (\mathbf{M}\mathbf{M}^\top)^\dagger$ (Penrose, 1955, Lemma 1 & Eq. 10), we obtain

$$y^* = (\mathbf{H})^{-1/2} \mathbf{A}^\top \mathbf{S}(\mathbf{S}^\top \mathbf{A}\mathbf{H}^{-1}\mathbf{A}^\top \mathbf{S})^\dagger \mathbf{S}^\top b.$$

Multiplying this equality by $\mathbf{H}^{-1/2}$ gives us the desired expression for x^*. □

This lemma is useful. Later it will be applied in Lemma D.1 and consequently provide the explicit updates of (10) and (11) of SAN in Section D.2. Thus this is a different way to obtain the closed form updates of SAN, compared to Section A.1.

B   Implementations for regularized GLMs

B.1   Definition and examples

Here we specify our algorithms for the case of regularized generalized linear models. Throughout this section, we assume that our finite sum minimization problem (1) is a GLM (generalized linear model) defined as follows.

Assumption B.1 (Regularized GLM). Our problem (1) writes as

$$\min_{w\in\mathbb{R}^d} \frac{1}{n} \sum_{i=1}^n f_i(w) \overset{\text{def}}{=} \phi_i(\langle a_i, w\rangle) + R(w), \tag{32}$$

where $\{a_i\}_{i=1}^n \subset \mathbb{R}^d$ are *data points*, $\{\phi_i\}_{i=1}^n$ are twice differentiable real convex *loss* functions with $\phi_i''(t) > 0$, and R is a separable *regularizer* with $R(w) = \sum_{j=1}^d R_j(w_j)$ where R_j is a twice differentiable real convex function with $R_j''(t) > 0$, for all $t \in \mathbb{R}$.

Some classic examples of GLMs include ridge regression where $\phi_i(t) = \frac{1}{2}(t - y_i)^2$ and $R_j(t) = \frac{\lambda}{2}t^2$ where $\lambda > 0$ is a regularization parameter. L2-regularized logistic regression, the example on which we perform most of our experiments, is also a GLM with

$$\phi_i(t) = \ln\left(1 + e^{-y_i t}\right) \quad \text{and} \quad R_j(t) = \frac{\lambda}{2}t^2. \tag{33}$$

We also consider other forms of separable regularizers such as the pseudo-huber regularizer where $R_j(t) = \lambda\delta^2 \left(\sqrt{1 + \left(\frac{t}{\delta}\right)^2} - 1\right)$ where δ is a parameter.

In the next section, we will show that for GLMs, our methods can be efficiently implemented. But first we need the following preliminary results.

Lemma B.2 (Simple computations with Regularized GLMs)**.** For GLMs (Assumption B.1) we have for all $j \in \{1, \ldots, n\}$, all $w \in \mathbb{R}^d$ and every $\mu \geq 0$ that

(1) $\quad \begin{aligned} \nabla R(w) &= [R_1'(w_1) \ldots R_d'(w_d)]^\top, \\ \nabla^2 R(w) &= \mathbf{Diag}\left(R_1''(w_1), \ldots, R_d''(w_d)\right). \end{aligned}$

(2) $\quad \begin{aligned} \nabla f_j(w) &= \nabla R(w) + \phi_j'(\langle a_j, w \rangle) a_j, \\ \nabla^2 f_j(w) &= \nabla^2 R(w) + \phi_j''(\langle a_j, w \rangle) a_j a_j^\top. \end{aligned}$

(3) With $\hat{a}_j := \left(\mu \mathbf{I}_d + \nabla^2 R(w_k)\right)^{-1} a_j$, we have

$$\left(\mu \mathbf{I}_d + \nabla^2 f_j(w)\right)^{-1} = \left(\mu \mathbf{I}_d + \nabla^2 R(w_k)\right)^{-1} - \frac{\phi_j''(\langle a_j, w \rangle)}{1 + \phi_j''(\langle a_j, w \rangle)\langle \hat{a}_j, a_j \rangle} \hat{a}_j \hat{a}_j^\top.$$

(4) If $R(w) = \frac{\lambda}{2}\|w\|^2$, with $\lambda > 0$, then

$$\left(\mu \mathbf{I}_d + \nabla^2 f_j(w)\right)^{-1} = \frac{1}{\mu + \lambda}\left(\mathbf{I}_d - \frac{\phi_j''(\langle a_j, w \rangle)}{\mu + \lambda + \phi_j''(\langle a_j, w \rangle)\|a_j\|^2} a_j a_j^\top\right).$$

Proof. (1) and (2) are trivial. For (3), let $\Phi := \phi_j''(\langle a_j, w \rangle)$, which is nonnegative because of the Assumption B.1. Consider now the Sherman–Morrison formula:

$$(\mathbf{M} + uu^\top)^{-1} = \mathbf{M}^{-1} - \frac{1}{1 + \langle \mathbf{M}^{-1}u, u \rangle}(\mathbf{M}^{-1}u)(\mathbf{M}^{-1}u)^\top.$$

This allows us to write, for $\mathbf{D} = (\mu \mathbf{I}_d + \nabla^2 R(w))^{-1}$ and $\mathbf{M} = \Phi^{-1}(\mu \mathbf{I}_d + \nabla^2 R(w))$, that

$$\begin{aligned} \left(\mu \mathbf{I}_d + \nabla^2 f_j(w)\right)^{-1} &= \left(\mu \mathbf{I}_d + \nabla^2 R(w) + \Phi a_j a_j^\top\right)^{-1} = \Phi^{-1}\left(\mathbf{M} + a_j a_j^\top\right)^{-1} \\ &= \Phi^{-1}\left(\mathbf{M}^{-1} - \frac{1}{1 + \langle \mathbf{M}^{-1}a_j, a_j \rangle}(\mathbf{M}^{-1}a_j)(\mathbf{M}^{-1}a_j)^\top\right) \\ &= \mathbf{D} - \frac{\Phi}{1 + \Phi\langle \mathbf{D}a_j, a_j \rangle}(\mathbf{D}a_j)(\mathbf{D}a_j)^\top. \end{aligned}$$

(4) is a direct consequence of the fact that $\mathbf{D} = \left(\mu \mathbf{I}_d + \nabla^2 R(w_k)\right)^{-1} = \frac{1}{\mu+\lambda}\mathbf{I}_d$. $\qquad \square$

B.2 SAN with GLMs

Here we give the detailed derivation of our implementation of SAN for GLMs, see Algorithm 3. Upon examination, we can see that every step of Algorithm 3 has a cost of $\mathcal{O}(d)$, except on line 4. As explained in Section 2.1, the averaging cost on line 4 costs $\mathcal{O}(d)$ in which π is of the order of $\mathcal{O}(1/n)$. The only step that we have left an implicit computation is on lines 9 and 10 which require inverting $(\mathbf{I}_d + \nabla^2 R(w^k))$. But this to comes at a cost of $\mathcal{O}(d)$ since in our Assumption B.1 the regularizer is separable, and thus the Hessian is a diagonal matrix whose

inversion also costs $\mathcal{O}(d)$.

Algorithm 3: SAN for regularized GLMs

Input: Data $\{a_i\}_{i=1}^n$, loss functions $\{\phi_i\}_{i=1}^n$, regularizer R, $\pi \in (0, 1)$, step size $\gamma \in (0, 1]$, max iteration T

1   Initialize $\alpha_1^0, \cdots, \alpha_n^0, w^0 \in \mathbb{R}^d$ and $\overline{\alpha}^0 = \frac{1}{n} \sum_{i=1}^n \alpha_i^0$.

2   **for** $k = 0, \ldots, T - 1$ **do**

3   **With probability π update:**

4         $\alpha_i^{k+1} = \alpha_i^k - \gamma \overline{\alpha}^k, \quad$ for all $i \in \{1, \cdots, n\}$

5         $w^{k+1} = w^k$

6   **Otherwise with probability $(1 - \pi)$:**

7   Sample $j \in \{1, \ldots, n\}$ uniformly

8         $g^k = \nabla R(w^k) + \phi_j'(\langle a_j, w^k \rangle) a_j - \alpha_j^k$

9         $\hat{a}_j = (\mathbf{I}_d + \nabla^2 R(w^k))^{-1} a_j$

10        $d^k = \frac{\phi_j''(\langle a_j, w^k \rangle) \langle \hat{a}_j, g^k \rangle}{1 + \phi_j''(\langle a_j, w^k \rangle) \langle \hat{a}_j, a_j \rangle} \hat{a}_j - (\mathbf{I}_d + \nabla^2 R(w^k))^{-1} g^k$

11        $w^{k+1} = w^k + \gamma d^k$

12        $\alpha_j^{k+1} = \alpha_j^k - \gamma d^k$

13        $\alpha_i^{k+1} = \alpha_i^k$ for $i \neq j$

14        $\overline{\alpha}^{k+1} = \overline{\alpha}^k - \frac{\gamma}{n} d^k$

Output: Last iterate w^T

Next we formalize the costs of Algorithm 3 in the following remark. By computational cost, we refer to the total number of floating point operations, that is the number of scalar multiplications and additions.

Remark B.3. The average costs of SAN (Algorithm 3) per iteration under Assumption B.1 are:

- `Memory storage` of $\mathcal{O}(nd)$ scalars.
- `Memory access` of $\mathcal{O}(\pi nd + (1 - \pi)d)$ which is $\mathcal{O}(d)$ when $\pi \simeq 1/n$.
- `Data access` of $\mathcal{O}(1)$.
- `Computational cost` of $\mathcal{O}(\pi dn + (1 - \pi)d)$ which is $\mathcal{O}(d)$ when $\pi \simeq 1/n$.

In calculating the average computational cost per iteration, we used that in expectation the updates on lines 4–5 occur with probability π, while the updates on lines 7–14 occur with probability $(1 - \pi)$.

Lemma B.4. The SAN Algorithm 1 applied to Regularized GLMs (in the sense of Assumption B.1) is Algorithm 3.

Proof. Let $k \in \{0, \ldots, T - 1\}$. With probability π from Algorithm 1 we have

$$\alpha_i^{k+1} \;=\; \alpha_i^k - \frac{\gamma}{n} \sum_{j=1}^n \alpha_j^k, \quad \text{for all } i \in \{1, \cdots, n\}.$$

This can be rewritten as $\alpha_i^{k+1} = \alpha_i^k - \gamma \bar{\alpha}^k$, which is the update on line 4 in Algorithm 3.

With probability $(1 - \pi)$ from Algorithm 1 we have

$$
\begin{aligned}
d^k &= -(\mathbf{I}_d + \nabla^2 f_j(w^k))^{-1} (\nabla f_j(w^k) - \alpha_j^k), \\
w^{k+1} &= w^k + \gamma d^k, \\
\alpha_j^{k+1} &= \alpha_j^k - \gamma d^k.
\end{aligned}
$$

Using Lemma B.2, we see that

$$g^k := \nabla f_j(w^k) - \alpha_j^k = \nabla R(w^k) + \phi_j'(\langle a_j, w^k \rangle) a_j - \alpha_j^k.$$

Still using Lemma B.2, and introducing the notation $\hat{a}_j = (\mathbf{I}_d + \nabla^2 R(w^k))^{-1} a_j$, we see that

$$(\mathbf{I}_d + \nabla^2 f_j(w^k))^{-1} \;=\; (\mathbf{I}_d + \nabla^2 R(w^k))^{-1} - \frac{\phi_j''(\langle a_j, w^k \rangle)}{1 + \phi_j''(\langle a_j, w^k \rangle) \langle \hat{a}_j, a_j \rangle} \hat{a}_j \hat{a}_j^\top.$$

Therefore,

$$d^k = \frac{\phi_j''(\langle a_j, w^k \rangle) \langle \hat{a}_j, g^k \rangle}{1 + \phi_j''(\langle a_j, w^k \rangle) \langle \hat{a}_j, a_j \rangle} \hat{a}_j - \left(\mathbf{I}_d + \nabla^2 R(w^k)\right)^{-1} g^k,$$

which concludes the proof. □

Finally, when the regularizer is the L2 norm, then we can implement SAN even more efficiently as follows.

Example B.5 (Ridge regularization). If $R(w) = \frac{\lambda}{2}\|w\|^2$ with $\lambda > 0$, then the stochastic Newton direction d^k can be computed explicitly (see also Lemma B.2):

$$d^k = \frac{\phi_j''(r^k)}{1 + \lambda} \cdot \frac{\langle a_j, \alpha_j^k \rangle - \phi_j'(r^k)\|a_j\|^2 - \lambda r^k}{1 + \lambda + \phi_j''(r^k)\|a_j\|^2} a_j - \frac{1}{1+\lambda}\left(\lambda w^k + \phi_j'(r^k)a_j - \alpha_j^k\right),$$

where $r^k = \langle a_j, w^k \rangle$.

B.3 SANA with GLMs

In Algorithm 4 we give the specialized implementation of SANA (Algorithm 2) for GLMs.

Algorithm 4: SANA for regularized GLMs

Input: Data $\{a_i\}_{i=1}^n$, loss functions $\{\phi_i\}_{i=1}^n$, regularizer R, step size $\gamma \in (0, 1]$, max iteration T

1 Initialize $\alpha_1^0, \cdots, \alpha_n^0, w^0 \in \mathbb{R}^d$, with $\sum_{i=1}^n \alpha_i^0 = 0$;

2 Pre-compute $\mu = \frac{n-1}{n}$;

3 **for** $k = 0, \ldots, T - 1$ **do**

4   Sample $j \in \{1, \ldots, n\}$ uniformly;

5      $g^k = \nabla R(w^k) + \phi_j'(\langle a_j, w^k \rangle)a_j - \alpha_j^k$

6      $\hat{a}_j = (\mu\mathbf{I}_d + \nabla^2 R(w^k))^{-1} a_j$

7      $d^k = \frac{\phi_j''(\langle a_j, w^k \rangle) \langle \hat{a}_j, g^k \rangle}{1 + \phi_j''(\langle a_j, w^k \rangle) \langle \hat{a}_j, a_j \rangle} \hat{a}_j - \left(\mu\mathbf{I}_d + \nabla^2 R(w^k)\right)^{-1} g^k$

8      $w^{k+1} = w^k + \gamma d^k$

9      $\alpha_j^{k+1} = \alpha_j^k + \gamma\mu d^k$

10     $\alpha_i^{k+1} = \alpha_i^k - \frac{\gamma}{n}d^k,$ for $i \neq j$

Output: Last iterate w^T

Next we formalize the costs of Algorithm 4 in the following remark.

Remark B.6. The costs of SANA (Algorithm 4) per iteration under Assumption B.1 are:

- `Memory storage` of $\mathcal{O}(nd)$ scalars.
- `Memory access` of $\mathcal{O}(nd)$.
- `Data access` of $\mathcal{O}(1)$.
- `Computational cost` of $\mathcal{O}(nd)$.

Lemma B.7. The SANA Algorithm 2 applied to Regularized GLMs (in the sense of Assumption B.1) is Algorithm 4.

Proof. Let $k \in \{0, \ldots, T - 1\}$, and $\mu := 1 - n^{-1}$. Let j be sampled over $\{1, \ldots, n\}$ uniformly. From Algorithm 2 we have

$$
\begin{aligned}
g^k &= \nabla f_j(w^k) - \alpha_j^k, \\
d^k &= -\left(\mu\mathbf{I}_d + \nabla^2 f_j(w^k)\right)^{-1} g^k, \\
w^{k+1} &= w^k + \gamma d^k, \\
\alpha_j^{k+1} &= \alpha_j^k - \gamma\mu d^k, \\
\alpha_i^{k+1} &= \alpha_i^k + \frac{\gamma}{n}d^k, \quad \text{for } i \neq j.
\end{aligned}
$$

Using Lemma B.2, we see that

$$g^k = \nabla R(w^k) + \phi_j'(\langle a_j, w^k \rangle) a_j - \alpha_j^k.$$

Still using Lemma B.2, and introducing the notation $\hat{a}_j = (\mu \mathbf{I}_d + \nabla^2 R(w^k))^{-1} a_j$, we have that

$$\left(\mu \mathbf{I}_d + \nabla^2 f_j(w^k) \right)^{-1} = \left(\mu \mathbf{I}_d + \nabla^2 R(w^k) \right)^{-1} - \frac{\phi_j''(\langle a_j, w^k \rangle)}{1 + \phi_j''(\langle a_j, w^k \rangle)\langle \hat{a}_j, a_j \rangle} \hat{a}_j \hat{a}_j^\top.$$

Therefore,

$$d^k = \frac{\phi_j''(\langle a_j, w^k \rangle)\langle \hat{a}_j, g^k \rangle}{1 + \phi_j''(\langle a_j, w^k \rangle)\langle \hat{a}_j, a_j \rangle} \hat{a}_j - \left(\mu \mathbf{I}_d + \nabla^2 R(w^k) \right)^{-1} g^k,$$

which concludes the proof.

$\square$

C   Experimental details in Section 3 and additional experiments

We present the details of the experiments in Section 3, in order to guide readers to reproduce the exact same results in Figure 1 and Figure 2. We also explain some grid search results about sensitivity of hyperparameters in Section C.3, showing in particular that SAN does not require parameter tuning. Then we provide additional experiments for SANA and SNM in Section C.4 which are not included in our main paper. These experimental results support that SANA introduced in Section 2.2 is also a reasonable method. Finally, we provide experiments to compare SAN and SAN without variable metric in Section C.5 to illustrate the importance of such variable metric.

C.1   Experimental details in Section 3

All experiments in Section 3 were run in Python 3.7.7 on a laptop with an Intel Core i9-9980HK CPU and 32 Gigabyte of DDR4 RAM running OSX 11.3.1.

All datasets were taken directly from LibSVM (Chang and Lin, 2011) on https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/ and the scaled versions were used if provided. All datasets were preprocessed by adding an intercept, i.e. a constant feature one. For the datasets whose binary labels are not in $\{-1, 1\}$, e.g., phishing, mushrooms and covtype, we assigned -1 to the smallest labels and $+1$ to those largest ones. All learnable parameters were initialized by zeros, e.g., $w^0 = 0 \in \mathbb{R}^d$ and $\alpha_i^0 = 0 \in \mathbb{R}^d$ for $i = 1, \cdots, n$ for SAN.

Table 2 provides the details of the datasets we used in Section 3, including the condition number and $L_{\max}$. For a given dataset, let $\mathbf{A} = [a_1 \cdots a_n] \in \mathbb{R}^{d \times n}$ be the data matrix, the condition number in Table 2 is computed by

$$\text{condition number of the dataset} \stackrel{\text{def}}{=} \sqrt{\frac{\lambda_{\max} \left(\mathbf{A} \mathbf{A}^\top \right)}{\lambda_{\min}^+ \left(\mathbf{A} \mathbf{A}^\top \right)}},$$

where $\lambda_{\max}$ and $\lambda_{\min}^+$ are the largest and smallest non-zero eigenvalue operators respectively. $L_{\max}$ is defined as $L_{\max} = \max_{i=1,\dots,n} L_i$, where $L_i = \frac{1}{4} \|a_i\|^2 + \lambda$ is the smoothness constant of the regularized logistic regression f_i. Notice that the step size's choice for SAG and SVRG is of the order of $\mathcal{O}(1/L_{\max})$.

From Table 2, note that we have datasets that are middle scale (top row of Figure 1) and large scale (bottom row of Figure 1), well conditioned (ijcnn1) and ill conditioned (webspam and rcv1), sparse (rcv1 and real-sim) and dense (epsilon), under-parametrized (phishing, mushrooms, ijcnn1, covtype, webspam, epsilon and real-sim) and over-parametrized (rcv1).

Pseudo-Huber function.   Recall the definition of the pseudo-Huber function used as the regularizer in our experiments in Figure 2: $R(w) = \sum_{i=1}^d R_i(w_i)$ with

$$R_i(w_i) = \delta^2 \left(\sqrt{1 + \left(\frac{w_i}{\delta} \right)^2} - 1 \right).$$

Table 2: Details of the binary data sets used in the logistic regression experiments

dataset	dimension (d)	samples (n)	$L_{\max}$	sparsity	condition number
phishing	$68 + 1$	11055	0.5001	0.5588	4.1065×10^{18}
mushrooms	$112 + 1$	8124	5.5001	0.8125	1.3095×10^{19}
ijcnn1	$22 + 1$	49990	1.2342	0.4091	25.6587
covtype	$54 + 1$	581012	2.154	0.7788	9.6926×10^{17}
webspam	$254 + 1$	350000	0.5	0.6648	6.9973×10^{255}
epsilon	$2000 + 1$	400000	0.5	0.0	3.2110×10^{10}
rcv1	$47236 + 1$	20242	0.5	0.9984	5.3915×10^{25}
real-sim	$20958 + 1$	72309	0.5	0.9976	1.3987×10^{20}

When w_i is large, $R_i(w_i) \to \delta|w_i|$ for all $i = 1, \cdots, n$, i.e. $R(w)$ approximates L1 loss with a factor δ; when w_i is closed to zero, $R_i(w_i) \to \frac{1}{2}w_i^2$ for all $i = 1, \cdots, n$, i.e. $R(w)$ approximates L2 loss. This function can be served as a regularizer to promote the sparsity of the solution (Fountoulakis and Gondzio, 2016).

Besides, the pseudo-Huber is $\mathcal{C}^\infty$. The gradient of the pseudo-Huber is given by

$$\nabla R(w) = \left[\frac{w_1}{\sqrt{1 + \left(\frac{w_1}{\delta}\right)^2}} \quad \cdots \quad \frac{w_d}{\sqrt{1 + \left(\frac{w_d}{\delta}\right)^2}} \right]^\top \in \mathbb{R}^d$$

and the Hessian is given by

$$\nabla^2 R(w) = \mathbf{Diag}\left(\left(1 + \left(\frac{w_1}{\delta}\right)^2\right)^{-3/2}, \cdots, \left(1 + \left(\frac{w_d}{\delta}\right)^2\right)^{-3/2} \right) \preceq \mathbf{I}_d.$$

Thus the pseudo-Huber is 1-smooth which is the same as L2 regularizer. Consequently, $L_{\max}$ for the pseudo-Huber regularized logistic regression is the same as the L2-regularized one.

C.2   Function sub-optimality plots

The performance of an algorithm for solving a convex problem is usually done by measuring one of the following quantities: the solution gap $\|w^k - w^*\|$ where w^* is the solution of the problem, the optimization gap $f(w^k) - \inf f$, and the stationarity gap $\|\nabla f(w^k)\|$. In this paper we choose to measure and compare performance of algorithms in terms of $\|\nabla f(w^k)\|$. The main reason for this is that the solution gap $\|w^k - w^*\|$ and the optimization gap $f(w^k) - \inf f$ both require to compute the solution of the problem to a high precision. While this is possible to do for small problems, it quickly becomes intractable for large problems (see Figure 3 for `covtype`), which we want to address in this paper (see Table 2 for more bigger datasets than `covtype`). The flat curves appeared in Figure 3 after certain effective passes, especially for `covtype` dataset, are due to the imprecise computation of $\inf f$ from the solver `scipy.optimize.fmin_l_bfgs_b`. Indeed, the curves in Figure 3 are in logarithmic scale. When $f(w^k) - \widehat{\inf f} < 0$ with $\widehat{\inf f}$ the tentative solution of the problem computed by the solver, it means that the solution $\widehat{\inf f}$ is imprecise, i.e. the solver performs worse than the tested algorithms. In this case, Figure 3 plots $\left| f(w^k) - \widehat{\inf f} \right| = \widehat{\inf f} - f(w^k) > 0$ where the curves remain flat in logarithmic scale.

We argue that the quantity $\|\nabla f(w^k)\|^2$ is a fair and good proxy for the more classical optimization gap $f(w^k) - \inf f$. Our argument for this is twofold. First, we observe empirically on small problems (for which we can compute $\inf f$ with precision) that the curves for $\|\nabla f(w^k)\|^2$ and $f(w^k) - \inf f$ behave the same (see Figure 3). Second, we verify theoretically that $\|\nabla f(w^k)\|^2$ and $f(w^k) - \inf f$ are of the same order. Indeed, Assumption 1.1 implies that f is strongly convex on every compact. In particular, it verifies on every compact a Lojasiewicz inequality:

$$(\forall R > 0)(\exists \mu > 0)(\forall w \in \mathbb{B}(0, R)) \quad f(w) - \inf f \leq \frac{1}{2\mu}\|\nabla f(w)\|^2.$$

Moreover, f is convex, so if we assume that f has a L-Lipschitz gradient, we obtain the following inequality:

$$(\forall w \in \mathbb{R}^d) \quad \frac{1}{2L}\|\nabla f(w)\|^2 \leq f(w) - \inf f.$$
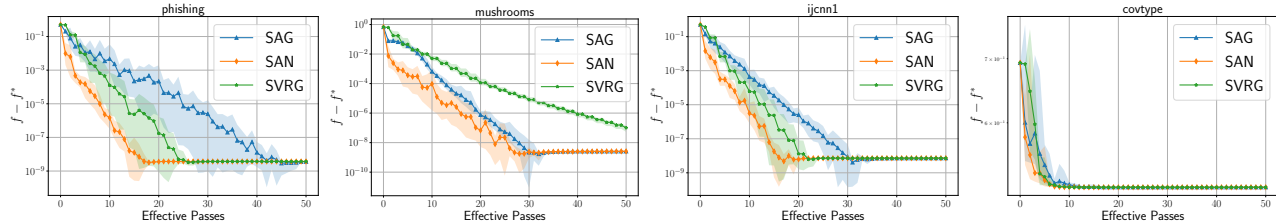
Figure 3: Function sub-optimality of logistic regression with L2 regularization.

Table 3: `covtype` dataset: grid search of π and γ for SAN

π $\backslash$ γ	0.7	0.8	0.9	1.0	1.1	1.2	1.3
$1/2n$	27	25	23	21	21	22	24
$1/n$	26	26	25	22	22	23	24
$10/n$	28	24	24	23	23	22	22
$100/n$	28	26	23	23	22	23	24
$1000/n$	28	26	27	27	25	24	26

Note that this assumption is verified for the functions considered in our experiments (see Section C.1).

C.3 Effect of hyperparameters

As we discussed in Section 3, SAN involves neither prior knowledge of the datasets (e.g., $L_{\max}$), nor the hyperparameter tuning, while both SAG (Schmidt et al., 2017) and SVRG (Johnson and Zhang, 2013) do. To support this conclusion, under different hyperparameters setting, we measure the performance of the given algorithm by monitoring the number of effective passes over the data required to reach below a threshold (e.g., 10^{-4} in our case) of $\|\nabla f\|$. We repeat this procedure 5 times and report the average results.

Grid search for SAN. SAN has two hyperparameters: the probability π doing the averaging step in Algorithm 1 and the step size γ. We searched π in a wide range, $\pi \in \{\frac{1}{2n}, \frac{1}{n}, \frac{10}{n}, \frac{100}{n}, \frac{1000}{n}\}$; as for γ, through our extensive experiments, we observed that SAN works consistently well when γ is around one as we expected for second order methods, thus we tried $\gamma \in \{0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3\}$.

Table 3 and 4 show the grid search results on datasets `covtype` and `ijcnn1`. We see that the average effective data passes required to reach the threshold is stable. It means that SAN is not sensitive to these hyperparameters. This advantage allows us to use $\pi = \frac{1}{n+1}$ and $\gamma = 1.0$ as default choice in our experiments shown in Section 3.

Grid search for SAG and SVRG. Additionally we evaluated the effect of step size γ which is a crucial hyperparameter for first order methods. Let f_i be L_i-smooth for all $i \in \{1, \ldots, n\}$ and $L_{\max} = \max_{i \in \{1,\ldots,n\}} L_i$. As $\gamma = \frac{1}{L_{\max}}$ is thought as the rule of thumb choice in practice (Pedregosa et al., 2011) for SAG and SVRG, we

Table 4: `ijcnn1` dataset: grid search of π and γ for SAN

π $\backslash$ γ	0.7	0.8	0.9	1.0	1.1	1.2	1.3
$1/2n$	13	13	14	12	11	12	13
$1/n$	14	13	12	12	12	12	13
$10/n$	13	13	12	12	12	12	13
$100/n$	13	11	12	13	11	12	14
$1000/n$	16	13	13	13	14	14	14

searched over the values given by

$$\gamma \in \left\{ \frac{1}{10L_{\max}}, \frac{1}{5L_{\max}}, \frac{1}{3L_{\max}}, \frac{1}{2L_{\max}}, \frac{1}{L_{\max}}, \frac{2}{L_{\max}}, \frac{5}{L_{\max}} \right\}$$

on different datasets.

Table 5: Grid search of the step size γ for SVRG on four datasets

Datasets $\diagdown$ γ	$\frac{1}{10L_{\max}}$	$\frac{1}{5L_{\max}}$	$\frac{1}{3L_{\max}}$	$\frac{1}{2L_{\max}}$	$\frac{1}{L_{\max}}$	$\frac{2}{L_{\max}}$	$\frac{5}{L_{\max}}$
covtype	44	24	18	**14**	20	$\times$	$\times$
ijcnn1	22	12	**10**	**10**	15	25	$\times$
phishing	14	11	**10**	14	18	44	$\times$
mushrooms	$\times$	$\times$	44	36	28	**20**	46

Table 6: Grid search of the step size γ for SAG on four datasets

Datasets $\diagdown$ γ	$\frac{1}{10L_{\max}}$	$\frac{1}{5L_{\max}}$	$\frac{1}{3L_{\max}}$	$\frac{1}{2L_{\max}}$	$\frac{1}{L_{\max}}$	$\frac{2}{L_{\max}}$	$\frac{5}{L_{\max}}$
covtype	21	**19**	23	24	40	$\times$	$\times$
ijcnn1	**14**	16	17	17	22	34	$\times$
phishing	**14**	17	21	21	30	48	$\times$
mushrooms	$\times$	47	32	24	**18**	25	$\times$

From our observations to Table 5 and 6 ,[7] we can draw the conclusions that compared to SAN, there is no universal step size choice for SAG and SVRG which gives a consistent good performance on different datasets. This point is one of our original motivations to develop a second order method that requires neither prior knowledge from datasets nor the hyperparameter tuning.

C.4   Additional experiments for SANA, SNM and IQN applied for L2 logistic regression

We present some additional results of SANA, SNM (Kovalev et al., 2019) and IQN (Mokhtari et al., 2018) compared to SAN on L2 logistic regression scenario.

First, we show the results on middle size datasets, phishing and mushrooms in Figure 4. On the one hand, in terms of effective passes of data, SANA has a similar performance as SAN despite the fact that SANA is unbiased and SAN is a biased estimate. Both methods are less efficient than SNM and IQN. Notice that the initialization process of SNM is expensive, as it requires a computation of the full Newton system. Such process is not counted into the effective passes. On the other hand, in terms of computational time, we observe that SAN does as well as IQN and SNM: SAN's cheap iteration cost compensates for its slower convergence rate. On the other hand, we observe for SANA that it is not competitive with respect to the other methods in terms of time taken. This is coherent in a regime where $d \ll n$ since SANA has a computation cost of $\mathcal{O}(nd)$ per iteration (see Table 1), while the cost for SAN and SNM, IQN is respectively $\mathcal{O}(d)$ and $\mathcal{O}(d^2)$, However, it shows that SANA is still a meaningful incremental second order method that satisfies our objective statement. This supports our general approach to design algorithms via function splitting.

In our second set of experiments, we compare those algorithms on large scale datasets. As shown in Figure 5, we tested two datasets webspam and epsilon. As we discuss below, both SNM and IQN are limited in this case, while SAN is able to efficiently solve the problem. IQN is disqualified in this large scale setting, because its memory cost of $\mathcal{O}(nd^2)$ is prohibitive and makes it impossible to run on a laptop. This cost comes from the fact that IQN maintains and updates n approximations of the hessians $\nabla^2 f_i(w^k)$, each of size d^2, and these matrices are not low-rank even for a GLM, preventing from using GLM implementation tricks (as it is the case for SNM, see Remark C.1 below). We also did not run SANA, since we already know that it performs similarly to SAN in

---

[7]The symbol $\times$ in these tables means that the algorithm can not reach below the threshold 10^{-4} after 50 data passes.
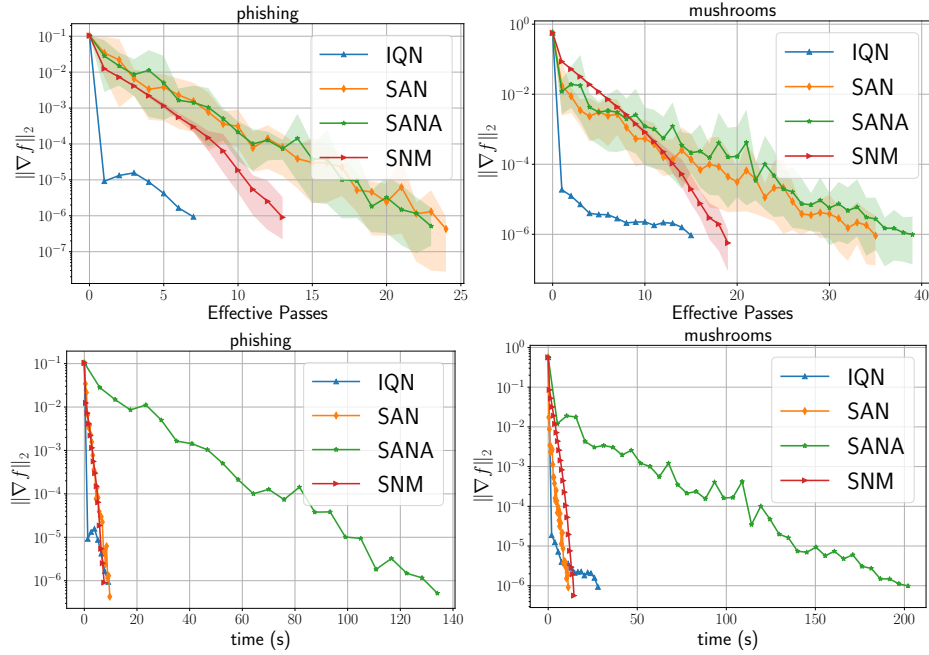
Figure 4: L2-regularized logistic regression for SAN, SANA, SNM and IQN on middle size datasets. Top row is evaluated in terms of effective data passes and bottom row is evaluated in terms of computational time.

terms of effective passes, but suffers from a cost per iteration scaling with n, which is too large here. It is possible to run SNM, but it is not efficient in terms of computational time due to its expensive cost per iteration. For the dataset epsilon, just after one pass over the data, the running time of SNM exceeded our maximum allowed time while at the same time SAN has run 25 data passes and reached a solution with a 10^{-6} precision.
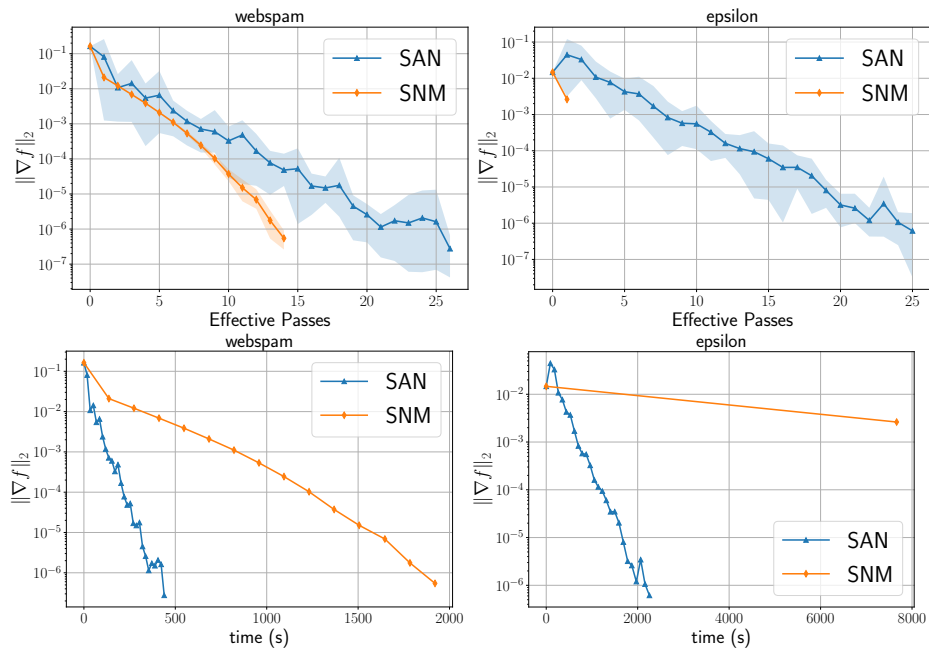

Figure 5: L2-regularized logistic regression for SAN and SNM on large size datasets. Top row is evaluated in terms of effective data passes and bottom row is evaluated in terms of computational time.

Furthermore, note that we are running experiments in a setting which is favorable to SNM. Indeed, its cost per iteration $\mathcal{O}(d^2)$ is only valid when using $L2$ regularization. If we were to consider another separable regularizer,

its cost per iteration would be $\mathcal{O}(d^3)$, making SNM infeasible for large dimensional problems. The next remark details those considerations about the complexity of SNM.

Remark C.1 (On the cost of SNM). The updates of SNM can be written in closed form as

$$w^{k+1} = \left(\frac{1}{n}\sum_{i=1}^{n}\nabla^2 f_i(\alpha_i^k)\right)^{-1}\left(\frac{1}{n}\sum_{i=1}^{n}\nabla^2 f_i(\alpha_i^k)\alpha_i^k - \nabla f_i(\alpha_i^k)\right), \ \alpha_j^{k+1} = w^{k+1}, \ \alpha_i^{k+1} = \alpha_i^k \text{ for } i \neq j, \quad (34)$$

where $w, \alpha_i \in \mathbb{R}^d$ for $i = 1, \cdots, n$ are variables defined in (5) using a variable splitting trick. The main cost of SNM is to update the following inverse matrix $\left(\frac{1}{n}\sum_{i=1}^{n}\nabla^2 f_i(\alpha_i^k)\right)^{-1}$ after updating a single α_j.

For L2-regularized GLMs, by using the *Sherman-Morrison formula*, the above term can be implemented efficiently in $\mathcal{O}(d^2)$ (See Algorithm 3 in Kovalev et al. (2019)), exploiting rank one updates of the matrix.

For other separable regularizers, such a formula is no longer available, as the perturbation becomes rank d due to the diagonal Hessian of the regularizer derived by Lemma B.2 (1). The inversion of the matrix, therefore costs $\mathcal{O}(d^3)$ over all. Note that the memory cost is also impacted in this case: for general separable regularizers the memory cost will be $\mathcal{O}(nd + d^2)$, instead of $\mathcal{O}(n + d^2)$ as can be seen in Table 1 for L2-regularized GLMs.

C.5 SAN vs SAN without the variable metric

One of the main design features of SAN is that at every iteration we project our iterates onto an affine space with respect to a metric induced by the Hessian of one sampled function. One could ask whether this is worth it, given that it makes the theoretical analysis much more difficult. Let us consider again the problem introduced in (11) where the Hessian induced norm has been replaced by the L2 norm as following:

$$\alpha_j^{k+1}, w^{k+1} \ = \ \arg\min_{\alpha_j \in \mathbb{R}^d, w \in \mathbb{R}^d} \left\|\alpha_j - \alpha_j^k\right\|^2 + \left\|w - w^k\right\|^2 \tag{35}$$
$$\text{subject to } \nabla f_j(w^k) + \nabla^2 f_j(w^k)(w - w^k) = \alpha_j.$$

Using Lemma A.5, we can compute the closed form update of (35) (the details are left to readers):

$$\alpha_j^{k+1} \ = \ \alpha_j^k - \left(\mathbf{I} + (\nabla^2 f_j(w^k))^2\right)^{-1}\left(\alpha_j^k - \nabla f_j(w^k)\right), \tag{36}$$
$$w^{k+1} \ = \ w^k - \nabla^2 f_j(w^k)\left(\alpha_j^{k+1} - \alpha_j^k\right). \tag{37}$$

We call this algorithm *SAN-id*[8] for short. However, we observe from Figure 6 that SAN-id only performs well at the early stage and stops converging to the optimum after the first few passes over the data. This motivated us to develop the version with the variable metric as introduced in the main text.
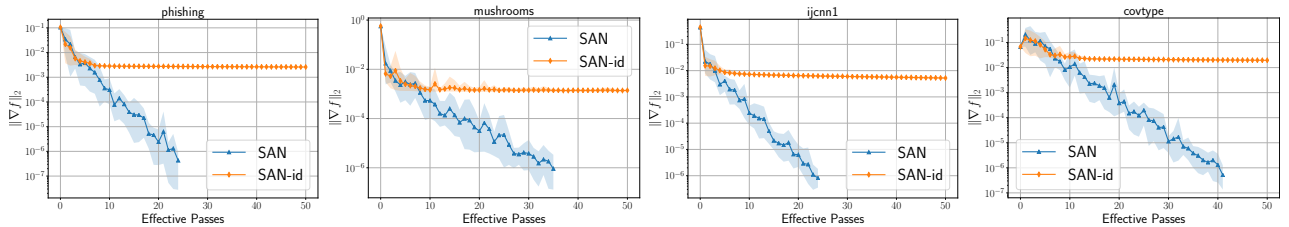

Figure 6: L2-regularized logistic regression for SAN and SAN without the variable metric.

D SAN and SANA viewed as a sketched Newton Raphson method with variable metric

Here we provide a more detailed, step by step, introduction of the SAN and SANA methods. We also detail how SAN and SANA are particular instances of the Variable Metric Sketched Newton Raphson method introduced in the Section 4.

---

[8]because this algorithm fits also in our SNRVM framework with $\mathbf{W}_k \equiv \mathbf{I}$ in (18).

D.1 A sketched Newton Raphson point of view

Here we clarify how the SAN and SANA methods are special cases of the sketched Newton Raphson method with a variable metric detailed in Section 4.

Let $x = \left[w \, ; \alpha_1 \, ; \cdots \, ; \alpha_n\right] \in \mathbb{R}^{(n+1)d}$ and $F : \mathbb{R}^{(n+1)d} \to \mathbb{R}^{(n+1)d}$ defined as

$$F(x) \overset{\text{def}}{=} \left[\tfrac{1}{n} \sum \alpha_i \, ; \nabla f_1(w) - \alpha_1 \, ; \cdots \, ; \nabla f_n(w) - \alpha_n\right]. \tag{38}$$

Therefore $w^* \in \mathbb{R}^d$ is a minimizer of (1) if and only if there exists $x^* = [w^*; \alpha_1^*; \dots; \alpha_n^*] \in \mathbb{R}^{(n+1)d}$ such that $F(x^*) = 0$. The Jacobian $\nabla F(x)$ is given by

$$\nabla F(x) \quad = \quad \begin{bmatrix} 0 & \nabla^2 f_1(w) & \cdots & \nabla^2 f_n(w) \\ \frac{1}{n}\mathbf{I}_d & & & \\ \vdots & & -\mathbf{I}_{nd} & \\ \frac{1}{n}\mathbf{I}_d & & & \end{bmatrix} \in \mathbb{R}^{(n+1)d \times (n+1)d}, \tag{39}$$

To find a zero of the function F, one could use the damped Newton Raphson method

$$x^{k+1} = x^k - \gamma \nabla F(x^k)^{\top \, \dagger} F(x^k), \quad \gamma \in (0, 1]. \tag{40}$$

This can be equivalently rewritten as a projection-and-relaxation step given by

$$\begin{cases} \bar{x}^{k+1} = \operatorname{argmin} \|x - x^k\|^2 & \text{subject to} \quad \nabla F(x^k)^{\top}(x - x^k) = -F(x^k), \\ x^{k+1} = (1 - \gamma)x^k + \gamma \bar{x}^{k+1}. \end{cases} \tag{41}$$

Using the definition of our function F in (6), we see that each iteration of the Newton Raphson method requires to project onto the following set of linear equations:

$$\begin{aligned} & \nabla F(x^k)^{\top}(x - x^k) = -F(x^k), \\ \Leftrightarrow \quad & \begin{cases} \frac{1}{n} \sum_{i=1}^{n}(\alpha_i - \alpha_i^t) = -\frac{1}{n} \sum_{i=1}^{n} \alpha_i^t, \\ \nabla^2 f_i(w^t)(w - w^t) - (\alpha_i - \alpha_i^t) = \alpha_i^t - \nabla f_i(w^t) \text{ for } i \in \{1, \dots, n\}, \end{cases} \\ \Leftrightarrow \quad & \begin{cases} \frac{1}{n} \sum_{i=1}^{n} \alpha_i = 0, \\ \nabla^2 f_i(w^t)(w - w^t) - \alpha_i = -\nabla f_i(w^t) \text{ for } i \in \{1, \dots, n\}. \end{cases} \end{aligned} \tag{42}$$

Projecting onto (42) is challenging for two reasons: first it accesses all of the data (every function f_i is involved) and second it requires solving a large linear system.

One approach to circumvent this bottleneck is to *sketch* this linear system: at every iteration, instead of considering (42), we will project onto a random row compression of this system. Sketching can be for instance as simple as sampling one of the equations appearing in (42). In its more general form, a sketch corresponds to any linear transformation of the equations. In our context, this can be written as

$$\mathbf{S}^{\top} \nabla F(x^k)^{\top}(x - x^k) = -\mathbf{S}^{\top} F(x^k), \tag{43}$$

where $\mathbf{S} \in \mathbb{R}^{(n+1)d \times \tau}$ is called the sketching matrix, and its number of columns τ is typically small.

This idea is at the core of the Sketched Newton Raphson method (Yuan et al., 2021), which aims at finding a zero of the function F by iterating:

$$\begin{cases} \bar{x}^{k+1} = \operatorname{argmin} \|x - x^k\|^2 & \text{subject to} \quad \mathbf{S}_k^{\top} \nabla F(x^k)^{\top}(x - x^k) = -\mathbf{S}_k^{\top} F(x^k), \\ x^{k+1} = (1 - \gamma)x^k + \gamma \bar{x}^{k+1}, \end{cases} \tag{44}$$

where $\mathbf{S}_k$ is a sketching matrix randomly sampled at each iteration with respect to some distribution.

As we detailed in Section 4, the algorithms proposed in this paper can be seen as particular instances of a *Variable Metric* Sketched Newton Raphson method. This more general framework allows, at every iteration, to project the previous iterate onto (43) with respect to some non-euclidean metric. The algorithm writes as follows:

$$\begin{cases} \bar{x}^{k+1} = \text{argmin } \|x - x^k\|_{\mathbf{W}_k}^2 \quad \text{subject to} \quad \mathbf{S}_k^\top \nabla F(x^k)^\top (x - x^k) = -\mathbf{S}_k^\top F(x^k), \\ x^{k+1} = (1-\gamma)x^k + \gamma \bar{x}^{k+1}. \end{cases} \tag{45}$$

Here, both $\mathbf{S}_k$ and $\mathbf{W}_k$ are randomly sampled with respect to a distribution which may depend on x^k. Besides, $\mathbf{W}_k$ is positive-definite. The closed form solution to (45) is given in (18), thanks to the following Lemma.

Lemma D.1. *If the iterates in (17) are well defined, then they are equivalent to (18).*

Proof. Let x^{k+1} be the iterate defined by (17), where we assumed that the linear system $\mathbf{S}_k^\top \nabla F(x^k)^\top (x - x^k) = -\mathbf{S}_k^\top F(x^k)$ has a solution. Let us do a change of variable $u = x - x^k$, and write $\bar{x}^{k+1} = x^k + u^*$ where

$$u^* = \text{argmin } \|u\|_{\mathbf{W}_k}^2 \quad \text{subject to} \quad \mathbf{S}_k^\top \nabla F(x^k)^\top u = -\mathbf{S}_k^\top F(x^k).$$

We can call Lemma A.5 to obtain that

$$u^* = -\mathbf{W}_k^{-1} \nabla F(x^k)\mathbf{S}_k \left(\mathbf{S}_k^\top \nabla F(x^k)^\top \mathbf{W}_k^{-1} \nabla F(x^k)\mathbf{S}_k\right)^\dagger \mathbf{S}_k^\top F(x^k).$$

The claim follows after writing that $x^{k+1} = (1-\gamma)x^k + \gamma(x^k + u^*) = x^k + \gamma u^*$. $\square$

D.2 SAN is a particular case of SNRVM

Let us consider SAN, described in Algorithm 1, and rewrite it as an instance of the Variable Metric Sketched Newton Raphson method (45). Given a probability $\pi \in (0, 1)$, we define for all $x \in \mathbb{R}^{(n+1)d}$ a distribution $\mathcal{D}_x^{\text{SAN}}$ as follows: $(\mathbf{S}, \mathbf{W}) \sim \mathcal{D}_x^{\text{SAN}}$ means that

- with probability π we have

$$\mathbf{S} = \begin{bmatrix} \mathbf{I}_d \\ \mathbf{0}_d \\ \vdots \\ \mathbf{0}_d \end{bmatrix} \quad \text{and} \quad \mathbf{W} = \mathbf{I}_{(n+1)d}, \tag{46}$$

- with probability $1 - \pi$, we sample $j \in \{1, \cdots, n\}$ uniformly and set

$$\mathbf{S} = \begin{bmatrix} \mathbf{0}_d \\ \vdots \\ \mathbf{I}_d \\ \vdots \\ \mathbf{0}_d \end{bmatrix} \leftarrow j+1 \quad \text{and} \quad \mathbf{W} = \begin{bmatrix} \nabla^2 f_j(w) & & & \\ & \mathbf{I}_d & & \\ & & \ddots & \\ & & & \mathbf{I}_d \end{bmatrix}. \tag{47}$$

Lemma D.2. *Let $\pi \in (0, 1)$ and $\gamma \in (0, 1]$ a step size. Algorithm 1 (SAN) is equivalent to the Variable Metric Sketched Newton Raphson method (45) applied to the function F defined in (6), where at each iteration $(\mathbf{S}_k, \mathbf{W}_k)$ is sampled with respect to $\mathcal{D}_{x^k}^{\text{SAN}}$, as defined in (46)-(47).*

Proof. Let us consider the Variable Metric Sketched Newton Raphson method described in this Lemma. We consider two cases, corresponding to the two classes of events described in (46) and (47).

Suppose that we are in the case (which holds with probability π) given by (46). In this case we have

$$\mathbf{S}_k^\top \nabla F(x^k)^\top (x - x^k) \overset{(39)+(46)}{=} \begin{bmatrix} \mathbf{0}_d & \frac{1}{n}\mathbf{I}_d & \cdots & \frac{1}{n}\mathbf{I}_d \end{bmatrix} (x - x^k)$$

$$= \frac{1}{n}\sum_{i=1}^n (\alpha_i - \alpha_i^k)$$

$$\mathbf{S}_k^\top F(x^k) \overset{(6)+(46)}{=} \frac{1}{n}\sum_{i=1}^n \alpha_i^k.$$

Those expressions mean that the linearized equation (43) is equivalent to $\frac{1}{n}\sum_{i=1}^{n}\alpha_i = 0$. So the update of the variables is exactly given by (24).

Let now j be in $\{1, \ldots, n\}$ sampled uniformly, and suppose that we are in the case given by (47). We can then compute

$$
\begin{aligned}
\mathbf{S}_k^\top \nabla F(x^k)^\top (x - x^k) &\overset{(39)+(47)}{=} \begin{bmatrix} \nabla^2 f_j(w^k) & \mathbf{0}_d & \cdots & -\mathbf{I}_d & \cdots & \mathbf{0}_d \end{bmatrix}(x - x^k) \\
&= \nabla^2 f_j(w^k)(w - w^k) - (\alpha_j - \alpha_j^k) \\
\mathbf{S}_k^\top F(x^k) &\overset{(6)+(47)}{=} \nabla f_j(w^k) - \alpha_j^k.
\end{aligned}
$$

Those expressions mean that the linearized equation (43) is equivalent to $\nabla^2 f_j(w^k)(w - w^k) - \alpha_j = -\nabla f_j(w^k)$. So the update of the variables is exactly given by (25). The conclusion follows Lemma A.2. □

D.3 SANA is a particular case of SNRVM

Let us consider SANA, described in Algorithm 2, and rewrite it as an instance of the Variable Metric Sketched Newton Raphson method (45). We define for all $x \in \mathbb{R}^{(n+1)d}$ a distribution $\mathcal{D}_x^{\text{SANA}}$ as follows: $(\mathbf{S}, \mathbf{W}) \sim \mathcal{D}_x^{\text{SANA}}$ means that, with probability $1/n$ we sample $j \in \{1, \cdots, n\}$ and we have

$$
\mathbf{S} = \begin{bmatrix} \mathbf{I}_d & \mathbf{0}_d \\ \mathbf{0}_d & \vdots \\ \vdots & \mathbf{I}_d \\ \vdots & \vdots \\ \mathbf{0}_d & \mathbf{0}_d \end{bmatrix} \leftarrow j+1 \quad \text{and} \quad \mathbf{W} = \begin{bmatrix} \nabla^2 f_j(w) & & & \\ & \mathbf{I}_d & & \\ & & \ddots & \\ & & & \mathbf{I}_d \end{bmatrix}. \tag{48}
$$

Lemma D.3. Let $\gamma \in (0, 1]$ be a step size. Algorithm 2 (SANA) is equivalent to the Variable Metric Sketched Newton Raphson method (45) applied to the function F defined in (6), where at each iteration $(\mathbf{S}_k, \mathbf{W}_k)$ is sampled with respect to $\mathcal{D}_{x^k}^{\text{SANA}}$, as defined in (48).

Proof. Let $k \in \mathbb{N}$, and suppose that we have sampled $j \in \{1, \ldots, n\}$ and $\mathbf{S}_k$ and $\mathbf{W}_k$ according to (48). Therefore,

$$
\begin{aligned}
\mathbf{S}_k^\top \nabla F(x^k)^\top (x - x^k) &\overset{(39)+(48)}{=} \begin{bmatrix} \mathbf{0}_d & \frac{1}{n}\mathbf{I}_d & \cdots & \cdots & \cdots & \frac{1}{n}\mathbf{I}_d \\ \nabla^2 f_j(w^k) & \mathbf{0}_d & \cdots & -\mathbf{I}_d & \cdots & \mathbf{0}_d \end{bmatrix}(x - x^k) \\
&= \begin{bmatrix} \frac{1}{n}\sum_{i=1}^{n}(\alpha_i - \alpha_i^k) \\ \nabla^2 f_j(w^k)(w - w^k) - (\alpha_j - \alpha_j^k) \end{bmatrix} \\
\mathbf{S}_k^\top F(x^k) &\overset{(6)+(48)}{=} \begin{bmatrix} \frac{1}{n}\sum_{i=1}^{n}\alpha_i^k \\ \nabla f_j(w^k) - \alpha_j^k \end{bmatrix}.
\end{aligned}
$$

Those expressions mean that the linearized equation (43) is equivalent to the two equations $\sum_{i=1}^{n}\alpha_i = 0$ and $\nabla^2 f_j(w^k)(w - w^k) - \alpha_j = -\nabla f_j(w^k)$. So the update of the variables is exactly given by (30). The conclusion follows Lemma A.4.

□

E Proofs for the results in Section 4, including Theorems 4.5 and 4.8

E.1 Proof of Proposition 4.3

Proof. Let us start by showing that Assumption 4.1 is satisfied for SAN and SANA. The distribution $\mathcal{D}_x^{\text{SAN}}$ (resp. $\mathcal{D}_x^{\text{SANA}}$) defined in the Section D.2 (resp. Section D.3) is clearly finite and proper so long as $\pi \in (0, 1)$. It

remains to compute $\mathbb{E}[\mathbf{SS}^\top]$. We can see that it is a block-diagonal matrix $\mathbf{Diag}\left(\pi \mathbf{I}_d, \frac{1-\pi}{n}\mathbf{I}_d, \cdots, \frac{1-\pi}{n}\mathbf{I}_d\right)$ (resp. $\mathbf{Diag}\left(\mathbf{I}_d, \frac{1}{n}\mathbf{I}_d, \cdots, \frac{1}{n}\mathbf{I}_d\right)$), which is invertible since $\pi \in (0,1)$.

Now let us turn on Assumption 4.2. To prove that $\nabla F(x)^\top \nabla F(x)$ is invertible, it is enough to show that $\nabla F(x)$ is injective. Let $x = (w\,; \alpha) \in \mathbb{R}^{d+dn}$, and let us first show that $\nabla F(x)$ is injective. Suppose there exists $\bar{x} = (\bar{w}\,; \bar{\alpha}) \in \mathbb{R}^{d+dn}$ such that $\nabla F(x)\bar{x} = 0$. Consequently from (39) we have that

$$\sum_{i=1}^n \nabla^2 f_i(w)\bar{\alpha}_i = 0$$

$$\frac{1}{n}\bar{w} = \bar{\alpha}_i, \quad \text{for } i = 1, \ldots, n. \tag{49}$$

Substituting out the $\bar{\alpha}_i$'s we have that

$$\nabla^2 f(w)\bar{w} = \frac{1}{n}\sum_{i=1}^n \nabla^2 f_i(w)\bar{w} = 0.$$

Consequently, since $\nabla^2 f(w)$ is positive definite (recall Assumption 1.1), and in particular injective, we have that $\bar{w} = 0$. Thus it follows from (49) that $\bar{\alpha}_i = 0$ for $i = 1, \ldots, n$. This all shows that $\bar{x} = 0$, and concludes the proof that $\nabla F(x)$ is injective.

Furthermore, $\nabla F(x)$ is a square matrix, thus invertible. We have $F(x) \in \mathbf{Im}\left(\nabla F(x)^\top\right)$.

Finally $\nabla F(x)^\top \nabla F(x)$ is invertible since

$$\mathbf{Null}\left(\nabla F(x)^\top \nabla F(x)\right) = \mathbf{Null}\left(\nabla F(x)\right) = \{0\}.$$

$\square$

E.2 SNRVM is equivalent to minimizing a quadratic function over a random subspace

Lemma E.1. (Lemma 10 in Gower et al. (2019)). For every matrix $\mathbf{M}$ and symmetric positive semi-definite matrix $\mathbf{G}$ such that $\mathbf{Null}(\mathbf{G}) \subset \mathbf{Null}(\mathbf{M})$, we have that $\mathbf{Null}\left(\mathbf{M}^\top\right) = \mathbf{Null}\left(\mathbf{MGM}^\top\right)$.

Lemma E.2. Let Assumptions 4.1 and 4.2 hold. Then the iterates of SNRVM are equivalent to

$$x^{k+1} = \operatorname{argmin}_{x \in \mathbb{R}^p} \hat{f}_k(x^k) + \left\langle \nabla \hat{f}_k(x^k), x - x^k \right\rangle + \frac{1}{2\gamma}\left\|x - x^k\right\|_{\mathbf{W}_k}^2 \tag{50}$$

$$\text{subject to } x \in x^k + \mathbf{Im}\left(\mathbf{W}_k^{-1}\nabla F(x^k)\mathbf{S}_k\right),$$

where $\hat{f}_k$ is defined in (19).

Proof. Start by observing that the problem in (50) is strongly convex, and therefore has a unique solution that we will note x^*. Let us prove that x^* is exactly x^{k+1} whose closed form expression is given in (18). For this, let τ be the number of columns for $\mathbf{S}_k$, and let $u \in \mathbb{R}^\tau$. We can then write that $x^* = x^k + \mathbf{W}_k^{-1}\nabla F(x^k)\mathbf{S}_k u^*$, where u^* is any solution of the following unconstrained optimization problem:

$$u^* \in \operatorname{argmin}_{u \in \mathbb{R}^\tau} \left\langle \nabla \hat{f}_k(x^k), \mathbf{W}_k^{-1}\nabla F(x^k)\mathbf{S}_k u^* \right\rangle + \frac{1}{2\gamma}\left\|\mathbf{W}_k^{-1}\nabla F(x^k)\mathbf{S}_k u^*\right\|_{\mathbf{W}_k}^2.$$

Writing down the optimality conditions for this convex quadratic problem, we see that u^* must verify:

$$\gamma \mathbf{S}_k^\top \nabla F(x^k)^\top \mathbf{W}_k^{-1}\nabla \hat{f}_k(x^k) + \mathbf{S}_k^\top \nabla F(x^k)^\top \mathbf{W}_k^{-1}\nabla F(x^k)\mathbf{S}_k u^*.$$

Let us choose the pseudo inverse solution of this linear system:

$$u^* = -\gamma \left(\mathbf{S}_k^\top \nabla F(x^k)^\top \mathbf{W}_k^{-1}\nabla F(x^k)\mathbf{S}_k\right)^\dagger \mathbf{S}_k^\top \nabla F(x^k)^\top \mathbf{W}_k^{-1}\nabla \hat{f}_k(x^k). \tag{51}$$

Using the definition of $\hat{f}_k$, we can write

$$\nabla \hat{f}_k(x^k) = \nabla F(x^k) \left(\nabla F(x^k)^\top \mathbf{W}_k^{-1} \nabla F(x^k)\right)^\dagger F(x^k).$$

All we need to prove now is that

$$\nabla F(x^k)^\top \mathbf{W}_k^{-1} \nabla F(x^k) \left(\nabla F(x^k)^\top \mathbf{W}_k^{-1} \nabla F(x^k)\right)^\dagger F(x^k) = F(x^k). \tag{52}$$

To see why (52) is true, first notice that $\nabla F(x^k)^\top \mathbf{W}_k^{-1} \nabla F(x^k) \left(\nabla F(x^k)^\top \mathbf{W}^{-1} \nabla F(x^k)\right)^\dagger$ is the orthogonal projector onto the range of $\nabla F(x^k)^\top \mathbf{W}_k^{-1} \nabla F(x^k)$. Moreover, using the fact that $\mathbf{W}_k^{-1}$ is injective together with Lemma E.1, we can write that

$$\begin{aligned}
\mathbf{Im}\left(\nabla F(x^k)^\top \mathbf{W}_k^{-1} \nabla F(x^k)\right) &= \left(\mathbf{Null}\left(F(x^k)^\top \mathbf{W}_k^{-1} \nabla F(x^k)\right)\right)^\perp \\
&= \left(\mathbf{Null}\left(\nabla F(x^k)\right)\right)^\perp \\
&= \mathbf{Im}\left(\nabla F(x^k)^\top\right).
\end{aligned}$$

Since we know from Assumption 4.2 that $\nabla F(x^k)^\top$ is surjective, and so that $F(x^k)$ belongs in the range of $\nabla F(x^k)^\top$, we deduce that (52) is true. We can now inject (52) into (51), and obtain finally that

$$\begin{aligned}
x^* &= x^k + \mathbf{W}_k^{-1} \nabla F(x^k) \mathbf{S}_k u^* \\
&= x^k - \gamma \mathbf{W}_k^{-1} \nabla F(x^k) \mathbf{S}_k \left(\mathbf{S}_k^\top \nabla F(x^k)^\top \mathbf{W}_k^{-1} \nabla F(x^k) \mathbf{S}_k\right)^\dagger \mathbf{S}_k^\top F(x^k),
\end{aligned}$$

which is exactly (18). $\qquad\square$

E.3   About ρ in Theorem 4.5

Lemma E.3. If A, B are two symmetric positive semi-definite matrices then $\mathbf{Null}(A + B) = \mathbf{Null}(A) \cap \mathbf{Null}(B)$.

Proof. If $x \in \mathbf{Null}(A) \cap \mathbf{Null}(B)$ then it is trivial to see that $x \in \mathbf{Null}(A + B)$. If $x \in \mathbf{Null}(A + B)$, then

$$0 = \langle (A + B)x, x \rangle = \langle Ax, x \rangle + \langle Bx, x \rangle,$$

where by positive semi-definiteness we have $\langle Ax, x \rangle \geq 0$ and $\langle Bx, x \rangle \geq 0$. The sum of nonnegative numbers being nonnegative, we deduce that

$$\langle Ax, x \rangle = \langle Bx, x \rangle = 0.$$

Since $\langle Ax, x \rangle = 0$, we deduce from the fact that A is symmetric that $Ax = 0$. Similarly, $Bx = 0$, which concludes the proof. $\qquad\square$

The following Lemma will be needed in the proof of Theorem 4.5.

Lemma E.4. Recall the definition of $\mathbf{H}(x)$ given by

$$\mathbf{H}(x) \stackrel{\text{def}}{=} \mathbb{E}\left[\mathbf{S}\left(\mathbf{S}^\top \nabla F(x)^\top \mathbf{W}^{-1} \nabla F(x) \mathbf{S}\right)^\dagger \mathbf{S}^\top\right], \tag{53}$$

If Assumption 4.1 and 4.2 hold, then $\mathbf{H}(x)$ is invertible. Moreover, for every symmetric positive definite matrix $\mathbf{W}$ and $x \in \mathbb{R}^p$ we have that

$$\min_{v \in \mathbf{Im}\left(\mathbf{W}^{-1/2} \nabla F(x)\right) \setminus \{0\}} \frac{\left\langle \mathbf{W}^{-1/2} \nabla F(x) \mathbf{H}(x) \nabla F(x)^\top \mathbf{W}^{-1/2} v, v \right\rangle}{\|v\|^2} \tag{54}$$

is exactly the smallest positive eigenvalue of $\mathbf{W}^{-1/2} \nabla F(x) \mathbf{H}(x) \nabla F(x)^\top \mathbf{W}^{-1/2}$.

Proof. Let $x \in \mathbb{R}^p$ and $(\mathbf{S}, \mathbf{W}) \sim \mathcal{D}_x$. Let $\mathbf{G} = \nabla F(x)^\top \mathbf{W}^{-1} \nabla F(x)$ which is symmetric positive semi-definite. Since $\nabla F(x)^\top \nabla F(x)$ and $\mathbf{W}$ are invertible we have that $\mathbf{G}$ is invertible. Consequently $\mathbf{Null}(\mathbf{G}) = \{0\} \subset \mathbf{Null}(\mathbf{S}^\top)$. Thus by Lemma E.1 (with $\mathbf{M} = \mathbf{S}^\top$) we have that

$$\mathbf{Null}\left(\left(\mathbf{S}^\top \nabla F(x)^\top \mathbf{W}^{-1} \nabla F(x)\mathbf{S}\right)^\dagger\right) = \mathbf{Null}\left(\mathbf{S}^\top \nabla F(x)^\top \mathbf{W}^{-1} \nabla F(x)\mathbf{S}\right) = \mathbf{Null}(\mathbf{S}).$$

Using Lemma E.1 once again with $\mathbf{G} = \left(\mathbf{S}^\top \nabla F(x)^\top \mathbf{W}^{-1} \nabla F(x)\mathbf{S}\right)^\dagger$ and $\mathbf{M} = \mathbf{S}$, we have that

$$\mathbf{Null}\Big(\underbrace{\mathbf{S}\left(\mathbf{S}^\top \nabla F(x)^\top \mathbf{W}^{-1} \nabla F(x)\mathbf{S}\right)^\dagger \mathbf{S}^\top}_{\overset{\text{def}}{=}\mathbf{H}_{\mathbf{S},\mathbf{W}}(x)}\Big) = \mathbf{Null}\left(\mathbf{S}^\top\right) = \mathbf{Null}\left(\mathbf{S}\mathbf{S}^\top\right). \tag{55}$$

Observe that with our notations and from Assumption 4.1,

$$\mathbf{H}(x) = \mathbb{E}_{\mathbf{S},\mathbf{W}\sim\mathcal{D}_x}\left[\mathbf{H}_{\mathbf{S},\mathbf{W}}(x)\right] = \sum_{i=1}^r p_i \mathbf{H}_{\mathbf{S}_i(x),\mathbf{W}_i(x)}(x).$$

As $\mathbf{H}_{\mathbf{S}_i(x),\mathbf{W}_i(x)}(x)$ is symmetric positive semi-definite, we can use Lemma E.3 to write

$$\begin{aligned}
\mathbf{Null}(\mathbf{H}(x)) = \mathbf{Null}\left(\sum_{i=1}^r p_i \mathbf{H}_{\mathbf{S}_i(x),\mathbf{W}_i(x)}(x)\right) &= \bigcap_{i=1}^r \mathbf{Null}\left(\mathbf{H}_{\mathbf{S}_i(x),\mathbf{W}_i(x)}(x)\right) \\
&\overset{(55)}{=} \bigcap_{i=1}^r \mathbf{Null}\left(\mathbf{S}_i(x)\mathbf{S}_i(x)^\top\right) \\
&= \mathbf{Null}\left(\mathbb{E}_{\mathbf{S}\sim\mathcal{D}_x}\left[\mathbf{S}\mathbf{S}^\top\right]\right) = \{0\}
\end{aligned}$$

This means that $\mathbf{H}(x)$ is invertible for all $x \in \mathbb{R}^p$.

Now, take any $x \in \mathbb{R}^p$, and a symmetric positive definite matrix $\mathbf{W}$. Then $\mathbf{W}^{-1/2}\nabla F(x)\mathbf{H}(x)\nabla F(x)^\top \mathbf{W}^{-1/2}$ is symmetric, semi-definite positive. Since $\mathbf{H}(x)$ and $\mathbf{W}$ are invertible, we can apply Lemma E.1 again to obtain

$$\mathbf{Null}\left(\nabla F(x)^\top \mathbf{W}^{-1/2}\right) = \mathbf{Null}\left(\mathbf{W}^{-1/2}\nabla F(x)\mathbf{H}(x)\nabla F(x)^\top \mathbf{W}^{-1/2}\right). \tag{56}$$

Consequently

$$\begin{aligned}
\mathbf{Im}\left(\mathbf{W}^{-1/2}\nabla F(x)\right) &= \left(\mathbf{Null}\left(\nabla F(x)^\top \mathbf{W}^{-1/2}\right)\right)^\perp \\
&\overset{(56)}{=} \left(\mathbf{Null}\left(\mathbf{W}^{-1/2}\nabla F(x)\mathbf{H}(x)\nabla F(x)^\top \mathbf{W}^{-1/2}\right)\right)^\perp. \tag{57}
\end{aligned}$$

Therefore, we conclude that (54) is equal to

$$\min_{v\in\left(\mathbf{Null}\left(\mathbf{W}^{-1/2}\nabla F(x)\mathbf{H}(x)\nabla F(x)^\top \mathbf{W}^{-1/2}\right)\right)^\perp\backslash\{0\}} \frac{\left\langle \mathbf{W}^{-1/2}\nabla F(x)\mathbf{H}(x)\nabla F(x)^\top \mathbf{W}^{-1/2}v, v\right\rangle}{\|v\|^2}$$

$$= \lambda_{\min}^+\left(\mathbf{W}^{-1/2}\nabla F(x)\mathbf{H}(x)\nabla F(x)^\top \mathbf{W}^{-1/2}\right) > 0.$$

$\square$

E.4 Proof of Theorem 4.5

Proof. Let $k \in \mathbb{N}$. In this proof, we will write ∇F_k as a shorthand for $\nabla F(x^k)$, and we introduce the notation $\nabla^{\mathbf{W}} F_k \overset{\text{def}}{=} \mathbf{W}_k^{-1/2}\nabla F(x^k)$. First we aim to establish a relationship between $\hat{f}_k(x^k) = \left\|F(x^k)\right\|_{(\nabla F_k^\top \mathbf{W}_k^{-1} \nabla F_k)^\dagger}^2$ and $\left\|F(x^k)\right\|_{\mathbf{H}(x^k)}^2$. Observe that Assumption 4.2 allows us to write that

$$(\forall x \in \mathbb{R}^p) \quad F(x) = \nabla F(x)^\top \mathbf{W}_k^{-1/2}(\nabla F(x)^\top \mathbf{W}_k^{-1/2})^\dagger F(x). \tag{58}$$

This is due to the fact that $\nabla F(x)^\top \mathbf{W}_k^{-1/2}(\nabla F(x)^\top \mathbf{W}_k^{-1/2})^\dagger$ is the projection matrix onto $\mathbf{Im}\left(\nabla F(x)^\top \mathbf{W}_k^{-1/2}\right)$, where $\mathbf{W}_k^{-1/2}$ is surjective, meaning that $\mathbf{Im}\left(\nabla F(x)^\top\right) = \mathbf{Im}\left(\nabla F(x)^\top \mathbf{W}_k^{-1/2}\right)$. From (58) we have that

$$
\begin{aligned}
\left\|F(x^k)\right\|_{\mathbf{H}(x^k)}^2 \quad &= \quad \langle F(x^k), \mathbf{H}(x^k)F(x^k)\rangle \\
&\overset{(58)}{=} \quad \langle \nabla^{\mathbf{W}} F_k^\top (\nabla^{\mathbf{W}} F_k^\top)^\dagger F(x^k), \mathbf{H}(x^k)\nabla^{\mathbf{W}} F_k^\top (\nabla^{\mathbf{W}} F_k^\top)^\dagger F(x^k)\rangle \\
&= \quad \langle (\nabla^{\mathbf{W}} F_k^\top)^\dagger F(x^k), \nabla^{\mathbf{W}} F_k \mathbf{H}(x^k)\nabla^{\mathbf{W}} F_k^\top \left((\nabla^{\mathbf{W}} F_k^\top)^\dagger F(x^k)\right)\rangle \\
&\overset{\text{Lemma E.4}}{\geq} \quad \rho\,\|(\nabla^{\mathbf{W}} F_k^\top)^\dagger F(x^k)\|^2 && (59) \\
&= \quad \rho\left\|F(x^k)\right\|_{\left(\nabla^{\mathbf{W}} F_k^\top \nabla^{\mathbf{W}} F_k\right)^\dagger}^2 && (60) \\
&\overset{(19)}{=} \quad 2\rho \hat{f}_k(x^k), && (61)
\end{aligned}
$$

where in (59) we used that $\mathbf{Im}\left((\nabla^{\mathbf{W}} F_k^\top)^\dagger\right) = \mathbf{Im}\left(\nabla^{\mathbf{W}} F_k\right)$ together with Lemma E.4, and in (60) we used that $(\mathbf{M})^{\dagger\top}(\mathbf{M})^\dagger = (\mathbf{M}^\top)^\dagger(\mathbf{M})^\dagger = (\mathbf{M}\mathbf{M}^\top)^\dagger$ for every matrix $\mathbf{M}$. Now, we turn onto the study the term $\hat{f}_k(x^k)$. Compute its gradient with respect to the metric induced by $\mathbf{W}_k$ at x^k:

$$
\begin{aligned}
\nabla^{\mathbf{W}_k} \hat{f}_k(x^k) \quad &= \quad \mathbf{W}_k^{-1}\nabla F_k(\nabla F_k^\top \mathbf{W}_k^{-1}\nabla F_k)^\dagger F(x^k) \\
&= \quad \mathbf{W}_k^{-1/2}(\nabla F_k^\top \mathbf{W}_k^{-1/2})^\dagger F(x^k). && (62)
\end{aligned}
$$

This, together with the Assumption 4.4, allows us to write that

$$
\begin{aligned}
\hat{f}_{k+1}(x^{k+1}) \quad &\leq \quad \hat{f}_k(x^k) + \left\langle \nabla\hat{f}_k(x^k), x^{k+1} - x^k\right\rangle + \frac{L}{2}\left\|x^{k+1} - x^k\right\|_{\mathbf{W}_k}^2 \\
&\overset{(18)+(62)}{=} \quad \hat{f}_k(x^k) - \gamma\left\langle n^{\mathbf{W}_k}(x^k), \mathbf{W}_k^{-1}\nabla F_k \mathbf{S}_k\left(\mathbf{S}_k^\top\nabla F_k^\top \mathbf{W}_k^{-1}\nabla F_k \mathbf{S}_k\right)^\dagger \mathbf{S}_k^\top F(x^k)\right\rangle_{\mathbf{W}_k} \\
&\qquad + \frac{\gamma^2 L}{2}\left\|\mathbf{W}_k^{-1}\nabla F_k \mathbf{S}_k\left(\mathbf{S}_k^\top\nabla F_k^\top \mathbf{W}_k^{-1}\nabla F_k \mathbf{S}_k\right)^\dagger \mathbf{S}_k^\top F(x^k)\right\|_{\mathbf{W}_k}^2 \\
&\overset{(58)}{=} \quad \hat{f}_k(x^k) - \gamma\left\langle F(x^k), \mathbf{S}_k\left(\mathbf{S}_k^\top\nabla F_k^\top \mathbf{W}_k^{-1}\nabla F_k \mathbf{S}_k\right)^\dagger \mathbf{S}_k^\top F(x^k)\right\rangle \\
&\qquad + \frac{\gamma^2 L}{2}\left\|\mathbf{W}_k^{-1}\nabla F_k \mathbf{S}_k\left(\mathbf{S}_k^\top\nabla F_k^\top \mathbf{W}_k^{-1}\nabla F_k \mathbf{S}_k\right)^\dagger \mathbf{S}_k^\top F(x^k)\right\|_{\mathbf{W}_k}^2 \\
&= \quad \hat{f}_k(x^k) - \gamma\left(1 - \frac{\gamma L}{2}\right)\left\|F(x^k)\right\|_{\mathbf{S}_k\left(\mathbf{S}_k^\top\nabla F_k^\top \mathbf{W}_k^{-1}\nabla F_k \mathbf{S}_k\right)^\dagger \mathbf{S}_k^\top}^2 \\
&\overset{\gamma=1/L}{=} \quad \hat{f}_k(x^k) - \frac{\gamma}{2}\left\|F(x^k)\right\|_{\mathbf{S}_k\left(\mathbf{S}_k^\top\nabla F_k^\top \mathbf{W}_k^{-1}\nabla F_k \mathbf{S}_k\right)^\dagger \mathbf{S}_k^\top}^2 && (63)
\end{aligned}
$$

where in (63) we use the identity $\mathbf{M}^\dagger \mathbf{M}\mathbf{M}^\dagger = \mathbf{M}^\dagger$ with $\mathbf{M} = \mathbf{S}_k^\top\nabla F_k^\top \mathbf{W}_k^{-1}\nabla F_k \mathbf{S}_k$. Taking the expectation conditioned on x_k in the inequality (63) gives

$$
\begin{aligned}
\mathbb{E}\left[\hat{f}_{k+1}(x^{k+1}) \mid x^k\right] \quad &\leq \quad \mathbb{E}\left[\hat{f}_k(x^k) \mid x^k\right] - \frac{\gamma}{2}\left\|F(x^k)\right\|_{\mathbf{H}(x^k)}^2 \\
&\leq \quad \mathbb{E}\left[\hat{f}_k(x^k) \mid x^k\right] - \rho\gamma\hat{f}_k(x^k).
\end{aligned}
$$

Taking full expectation and expanding the recurrence gives finally

$$
\mathbb{E}\left[\hat{f}_{k+1}(x^{k+1})\right] \leq (1 - \rho\gamma)\mathbb{E}\left[\hat{f}_k(x^k)\right].
$$

$\square$

E.5   SNRVM for solving linear systems

Here we consider the simplified case in which our objective function (1) is a quadratic function. In this case, the stationarity condition (5) is a linear system. To simplify the notation, let us denote in this section the resulting linear system as

$$
\mathbf{A}x = b, \quad \text{where } \mathbf{A} \in \mathcal{M}_p(\mathbb{R}),\ p = d(n+1). \tag{64}
$$

In other words, our nonlinear map is given by $F(x) = \mathbf{A}x - b$. Because $\nabla F(x) = \mathbf{A}^\top$, where $\mathbf{A}$ is a square matrix, we see that the assumptions on F in Assumption 4.2 are verified if and only if $\mathbf{A}$ is invertible.

In this setting the SNR method (16) is known as the sketch-and-project method (Gower and Richtárik, 2015a). The sketch-and-project method has been shown to converge linearly at a fast rate (Gower and Richtárik, 2015a,b). Thus this quadratic case serves as a good sanity check to verify if our rate of convergence in Theorem 4.5 recovers the well known fast linear rate of the sketch-and-project method. This is precisely what we investigate in the next lemma. It only remains to reformulate Assumption 4.4, which we do in the following lemma.

Lemma E.5. If $\{\mathbf{W}_k\}_{k\in\mathbb{N}}$ is a sequence of invertible matrices such that

$$\mathbf{W}_{k+1} \preceq \mathbf{W}_k, \tag{65}$$

and $\mathbf{A}$ is invertible, then Assumption 4.4 holds with $L = 1$.

Proof. Using the fact that $F(x) = \mathbf{A}x - b$ and $\nabla F(x) = \mathbf{A}^\top$, we can rewrite definition (19) as

$$\hat{f}_k(x) = \frac{1}{2}\|\mathbf{A}x - b\|^2_{(\mathbf{A}\mathbf{W}_k^{-1}\mathbf{A}^\top)^\dagger}. \tag{66}$$

Now, since $\mathbf{A}$ is invertible, we have $(\mathbf{A}\mathbf{W}_k^{-1}\mathbf{A}^\top)^\dagger = \mathbf{A}^{\top^{-1}}\mathbf{W}_k\mathbf{A}^{-1}$. So, if x^* is the unique solution to (64), then we obtain

$$\hat{f}_k(x) = \frac{1}{2}\|x - x^*\|^2_{\mathbf{W}_k}. \tag{67}$$

Using Assumption 65 together with the fact that $\hat{f}_k$ is quadratic to conclude that

$$\begin{aligned}
\hat{f}_{k+1}(x^{k+1}) &\overset{(65)}{\leq} \hat{f}_k(x^{k+1}) \\
&= \hat{f}_k(x^k) + \langle \nabla \hat{f}_k(x^k), x^{k+1} - x^k \rangle + \frac{1}{2}\|x^{k+1} - x^k\|^2_{\nabla^2 \hat{f}_k(x^k)} \\
&= \hat{f}_k(x^k) + \langle \nabla \hat{f}_k(x^k), x^{k+1} - x^k \rangle + \frac{1}{2}\|x^{k+1} - x^k\|^2_{\mathbf{W}_k}.
\end{aligned}$$

$\square$

Proposition E.6. Let $\mathbf{A} \in \mathcal{M}_p(\mathbb{R})$ be invertible, $b \in \mathbb{R}^p$, and x^* be the solution to (64). Let $(x^k)_{k\in\mathbb{N}}$ be a sequence generated from SNRVM (18), with $F(x) = \mathbf{A}x - b$, and $\gamma = 1$. We assume that, at every iteration $k \in \mathbb{N}$, the matrices $(\mathbf{S}_k, \mathbf{W}_k)$ are sampled from a finite proper distribution (see Assumption 4.1) such that for all x, $\mathcal{D}_x$ is independant of x, that $\mathbb{E}_{\mathcal{D}_x}[\mathbf{S}\mathbf{S}^\top]$ is invertible and $\mathbf{W}_k$ is constant and equal to some invertible matrix $\mathbf{W} \in \mathcal{M}_p(\mathbb{R})$.

Let

$$\rho \overset{\text{def}}{=} \lambda_{\min}\left(\mathbf{W}^{-1/2}\mathbf{A}^\top\mathbb{E}_{\mathcal{D}_{x^0}}[\mathbf{S}(\mathbf{S}^\top\mathbf{A}\mathbf{W}^{-1}\mathbf{A}^\top\mathbf{S})^\dagger\mathbf{S}^\top]\mathbf{A}\mathbf{W}^{-1/2}\right).$$

It follows that $\rho \in (0, 1)$, and

$$\mathbb{E}\left[\|x^k - x^*\|^2_{\mathbf{W}}\right] \leq (1 - \rho)\|x^2 - x^*\|^2_{\mathbf{W}}. \tag{68}$$

Proof. We are going to apply the result in Theorem 4.5, so we start by checking its assumptions. First, our assumptions on the sampling ensure that Assumption 4.1 is verified. Second, as discussed earlier in this section, the fact that $\mathbf{A}$ is invertible ensures that Assumption 4.2 holds true. Third, our assumption that $\mathbf{W}_k \equiv \mathbf{W}$ together with Lemma E.5 tells us that Assumption 4.4 holds with $L = 1$, meaning that we take a stepsize $\gamma = 1$. Let $\mathbf{H} \overset{\text{def}}{=} \mathbb{E}_{\mathcal{D}_{x^0}}[\mathbf{S}(\mathbf{S}^\top\mathbf{A}\mathbf{W}^{-1}\mathbf{A}^\top\mathbf{S})^\dagger\mathbf{S}^\top]$. Note that this matrix is independant of k, because we assumed the distribution $\mathcal{D}_x$ to be independant of x. We also know that $\mathbf{H}$ is invertible, thanks to Lemma E.4. Therefore, $\rho = \lambda_{min}\left(\mathbf{W}^{-1/2}\mathbf{A}^\top\mathbf{H}\mathbf{A}\mathbf{W}^{-1/2}\right) > 0$.

To prove that $\rho \leq 1$, observe that

$$\mathbf{W}^{-1/2}\mathbf{A}^\top \mathbb{E}[\mathbf{S}(\mathbf{S}^\top \mathbf{A}\mathbf{W}^{-1}\mathbf{A}^\top \mathbf{S})^\dagger \mathbf{S}^\top]\mathbf{A}\mathbf{W}^{-1/2}$$
$$= \mathbb{E}[\mathbf{W}^{-1/2}\mathbf{A}^\top \mathbf{S}(\mathbf{S}^\top \mathbf{A}\mathbf{W}^{-1/2}\mathbf{W}^{-1/2}\mathbf{A}^\top \mathbf{S})^\dagger \mathbf{S}^\top \mathbf{A}\mathbf{W}^{-1/2}]$$
$$= \mathbb{E}[(\mathbf{S}^\top \mathbf{A}\mathbf{W}^{-1/2})^\top ((\mathbf{S}^\top \mathbf{A}\mathbf{W}^{-1/2})(\mathbf{S}^\top \mathbf{A}\mathbf{W}^{-1/2}))^\dagger (\mathbf{S}^\top \mathbf{A}\mathbf{W}^{-1/2})]$$
$$= \mathbb{E}[(\mathbf{S}^\top \mathbf{A}\mathbf{W}^{-1/2})^\dagger (\mathbf{S}^\top \mathbf{A}\mathbf{W}^{-1/2})],$$

where $(\mathbf{S}^\top \mathbf{A}\mathbf{W}^{-1/2})^\dagger (\mathbf{S}^\top \mathbf{A}\mathbf{W}^{-1/2})$ is the orthogonal projection onto the range of $\mathbf{W}^{-1/2}\mathbf{A}^\top \mathbf{S}$. Consequently $\lambda_{max}((\mathbf{S}^\top \mathbf{A}\mathbf{W}^{-1/2})^\dagger (\mathbf{S}^\top \mathbf{A}\mathbf{W}^{-1/2})) \leq 1$, and from Jensen's inequality, we deduce that the eigenvalues of its expectation also are in $(0, 1]$. Whence $\rho \in (0, 1]$.

To conclude the proof, see that under our assumptions, the quantity $\rho(x)$ defined in Theorem 4.5 is independent of x, and equal to ρ. We have verified all the assumptions needed to call Theorem 4.5, which proves the claim. $\square$

The rate of convergence given in (68) is exactly the well known linear rate of convergence given in Theorem 4.6 in (Gower and Richtárik, 2015b). For example, if $\mathbf{A}$ is symmetric positive definite, and we can set $\mathbf{W}_k \equiv \mathbf{A}$ and sample the sketching matrix $\mathbf{S} \in \mathbb{R}^{p \times 1}$ according to

$$\mathbb{P}[\mathbf{S} = e_i] = \frac{\mathbf{A}_{ii}}{\mathbf{Trace}(\mathbf{A})}, \quad \text{for } i = 1, \ldots, m.[9]$$

With this choice of sketch and metric, the resulting method (18) is known as coordinate descent (Leventhal and Lewis, 2010; Gower and Richtárik, 2015a). In this case, our resulting rate in (68) is controlled by

$$\rho = \lambda_{\min}\left(\mathbf{A}^{1/2}\mathbb{E}\left[e_i (\mathbf{A}_{ii})^\dagger e_i^\top\right] \mathbf{A}^{1/2}\right) = \frac{\lambda_{\min}(\mathbf{A})}{\mathbf{Trace}(\mathbf{A})},$$

which is exactly the celebrated linear convergence rate of coordinate descent first given in (Leventhal and Lewis, 2010).

Proposition E.6 shows that our main convergence theory in Theorem 4.5 is *tight* in this quadratic setting. That is, when specialized to a linear mapping $F(x)$ and a fixed metric $\mathbf{W} \equiv \mathbf{W}_k$, our Theorem 4.5 recovers the best known convergences results as a special case.

E.6   Proof of Theorem 4.7

Lemma E.7. Let Assumption 4.6 hold. Let $x \in \Omega$, let $(\hat{\mathbf{S}}, \hat{\mathbf{W}})$ be in the domain of $\mathcal{D}_x$, and consider $\mathbf{A} := \hat{\mathbf{W}}^{-1/2}\nabla F(x)\mathbf{H}(x)\nabla F(x)^\top \hat{\mathbf{W}}^{-1/2}$, where $\mathbf{H}(x)$ is defined in (53). Then

$$\lambda_{min}(\mathbf{A}) \geq \frac{\mu_{\nabla F}^2}{L_{\nabla F}^2}\frac{\mu_W}{L_W}\frac{\bar{\mu}_S}{L_S}.$$

Proof. Let us write $\mathbf{H} := \mathbf{H}(x)$, $J := \nabla F(x)$ and $U := J^\top \hat{\mathbf{W}}^{-1/2}$, so that $\mathbf{A} = U^\top \mathbf{H} U$. Therefore,

$$\lambda_{min}(\mathbf{A}) \geq \lambda_{min}(U^\top U)\lambda_{min}(\mathbf{H}).$$

From Assumption 4.2 we know that $\hat{\mathbf{W}}$ is invertible, and also that J is injective, so from Assumption 4.6 and the fact that J is a square matrix, we deduce that J is invertible, and therefore deduce that U is invertible as well. This means that $\lambda_{min}(U^\top U) > 0$ and that

$$\lambda_{min}(U^\top U) = \lambda_{min}(UU^\top) = \lambda_{min}(J^\top \hat{\mathbf{W}}^{-1}J) \geq \lambda_{min}(J^\top J)\lambda_{min}(\hat{\mathbf{W}}^{-1}) = \frac{\sigma_{min}(J)^2}{\lambda_{max}(\mathbf{W})} \geq \frac{\mu_{\nabla F}^2}{L_W}.$$

Now we turn to $\mathbf{H}$, and write $\mathbf{H} = \mathbb{E}[\mathbf{S}B\mathbf{S}^\top]$, where $B = (\mathbf{S}^\top G\mathbf{S})^\dagger$, with $G = J^\top \mathbf{W}^{-1}J$. From the same arguments as above, we know that that G is invertible under our assumptions. So, using properties of the pseudo inverse with the fact that G is symmetric and Lemma E.1, we can write that

$$\mathbf{Null}(B) = \mathbf{Null}((\mathbf{S}^\top G\mathbf{S})^\top) = \mathbf{Null}(\mathbf{S}^\top G\mathbf{S}) = \mathbf{Null}(\mathbf{S}).$$

---

[9]Here $e_i \in \mathbb{R}^m$ is the i-th unit coordinate vector and $\mathbf{Trace}(\mathbf{A}) = \sum_{i=1}^m \mathbf{A}_{ii}$ is the trace of $\mathbf{A}$.

Therefore, for all $x \in \mathbb{R}^p$ we have $\mathbf{S}^\top x \in \mathbf{Null}\,(B)^\perp$. So, by noting $\lambda^*_{min}(B)$ the smallest nonzero eigenvalue of B, we can write that

$$\langle \mathbf{S} B \mathbf{S}^\top x, x \rangle = \langle B\mathbf{S}^\top x, B\mathbf{S}^\top \rangle \geq \lambda^*_{min}(B)\|\mathbf{S}^\top x\|^2 = \lambda^*_{min}(B)\langle \mathbf{S}\mathbf{S}^\top x, x \rangle.$$

Here

$$\lambda^*_{min}(B) = \|B^\dagger\|^{-1} = \|\mathbf{S}^\top G \mathbf{S}\|^{-1} \geq \|\mathbf{S}\mathbf{S}^\top\|^{-1}\|G\|^{-1} \geq \|\mathbf{S}\mathbf{S}^\top\|^{-1}\|J^\top J\|^{-1}\|\mathbf{W}^{-1}\|^{-1} \geq L_S^{-1}L_{\nabla F}^{-2}\mu_W,$$

where we used the fact that $\|\mathbf{W}^{-1}\|^{-1} = \lambda_{min}(\mathbf{W})$ and $\|J^\top J\|^{-1} = \sigma_{max}(J)^{-2}$. By combining those last inequalities we obtain that

$$
\begin{aligned}
\langle \mathbf{H}x, x \rangle &= \mathbb{E}\left[\langle \mathbf{S}B\mathbf{S}^\top x, x \rangle\right] \\
&\geq L_S^{-1}L_{\nabla F}^{-2}\mu_W \mathbb{E}\left[\langle \mathbf{S}\mathbf{S}^\top x, x \rangle\right] \\
&= L_S^{-1}L_{\nabla F}^{-2}\mu_W \langle \mathbb{E}\left[\mathbf{S}\mathbf{S}^\top\right] x, x \rangle \\
&\geq L_S^{-1}L_{\nabla F}^{-2}\mu_W \bar{\mu}_S\|x\|^2.
\end{aligned}
$$

This means that $\lambda_{min}(\mathbf{H}) \geq L_S^{-1}L_{\nabla F}^{-2}\mu_W\bar{\mu}_S$. If we recombine all our inequalities, we ultimately obtain that

$$\lambda_{min}(\mathbf{A}) \geq \frac{\mu^2_{\nabla F}}{L_W}L_S^{-1}L_{\nabla F}^{-2}\mu_W\bar{\mu}_S,$$

which is what we needed. $\qquad\square$

Lemma E.8. Let Assumption 4.6 hold. Let $x \in \Omega$, let $(\hat{\mathbf{S}}, \hat{\mathbf{W}})$ be in the domain of $\mathcal{D}_x$. Then

$$\lambda_{min}\left((\nabla F(x)^\top \mathbf{W}^{-1}\nabla F(x))^\dagger\right) \geq L_{\nabla F}^{-2}\mu_W > 0.$$

In particular, for all $k \in \mathbb{N}$, if $x^k \in \Omega$ almost surely then

$$\mathbb{E}\left[\hat{f}_k(x^k)\right] \geq \frac{\mu_W}{2L_{\nabla F}^2}\mathbb{E}\left[\|F(x^k)\|^2\right] \text{ a.s.}$$

Proof. We have $\lambda_{min}\left((\nabla F(x)^\top \mathbf{W}^{-1}\nabla F(x))^\dagger\right) = \|\nabla F(x)^\top \mathbf{W}^{-1}\nabla F(x)\|^{-1}$ where

$$\|\nabla F(x)^\top \mathbf{W}^{-1}\nabla F(x)\| \leq \sigma_{max}(\nabla F(x))^2\lambda_{min}(\mathbf{W})^{-1} \leq L_{\nabla F}^2\mu_W^{-1},$$

which gives the desired lower bound on the eigenvalues. Now, given $x^k \in \Omega$ we immediately deduce that

$$\hat{f}_k(x^k) \geq \frac{\mu_W}{2L_{\nabla F}^2}\|F(x^k)\|^2.$$

The conclusion follows by taking the expectation on this inequality. $\qquad\square$

Proof of Theorem 4.7. Keeping the notations of Theorem 4.5, we see from Lemma E.7 that we can take

$$\rho = \frac{\mu^2_{\nabla F}}{L^2_{\nabla F}}\frac{\mu_W}{L_W}\frac{\bar{\mu}_S}{L_S} > 0,$$

from which we obtain that

$$(\forall k \in \mathbb{N}) \quad \mathbb{E}\left[\hat{f}_k(x^k)\right] \leq (1-\rho\gamma)^k \mathbb{E}\left[\hat{f}_0(x^0)\right] \quad \text{almost surely.}$$

We now can use Lemma E.8 to lower bound the left member of that inequality, and obtain

$$(\forall k \in \mathbb{N}) \quad \frac{\mu_W}{2L_{\nabla F}^2}\mathbb{E}\left[\|F(x^k)\|^2\right] \leq (1-\rho\gamma)^k \mathbb{E}\left[\hat{f}_0(x^0)\right] \quad \text{almost surely.}$$

The conclusion follows by taking

$$C = \mathbb{E}\left[\hat{f}_0(x^0)\right]\frac{L^2_{\nabla F}}{\mu_W}.$$

$\qquad\square$

E.7 Proof of Theorem 4.8

The proof of Theorem 4.8, which can be found at the end of this section, will combine Theorem 4.7 with the forthcoming Propositions E.9 and E.13.

Proposition E.9. Let Assumption 1.1 hold. For SAN and SANA, Assumption 4.6 holds on every compact set Ω.

Proof. First, remember that Assumption 4.2 holds for SAN and SANA (see Proposition 4.3), and that $m = p = (n+1)d$. Now, let Ω be a compact set, and verify that the bounds in Assumption 4.6 hold.

We start with the sketching matrices $\mathbf{S}$, for which we know (see the proof of Proposition 4.3 in Section E.1) that

$$\|\mathbf{S}\mathbf{S}^\top\| = 1 \quad \text{and} \quad \mathbb{E}\left[\mathbf{S}\mathbf{S}^\top\right] = \mathbf{Diag}\left(\pi \mathbf{I}_d, \frac{1-\pi}{n}\mathbf{I}_d, \cdots, \frac{1-\pi}{n}\mathbf{I}_d\right) \text{ or } \mathbf{Diag}\left(\mathbf{I}_d, \frac{1}{n}\mathbf{I}_d, \cdots, \frac{1}{n}\mathbf{I}_d\right).$$

In both cases, we see that we can take $L_S = 1$ and $\bar{\mu}_L = \min\{\frac{1}{n}, \frac{1-\pi}{n}, \pi\}$.

Second, let $\mathbf{W}_i$ be in the domain of $\mathcal{D}_x$. According to their definition in (47, 48), and because the f_i is of class C^2 (see Assumption 1.1), we know that each $\mathbf{W}_i$ is continuous with respect to x. Moreover, we know (again from Assumption 1.1) that $\mathbf{W}_i$ is definite positive : $\lambda_{min}(\mathbf{W}_i) > 0$. This is true for every $x \in \Omega$, so by continuity of λ_{min} and the compactness of Ω, we deduce that $\inf_{x \in \Omega} \lambda_{min}(\mathbf{W}_i) > 0$. Similarly, $\sup_{x \in \Omega} \lambda_{max}(\mathbf{W}_i) < +\infty$. This means that the constants μ_W and L_W are well defined in $(0, +\infty)$.

Finally, we need to control the singular values of $\nabla F(x)$ over Ω. We use here the same arguments that we used for $\mathbf{W}_i$: $\nabla F(x)$ is continuous with respect to x, and it is invertible (because it is square and injective, see Proposition 4.3). $\qquad\square$

Now we will show that Assumption 4.6 holds if f is smooth and strongly convex.

Assumption E.10. There exists $0 < \mu_f \le L_f$ such that for all $i \in \{1, \ldots, n\}$, the function $f_i : \mathbb{R}^d \longrightarrow \mathbb{R}$ is of class C^2, μ_f-strongly convex and has a L_f-Lipschitz continuous gradient.

Lemma E.11. Let $\varphi : [0, +\infty) \longrightarrow [1, +\infty)$ be defined as

$$\varphi(t) := \sqrt{1 + \frac{1}{2}\left(t + \sqrt{4t + t^2}\right)}. \tag{69}$$

1. $\varphi(t)$ is well defined and increasing on $[0, +\infty[$.

2. $\varphi(t)^{-1} = \sqrt{1 + \frac{1}{2}\left(t - \sqrt{4t + t^2}\right)}$.

3. For all $a \in (0, +\infty)$, $\varphi(at)\varphi(t^{-1})t^{-1/2}$ is decreasing on $(0, +\infty)$.

4. For all $a \in (0, +\infty)$, and all $t \ge 1$, $\varphi(at)\varphi(t^{-1}) \le \varphi(1)\sqrt{t}\sqrt{2 + a}$.

Proof. 1 : It is well defined because $t + \sqrt{4t + t^2} \ge 0$. It is increasing because it is the composition, sum and product of increasing functions on $[0, +\infty[$. Point 2 is a simple exercise. Point 3 is a bit more technical. Let $\phi(t) = \varphi(at)^2\varphi(t^{-1})^2t^{-1}$, which is the square of the quantity of interest. We can compute its derivative, and a some effort we obtain that

$$t^3\phi'(t) = -\left[t + \frac{1}{2}\left(1 + \sqrt{1 + 4t}\right)\right]\left[1 + \frac{at}{\sqrt{4at + a^2t^2}}\right] - \frac{1}{2}\left[1 + \frac{1 + 2t}{\sqrt{1 + 4t}}\right]\left[1 + \frac{1}{2}\left(at + \sqrt{4at + a^2t^2}\right)\right].$$

It is clear that the above expression is the sum of two negative terms, implying that ϕ is decreasing. For item 4, we use the monotonicity of item 3 to get

$$\varphi(at)\varphi(t^{-1}) \le \varphi(a \cdot 1)\varphi(1)\sqrt{t}.$$

From the inequality $a + \sqrt{a^2 + 4a} \le 2a + 2$ (it is easy to prove it by rearranging the terms and taking the square), we deduce that

$$\varphi(a) \le \sqrt{1 + \frac{1}{2}(2a + 2)} = \sqrt{2 + a}.$$

$\square$

Lemma E.12. Let $\mathbf{A} \in \mathbb{R}^{m \times d}$ be an injective matrix. Let φ be defined as in (69) and consider:

$$A := \begin{bmatrix} \mathbf{I}_d & \mathbf{0}_{d,m} \\ \mathbf{A} & \mathbf{I}_m \end{bmatrix}.$$

Then $\|A\| = \varphi\left(\|\mathbf{A}^\top \mathbf{A}\|\right)$.

Proof. We start by remembering that $\|A\|$ is the largest singular value of A. The singular values of A are exactly the square root of the eigenvalues of $A^\top A$, which is given by

$$A^\top A := \begin{bmatrix} \mathbf{I}_d + \mathbf{A}^\top \mathbf{A} & \mathbf{A}^\top \\ \mathbf{A} & \mathbf{I}_m \end{bmatrix}.$$

We compute its eigenvalues by finding the roots of its characteristic polynomial, that we note $P \in \mathbb{R}[X]$. Using a simple formula for computing the determinant of a 2×2 block matrix, we can write, for all $X \neq -1$:

$$
\begin{aligned}
P(X) &= \det \begin{bmatrix} (1 - X)\mathbf{I}_d + \mathbf{A}^\top \mathbf{A} & \mathbf{A}^\top \\ \mathbf{A} & (1 - X)\mathbf{I}_m \end{bmatrix} \\
&= \det\left((1 - X)\mathbf{I}_m\right) \det\left((1 - X)\mathbf{I}_d + \mathbf{A}^\top \mathbf{A} - \mathbf{A}^\top((1 - X)\mathbf{I}_m)^{-1}\mathbf{A}\right) \\
&= (1 - X)^m \det\left((1 - X)\mathbf{I}_d + \mathbf{A}^\top \mathbf{A} - \frac{1}{1 - X}\mathbf{A}^\top \mathbf{A}\right) \\
&= (1 - X)^{m-d} \det\left((1 - X)^2\mathbf{I}_d + (1 - X)\mathbf{A}^\top \mathbf{A} - \mathbf{A}^\top \mathbf{A}\right) \\
&= (1 - X)^{m-d} \det\left((1 - X)^2\mathbf{I}_d - X\mathbf{A}^\top \mathbf{A}\right).
\end{aligned}
$$

The right member of this equality is polynomial in X, since the determinant of a matrix is polynomial in its coefficients, and our assumption that $\mathbf{A}$ is injective implies that $m - d \ge 0$. In particular this right member is well defined and continuous at $X = -1$, which means that the equality holds true for every $X \in \mathbb{R}$.

We see that 1 is a root of P, with multiplicity $m - d$. The other roots are the zeroes of $\det\left((1 - X)^2\mathbf{I}_d - X\mathbf{A}^\top \mathbf{A}\right)$, for which we see that

$$
\begin{aligned}
\det\left((1 - X)^2\mathbf{I}_d - X\mathbf{A}^\top \mathbf{A}\right) = 0 \quad &\Leftrightarrow \quad (1 - X)^2 \text{ is an eigenvalue of } X\mathbf{A}^\top \mathbf{A} \\
&\Leftrightarrow \quad (1 - X)^2 = X\lambda \text{ for } \lambda \in \mathrm{spec}(\mathbf{A}^\top \mathbf{A}) \\
&\Leftrightarrow \quad X = 1 + \frac{1}{2}\left(\lambda \pm \sqrt{4\lambda + \lambda^2}\right) \text{ for } \lambda \in \mathrm{spec}(\mathbf{A}^\top \mathbf{A}),
\end{aligned}
$$

which gives us the remaining $2d$ roots (counted with multiplicity). This proves that the singular values of A are 1 (with multiplicity $m - d$) and (see Lemma E.11.2)

$$\{\varphi(\lambda), \varphi(\lambda)^{-1} \mid \lambda \in \mathrm{spec}\left(\mathbf{A}^\top \mathbf{A}\right)\}.$$

Since φ is increasing (Lemma E.11.1), and $\varphi(\lambda) \ge 1$, we conclude that the largest singular value of A is $\varphi\left(\|\mathbf{A}^\top \mathbf{A}\|\right)$.

$\square$

Proposition E.13. Let Assumption E.10 hold, and consider the SAN (resp. SANA) algorithm. Let $c = \sqrt{\frac{3 + \sqrt{5}}{2}}$.

Then Assumption 4.6 is verified, with $\Omega = \mathbb{R}^p$ and:

$$\mu_W = \min\{1, \mu_f\}, \qquad\qquad L_W = \max\{1, L_f\},$$
$$\bar{\mu}_S = \min\left\{\tfrac{1-\pi}{n}, \pi\right\} \left(\text{resp. } \bar{\mu}_S = \tfrac{1}{n}\right), \quad L_S = 1,$$
$$\mu_{\nabla F} = \frac{\mu_W}{c\sqrt{n}\sqrt{2+L^2}}, \qquad\qquad L_{\nabla F} = L_W c\sqrt{n}\sqrt{2+L^2}.$$

Proof. Let $x \in \mathbb{R}^p$ be fixed, $J := \nabla F(x) \in \mathbb{R}^{p \times p}$, and $(\mathbf{S}, \mathbf{W})$ in the domain of $\mathcal{D}_x$. We need to find uniform spectral bounds on those three matrices. For $\mathbf{S}$, we have seen already in the proof of Proposition E.9 that we can take $L_S = 1$, and $\bar{\mu}_L = \frac{1}{n}$ for SAN, or $\min\{\frac{1-\pi}{n}, \pi\}$ for SANA. For $\mathbf{W}$, we see directly from (47, 48) that it is a block-diagonal matrix, whose eigenvalues are included in $[\mu_W, L_W]$, with $\mu_W = \min\{1, \mu_f\}$ and $L_W = \max\{1, L_f\}$. The rest of this proof is dedicated to the study of J, which requires more work.

Remember that the expression for J is given in (39). We write for convenience that

$$J = \begin{bmatrix} \mathbf{0}_d & \mathbf{H} \\ \frac{1}{n}\mathbf{E}^\top & -\mathbf{I}_{nd} \end{bmatrix} \quad \text{with} \quad \mathbf{E} := [\mathbf{I}_d, \cdots, \mathbf{I}_d] \in \mathbb{R}^{d \times nd}, \quad \mathbf{H} := [H_1, \cdots, H_n] \in \mathbb{R}^{d \times nd}, \quad H_i := \nabla^2 f_i(w).$$

Let us now introduce a few more matrices. Let $\bar{H} := \nabla^2 f(w)$, which can equivalently be written as $\bar{H} = \frac{1}{n}\sum_i H_i = \frac{1}{n}\mathbf{H}\mathbf{E}^\top$. Now consider

$$U := \begin{bmatrix} \mathbf{I}_d & \mathbf{0}_{d,nd} \\ -\mathbf{H}^\top & \mathbf{I}_{nd} \end{bmatrix}, \quad D := \begin{bmatrix} \bar{H} & \mathbf{0}_{d,nd} \\ \mathbf{0}_{nd,d} & -\mathbf{I}_{nd} \end{bmatrix}, \quad V := \begin{bmatrix} \mathbf{I}_d & \mathbf{0}_{d,nd} \\ \frac{-1}{n}\mathbf{E}^\top & \mathbf{I}_{nd} \end{bmatrix}.$$

Note that those three matrices are triangular, and invertible because $\bar{H}$ is invertible. It is easy to see that $J = U^\top DV$. Indeed,

$$DV = \begin{bmatrix} \bar{H} & \mathbf{0}_{d,nd} \\ \frac{1}{n}\mathbf{E}^\top & -\mathbf{I}_{nd} \end{bmatrix}, \quad U^\top DV = \begin{bmatrix} \bar{H} - \mathbf{H}\frac{1}{n}\mathbf{E}^\top & \mathbf{H} \\ \frac{1}{n}\mathbf{E}^\top & -\mathbf{I}_{nd} \end{bmatrix} = \begin{bmatrix} \mathbf{0}_d & \mathbf{H} \\ \frac{1}{n}\mathbf{E}^\top & -\mathbf{I}_{nd} \end{bmatrix},$$

where the last equality comes from the fact that $\mathbf{H}\frac{1}{n}\mathbf{E}^\top = \bar{H}$. Therefore, it remains to upper bound the right member of

$$\sigma_{max}(J) = \sqrt{\lambda_{max}(J^\top J)} = \|J\| \le \|D\|\|U\|\|V\|.$$

Bounding the smallest singular value will follow the same argument. Indeed, from our assumptions, $J^T J$ is invertible (see Proposition 4.3), but J is a square matrix, therefore J itself is invertible. In consequence, we can write that $J^{-1} = V^{-1\top}D^{-1}U^{-1}$, so that

$$\sigma_{min}(J) = \frac{1}{\|J^{-1}\|} \ge \frac{1}{\|D^{-1}\|\|U^{-1}\|\|V^{-1}\|},$$

where one easily computes that

$$U^{-1} = \begin{bmatrix} \mathbf{I}_d & \mathbf{0}_{d,nd} \\ \mathbf{H}^\top & \mathbf{I}_{nd} \end{bmatrix}, \quad D^{-1} = \begin{bmatrix} \bar{H}^{-1} & \mathbf{0}_{d,nd} \\ \mathbf{0}_{nd,d} & -\mathbf{I}_{nd} \end{bmatrix}, \quad V^{-1} = \begin{bmatrix} \mathbf{I}_d & \mathbf{0}_{d,nd} \\ \frac{1}{n}\mathbf{E}^\top & \mathbf{I}_{nd} \end{bmatrix}.$$

It is easy to see, given our smoothness and strong convexity assumptions, that

$$\|D\| = \max\{1, L\} = L_W \quad \text{and} \quad \|D^{-1}\| = \max\{1, \mu^{-1}\} = \mu_W^{-1}.$$

Now, observe that V and V^{-1} share the same structure, so we can call Lemma E.12 with $\mathbf{A} = \frac{1}{n}\mathbf{E}^T$ or $-\frac{1}{n}\mathbf{E}^T$. In both cases $\mathbf{A}^\top\mathbf{A} = n^{-1}\mathbf{I}_d$, meaning that $\|\mathbf{A}^\top\mathbf{A}\| = n^{-1}$, and so we deduce that

$$\|V\| = \|V^{-1}\| = \varphi(n^{-1}).$$

Finally, we do the same for U and U^{-1} : we use Lemma E.12 with $\mathbf{A} = \pm\mathbf{H}^\top$. In both cases $\mathbf{A}^\top\mathbf{A} = \mathbf{H}\mathbf{H}^\top = \sum_{i=1}^n H_i^2$, whose eigenvalues belong to $[n\mu^2, nL^2]$. Due to the monotonicity of φ, we deduce that

$$\|U\| = \|U^{-1}\| \le \varphi(nL^2).$$

As a result, we conclude that

$$\sigma_{min}(J) \geq \frac{1}{\mu_W^{-1}\varphi(n^{-1})\varphi(nL^2)} = \frac{\mu_W}{\varphi(n^{-1})\varphi(nL^2)},$$

while

$$\sigma_{max}(J) \leq L_W\varphi(n^{-1})\varphi(nL^2).$$

We can then conclude by using Lemma E.11.4 and noting $c = \varphi(1)$. $\qquad\square$

Proof of Theorem 4.8. Suppose first that a) holds. Then we can use Theorem 4.7 together with Proposition E.13, and combine the constants into play. For the sake of the presentation, we assume $\pi = 1/(n+1)$, so that we can have a unique lower bound for SAN and SANA : $\bar{\mu}_S = \min\{\frac{1}{n}, \frac{1}{n+1}\} \geq \frac{1}{2n}$. We also simplify the expression of c, by using the bounds $c^2 \leq 3$ and $c^4 \leq 7$. If instead b) holds, then there exists a compact set Ω containing almost surely the sequence. So it remains to combine Theorem 4.7 together with Proposition E.9. $\qquad\square$