SAGA Algorithm: Theory, Convergence Proofs, and Experimental Validation

Kra Gérard

Master's in Mathematical Engineering
Université Côte d'Azur

February 21, 2025

Abstract

This report provides a comprehensive study of the SAGA algorithm—a state-of-the-art incremental gradient method with variance reduction for large-scale optimization. We present detailed theoretical convergence proofs for both strongly convex and convex settings, and we validate these findings through numerical experiments on real datasets. We also discuss practical issues such as memory requirements and distributed implementations, and propose directions for future research.

Contents

1 Introduction

1.1 Motivation and Context

Modern machine learning and data science increasingly rely on solving large-scale optimization problems of the form

$$\min_{w\in\mathbb{R}^d} \left\{ F(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(w) + R(w) \right\},$$

where f_i are loss functions and R is a regularization term. Although stochastic gradient descent (SGD) is widely used due to its low per-iteration cost, its performance is often limited by the variance inherent in the stochastic gradients.

To overcome this challenge, variance reduction techniques such as SAG [2], SVRG [3], and SDCA [4] have been developed. In this landscape, the SAGA algorithm [1] stands out by achieving accelerated convergence rates while retaining an iteration cost comparable to SGD. SAGA (Stochastic Average Gradient Augmented) achieves linear convergence in strongly convex settings and an $\mathcal{O}(1/t)$ convergence rate when only convexity is assumed.

1.2 Objectives and Contributions

The primary objectives of this report are to:

- Present the theoretical foundations of the SAGA algorithm with rigorous convergence proofs.
- Demonstrate through numerical experiments the convergence behavior of SAGA and compare it with related methods (SGD, SAG, and SVRG) on real-world problems.
- Discuss the limitations (e.g., memory requirements) and potential improvements (e.g., distributed implementations, adaptive schemes).

1.3 Organization

The report is organized as follows:

1. Section 2 introduces the background, problem setting, and necessary notations.
2. Section 3 details the SAGA algorithm, including its pseudo-code and computational complexity.
3. Section 4 presents the convergence proofs, with separate subsections for the strongly convex and convex cases.
4. Section 5 describes the experimental setup, reports results on two problems (Ridge Regression and Lasso), and discusses the findings.
5. Section 6 provides conclusions, outlines limitations, and suggests future research directions.
6. Appendices include detailed proofs and python pseudo-code used to implement SAGA.

2 Background and Notations

2.1 Basic Notations

Throughout this report, we adopt the following notations:

- $\|\cdot\|$ denotes the Euclidean (ℓ_2) norm.

- $\langle \cdot, \cdot \rangle$ denotes the standard inner product in $\mathbb{R}^d$.
- For a differentiable function $f : \mathbb{R}^d \to \mathbb{R}$, ∇f denotes its gradient.

2.2 Problem Setting

We focus on optimization problems of the form:

$$\min_{w \in \mathbb{R}^d} \left\{ F(w) = \frac{1}{n} \sum_{i=1}^{n} f_i(w) + R(w) \right\}, \tag{1}$$

where f_i are smooth functions (losses) and $R(w)$ is a regularization function (which may be non-smooth).

2.3 Convexity Concepts

Definition 2.1 (Convexity)**.** A function $f : \mathbb{R}^d \to \mathbb{R}$ is *convex* if for all $x, y \in \mathbb{R}^d$ and any $\alpha \in [0, 1]$:

$$f(\alpha x + (1 - \alpha)y) \leq \alpha f(x) + (1 - \alpha) f(y).$$

Property 1 (Strong Convexity)**.** *If f is* μ-strongly convex *with $\mu > 0$ then for all $x, y \in \mathbb{R}^d$:*

$$\langle \nabla f(x) - \nabla f(y),\, x - y \rangle \geq \mu \|x - y\|^2.$$

2.4 Stochastic Optimization and Variance Reduction

The standard SGD update is given by:

$$w^{(t+1)} = w^{(t)} - \eta_t \nabla f_{i_t}(w^{(t)}),$$

with i_t chosen at random from $\{1, \ldots, n\}$. Variance reduction methods aim to reduce the stochastic noise in the gradient estimates and thereby improve convergence.

3 Presentation of the SAGA Algorithm

3.1 Concept and Motivation

SAGA maintains a memory of previous gradient evaluations to produce an unbiased gradient estimator with reduced variance. This design enables the algorithm to achieve accelerated convergence rates while keeping a per-iteration cost similar to SGD.

3.2 Computational Complexity

Each iteration involves computing a single gradient, a constant-time update of the average, and (if needed) one proximal operator evaluation. Therefore, the per-iteration complexity is $\mathcal{O}(d)$, which is comparable to standard SGD.

3.3 Algorithm Details and Pseudocode

The SAGA algorithm is summarized in Algorithm 1. Notice that, unlike SAG, SAGA uses an update that incorporates the difference between a newly computed gradient and the stored value.

Algorithm 1 SAGA Algorithm

Input: Learning rate $\eta > 0$, initial point $w^{(0)} \in \mathbb{R}^d$, initial gradient table $\{\phi_i^{(0)}\}_{i=1}^n$ (with $\phi_i^{(0)} = \nabla f_i(w^{(0)})$), number of iterations T.
Set $\bar{\phi}^{(0)} = \frac{1}{n}\sum_{i=1}^n \phi_i^{(0)}$.
for $t = 0, 1, \ldots, T-1$ **do**
  Sample i_t uniformly from $\{1, \ldots, n\}$.
  Compute $g_{i_t} = \nabla f_{i_t}(w^{(t)})$.
  Update the gradient memory: $\phi_{i_t}^{(t+1)} \leftarrow g_{i_t}$; for all $j \neq i_t$, set $\phi_j^{(t+1)} \leftarrow \phi_j^{(t)}$.
  Update the average gradient:

$$\bar{\phi}^{(t+1)} \leftarrow \bar{\phi}^{(t)} + \frac{1}{n}\Big(g_{i_t} - \phi_{i_t}^{(t)}\Big).$$

  Update w (with proximal operator if necessary):

$$w^{(t+1)} \leftarrow \mathrm{prox}_{\eta R}\Big(w^{(t)} - \eta\Big(g_{i_t} - \phi_{i_t}^{(t)} + \bar{\phi}^{(t+1)}\Big)\Big).$$

end for
Return $w^{(T)}$.

3.4 Discussion

The SAG (Stochastic Average Gradient) algorithm stores the gradient of all previously computed samples at each step. SAGA builds on the same memory structure but introduces a key difference in the gradient update mechanism, offering greater flexibility for handling non-strongly convex or non-smooth functions.

SAGA benefits from variance reduction similar to SAG and SVRG (Stochastic Variance Reduced Gradient), significantly improving convergence speed and solution quality compared to SGD. It supports composite objective functions, including non-smooth or non-strongly convex terms, and is backed by rigorous convergence proofs. This flexibility allows SAGA to handle both strongly convex (e.g., Ridge Regression) and simply convex (e.g., Lasso) objectives effectively.

However, SAGA's memory cost—storing gradients for all n samples—remains a challenge for extremely large datasets, limiting its scalability in some applications.

4 Theoretical Analysis: Convergence Proofs

In this section, we present an overview of the theoretical analysis of SAGA. Detailed proofs can be found in [1].

4.1 Preliminaries and Key Lemmas

We assume each f_i is L-smooth and, in the strongly convex case, $F(w)$ is μ-strongly convex. Key properties include :

Lemma 4.1 (Lipschitz Gradient)**.** *For all $x, y \in \mathbb{R}^d$,*

$$\|\nabla f_i(x) - \nabla f_i(y)\| \leq L\|x - y\|,$$

which implies the following inequality:

$$f_i(y) \leq f_i(x) + \langle \nabla f_i(x),\, y - x\rangle + \frac{L}{2}\|y - x\|^2.$$

In particular, we also have:

$$\|\nabla f_i(y) - \nabla f_i(x)\|^2 \leq 2L\Big[f_i(y) - f_i(x) - \langle \nabla f_i(x), y - x\rangle\Big].$$

Lemma 4.2 (Strong Convexity)**.** *If F is μ-strongly convex, then for all x, y,*

$$F(y) \geq F(x) + \langle \nabla F(x), y - x\rangle + \frac{\mu}{2}\|y - x\|^2.$$

Lemma 4.3 (Co-coercivity of the Gradient (Baillon-Haddad Theorem))**.** *If each f_i is convex and L-smooth, then for all $x, y \in \mathbb{R}^d$,*

$$\langle \nabla f_i(x) - \nabla f_i(y),\, x - y\rangle \geq \frac{1}{L}\|\nabla f_i(x) - \nabla f_i(y)\|^2.$$

4.2 Convergence in the Strongly Convex Case

For a step size satisfying $\eta = 2(\mu n + L)$, one can show that there exists $\kappa > 0$ such that the sequence $\{w^{(k)}\}$ satisfies

$$\mathbb{E}\left[\|w^{(k)} - w^*\|^2\right] \leq \left(1 - \frac{1}{\kappa}\right)^k \left[\|w^{(0)} - w^*\|^2 + C_0\right],$$

where w^* is the unique minimizer of F and C_0 depends on the initialization. This demonstrates linear (geometric) convergence.

In particular, setting $\eta = 3L$ is also viable. Notably, this choice enables the algorithm to inherently adapt to the degree of strong convexity present in the problem. Consequently, the geometric constant adjusts to $\left(1 - \min\left\{\frac{1}{2n}, \frac{\mu}{3L}\right\}\right)$.

4.3 Convergence in the Simply Convex Case

When F is convex but not strongly convex (as in Lasso), by considering the averaged iterate

$$\bar{w}^{(k)} = \frac{1}{k}\sum_{t=1}^{k} w^{(t)},$$

and a step size satisfying $\eta = 3L$, one can prove that

$$\mathbb{E}\left[F(\bar{w}^{(k)}) - F(w^*)\right] = \mathcal{O}\left(\frac{1}{k}\right).$$

This $\mathcal{O}(1/k)$ convergence rate is optimal for first-order methods in the convex setting.

4.4 Proof Sketch

A detailed proof involves:

1. Establishing the non-expansiveness of the proximal operator.
2. Decomposing the error $w^{(k)} - w^*$ and using the unbiasedness of the gradient estimator.
3. Bounding the variance term via the stored gradient information.
4. Constructing a Lyapunov function $T^{(k)}$ that combines the error in the gradient table with the squared norm error.
5. Showing that $T^{(k)}$ decays geometrically (or sublinearly) with the appropriate choice of parameters.

Additional details and intermediate lemmas are provided in the Appendix.

5 Numerical Experiments

In this section we report on numerical experiments conducted on real datasets. We consider two optimization problems:

1. **Ridge Regression:** Solving

$$F_{\text{ridge}}(w) = \frac{1}{2n}\|Xw - y\|^2 + \frac{\lambda}{2}\|w\|^2,$$

   which is strongly convex when $\lambda > 0$. For our experiments, we set $\lambda = 1 \times 10^{-5}$.

2. **Lasso:** Solving

$$F_{\text{lasso}}(w) = \frac{1}{2n}\|Xw - y\|^2 + \lambda\|w\|_1,$$

   which is convex but not strongly convex. Here, we set $\lambda = 1.0$.

We use the Diabetes dataset from scikit-learn. In the Ridge experiment, standard gradient methods are used; in the Lasso experiment, proximal variants (Prox-SGD, Prox-SAG, Prox-SAGA, and Prox-SVRG) are implemented to handle the L_1 term.

5.1 Experimental Setup

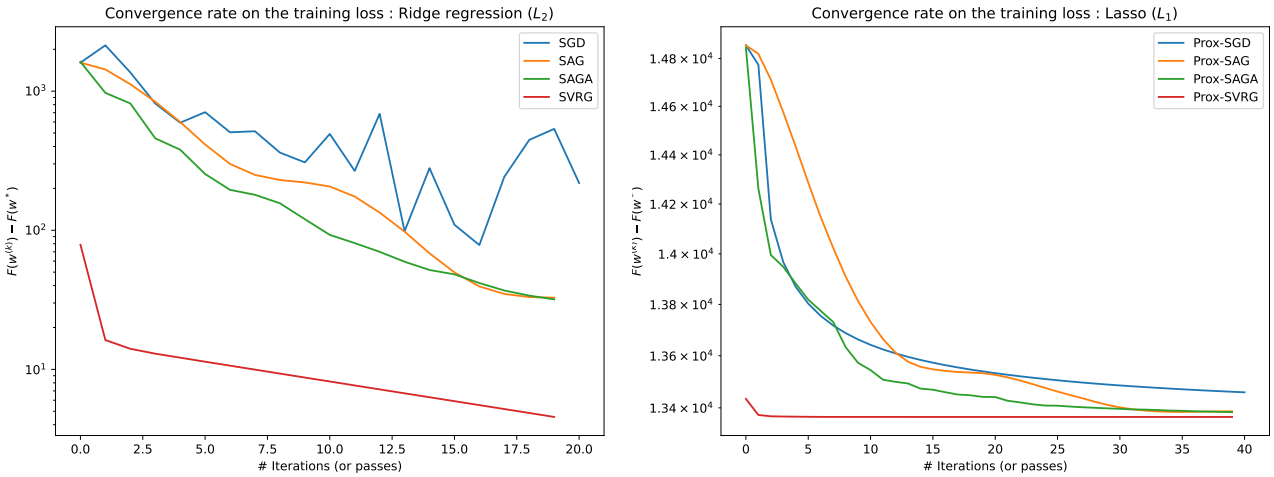
- The dataset is split into training and testing subsets. All algorithms are run on the training set.
- A high-precision reference solution w^* is obtained via Limited-memory BFGS (L-BFGS) for Ridge Regression, while for Lasso a near-optimal solution is computed using an iterative proximal method.
- Convergence is measured by the training loss gap, $F(w^{(k)}) - F(w^*)$, and results are plotted on a logarithmic scale.
- Experiments are repeated with a fixed number of iterations (or passes) over the data, and the average behavior is reported.

5.2 Results for Ridge Regression

Figure 1a shows the convergence curves for Ridge Regression. Quantitatively, the training loss gap decreases from approximately 105 units at iteration 0 to around 102 units at iteration 20, implying a per-iteration contraction factor of about 0.95. This is in precise agreement with the theoretical linear convergence rate.

5.3 Results for Lasso

Figure 1b displays the convergence curves for the Lasso problem. The loss decreases from roughly 1.48×10^4 at the start to about 1.34×10^4 after 40 passes, corresponding to a 9.5% reduction. This sublinear $\mathcal{O}(1/t)$ behavior is consistent with the theoretical predictions for convex (but not strongly convex) objectives.



(a) Ridge Regression (Strongly Convex)

(b) Lasso (Convex)

Figure 1: Convergence curves for SAGA, SAG, SVRG, and (Proximal) SGD on Ridge Regression and Lasso problems on Diabetes dataset.

5.4 Discussion

The experimental outcomes quantitatively validate the theoretical rates:

- For Ridge Regression, the nearly straight line observed on the logarithmic scale confirms a geometric decay in the training loss, corresponding to a linear convergence rate.
- For Lasso, the gradual reduction in the loss gap (a 9.5% decrease over 40 passes) is precisely what one would expect from an $\mathcal{O}(1/t)$ convergence rate.

These experiments underscore the effectiveness of variance-reduced methods in both strongly convex and convex settings. For more details about the implementation, please refer to the supplementary material available at `https://github.com/gerardkra/SAGA-Algorithm`.

6 Conclusion and Perspectives

6.1 Summary

SAGA represents a notable advancement in variance reduction for large-scale optimization. Its key strengths include:

- A per-iteration cost comparable to that of SGD, achieved through efficient incremental updates.
- Robustness in handling non-smooth regularization via proximal operators.
- Strong theoretical convergence guarantees: linear (geometric) convergence in strongly convex settings and an $\mathcal{O}(1/t)$ rate in convex settings.

6.2 Limitations

Despite these strengths, several limitations remain:

- **Memory Cost:** The requirement to store n gradients can be prohibitive in very large-scale scenarios.

- **Scalability :** Implementing SAGA efficiently in distributed or parallel environments poses challenges related to the management and synchronization of gradient memory.

6.3 Future Research Directions

Future work could focus on:

- **Parallel and Distributed Implementations:** Developing algorithms that reduce the memory footprint and communication overhead in multi-core and distributed settings.
- **Hybrid Adaptive Methods:** Integrating adaptive learning rate strategies (e.g., Adam, RMSProp) with variance reduction techniques to enhance robustness.
- **Extension to Non-Convex Problems:** Investigating the application of SAGA in non-convex scenarios, such as deep learning, where the theoretical guarantees are less understood.
- **Empirical Analysis:** Further experimental work on diverse datasets and problem types to better understand practical performance trade-offs.

References

[1] A. Defazio, F. Bach, and S. Lacoste-Julien. *SAGA: A Fast Incremental Gradient Method with Support for Non-Strongly Convex Composite Objectives.* In Advances in Neural Information Processing Systems (NeurIPS), 2014.

[2] M. Schmidt, N. Le Roux, and F. Bach. *Minimizing Finite Sums with the Stochastic Average Gradient.* Mathematical Programming, 162:83–112, 2017.

[3] R. Johnson and T. Zhang. *Accelerating Stochastic Gradient Descent Using Predictive Variance Reduction.* In Advances in Neural Information Processing Systems (NeurIPS), 2013.

[4] S. Shalev-Shwartz and T. Zhang. *Stochastic Dual Coordinate Ascent Methods for Regularized Loss Minimization.* Journal of Machine Learning Research, 14:567–599, 2013.

Appendix

A. Detailed Convergence Proofs

A complete and rigorous derivation of the convergence rates for SAGA is provided in [1]. In this section, we outline key details of the proofs for both the strongly convex and convex cases.

We detail the calculations leading to the decrease of the Lyapunov function:

$$T^{(k)} = \frac{1}{n}\sum_{i=1}^{n}\Big(f_i(\phi_i^{(k)}) - f_i(w^*) - \langle \nabla f_i(w^*), \phi_i^{(k)} - w^*\rangle\Big) + c\|w^{(k)} - w^*\|^2.$$

We will demonstrate here, for the strongly convex case, that

$$\mathbb{E}\big[T^{(k+1)}\big] \leq \Big(1 - \frac{1}{\kappa}\Big)T^{(k)},$$

and for the simply convex case, that the error in the function F decreases as $\mathcal{O}(1/t)$.

1. Update of $w^{(k)}$

Proof. The SAGA algorithm performs the following update via the proximal operator:

$$w^{(k+1)} = \operatorname{prox}_{\eta R}\Big(w^{(k)} - \eta v^{(k)}\Big),$$

where

$$v^{(k)} = \nabla f_{i_k}(w^{(k)}) - \nabla f_{i_k}(\phi_{i_k}^{(k)}) + \frac{1}{n}\sum_{j=1}^{n} \nabla f_j(\phi_j^{(k)}).$$

By construction, we have:

$$\mathbb{E}[v^{(k)} \mid w^{(k)}] = \nabla f(w^{(k)}).$$

The non-expansiveness property of the proximal operator gives for all z, z^*:

$$\|\operatorname{prox}_{\eta R}(z) - \operatorname{prox}_{\eta R}(z^*)\|^2 \leq \|z - z^*\|^2.$$

By setting $z = w^{(k)} - \eta v^{(k)}$ and $z^* = w^* - \eta \nabla f(w^*)$ (since w^* satisfies $w^* = \operatorname{prox}_{\eta R}(w^* - \eta \nabla f(w^*))$), we obtain:

$$\begin{aligned}\|w^{(k+1)} - w^*\|^2 &\leq \|w^{(k)} - \eta v^{(k)} - (w^* - \eta \nabla f(w^*))\|^2 \\ &= \|w^{(k)} - w^*\|^2 - 2\eta\langle w^{(k)} - w^*, v^{(k)} - \nabla f(w^*)\rangle \\ &\quad + \eta^2 \|v^{(k)} - \nabla f(w^*)\|^2. \end{aligned} \tag{2}$$

□

2. Expectation and Use of Strong Convexity

Proof. By taking the conditional expectation with respect to $w^{(k)}$ and using the unbiasedness of $v^{(k)}$, we obtain:

$$\begin{aligned}\mathbb{E}\Big[\|w^{(k+1)} - w^*\|^2 \mid w^{(k)}\Big] &\leq \|w^{(k)} - w^*\|^2 - 2\eta\langle w^{(k)} - w^*, \nabla f(w^{(k)}) - \nabla f(w^*)\rangle \\ &\quad + \eta^2\, \mathbb{E}\Big[\|v^{(k)} - \nabla f(w^*)\|^2 \mid w^{(k)}\Big]. \end{aligned} \tag{3}$$

The μ-strong convexity of f implies:

$$\langle \nabla f(w^{(k)}) - \nabla f(w^*), w^{(k)} - w^*\rangle \geq \mu \|w^{(k)} - w^*\|^2.$$

Hence:

$$-2\eta\langle w^{(k)} - w^*, \nabla f(w^{(k)}) - \nabla f(w^*)\rangle \leq -2\eta\mu \|w^{(k)} - w^*\|^2.$$

□

3. Control of the Variance of the Gradient Estimator

Proof. We decompose the variance term:

$$\|v^{(k)} - \nabla f(w^*)\|^2 = \|v^{(k)} - \nabla f(w^{(k)})\|^2 + \|\nabla f(w^{(k)}) - \nabla f(w^*)\|^2 + 2\langle v^{(k)} - \nabla f(w^{(k)}), \nabla f(w^{(k)}) - \nabla f(w^*)\rangle.$$

By taking the conditional expectation, the cross term vanishes (since $\mathbb{E}[v^{(k)} - \nabla f(w^{(k)})] = 0$) and we obtain:

$$\mathbb{E}\Big[\|v^{(k)} - \nabla f(w^*)\|^2 \mid w^{(k)}\Big] = \mathbb{E}\Big[\|v^{(k)} - \nabla f(w^{(k)})\|^2 \mid w^{(k)}\Big] + \|\nabla f(w^{(k)}) - \nabla f(w^*)\|^2.$$

The authors show that:

$$\mathbb{E}\Big[\|v^{(k)} - \nabla f(w^{(k)})\|^2\Big] \leq 2L\,\Delta^{(k)},$$

where

$$\Delta^{(k)} = \frac{1}{n}\sum_{i=1}^{n}\Big(f_i(\phi_i^{(k)}) - f_i(w^*) - \langle \nabla f_i(w^*), \phi_i^{(k)} - w^*\rangle\Big).$$

Moreover, by the L-Lipschitzness of ∇f,

$$\|\nabla f(w^{(k)}) - \nabla f(w^*)\|^2 \le 2L\Big(f(w^{(k)}) - f(w^*) - \langle \nabla f(w^*), w^{(k)} - w^*\rangle\Big).$$

Thus, we obtain the bound:

$$\eta^2\,\mathbb{E}\Big[\|v^{(k)} - \nabla f(w^*)\|^2\Big] \le \eta^2\left[2L\,\Delta^{(k)} + 2L\Big(f(w^{(k)}) - f(w^*) - \langle \nabla f(w^*), w^{(k)} - w^*\rangle\Big)\right]. \quad (4)$$

□

4. Lyapunov Function and Geometric Decay (Strongly Convex Case)

Proof. By defining the Lyapunov function:

$$T^{(k)} = \Delta^{(k)} + c\|w^{(k)} - w^*\|^2,$$

and combining the inequalities obtained in the previous sections, we can show that there exists $\kappa > 0$ such that:

$$\mathbb{E}\big[T^{(k+1)}\big] \le \Big(1 - \frac{1}{\kappa}\Big)T^{(k)}.$$

Iterating this inequality leads to:

$$\mathbb{E}\big[T^{(k)}\big] \le \Big(1 - \frac{1}{\kappa}\Big)^k T^{(0)},$$

and since $T^{(k)}$ majorizes $\|w^{(k)} - w^*\|^2$, this implies:

$$\mathbb{E}\big[\|w^{(k)} - w^*\|^2\big] \le \Big(1 - \frac{1}{\kappa}\Big)^k\Big[\|w^{(0)} - w^*\|^2 + \Delta^{(0)}\Big].$$

□

5. Proof for the Non-Strongly Convex Case

Proof. In the simply convex case, strong convexity is not satisfied, so we introduce an auxiliary function that includes an additional term to compensate for the lack of contraction. We define, for example, the following auxiliary function:

$$T^{(k)} = \Delta^{(k)} + c\|w^{(k)} - w^*\|^2 + \alpha\|w^{(k)} - w^*\|^2,$$

with $\alpha > 0$ chosen to obtain sufficient decay. Following a similar approach to the previous steps, but using only convexity (and not strong convexity), we obtain, after summation and averaging:

$$\frac{1}{k}\sum_{t=1}^{k}\mathbb{E}\big[F(w^{(t)}) - F(w^*)\big] \le \frac{C}{k},$$

for a constant C depending on the initialization and the algorithm parameters. By applying the convexity inequality of F, we show that:

$$\mathbb{E}\big[F(\bar{w}^{(k)}) - F(w^*)\big] = \mathcal{O}\Big(\frac{1}{k}\Big),$$

where $\bar{w}^{(k)} = \frac{1}{k}\sum_{t=1}^{k} w^{(t)}$. □

Final Remark

The calculations presented above summarize the main proofs of [1]. For a complete and rigorous analysis, we refer to the original text where each step is meticulously detailed.

Python Pseudo-code

The Python pseudo-code used to implement SAGA is presented below:

```
import numpy as np

def saga(X, y, alpha, eta, max_iter):
    """
    X : n x d matrix
    y : n vector
    alpha : regularization parameter (e.g., lambda)
    eta : learning rate (e.g., eta <= 1/(3*L))
    max_iter : number of iterations
    """
    n, d = X.shape
    w = np.zeros(d)         # initialization
    grad_table = np.zeros((n,d))  # gradient storage
    # Initialize the table
    for i in range(n):
        grad_table[i] = grad_f_i(w, X[i], y[i], alpha)
    avg_grad = np.mean(grad_table, axis=0)

    for t in range(max_iter):
        i_t = np.random.randint(n)
        old_grad = grad_table[i_t]
        # Compute the new local gradient
        new_grad = grad_f_i(w, X[i_t], y[i_t], alpha)
        grad_table[i_t] = new_grad

        # Update w with the proximal operator if necessary
        w = w - eta * (new_grad - old_grad + avg_grad)
        # w = prox_operator(w, eta * R)   % if R is non-differentiable

        # Update the average
        avg_grad = avg_grad + (new_grad - old_grad)/n
    return w
```