Non-Convex Self-Concordant Functions: Practical Algorithms and Complexity Analysis

Donald Goldfarb[*] Lexiao Lai[†] Tianyi Lin[‡] Jiayu Zhang[§]

Abstract

We extend the standard notion of self-concordance to non-convex optimization and develop a family of second-order algorithms with global convergence guarantees. In particular, two function classes – *weakly self-concordant* functions and *F-based self-concordant* functions – generalize the self-concordant framework beyond convexity, without assuming the Lipschitz continuity of the gradient or Hessian. For these function classes, we propose a regularized Newton method and an adaptive regularization method that achieve an ϵ-approximate first-order stationary point in $O(\epsilon^{-2})$ iterations. Equipped with an oracle capable of detecting negative curvature, the adaptive algorithm can further attain convergence to an approximate second-order stationary point. Our experimental results demonstrate that the proposed methods offer superior robustness and computational efficiency compared to cubic regularization and trust-region approaches, underscoring the broad potential of self-concordant regularization for large-scale and neural network optimization problems.

1 Introduction

Newton's method is one of the cornerstones of optimization, dating back to the early development of iterative methods. In convex optimization, it is known for its local quadratic convergence when the objective function is smooth and strongly convex [48]. Despite its efficiency in the local sense, the Newton method often lacks robustness in non-convex settings: without appropriate safeguards, it converges to saddle points or exhibit global convergence slower than first-order methods [9]. To address these limitations, extensive research has focused on enhancing its global behavior through various regularization techniques. Among them, cubic regularization methods [47, 44, 10, 11] control step size by penalizing large curvature through a cubic term; trust-region methods [13, 16, 14, 17] restrict each update to a region where the local quadratic model is accurate; and regularized Newton methods [41, 20, 26] balance curvature exploitation and stability via adaptive damping terms. Collectively, these approaches have established rigorous global convergence guarantees and strong empirical performance across a broad range of non-convex optimization problems.

However, these approaches suffer from some drawbacks [41, 26]. A primary one is computational: both cubic regularization methods [47] and trust-region methods [13] require solving a nontrivial subproblem at each iteration that involves cubic or constrained quadratic models rather than updating via a simple Hessian inversion. This additional subproblem limits scalability to high-dimensional or large-scale problems. Moreover, their theoretical guarantees depend on strong smoothness conditions, such as the Lipschitz continuity of the Hessian [47, 44, 10, 11, 13, 16, 17, 41, 20, 26], and sometimes even of

[*]Department of Industrial Engineering and Operations Research, Columbia University (goldfarb@columbia.edu).
[†]Department of Mathematics, The University of Hong Kong (lai.lexiao@hku.hk).
[‡]Department of Industrial Engineering and Operations Research, Columbia University (tl3335@columbia.edu).
[§]Department of Industrial Engineering and Operations Research, Columbia University (jz3824@columbia.edu).

the gradient [16, 17, 26]. These conditions are often unrealistic in various applications, where Hessian information may be ill-behaved or expensive to approximate accurately.

In the convex setting, these challenges can be effectively mitigated through the notion of self-concordance [46, 45], which provides a unified framework for studying second-order methods without strong smoothness. Self-concordant functions may possess unbounded third-order derivatives, yet the growth of derivatives is locally controlled by the curvature encoded in the Hessian. This structure ensures that Newton steps remain well-behaved in regions where strong smoothness fails. A canonical example is the logarithmic barrier function, which is self-concordant but lacks a Lipschitz continuous gradient or Hessian. For such functions, second-order methods (e.g., the damped Newton) achieve global convergence and local quadratic convergence, offering both theoretical elegance and computational practicality. Despite its success in convex optimization, extending self-concordance to non-convex settings remains nontrivial. Although its definition appears superficially independent of convexity, it involves the square root of local curvature terms, which can become ill-defined when the Hessian has negative eigenvalues. One might attempt to circumvent this by taking absolute values before square roots, but this modification imposes restrictive curvature bounds that fail to capture simple non-convex structures; for example, the sigmoid function $x \mapsto 1/(1+e^{-x})$ violates this property. This gap motivates developing a generalized notion of self-concordance suitable for non-convex objectives, preserving its curvature control while accommodating regions of negative curvature.

Related works The classical notion of self-concordance was introduced by [46] as a fundamental tool for analyzing interior-point methods in convex optimization. Since then, this notion was extended in various contexts, including composite optimization problems [61], distributed Newton methods [67], randomized Newton updates [35] and stochastic quasi-Newton methods [24]. In parallel, other works studied stronger variants of this notion [52, 27], which impose tighter curvature control to improve numerical stability. A recent line of works have began to adopt the generalized notion of quasi-self-concordance, which encompasses important non-quadratic objectives such as logistic regression [4, 61, 5, 68, 33, 19]. Further generalization was proposed by [62], who introduced the notion of generalized self-concordance. Building upon this formulation, subsequent works developed and analyzed Frank-Wolfe methods, demonstrating the versatility of self-concordance-based curvature control beyond the standard convex setting [22, 7, 23].

Our proposed algorithms can also be viewed as second-order methods for non-convex optimization. In this setting, numerous regularization techniques have been developed to ensure global convergence of Newton-type methods. Classical strategies include cubic regularization [47, 44], trust-region frameworks [13], and more recent regularized Newton variants [41, 20], all designed to stabilize Newton steps in the presence of non-convex curvature. A major milestone in this line of work is the adaptive cubic regularization method [10, 11], which established a global iteration complexity of $O(\epsilon^{-1.5})$ for minimizing functions with Lipschitz continuous Hessian. This framework has inspired extensive research aimed at improving regularization strategies, reducing computational cost, and refining complexity guarantees [16, 14, 15, 17, 21, 26]. To enhance computational efficiency, several works have exploited Hessian-vector products as major oracles, achieving convergence rates on the order of $O(\epsilon^{-7/4})$ while avoiding explicit Hessian computations [3, 8]. A complementary line of research [40, 42, 25] has leveraged negative curvature directions to escape saddle points and approach approximate second-order stationary points. Building on these ideas, [54, 53] proposed hybrid line-search algorithms that alternate between Newton updates and negative-curvature steps identified through the Lanczos method [31], offering a robust trade-off between curvature exploitation and global progress.

Contributions This paper extends the notion of self-concordance to non-convex optimization and develops corresponding second-order methods with theoretical guarantees and empirical validation. As in convex settings, our framework accommodates functions with unbounded higher-order derivatives, thereby covering a broad range of applications. Our contributions can be summarized as follows:

1. We introduce two function classes – weakly self-concordant and F-based self-concordant functions – that extend the classical notion of self-concordance to non-convex settings. These classes preserve key curvature control properties while encompassing many objectives arising in machine learning.

2. We propose regularized Newton and adaptive regularization methods and prove their global and local convergence guarantees without the gradient or Hessian Lipschitz continuity. We extend the adaptive regularization methods [10, 11, 12] to constrained problems with self-concordant barriers.

3. We conduct experiments on non-negative matrix factorization and neural network training, demonstrating that our methods achieve superior robustness and efficiency compared with existing second-order approaches.

Organization In Section 2, we define non-convex self-concordant functions, and provide three examples of their applicability. In Section 3, we describe our methods and present convergence results. In Section 4, we provide several lemmas and prove our convergence results. In Section 5, we present empirical results complementing our theoretical results. We conclude in Section 6.

2 Non-Convex Self-Concordant Functions

Letting $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ where $\mathbf{dom}(f) = \{x \in \mathbb{R}^n \mid f(x) < +\infty\}$, the problem of interest is given by

$$\min_{x\in\mathbb{R}^n} f(x). \tag{1}$$

In the convex setting, the self-concordant function is defined in [45, Definition 5.1.1]. For the sake of completeness, we summarize it as follows,

Definition 2.1. *A closed and convex function $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is κ-self-concordant if $\mathbf{dom}(f)$ is open, f is three times continuously differentiable on $\mathbf{dom}(f)$, and $\nabla^3 f(x)[h,h,h] \leq 2\kappa\big(\nabla^2 f(x)[h,h]\big)^{3/2}$ for all $x \in \mathbf{dom}(f)$ and $h \in \mathbb{R}^n$.*

We next extend this notion to non-convex settings.

Definition 2.2. *A function $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is (κ, ℓ)-weakly self-concordant if the function $f(x) + \frac{\ell}{2}\|x\|^2$ is κ-self-concordant.*

The weakly self-concordant functions form a subclass of weakly convex functions [18]. Indeed, f is (κ, ℓ)-weakly self-concordant if and only if $f(x) + \frac{\ell}{2}\|x\|^2$ is closed and convex, and for all $x \in \mathbf{dom}(f)$ and $h \in \mathbb{R}^n$, we have

$$\nabla^3 f(x)[h,h,h] = \nabla^3 \left(f + \tfrac{\ell}{2}\|\cdot\|^2\right)(x)[h,h,h] \leq 2\kappa\left(\nabla^2 f(x)[h,h] + \ell\|h\|^2\right)^{3/2}.$$

In particular, if f is ℓ-weakly convex and satisfies $|\nabla^3 f(x)[h,h,h]| \leq m\|h\|^3$, then f is $(\frac{m}{2a^{3/2}}, \ell + a)$-weakly self-concordant for any $a > 0$.

Definition 2.3. *Given a reference function $F : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$, we say that f is F-based* κ-self-concordant *if $\mathbf{dom}(f) \subseteq \mathbf{dom}(F)$ and $f + F$ is κ-self-concordant.*

Classical and weakly self-concordant functions are two special instances of general F-based self-concordant functions. This formulation unifies analysis across different settings and enjoys convenient closure properties.

Proposition 2.4. *Let $\alpha_1, \alpha_2 > 0$. If f_i is F_i-based κ_i-self-concordant for $i = 1, 2$, then $\alpha_1 f_1 + \alpha_2 f_2$ is $(\alpha_1 F_1 + \alpha_2 F_2)$-based $\max(\frac{\kappa_1}{\sqrt{\alpha_1}}, \frac{\kappa_2}{\sqrt{\alpha_2}})$-self-concordant.*

Proof. By definition, we have $f_i + F_i$ is κ_i-self-concordant for $i = 1, 2$. Then, [45, Theorem 5.1.1] shows that $\alpha_1(f_1 + F_1) + \alpha_2(f_2 + F_2)$ is $\max(\frac{\kappa_1}{\sqrt{\alpha_1}}, \frac{\kappa_2}{\sqrt{\alpha_2}})$-self-concordant. This implies the desired result. □

2.1 Examples

In what follows, we present three representative examples of F-based self-concordant functions with full details.

Generalized phase retrieval As shown in [58], the loss function $f : \mathbb{R}^{2n} \to \mathbb{R}$ is defined as follows (a_k^* represents the conjugate transpose of a_k),

$$f(z^R, z^I) = \tfrac{1}{2m}\sum_{k=1}^{m}(|a_k^* z|^2 - y_k^2)^2 \text{ for } a_k = \tfrac{1}{\sqrt{2}}(a_k^R + a_k^I\sqrt{-1}) \in \mathbb{C}^n, \ z = z^R + z^I\sqrt{-1}.$$

This is a (κ, ℓ)-weakly self-concordant function for some $\kappa > 0$ and $\ell > 0$. Indeed, we can rewrite the loss function as

$$f(z^R, z^I) = \tfrac{1}{2m}\sum_{k=1}^{m}\left(\tfrac{1}{2}((a_k^R)^\top z^R + (a_k^I)^\top z^I)^2 + \tfrac{1}{2}((a_k^R)^\top z^I - (a_k^I)^\top z^R)^2 - y_k^2\right)^2.$$

By Proposition 2.4, we claim that $g(x) = ((c_1^\top x)^2 + (c_2^\top x)^2)^2$ is weakly self-concordant. Indeed, we denote $c_i^\top x$ by x_i and $c_i^\top h$ by h_i for $i = 1, 2$. Then, we have $\nabla^2 g(x)[h, h] = 8(x_1h_1 + x_2h_2)^2 + 4(x_1^2 + x_2^2)(h_1^2 + h_2^2)$. For some $\ell \geq \|c_1\|^2 + \|c_2\|^2$, we have

$$(\nabla^2 g(x)[h, h] + \ell\|h\|^2)^{\frac{3}{2}} \geq \sum_{i,j\in\{1,2\}}(|x_i|^3|h_j|^3 + |h_i|^3 + |h_j|^3).$$

We also have

$$\nabla^3 g(x)[h, h, h] = 24(x_1h_1 + x_2h_2)(h_1^2 + h_2^2) \leq 8\sum_{i,j\in\{1,2\}}(|x_i|^3|h_j|^3 + |h_i|^3 + |h_j|^3)$$

Putting these pieces together yields that g is $(4, \ell)$-weakly self-concordant.

The above results can be generalized; indeed, any polynomial functions are F-based self-concordant for some functions F.

Proposition 2.5. *Let $p \geq 2$ be integer-valued. Suppose that $f : \mathbb{R}^n \to \mathbb{R}$ is a multivariate polynomial with degree $\deg(f) \leq 2p$. Then, $F(x) = (\|x\|^2 + 1)^p$ is convex and there exists $m \geq 0$ such that f is mF-based 1-self-concordant.*

Proof. Since $\deg(f) \leq 2p$, each entry of $\nabla^2 f(x)$ (resp., $\nabla^3 f(x)$) is a polynomial of degree at most $2p-2$ (resp., at most $2p-3$). There exist $c_1, c_2 \geq 0$ such that, for all $x, h \in \mathbb{R}^n$, we have $|\nabla^2 f(x)[h, h]| \leq c_1(1+\|x\|^2)^{p-1}\|h\|^2$ and $|\nabla^3 f(x)[h, h, h]| \leq c_2(1+\|x\|^2)^{p-3/2}\|h\|^3$. In addition, we have $|\nabla^3 F(x)[h, h, h]| \leq c_3(1 + \|x\|^2)^{p-\frac{3}{2}}\|h\|^3$.

We define $\varphi(r) = (r+1)^p$. Then, we have $\nabla F(x) = 2\varphi'(\|x\|^2)x$, $\nabla^2 F(x) = 2\varphi'(\|x\|^2)I + 4\varphi''(\|x\|^2)xx^\top$. For all $h \in \mathbb{R}^n$, we have

$$\nabla^2 F(x)[h,h] = 2\varphi'(\|x\|^2)\|h\|^2 + 4\varphi''(\|x\|^2)(x^\top h)^2 \geq 2p(\|x\|^2+1)^{p-1}\|h\|^2.$$

Thus, F is convex. Define $g = f + mF$, we have $\nabla^2 g(x)[h,h] \geq (2pm - c_1)(\|x\|^2+1)^{p-1}\|h\|^2$. Moreover, $|\nabla^3 g(x)[h,h,h]| \leq (c_2 + mc_3)(\|x\|^2+1)^{p-3/2}\|h\|^3$. Thus, there exists $m \geq 0$ such that $g = f + mF$ is 1-self-concordant. $\square$

We consider the constrained polynomial optimization problem $\min_{x \in C} f(x)$, where f is a polynomial, and suppose there exists a self-concordant function $g : \mathbb{R}^n \to \mathbb{R}$ such that $\mathbf{dom}(g) = C$. By Proposition 2.5, we have $f + F$ is self-concordant on $\mathbb{R}^n$. Thus, we have $f + F + g$ is self-concordant on C and $f \mid_C$ is $(F+g)$-based self-concordant. This perspective provides numerous examples of F-based self-concordant functions, such as linear regression, matrix factorization, sensor network localization [59], and D-optimal design [60], without applying the logarithm in the objective and with the sum of decision variables being less than 1 rather than equal to 1.

Many problems admit loss functions that can be modified to be F-based self-concordant for some function F. For our next example, we define $h_\alpha(x) = \alpha^{-1}(\log(1+e^{\alpha x}) + \log(1+e^{-\alpha x}))$. For small $\alpha > 0$, h_α serves as a smooth approximation to $|x|$ [55], and it is 0.5α-self-concordant. After some calculation, it can be verified that $\frac{\partial^2}{\partial x^2}h_\alpha(x) = \frac{\alpha}{2}(1 - \tanh^2(\frac{\alpha x}{2}))$, $\frac{\partial^3}{\partial x^3}h_\alpha(x) = -\frac{\alpha^2 \tanh(\alpha x/2)}{2\cosh^2(\alpha x/2)}$, and that $|\frac{\partial^3}{\partial x^3}h_\alpha(x)| \leq \alpha(\frac{\partial^2}{\partial x^2}h_\alpha(x))$ for $x \geq 0$.

Sparse dictionary learning As introduced in [36], we consider the optimization problem in the following form of

$$\min_{\|d_i\| \leq 1} f(D, r_1, \ldots, r_K) = \sum_{i=1}^K \|x_i - Dr_i\|^2 + \lambda \left(\sum_{i=1}^K \sum_{j=1}^n |r_i^{(j)}| \right),$$

where $x_i \in \mathbb{R}^m$, $r_i \in \mathbb{R}^n$, $D \in \mathbb{R}^{m\times n}$, d_i denotes the i-th column of D, $r_i^{(j)}$ denotes the j-th entry of r_i, and $\lambda > 0$. We apply h_α to smooth the absolute value function and the self-concordant function $g(d_i) = -\log(1 - \|d_i\|^2)$ (see [45, Theorem 5.1.4]) to impose $\|d_i\| < 1$. This yields the approximation in the following form of

$$f_{\alpha,\mu}(D, r_1, \ldots, r_K) = \sum_{i=1}^K \|x_i - Dr_i\|_2^2 + \lambda \left(\sum_{i=1}^K \sum_{j=1}^n h_\alpha(r_i^{(j)}) \right) + \mu \left(\sum_{i=1}^n g(d_i) \right).$$

Propositions 2.4 and 2.5 imply that $f_{\alpha,\mu}$ is F-based 1-self-concordant.

Nonnegative matrix factorization Let $\mathbb{R}_+ = \{x \in \mathbb{R} : x \geq 0\}$, $\mathbb{R}_{++} = \{x \in \mathbb{R} : x > 0\}$. We show that two NMF formulations [32] using Frobenius norm in Eq. (2) and Kullback–Leibler (KL) divergence in Eq. (3), both fit into our nonconvex self-concordant framework. Indeed, we have

$$\min_{X \in \mathbb{R}_{++}^{m\times r}, Y \in \mathbb{R}_{++}^{r\times n}} f_1(X,Y) = \tfrac{1}{2mn}\|Z - XY\|_2^2 = \tfrac{1}{2mn}\left(\sum_{i=1}^m \sum_{j=1}^n \left(Z_{ij} - \sum_{k=1}^r X_{ik}Y_{kj} \right)^2 \right), \tag{2}$$

where $Z \in \mathbb{R}_+^{m\times n}$. By Proposition 2.5, f_1 is ℓF-based 1-self-concordant for $F(X,Y) = (\|X\|_F^2 + \|Y\|_F^2 + 1)^2 - \sum_{i,k}\log(X_{ik}) - \sum_{k,j}\log(Y_{kj})$ and some $\ell > 0$. Another formulation using KL divergence is in the following form of

$$\min_{X\in\mathbb{R}_{++}^{m\times r}, Y\in\mathbb{R}_{++}^{r\times n}} f_2(X,Y) = \tfrac{1}{mn}\left(\sum_{i=1}^m\sum_{j=1}^n D\left(Z_{ij}\middle\| \sum_{k=1}^r X_{ik}Y_{kj}\right)\right), \tag{3}$$

where $D(x\|y) = x\log(x/y) - x + y$.

We prove that $f_2(X,Y)$ is F-based 1-self-concordant for some convex function F on $\mathbb{R}_{++}^{m\times r}\times\mathbb{R}_{++}^{r\times n}$. Indeed, we let $g(x,y) = -\log(x^\top y)$ and $G(x,y) = -\sum_{i=1}^r \log(x_i y_i)$. By Proposition 2.4, it suffices to prove that g is τG-based 1-self-concordant for some $\tau > 3$. Fixing $(x,y)\in\mathbb{R}_{++}^r\times\mathbb{R}_{++}^r$ and $h = (h_x, h_y)\in\mathbb{R}^r\times\mathbb{R}^r$, we define

$$s = x^\top y, \quad L = h_x^\top y + h_y^\top x, \quad u_i = \tfrac{|(h_x)_i|}{x_i}, \quad v_i = \tfrac{|(h_y)_i|}{y_i}, \quad w_i = \tfrac{x_i y_i}{s}.$$

Then, we have

$$\tfrac{|L|}{s} \le \sum_{i=1}^r w_i(|u_i| + |v_i|) \le \sqrt{2\left(\sum_{i=1}^r w_i(u_i^2+v_i^2)\right)} \le \sqrt{2\left(\sum_{i=1}^r (u_i^2+v_i^2)\right)},$$

and

$$\tfrac{|h_x^\top h_y|}{s} \le \sum_{i=1}^r \tfrac{|(h_x)_i||(h_y)_i|}{x_i y_i} \le \tfrac{1}{2}\left(\sum_{i=1}^r (u_i^2+v_i^2)\right).$$

Thus, we have

$$|\nabla^2 g(x,y)[h,h]| = \left|\tfrac{L^2}{s^2} - \tfrac{2h_x^\top h_y}{s}\right| \le 3\left(\sum_{i=1}^r (u_i^2+v_i^2)\right),$$

and

$$|\nabla^3 g(x,y)[h,h,h]| = \left|-2\tfrac{L^3}{s^3} + \tfrac{6Lh_x^\top h_y}{s^2}\right| \le 7\sqrt{2}\left(\sum_{i=1}^r (u_i^2+v_i^2)\right)^{3/2}.$$

In addition, we obtain that $\nabla^2 G(x,y)[h,h] = \sum_{i=1}^r (u_i^2+v_i^2)$ and $|\nabla^3 G(x,y)[h,h,h]| \le 2(\sum_{i=1}^r (u_i^3+v_i^3)) \le 2(\sum_{i=1}^r (u_i^2+v_i^2))^{3/2}$. Thus, we have

$$\begin{aligned}\nabla^2(g+\tau G)[h,h] &\ \ge\ (\tau-3)\left(\sum_{i=1}^r (u_i^2+v_i^2)\right), \\ |\nabla^3(g+\tau G)[h,h,h]| &\ \le\ (2\tau+7\sqrt{2})\left(\sum_{i=1}^r (u_i^2+v_i^2)\right)^{3/2}.\end{aligned}$$

This implies that $|\nabla^3(g+\tau G)[h,h,h]| \le 2(\nabla^2(g+\tau G)[h,h])^{3/2}$ for some $\tau > 3$. This yields the desired result.

2.2 Descent Inequality

We establish the descent inequality for F-based κ-self-concordant functions where F is a convex function. For simplicity, we define two auxiliary functions $\omega(z)$ and $\omega_\star(z)$ on $\mathbb{R}_+$ as follows,

$$\omega(z) = z - \log(1+z) \text{ and } \omega_\star(z) = \begin{cases} -z - \log(1-z) & \text{if } z < 1, \\ +\infty & \text{otherwise.}\end{cases}$$

In addition, suppose that f is lower bounded, we define

$$\Gamma_f(x) = \sup\left\{t \geq 0 : t - \log(1+t) \leq \kappa^2 \left(f(x) - \inf_{z \in \mathbb{R}^n} f(z)\right)\right\}. \tag{4}$$

Proposition 2.6. *Suppose that f is F-based κ-self-concordant. Then, for any $x \in \textbf{dom}(f)$ and $d \in \mathbb{R}^n$ satisfying $\nabla f(x)^\top d \leq 0$, we define*

$$\rho = -\nabla f(x)^\top d, \quad \delta = \nabla^2 f(x)[d,d], \quad \Delta = \nabla^2 F(x)[d,d], \quad \eta = \tfrac{\rho}{\sqrt{\Delta+\delta}}.$$

For any $t \in [0, \frac{1}{\kappa\sqrt{\delta+\Delta}})$, we have $x + td \in \textbf{dom}(f)$ and

$$f(x+td) - f(x) + \rho t \leq F(x) + t\nabla F(x)^\top d - F(x+td) + \kappa^{-2}\omega_\star(\kappa t\sqrt{\delta+\Delta}).$$

If F is convex, we have

$$f(x+td) \leq f(x) - \rho t + \kappa^{-2}\omega_\star(\kappa t\sqrt{\delta+\Delta}). \tag{5}$$

Suppose that $\bar{t} = \frac{\rho}{\delta+\Delta+\kappa\rho\sqrt{\delta+\Delta}} < \frac{1}{\kappa\sqrt{\delta+\Delta}}$ minimizes the right-hand side of Eq. (5). Then, we have $f(x+td) \leq f(x)$ for all $t \in [0,\bar{t}]$ and

$$f(x+\bar{t}d) \leq f(x) - \rho\bar{t} + \kappa^{-2}\omega_\star(\kappa\bar{t}\sqrt{\delta+\Delta}) = f(x) - \kappa^{-2}\omega(\kappa\eta). \tag{6}$$

If f is lower bounded, we have $\kappa^{-2}\omega(\kappa\eta) \geq \frac{\eta^2}{2(1+\Gamma_f(x))}$. If F is convex and κ_F-self-concordant, we have

$$f(x+td) \leq f(x) - \rho t + \kappa^{-2}\omega_\star(\kappa t\sqrt{\delta+\Delta}) - \kappa_F^{-2}\omega(\kappa_F t\sqrt{\Delta}). \tag{7}$$

Proof. It follows from [45, Theorems 5.1.5 and 5.1.9] that $x + td \in \textbf{dom}(f)$ and

$$f(x+td) - f(x) + \rho t \leq F(x) + t\nabla F(x)^\top d - F(x+td) + \kappa^{-2}\omega_\star(\kappa t\sqrt{\delta+\Delta}).$$

Since F is convex, we have $F(x) + t\nabla F(x)^\top d - F(x+td) \leq 0$. Plugging this into the above inequality yields Eq. (5). We define

$$g(t) = f(x) - \rho t - \tfrac{1}{\kappa^2}\left(\kappa t\sqrt{\delta+\Delta} + \log(1 - \kappa t\sqrt{\delta+\Delta})\right).$$

Taking the derivative of $g(t)$ yields

$$g'(t) = -\rho + \left(\tfrac{t(\delta+\Delta)}{1-\kappa t\sqrt{\delta+\Delta})}\right).$$

Thus, $g'(t) \leq 0$ for all $t \in [0,\bar{t}]$. Since $g(0) = f(x)$, we have $f(x+td) \leq g(t) \leq f(x)$ for all $t \in [0,\bar{t}]$. Plugging $\bar{t}$ into (5) yields Eq. (6). In addition, Eq. (6) implies

$$\kappa\eta - \log(1+\kappa\eta) = \omega(\kappa\eta) \leq \kappa^2(f(x) - f(x+\bar{t}d)) \leq \kappa^2\left(f(x) - \inf_{z\in\mathbb{R}^n} f(z)\right).$$

Thus, $\kappa\eta \leq \Gamma_f(x)$. Since $w(z) \geq \frac{z^2}{2(1+\Gamma_f(x))}$ for $0 \leq z \leq \Gamma_f(x)$, we have $\kappa^{-2}\omega(\kappa\eta) \geq \frac{\eta^2}{2(1+\Gamma_f(x))}$. It follows from [45, Theorem 5.1.8] that

$$F(x) + t\nabla F(x)^\top d - F(x+td) \leq -\kappa_F^{-2}\omega(\kappa_F t\sqrt{\Delta}).$$

which implies Eq. (7). $\square$

3 Algorithms

We develop two algorithms for solving the problem in Eq. (1), where the objective function f is F-based κ-self-concordant and bounded below. Compared to existing methods [47, 10, 11, 26], our two algorithms rely on neither the Lipschitz Hessian condition nor the unconstrained domain (i.e., $\mathbf{dom}(f) = \mathbb{R}^n$). Throughout this paper, we make Assumption 3.1 and impose additional assumptions (e.g., F is self-concordant or quadratic) if necessary.

Assumption 3.1. *The function $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is bounded below and is F-based κ-self-concordant with respect to a convex function $F : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ for some $\kappa \geq 0$. In addition, $\nabla^2(f + F)(x)$ is positive definite for any $x \in \mathbf{dom}(f)$.*

3.1 Regularized Newton's Method (RNM)

The scheme is as follows,

$$x_{j+1} = x_j - \tfrac{1}{1+\kappa\nu_{f,F}(x_j)}(\nabla^2(f + F)(x_j))^{-1}\nabla f(x_j), \quad \text{for all } j = 0, 1, 2, \ldots \tag{8}$$

where $\nu_{f,F}(x) = \sqrt{\nabla f(x)^\top(\nabla^2(f + F)(x))^{-1}\nabla f(x)}$[1]. At each iteration, we compute a direction $d \in \mathbb{R}^n$ that minimizes the right-hand side of Eq. (6). This is equivalent to maximizing $\eta = \frac{-\nabla f(x_j)^\top d}{\sqrt{d^\top(\nabla^2(f+F)(x_j))d}}$ and yields Eq. (8). In addition, we have $\rho = \delta + \Delta = (\nu_{f,F}(x_j))^2$ and $\bar{t}_j = \frac{1}{1+\kappa\nu_{f,F}(x_j)}$ with $f(x_j + \bar{t}_j d_j) \leq f(x_j) - \kappa^{-2}\omega(\kappa\nu_{f,F}(x_j))$.

Under Assumption 3.1, we derive the convergence guarantees in terms of $\nu_{f,F}(x)$ rather than gradient norm, which is insufficient to measure the stationarity due to the lack of Lipschitz gradients. Our results can be interpreted as a generalization of [45, Section 5.2] and $\nu_{f,F}(x) = \lambda_f(x)$ when $F = 0$ (see [45, Page 354]). For functions that are weakly self-concordant, this measure relates to gradient norms of Moreau envelopes (see Appendix A.1). We summarize our results in the following theorems and defer the proofs to Section 4.

Theorem 3.2. *Suppose that Assumption 3.1 holds. Then, the sequence of iterates $\{x_j\}_{j\geq 0}$ generated by Algorithm 8 satisfy*

$$\min_{0\leq j\leq k} \nu_{f,F}(x_j) \leq \sqrt{\tfrac{2(1+\Gamma_f(x_0))(f(x_0)-\inf_{x\in\mathbb{R}^n} f(x))}{k+1}}.$$

where $\Gamma_f(x)$ is defined in Eq. (4).

If the Polyak-Łojasiewicz condition [34, 50] holds at a limit point of $\{x_j\}_{j\geq 0}$, we show that the function value locally converges linearly in the following theorem.

Theorem 3.3. *Suppose that Assumption 3.1 holds, $f + F$ is α-strongly convex and the Polyak-Łojasiewicz condition holds true at an interior optimal solution $x^\star \in \operatorname{int}(\mathbf{dom}(f))$ (i.e., there exists $\mu > 0$ and a neighborhood U of $x^\star$ in $\mathbf{dom}(f)$ such that $\frac{1}{2}\|\nabla f(x)\|^2 \geq \mu(f(x) - f(x^\star))$ for all $x \in U$). If $x^\star$ is a limit point of $\{x_j\}_{j\geq 0}$, we have $x_j \to x^\star$ and $f(x_j) - f(x^\star)$ converges Q-linearly to 0.*

Remark 3.4. *The RNM scheme performs well for training convolutional neural networks (CNNs) but is less effective for nonnegative matrix factorization (NMF). An explanation lies in different choices and magnitudes of F. In our CNN experiments, F is quadratic with the coefficients on the order of 10^{-2}, which is consistent with the asymmetric Hessian spectra reported in neural network training [66], where*

[1] $\sqrt{a} = \infty$ when $a < 0$.

Algorithm 1 Adaptive Regularization Method

```
1: Input: $0 < \sigma_{\min} \leq \sigma_0$, $0 < \eta_1 \leq \eta_2 < 1$ and $0 < \gamma_1 < 1 < \gamma_2 < \gamma_3$.
2: Initialization: $x_0 \in \mathbf{dom}(f)$.
3: for $j = 0, 1, 2, \ldots$ do
4:     Compute $d_j \in \mathbb{R}^n$ and construct the model $m_j : \mathbb{R}_+ \to \mathbb{R} \cup \{+\infty\}$ according to one of two options: (i)
       $d_j$ satisfies Eq. (9) and $m_j$ is defined in Eq. (10); and (ii) $d_j$ is defined in Eq. (17) and $m_j$ is defined in
       Eq. (18).
5:     if $\nabla f(x_j)^\top d_j > 0$ then
6:         $t_j \leftarrow 0$.
7:     else
8:         $t_j \leftarrow \operatorname{argmin}_{t \geq 0} m_j(t)$.
9:     end if
10:    Compute $r_j \leftarrow \frac{f(x_j) - f(x_j + t_j d_j)}{f(x_j) - m_j(t_j)}$.
       $$x_{j+1} \leftarrow \begin{cases} x_j + t_j d_j, & \text{if } r_j \geq \eta_1, \\ x_j, & \text{otherwise.} \end{cases} \quad \sigma_{j+1} \in \begin{cases} [\max(\sigma_{\min}, \gamma_1 \sigma_j), \sigma_j], & \text{if } r_j \geq \eta_2, \\ [\sigma_j, \gamma_2 \sigma_j], & \text{if } r_j \in (\eta_1, \eta_2), \\ [\gamma_2 \sigma_j, \gamma_3 \sigma_j], & \text{if } r_j \leq \eta_1. \end{cases}$$
11:    if the termination condition depending on $(x_j, d_j, \sigma_j)$ is satisfied then
12:        return $x_j$.
13:    end if
14: end for
```

negative curvature is mild. In these settings, a small regularizer is sufficient to stabilize the updates and capture local curvature. In contrast, the objective function f and its Hessian entries are much larger in scale for NMF. Near a local minimizer, applying the same globally chosen F introduces excessive regularization, which distorts the update direction and ultimately degrades performance.

3.2 Adaptive Regularization Method (ARM)

The scheme shares a similar spirit with the methods from [10, 11, 12] and has been summarized in Algorithm 1. At each iteration, we compute a direction $d_j \in \mathbb{R}^n$ and a step-size t_j by minimizing the model $m_j(t)$. Then, we compute the ratio r_j to determine if a sufficient decrease is made. If $r_j \geq \eta_1$, we accept the update; otherwise, the iterate remains the same. Finally, we adjust the parameter σ_j based on r_j as described in Algorithm 1.

The key ingredients of Algorithm 1 consist of a specific rule for selecting the direction d_j. The first option can yield approximate first-order stationary points (see Theorem 3.5), while the second option, equipped with an oracle that detects negative curvature, can yield approximate second-order stationary points (see Theorem 3.8). In what follows, we provide all of details and convergence results.

First option: general descent directions To begin, we derive a convergence result for general update directions d_j. The only requirement is that $d_j = d(x_0, \ldots, x_j, \sigma_j)$ is a descent direction when the parameter σ_j is sufficiently large:

$$\exists \bar{\sigma} > 0, \text{ s.t. } \nabla f(x)^\top d_j \leq 0 \text{ if } \sigma_j \geq \bar{\sigma}. \tag{9}$$

Under the same notation of Proposition 2.6, we view f as $\sigma_j F$-based κ-self-concordant and define the local model $m_j(t)$ as follows,

$$m_j(t) = \begin{cases} f(x_j) - \rho_j t + \kappa^{-2}\,\omega_\star(\kappa t\sqrt{\delta_j + \sigma_j \Delta_j}), & \text{if } \delta_j + \sigma_j\Delta_j \geq 0, \\ 0, & \text{if } \delta_j + \sigma_j\Delta_j < 0 \text{ and } t = 0, \\ +\infty, & \text{otherwise.} \end{cases} \tag{10}$$

Thus, we only compute t_j when $\delta_j + \sigma_j\Delta_j \geq 0$ and $\rho_j \geq 0$. In this case, minimizing Eq. (10) with respect to t yields the solution in the form of

$$t_j = \frac{\rho_j}{\delta_j + \sigma_j\Delta_j + \kappa\rho_j\sqrt{\delta_j + \sigma_j\Delta_j}}. \tag{11}$$

Theorem 3.5. *Suppose that Assumption 3.1 holds and Eq.* (9) *is satisfied. Let m_j be defined in Eq.* (10), *and the termination condition be*

$$\frac{\nabla f(x_j)^\top d_j}{\sqrt{\nabla^2(f + \sigma_j F)(x_j)[d_j, d_j]}} \leq \epsilon. \tag{12}$$

Then, we have $\sigma_j \leq \max(\gamma_3\bar{\sigma}, \sigma_0, \gamma_3)$ for all j, and Algorithm 1 terminates within at most $O(\epsilon^{-2})$ iterations.

Note that it is possible to terminate at a point x_j where both $\nabla f(x_j)$ and $\nabla^2(f + \sigma_j F)$ have large magnitudes. To mitigate this issue, we introduce two specific choices for d_j and refine the termination conditions accordingly.

The first choice is *preconditioned gradient descent.* Indeed, we define

$$d_j = -H_j \nabla f(x), \quad H_j \succ 0, \tag{13}$$

where H_j is the preconditioner that depends on $x_0, \ldots, x_j$. This framework covers both gradient descent ($H_j = I$) and quasi-Newton methods. Since $H_j \succ 0$, we have $\nabla f(x_j)^\top d_j \leq 0$, ensuring that Eq. (9) is satisfied.

Corollary 3.6. *Suppose that Assumption 3.1 holds, and there exists $\Lambda > 0$ such that $\|H_j^{1/2}\nabla^2(f + \sigma_j F)H_j^{1/2}\|_{\mathrm{op}} \leq \Lambda$ for all j. Let d_j and m_j be defined in Eq.* (13) *and Eq.* (10) *and the termination condition be $\sqrt{\nabla f(x_j)^\top H_j \nabla f(x_j)} \leq \epsilon$. Then, we have $\sigma_j \leq \max(\sigma_0, \gamma_3)$, and Algorithm 1 terminates within at most $O(\epsilon^{-2})$ iterations.*

The second choice is *adaptive regularized Newton's method.* Indeed, we define

$$d_j = -(\nabla^2(f + \sigma_j F)(x_j))^{-1}\nabla f(x_j), \tag{14}$$

which yields

$$m_j(t) = f(x_j) - t(\nu_{f,\sigma_j F}(x_j))^2 + \kappa^{-2}\omega_\star(\kappa t \nu_{f,\sigma_j F}(x_j)). \tag{15}$$

The termination condition in Eq. (12) becomes $\nu_{f,\sigma_j F}(x_j) \leq \epsilon$. If $\delta_j + \sigma_j\Delta_j \geq 0$ and $\rho_j \geq 0$, the optimal step size is

$$t_j = \frac{1}{1 + \kappa\nu_{f,\sigma_j F}(x_j)}. \tag{16}$$

Corollary 3.7. *Suppose that Assumption 3.1 holds. Let d_j and m_j be defined in Eq.* (14) *and Eq.* (15), *and the termination condition be $\nu_{f,\sigma_j F}(x_j) \leq \epsilon$. Then, we have Algorithm 1 terminates within at most $O(\epsilon^{-2})$ iterations.*

Second option: negative curvature descent Suppose that F is κ_F-self-concordant. Then, we exploit negative curvature to escape from saddle points and further decrease the objective. For each iterate x_j, we have access to the smallest eigenvalue of Hessian $\lambda_{\min}(\nabla^2 f(x_j))$ and a corresponding unit eigenvector v_j. If the Hessian exhibits strong negative curvature, Algorithm 1 switches from a regularized Newton step to a curvature-exploiting step as follows,

$$d_j = \begin{cases} -(\nabla^2(f+\sigma_j F)(x_j))^{-1}\nabla f(x_j), & \text{if } \lambda_{\min}(\nabla^2 f(x_j)) \geq -\lambda_{\text{nc}}, \\ v_j, & \text{if } \nabla f(x_j)^\top v_j \leq 0,\ \lambda_{\min}(\nabla^2 f(x_j)) < -\lambda_{\text{nc}}, \\ -v_j, & \text{otherwise}, \end{cases} \tag{17}$$

where $\lambda_{\text{nc}} = \sigma_j\sqrt{\epsilon_H}\nabla^2 F(x_j)[v_j, v_j]$. This rule ensures that, if the significant negative curvature is detected, a descent direction aligned with it will be used to escape saddle regions. Under the same notation of Proposition 2.6, we view f as $\sigma_j F$-based κ-self-concordant and define the local model $m_j(t)$ as follows,

$$m_j(t) = \begin{cases} f(x_j) - t(\nu_{f,\sigma_j F}(x_j))^2 + \kappa^{-2}\omega_\star(\kappa t\nu_{f,\sigma_j F}(x_j)), & \text{if } \lambda_{\min}(\nabla^2 f(x_j)) \geq -\lambda_{\text{nc}}, \\ f(x_j) - \kappa_F^{-2}\omega(\kappa_F t\sqrt{\sigma_j\Delta_j}) + \kappa^{-2}\omega_\star(\kappa t\sqrt{\delta_j + \sigma_j\Delta_j}), & \text{otherwise}. \end{cases} \tag{18}$$

If $\lambda_{\min}(\nabla^2 f(x_j)) \geq -\lambda_{\text{nc}}$, we have that $m_j(t)$ reduces to the one used for adaptive regularized Newton's methods. Thus, we can use Eq. (16). If $\lambda_{\min}(\nabla^2 f(x_j)) < -\lambda_{\text{nc}}$ and $\delta_j + \sigma_j\Delta_j \geq 0$, Algorithm 1 performs a negative curvature step where $\delta_j = \lambda_{\min}(\nabla^2 f(x_j)) < 0$. Minimizing $m_j(t)$ with respect to t yields

$$t_j = -\frac{\delta_j}{\sqrt{\sigma_j\Delta_j}\sqrt{\delta_j+\sigma_j\Delta_j}(\kappa_F\sqrt{\delta_j+\sigma_j\Delta_j}+\kappa\sqrt{\sigma_j\Delta_j})}. \tag{19}$$

Theorem 3.8. *Suppose that Assumption 3.1 holds with a κ_F-self-concordant F. Let d_j and m_j be defined in Eq.* (17) *and Eq.* (18)*, and the termination condition be $\nu_{f,\sigma_j F}(x_j) \leq \epsilon_g$ and $\lambda_{\min}(\nabla^2 f(x_j)) \geq -\sigma_j\sqrt{\epsilon_H}\nabla^2 F(x_j)[v_j, v_j]$. Then, we have $\sigma_j \leq \max(\sigma_0, \gamma_3)$ for all j, and Algorithm 1 terminates within at most $O(\epsilon_g^{-2} + \epsilon_H^{-1.5})$ iterations.*

Remark 3.9. *The ARM improves the RNM in two dimensions. First, it introduces a parameter σ_j that controls the magnitude of regularization, mitigating the issue of excessively large regularization, as discussed in Remark 3.4. Second, the ARM is effective when f is ℓF-based κ-self-concordant, but we have no prior knowledge of ℓ.*

Remark 3.10. *The key advantage of the ARM over the trust region (TR) method and [12, Algorithm 2.4.1] is to compute a better direction d_j and construct a better model $m_j(\cdot)$ by leveraging the special self-concordance structure. Indeed, the TR method computes d_j based on a local model $m_j(d) = f(x_j) + \nabla f(x_j)^\top d + \frac{1}{2}d^\top B_j d$ where B_j is symmetric such that $f(x_j + d_j) < f(x_j)$. The corresponding ratio of reduction is $r_j = \frac{f(x_j)-f(x_j+d_j)}{m_j(0)-m_j(d_j)}$. The TR method either accepts or rejects but always scales the radius based on r_j. In [12, Algorithm 2.4.1], a local model $m_j(d) = f(x_j) + \nabla f(x_j)^\top d + \frac{1}{2}\sigma_j\|d\|^2$ is constructed and $d_j = \text{argmin}_d\, m_j(d)$. All of other steps are similar to that used in the ARM. Another advantage of the ARM is its ability to handle the constrained case where $\textbf{dom}(f) \neq \mathbb{R}^n$. It rejects when $x_j + d_j \notin \textbf{dom}(f)$, while maintaining convergence guarantees.*

4 Convergence Analysis

We provide the convergence analysis for the RNM and ARM algorithms proposed in Section 3 by proving all theorems.

4.1 Technical Lemmas

We define a set of successful iterations up to k by $\mathcal{S}_k = \{j : 0 \leq j \leq k, r_j \geq \eta_1\}$[2], unsuccessful iterations up to k by $\mathcal{U}_k = \{0, \ldots, k\} \backslash \mathcal{S}_k$, and all successful iterations by $\mathcal{S} = \bigcup_{k=0}^{\infty} \mathcal{S}_k$.

Lemma 4.1. *Suppose that f is bounded below and continuously differentiable. If there exists $\sigma_{\max} > 0$ such that $\sigma_j \leq \sigma_{\max}$ for all $j \geq 0$ and $\tilde{\epsilon} > 0$ such that $f(x_j) - m_j(t_j) \geq \tilde{\epsilon}$ for all $j \in \mathcal{S}$, Algorithm 1 terminates within $O(\tilde{\epsilon}^{-1})$ iterations.*

Proof. If Algorithm 1 does not terminate up to k, we have $f(x_0) - \inf_{z \in \mathbb{R}^n} f(z) \geq \sum_{j \in \mathcal{S}_k} f(x_j) - f(x_{j+1}) \geq \sum_{j \in \mathcal{S}_k} \eta_1 \tilde{\epsilon} = \eta_1 |\mathcal{S}_k| \tilde{\epsilon}$. This implies that $|\mathcal{S}_k| = O(\tilde{\epsilon}^{-1})$. Then, we see from Algorithm 1 that $\gamma_1 \sigma_j \leq \max(\gamma_1 \sigma_j, \sigma_{\min}) \leq \sigma_{j+1}$ for $j \in \mathcal{S}_k$ and $\gamma_2 \sigma_j \leq \sigma_{j+1}$ for $j \in \mathcal{U}_k$. Thus, $\sigma_0 \gamma_1^{|\mathcal{S}_k|} \gamma_2^{|\mathcal{U}_k|} \leq \sigma_k$. Since $\sigma_k \leq \sigma_{\max}$, we have $|\mathcal{S}_k| \log(\gamma_1) + |\mathcal{U}_k| \log(\gamma_2) \leq \log(\sigma_{\max} \sigma_0^{-1})$. This together with $k = |\mathcal{S}_k| + |\mathcal{U}_k|$ yields $k \leq |\mathcal{S}_k| \left(1 - \frac{\log(\gamma_1)}{\log(\gamma_2)}\right) + \frac{1}{\log(\gamma_2)} \log(\frac{\sigma_{\max}}{\sigma_0})$. Since $|S_k| = O(\tilde{\epsilon}^{-1})$, we have $k = O(\tilde{\epsilon}^{-1})$. □

Lemma 4.2. *Suppose that Assumption 3.1 holds and Eq.* (9) *is satisfied. Let m_j be defined in Eq.* (10), *and the termination condition be $\frac{\nabla f(x_j)^\top d_j}{\sqrt{\nabla^2 (f + \sigma_j F)(x_j)[d_j, d_j]}} \leq \epsilon$. Then, we have $\sigma_j \leq \max(\gamma_3 \bar{\sigma}, \sigma_0, \gamma_3)$ for all j and $f(x_j) - m_j(t_j) \geq \frac{\epsilon^2}{2(1+\Gamma_f(x_0))}$ for all $j \in \mathcal{S}$.*

Proof. We prove by contradiction. Suppose that there exists j such that $\sigma_{j+1} > \max(\gamma_3 \bar{\sigma}, \sigma_0, \gamma_3) \geq \sigma_j$. Then, we have $\sigma_j \geq \frac{\sigma_{j+1}}{\gamma_3} \geq \max(\bar{\sigma}, 1)$. We let $\rho_j = -\nabla f(x_j)^\top d_j$, $\delta_j = \nabla^2 f(x_j)[d_j, d_j]$, and $\Delta_j = \nabla^2 F(x_j)[d_j, d_j]$. We obtain from Assumption 3.1 that $\rho_j \geq 0$ and $\delta_j + \sigma_j \Delta_j \geq \delta_j + \Delta_j \geq 0$. We also have

$$f(x_j) - m_j(t_j) = \rho_j t_j - \kappa^{-2} \omega_\star(\kappa t_j \sqrt{\delta_j + \sigma_j \Delta_j}).$$

By Proposition 2.6, we have $x_j + t_j d_j \in \mathbf{dom}(f)$ and

$$f(x_j) - f(x_j + t_j d_j) \geq \rho_j t_j - \kappa^{-2} \omega_\star(\kappa t_j \sqrt{\delta_j + \Delta_j}).$$

Since $\omega_\star(\cdot)$ is increasing, we have $r_j \geq 1 > \eta_2$ and thus $\sigma_{j+1} \leq \sigma_j \leq \max(\gamma_3 \bar{\sigma}, \sigma_0, \gamma_3)$. This yields a contradiction. Thus, $\sigma_j \leq \max(\gamma_3 \bar{\sigma}, \sigma_0, \gamma_3)$ for all j.

For all $j \in \mathcal{S}$, we provide a lower bound of $f(x_j) - m_j(t_j)$. Indeed, by combining Eq. (11) with Eq. (10), we have

$$f(x_j) - m_j(t_j) \geq \frac{(\nabla f(x_j)^\top d_j)^2}{2(1+\Gamma_f(x_0)) \nabla^2 (f + \sigma_j F)[d_j, d_j]} > \frac{\epsilon^2}{2(1+\Gamma_f(x_0))}.$$

This completes the proof. □

Lemma 4.3. *Suppose that Assumption 3.1 holds with a κ_F-self-concordant F. Let d_j and m_j be defined in Eq.* (17) *and Eq.* (18), *and the termination condition be $\nu_{f,\sigma_j F}(x_j) \leq \epsilon_g$ and $\lambda_{\min}(\nabla^2 f(x_j)) \geq -\sigma_j \sqrt{\epsilon_H} \nabla^2 F(x_j)[v_j, v_j]$. Then, we have $\sigma_j \leq \max(\gamma_3, \sigma_0)$ for all j and $f(x_j) - m_j(t_j) \geq \min\left(\frac{\epsilon_H^{1.5}}{6(\kappa_f + \kappa_F)^2}, \frac{\epsilon_g^2}{2(1+\Gamma_f(x_0))}\right)$ for all $j \in \mathcal{S}$.*

Proof. We prove by contradiction. Suppose that there exists j such that $\sigma_{j+1} > \max(\sigma_0, \gamma_3) \geq \sigma_j$. Then, we have $\sigma_j \geq \frac{\sigma_{j+1}}{\gamma_3} \geq 1$. Using the same idea proving Lemma 4.2, we have $\rho_j \geq 0$ and $\delta_j + \sigma_j \Delta_j \geq \delta_j + \Delta_j \geq 0$. We divide the remaining proof into two cases: (i) $\lambda_{\min}(\nabla^2 f(x_j)) \geq -\lambda_{\mathrm{nc}}$; and (ii) $\lambda_{\min}(\nabla^2 f(x_j)) < -\lambda_{\mathrm{nc}}$.

For the first case, we have

$$f(x_j) - m_j(t_j) = \rho_j t_j - \kappa^{-2} \omega_\star(\kappa t_j \sqrt{\delta_j + \sigma_j \Delta_j}).$$

[2] We define $r_i = 0$ if Algorithm 1 terminates when $j \leq i$.

By Proposition 2.6, we have $x_j + t_j d_j \in \mathbf{dom}(f)$ and

$$f(x_j) - f(x_j + t_j d_j) \geq \rho_j t_j - \kappa^{-2}\omega_\star(\kappa t_j \sqrt{\delta_j + \Delta_j}).$$

Since $\omega_\star(\cdot)$ is increasing, we have $r_j \geq 1 > \eta_2$ and then $\sigma_{j+1} \leq \sigma_j \leq \max(\sigma_0, \gamma_3)$. This yields the contradiction.

For the second case, we have $\delta_j = \lambda_{\min}(\nabla^2 f(x_j)) < 0$ and

$$f(x_j) - m_j(t_j) = -\kappa^{-2}\omega_\star(\kappa t_j \sqrt{\delta_j + \sigma_j \Delta_j}) + \kappa_F^{-2}\omega(\kappa_F t_j \sqrt{\sigma_j \Delta_j}).$$

By Proposition 2.6, we have $x_j + t_j d_j \in \mathbf{dom}(f)$ and

$$f(x_j) - f(x_j + t_j d_j) \geq -\kappa^{-2}\omega_\star(\kappa t_j \sqrt{\delta_j + \Delta_j}) + \kappa_F^{-2}\omega(\kappa_F t_j \sqrt{\Delta_j}).$$

Defining $g(\sigma) = -\kappa^{-2}\omega_\star(\kappa t_j \sqrt{\delta_j + \sigma\Delta_j}) + \kappa_F^{-2}\omega(\kappa_F t_j \sqrt{\sigma\Delta_j})$, we have

$$g'(\sigma) = \tfrac{t_j^2 \Delta_j}{2}\left(\tfrac{-1}{1-\kappa t_j\sqrt{\delta_j+\sigma\Delta_j}} + \tfrac{1}{1+\kappa_F t_j \sqrt{\sigma\Delta_j}}\right) \leq 0.$$

Thus, $f(x_j) - m_j(t_j) = g(\sigma_j) \leq g(1) \leq f(x_j) - f(x_j + t_j d_j)$, which implies $r_j \geq 1 > \eta_2$. This yields a contradiction.

Since Algorithm 1 does not terminate at x_j for $j \in \mathcal{S}$, we have $\nu_{f,\sigma_j F}(x_j) > \epsilon_g$ or $\lambda_{\min}(\nabla^2 f(x_j)) < -\lambda_{\text{nc}}$. Indeed, if $\lambda_{\min}(\nabla^2 f(x_j)) \geq -\lambda_{\text{nc}}$ and $\nu_{f,\sigma_j F}(x_j) > \epsilon_g$, we obtain from $x - \log(1+x) \geq \frac{x^2}{2(1+\Gamma_f(x_0))}$ for $0 \leq x \leq \Gamma_f(x_0)$ that

$$f(x_j) - m_j(t_j) \geq \tfrac{\epsilon_g^2}{2(1+\Gamma_f(x_0))}.$$

Otherwise, we consider the case of $\lambda_{\min}(\nabla^2 f(x_j)) < -\lambda_{\text{nc}}$. For simplicity, we define $\gamma = \frac{\kappa_F}{\kappa}$ and $z = -\frac{\delta_j}{\sigma_j \Delta_j}$. Then, Eq. (18) implies

$$f(x_j) - m_j(t_j) = \kappa^{-2}\left(\gamma^{-2}\omega\left(\tfrac{\gamma z}{\gamma(1-z)+\sqrt{1-z}}\right) - \omega_\star\left(\tfrac{z}{\gamma\sqrt{1-z}+1}\right)\right) \geq \kappa^{-2} g_\gamma(z),$$

where

$$g_\gamma(z) = \tfrac{z}{\gamma\sqrt{1-z}} + \log\left(1 - \tfrac{z}{\gamma\sqrt{(1-z)}+1}\right) - \tfrac{1}{\gamma^2}\log\left(1 + \tfrac{\gamma z}{\gamma(1-z)+\sqrt{1-z}}\right).$$

We also have $g_\gamma(0) = 0$ and $g'_\gamma(z) \geq \frac{z^2}{2(\gamma+1)^2}$. This implies $g_\gamma(z) \geq \frac{z^3}{6(\gamma+1)^2}$ and

$$f(x_j) - m_j(t_j) \geq \tfrac{z^3}{6(\kappa+\kappa_F)^2} \geq \tfrac{\epsilon_H^{1.5}}{6(\kappa_f+\kappa_F)^2}.$$

This completes the proof. □

4.2 Main Results

We are ready to provide the proofs for all theorems.

Proof of Theorem 3.2. Proposition 2.6 guarantees that $x_j \in \mathbf{dom}(f)$, $f(x_j) \leq f(x_0)$ for all $j \geq 0$ and $f(x_j) - f(x_{j+1}) \geq \frac{(\nu_{f,F}(x_j))^2}{2(1+\Gamma_f(x_0))}$. Summing this inequality over $j = 0, 1, 2, \ldots$ yields

$$f(x_0) - \inf_{x \in \mathbb{R}^n} f(x) \geq \sum_{i=0}^{k} f(x_j) - f(x_{j+1}) \geq \sum_{i=0}^{k} \tfrac{\nu_{f,F}(x_j)^2}{2(1+\Gamma_f(x_0))} \geq \tfrac{(k+1)(\min_{0\leq j\leq k}\nu_{f,F}(x_j))^2}{2(1+\Gamma_f(x_0))}.$$

This completes the proof. □

Proof of Theorem 3.3. We denote by U the neighborhood of $x^\star$ where the Polyak-Łojasiewicz condition holds. There exists $L > 0$ such that $\|\nabla^2(f+F)(x)\|_{\mathrm{op}} \le L$ for all $x \in U$. By [2, Proposition 3.3 and Theorem 3.4], we have $x_j \to x_\star$ if there exist $c_1, c_2 > 0$ such that $f(x_j) - f(x_{j+1}) \ge c_1 \|\nabla f(x_j)\|^2$ and $\|\nabla f(x_j)\| \ge c_2 \|x_j - x_{j+1}\|$ for all $x_j, x_{j+1} \in U$. Indeed, by Proposition 2.6 and using the fact that $f(x_j) \le f(x_0)$ and $\|\nabla^2(f+F)(x_j)\|_{\mathrm{op}} \le L$ for all j, we have

$$f(x_j) - f(x_{j+1}) \ge \tfrac{1}{2(1+\Gamma_f(x_0))} \left(\nu_{f,F}(x_j)\right)^2 \ge \tfrac{1}{2(1+\Gamma_f(x_0))L} \|\nabla f(x_j)\|^2, \text{ for any } x_j \in U. \tag{20}$$

In addition, we have $\|\nabla f(x_j)\| \ge \alpha \|x_{j+1} - x_j\|$. Putting these pieces together yields the desired result.

It remains to prove the linear convergence of function values. As $x_j \to x^\star$, there exists $K \in \mathbb{N}$ such that $x_j \in U$ for all $j \ge K$. Applying Eq. (20) and the Polyak-Łojasiewicz inequality yields $f(x_j) - f(x_{j+1}) \ge \frac{\mu}{(1+\Gamma_f(x_0))L}(f(x_j) - f(x^\star))$ for all $j \ge K$. This implies

$$f(x_{j+1}) - f(x^\star) \le \left(1 - \tfrac{\mu}{(1+\Gamma_f(x_0))L}\right)(f(x_j) - f(x^\star)).$$

which implies that $f(x_j) - f(x^\star)$ converges Q-linearly to 0. □

Proof of Theorem 3.5. Suppose that Assumption 3.1 holds, Lemma 4.2 implies that Lemma 4.1 holds with $\tilde\epsilon = \frac{\epsilon^2}{2(1+\Gamma_f(x_0))}$ and $\sigma_{\max} = \max(\gamma_3\bar\sigma, \sigma_0, \gamma_3)$. Thus, Algorithm 1 terminates within $O(\tilde\epsilon^{-1}) = O(\epsilon^{-2})$ iterations. □

Proof of Corollary 3.6. It follows from Theorem 3.5 that within $O(\epsilon^{-2})$ iterations, we can output x_j satisfying

$$\frac{\sqrt{\nabla f(x_j)^\top H_j \nabla f(x_j)}}{\Lambda} \le \frac{\nabla f(x_j)^\top d_j}{\sqrt{\nabla^2(f+\sigma_j F)[d_j, d_j]}} \le \frac{\epsilon}{\Lambda}.$$

This completes the proof. □

Proof of Corollary 3.7. The direction d_j and the model m_j given by Eq. (14) and Eq. (15), and the termination condition given by Eq. (12) satisfy the conditions of Theorem 3.5 with $\bar\sigma = 1$. This completes the proof. □

Proof of Theorem 3.8. Suppose that Assumption 3.1 holds, Lemma 4.3 implies that the conditions of Lemma 4.1 hold with $\tilde\epsilon = \min\left(\frac{\epsilon_H^{1.5}}{6(\kappa_f+\kappa_F)^2}, \frac{\epsilon_g^2}{2(1+\Gamma_f(x_0))}\right)$ and $\sigma_{\max} = \max(\sigma_0, \gamma_3)$. Thus, Algorithm 1 terminates within $O(\tilde\epsilon^{-1}) = O(\epsilon_g^{-2} + \epsilon_H^{-1.5})$ iterations. This completes the proof. □

5 Experiments

We conduct experiments on nonnegative matrix factorization and training convolutional neural networks to evaluate our proposed methods.

Nonnegative matrix factorization (NMF) We denote $\mathcal{U}[a,b]$ as the uniform distribution on $[a,b]$. For the KL divergence, we generate the data matrix $Z = \hat X \hat Y + 0.01\hat Z$, where $\hat X \in \mathbb{R}^{m\times r}$, $\hat Y \in \mathbb{R}^{r\times n}$, $\hat Z \in \mathbb{R}^{m\times n}$ and their entries are sampled i.i.d. from $\mathcal{U}[0,1]$. For the MSE loss, we generate $M = \hat X \hat Y$, where $\hat X \in \mathbb{R}^{m\times r}$, $\hat Y \in \mathbb{R}^{r\times n}$ and their entries are sampled i.i.d. from $\mathcal{U}[0,1]$. We compute the singular value decomposition of $M = U \operatorname{diag}(\sigma_1, \dots, \sigma_r, 0, \dots, 0)V^\top$ and construct the data matrix $Z = U\operatorname{diag}(\sigma_1, \dots, \sigma_r, \sigma_{r+1}, \dots, \sigma_{\min(m,n)})V^\top$, where $\sigma_{r+1}, \dots, \sigma_{\min(m,n)}$ are sampled i.i.d. from $\mathcal{U}[0, 0.1\sigma_r]$. The optimal function value is given by $\frac{1}{2mn}\|Z - \hat X \hat Y\|_F^2$. Thus, the optimality gap is defined as the difference between the current function value and the optimal function value. Throughout, we set

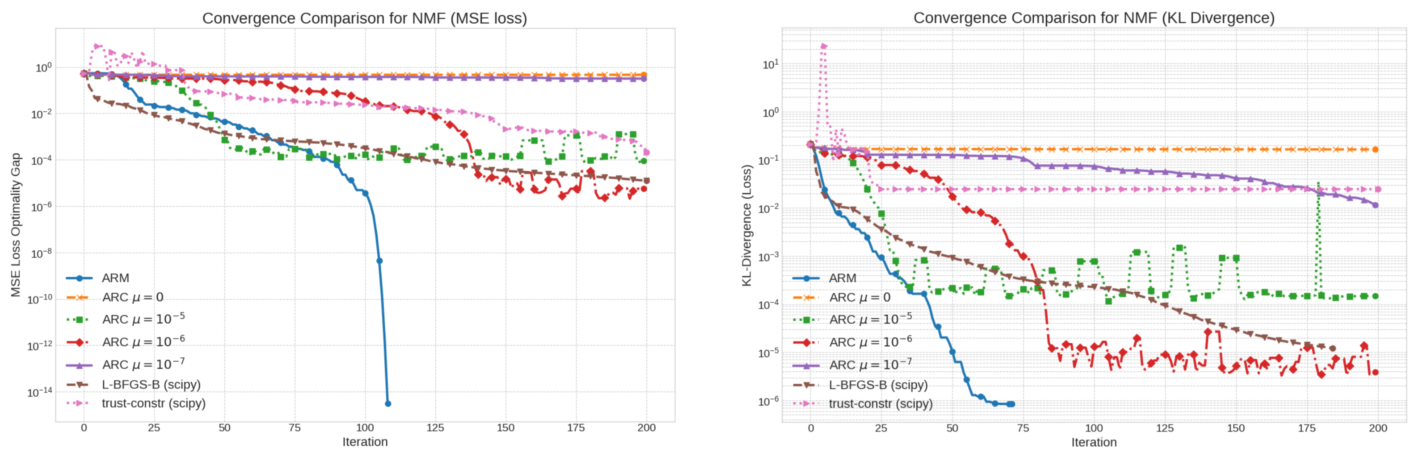

Figure 1: Performance comparison on NMF with MSE loss (left) and KL divergence (right).

$m = 100$, $n = 20$, and $r = 10$. We consider the case where the initialization of X and Y are independent, with each entry drawn i.i.d. from $\mathcal{U}[0, 1]$.

We compare the performance of Algorithm 1 (with configurations in Eq. (14) and Eq. (15)) against `L-BFGS-B` and `trust-constr` methods from `scipy` [63]. We also implement several configurations of `ARC` [10, 11] to illustrate the advantages of our method over cubic regularization method. Since `ARC` is designed for unconstrained optimization problems, we define $f(X, Y) = +\infty$ whenever $X \notin \mathbb{R}_+^{m\times r}$ or $Y \notin \mathbb{R}_+^{r\times n}$. In this setting, when some entries of X or Y approach zero, `ARC` tends to stagnate, with the regularization parameter blowing up since the direction drives those entries toward zero. To mitigate this issue, we incorporate a logarithmic barrier into the objective function when running `ARC`, defined as

$$f_\mu(X, Y) = f(X, Y) + \mu \left(\sum_{i,j=1}^{m,r} \log X_{ij} \right) + \mu \left(\sum_{i,j=1}^{r,n} \log Y_{ij} \right).$$

We report the original loss function $f(X, Y)$ and tune μ to achieve the best performance. In both Algorithm 1 and `ARC`, we use $\sigma_0 = 1$, $\eta_1 = 0.01$, $\eta_2 = 0.9$, $\gamma_1 = 0.5$, and $\gamma_2 = \gamma_3 = 2$, following [12, Section 2.4]. In Algorithm 1, we set $\kappa = 1$.

As shown in Figure 1, Algorithm 1 consistently outperforms other methods. For large μ, although `ARC` converges faster, it fails to reach high accuracy, while for small μ, `ARC` may fail to converge. These observations align with the strong performance of `ARC` in unconstrained optimization and suggest that more refined algorithmic designs may be required for constrained optimization problems.

Training convolutional neural networks (CNNs) The use of Newton's methods in large-scale neural network training is prohibitively expensive, as computing and inverting the Hessian is costly. To overcome this limitation, the Kronecker-Factored Approximate Curvature (KFAC) method was introduced in [38] as a scalable alternative to Newton's method and natural gradient descent.

KFAC is a Kronecker-factorized version of a positive definite Fisher Information Matrix (FIM), which is equivalent to the generalized Gauss-Newton matrix (GGN) in certain cases [39, 49, 37, 1], to approximate the Hessian. This avoids doing multiple back propagation steps to compute the exact Hessian and additional storage. KFAC approximates the FIM by exploiting the layer-wise structure of feed-forward neural networks. Indeed, we consider a supervised learning problem with data distribution $(x, y) \sim \mathcal{D}$, where x denotes the input and y the target. A feed-forward neural network defines a parametric function $f_\theta : \mathbb{R}^{d_x} \to \mathbb{R}^{d_y}$, with θ consisting of weight matrices and biases across all layers. In particular, we let the l-th layer be $s_l = W_l a_l + b_l$, $h_l = \phi(s_l)$, where $a_l \in \mathbb{R}^{d_{\text{in}}}$ are the input activation nodes to the layer, $W_l \in \mathbb{R}^{d_{\text{out}}\times d_{\text{in}}}$ and $b_l \in \mathbb{R}^{d_{\text{out}}}$ are the parameters, $s_l \in \mathbb{R}^{d_{\text{out}}}$ are the pre-activations, and $h_l \in \mathbb{R}^{d_{\text{out}}}$ are the output activation after applying the elementwise transformation ϕ.

Algorithm 2 KFAC **(i)** [38, 65] and KFAC with weak self-concordance (KFAC-WSC) **(ii)**.

```
1: Input: $\eta_k$: learning rate; $\omega$: weight decay; $\gamma$: damping; $\mu$: momentum, default 0.9.
2: Input: If choosing (ii), input $\kappa$: weak self-concordance parameter, default 1; input $\beta$: layerwise leaning rate momentum, default 0.99.
3: Initialization: $k \leftarrow 0$ and initialize $\{W_l\}_{l=0}^{L-1}$.
4: while stopping criterion not met do
5:     $k \leftarrow k + 1$.
6:     Sample a mini-batch and update $\{S_l\}_{l=0}^{L-1}$, $\{A_l\}_{l=0}^{L-1}$ with moving average.
7:     Compute the inverses $\{S_l^{-1}\}_{l=0}^{L-1}$, $\{A_l^{-1}\}_{l=0}^{L-1}$.
8:     for $l = 0, \dots, L-1$ do
9:         Sample a mini-batch $B$.
10:        $G_l \leftarrow \nabla_{W_l}\mathcal{L}_B(W)$, $D_l \leftarrow (A_l + \gamma I)^{-1} G_l (S_l + \gamma I)^{-1}$, $M_l \leftarrow \mu M_l + (1-\mu) D_l$.
11:        Compute $\lambda_l$ according to one of two options: (i) $\lambda_l \leftarrow 0$. (ii) $\lambda_l^2 \leftarrow \beta\lambda_l^2 + (1-\beta)\mathrm{vec}(G_l)^\top \mathrm{vec}(D_l)$.
12:        $W_l \leftarrow W_l - \frac{\eta_k}{1+\kappa\lambda_l} M_l - \eta_k \omega W_l$.
13:    end for
14: end while
```

Table 1: Test accuracy of different neural network optimizers. We replicate the KFAC experiments from [65] using 10 random seeds. Values following "±" denote standard deviations across seeds.

Dataset	Model	SGD	Adam [29]	KFAC	KFAC-WSC
CIFAR10	VGG16	93.39	93.62	93.88±0.16	**94.13±0.17**
CIFAR10	ResNet32	95.14	94.66	**95.24±0.13**	95.07±0.14
CIFAR100	VGG16	73.31	74.22	73.53±0.47	**74.27±0.19**
CIFAR100	ResNet32	77.73	77.40	**78.08±0.21**	77.07±0.39

The network is trained by minimizing $\mathcal{L}(\theta) = \mathbb{E}_{(x,y)\sim\mathcal{D}}[L(f_\theta(x), y)]$, where L is a loss function, and the FIM is $F(\theta) = \mathbb{E}_{(x,y)\sim\mathcal{D}}[\nabla_\theta \log p_\theta(y \mid x)(\nabla_\theta \log p_\theta(y \mid x))^\top]$ where $p_\theta(y \mid x)$ denotes the model's predictive distribution. For the weight matrix W_l, the gradient of $\mathcal{L}$ with respect to W_l can be written in terms of the input activation and the back-propagated signal. In particular, $\nabla_{W_l} L = g_l a_l^\top$, where $g_l = \nabla_{s_l} L \in \mathbb{R}^{d_{\text{out}}}$ is the gradient of L with respect to the pre-activations s. The Fisher block corresponding to W_l is $F_l = \mathbb{E}[\mathrm{vec}(\nabla_{W_l} L)\,\mathrm{vec}(\nabla_{W_l} L)^\top] = \mathbb{E}[(a_l \otimes g_l)(a_l \otimes g_l)^\top]$. The direct computation of F_l is intractable, as it scales with the square of the number of parameters in W. The key idea behind KFAC is to treat a_l and g_l as independent, yielding $F_l \approx \mathbb{E}[a_l a_l^\top] \otimes \mathbb{E}[g_l g_l^\top]$. For any $X \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, we have $F_l^{-1}\mathrm{vec}(X) \approx (\mathbb{E}[a_l a_l^\top])^{-1} X (\mathbb{E}[g_l g_l^\top])^{-1} = A_l^{-1} X S_l^{-1}$, where $A_l = \mathbb{E}[a_l a_l^\top]$ and $S_l = \mathbb{E}[g_l g_l^\top]$. Thus, if the partial gradient of $\mathcal{L}$ with respect to W_l is denoted as $\nabla_{W_l}\mathcal{L}(W)$ (as is common in PyTorch), the natural gradient is given by $A_l^{-1}\nabla_{W_l}\mathcal{L}(W) S_l^{-1}$.

When implementing KFAC, it is common to add regularization to A_l and S_l to improve both generalization and numerical stability [38]. The natural gradient with Tikhonov regularization [38] is $D_l = (A_l + \gamma I)^{-1}\nabla_{W_l}\mathcal{L}(W)(S_l + \gamma I)^{-1}$. We can interpret D_l as an approximate Newton direction $(\nabla^2(f + F)(x_j))^{-1}\nabla f(x_j)$ in the sense that $\nabla^2 F(x_j)$ is replaced by γI. We interpret γ as the weak self-concordance parameter ℓ and $\lambda_l = \sqrt{\mathrm{vec}(D_l)^\top \mathrm{vec}(\nabla_{W_l}\mathcal{L}(W))}$ as an approximation of $\nu_{f,F}(x_k)$.

We follow the techniques of [65] and evaluate our algorithms by training VGG16 [56] and ResNet32 [28, 64] on CIFAR-10 and CIFAR-100 [30]. We conduct experiments on a single NVIDIA V100 SXM2 GPU from San Diego Supercomputer Center [57]. The results are summarized in Table 1. We performed a grid search to identify the best hyperparameters that maximize test accuracy during training. Both KFAC and KFAC-WSC were trained for 100 epochs. The initial learning rate was selected from the set $\eta_0 \in \{10^{-3}, 3 \times 10^{-2}, 10^{-2}\}$ and the learning rate decay by 0.1 was applied at the 40th and 80th epochs. The weight decay factor from $\omega \in \{10^{-2}, 3 \times 10^{-1}, 10^{-1}\}$, and the damping factor from

$\gamma \in \{3 \times 10^{-3}, 10^{-2}, 3 \times 10^{-2}\}$. Our method improved training efficiency on VGG16 for CIFAR-10 and CIFAR-100, while being outperformed by KFAC on ResNet32.

6 Conclusions

We generalized the notion of self-concordance to non-convex settings by introducing weakly self-concordant functions and F-based self-concordant functions. Based on these two classes, we proposed regularized Newton and adaptive regularization methods and proved their global and local convergence guarantees. We also conducted numerical experiments to evaluate their effectiveness. Indeed, our adaptive regularization method outperformed adaptive cubic regularization, L-BFGS, and trust-region methods on non-negative matrix factorization problems in terms of both MSE and KL divergence losses. Our KFAC-based regularized Newton method improved training efficiency on VGG16 for CIFAR-10 and CIFAR-100, while being outperformed by KFAC on ResNet32. In summary, our results demonstrate the importance of extending self-concordance for large-scale non-convex optimization.

Acknowledgments

The second author is partially supported by a start-up fund at the University of Hong Kong and by the Hong Kong Research Grants Council under ECS project 27301425. The third author is partially supported by a start-up fund at Columbia University. We also thank the support of GPU hours from NSF ACCESS [6].

A Additional Results on Weak Self-Concordant Functions

In Section A.1, we present an alternative way to measure stationarity for weakly self-concordant functions. In Section A.2, we introduce an inexact proximal point method for minimizing weakly self-concordant functions that relies on Hessian–vector products (HVPs) and achieves an iteration complexity comparable to that of RNM and ARM, with high probability. In Section A.3, we prove our convergence results.

For a self-concordant function $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ and $x \in \mathbf{dom}(f)$, we define $\lambda_f(x) = \sqrt{\nabla f(x)^\top (\nabla^2 f(x))^{-1} \nabla f(x)}$. For a matrix $A \succ 0$, we denote its condition number by $\mathrm{cond}(A)$ and define $\|x\|_A = \sqrt{x^\top A x}$. We also define the following absolute constants $\alpha_\star = 0.0001$, $R_1 = 0.49$, $R_2 = \frac{1}{\sqrt{1-\alpha_\star}} R_1$, $R_3 = \frac{\sqrt{1-\alpha_\star}}{1+\alpha_\star} R_1$, $C_1 = 9$, $C_2 = 0.95$, and $C_3 = \frac{R_2^{-1}-1}{R_2^{-1}-2}$.

A.1 Convergence to a Proximal Stationary Point

For any $\eta > 0$, we define the Moreau envelope $f_\eta : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ [43] of f by $f_\eta(y) = \inf_{x \in \mathbb{R}^n} \{f(x) + \frac{1}{2\eta}\|y - x\|^2\}$. The gradient norms of the Moreau envelope, $\|\nabla f_\eta(x)\|$, have been used as a measure of stationarity for weakly convex functions [18]. For weakly self-concordant functions, we show next that RNM (see Eq. (8)) and Algorithm 1 can find a point with $\|\nabla f_\eta(x)\| \le \epsilon$ (for any small enough η) in $O(\epsilon^{-2})$ iterations.

Lemma A.1. *Suppose that $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is (κ, ℓ)-weakly self-concordant and lower-bounded. For any $\mu > \ell$, $\epsilon > 0$, and $x \in$ **dom**(f), if $\nu_{f, \frac{1}{2}\mu\|\cdot\|^2}(x) \le \frac{\sqrt{\mu-\ell}\epsilon}{\mu+\kappa\sqrt{\mu-\ell}\epsilon}$, then we have $\|\nabla f_{1/\mu}(x)\| \le \epsilon$.*

Proof. Let $\epsilon' = \frac{\sqrt{\mu-\ell}\epsilon}{\mu+\kappa\sqrt{\mu-\ell}\epsilon}$ and $\varphi(y) = f(y) + \frac{1}{2}\mu\|y - x\|^2$. Thus, we assume $\nu_{f, \frac{1}{2}\mu\|\cdot\|^2}(x) = \lambda_\varphi(x) \le \epsilon'$. It follows from [18, Lemma 2.2] that $\frac{1}{\mu}\|\nabla f_{1/\mu}(x)\| = \|x - \arg\min \varphi\|$. Moreover, $\nabla^2 \varphi \succeq (\mu - \ell) I$

implies that $\sqrt{\mu-\ell}\|x - \arg\min\varphi\| \le \|x - \arg\min\varphi\|_{\nabla^2\varphi(x)}$. We define $\omega_\star'(t) = \frac{t}{1-t}$ for $t \in [0,1)$ and apply [45, Eq. (5.2.4)] with $f = \varphi$. In the notation of [45], we note that $x_f^\star = \arg\min\varphi$, $M_f = \kappa$, $\|\cdot\|_x = \|\cdot\|_{\nabla^2\varphi(x)}$. Thus, we have $\|x - \arg\min_y \varphi(y)\|_{\nabla^2\varphi(x)} \le \kappa^{-1}\omega_\star'(\kappa\lambda_\varphi(x))$. Therefore, we obtain $\frac{\sqrt{\mu-\ell}}{\mu}\|\nabla f_{1/\mu}(x)\| \le \|x - \arg\min\varphi\|_{\nabla^2\varphi(x)} \le \kappa^{-1}\omega_\star'(\kappa\lambda_\varphi(x)) \le \kappa^{-1}\omega_\star'(\kappa\epsilon') = \frac{\sqrt{\mu-\ell}}{\mu}\epsilon$. □

A.2 Inexact Proximal Point Method (IPPM)

We develop an algorithm for minimizing a (κ,ℓ)-weakly self-concordant function f using only gradient and Hessian–vector product (HVP) oracles. Our approach is based on the proximal point method [51]. Given the current iterate $z_j \in \mathbf{dom}(f)$, we consider the regularized objective $f_j : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ defined as $f_j(y) = f(y) + \frac{1}{2}\mu\|y - z_j\|^2$. The update of the proximal point method is given by $z_{j+1} = \arg\min f_j$. When $\mu > \ell$, each function f_j is self-concordant.

To solve these subproblems inexactly, we introduce Algorithm 3. The algorithm combines the Newton–CG (conjugate gradient) method [48, Algorithm 7.1] with the Lanczos method [31]. By exploiting self-concordance, it avoids line searches, which is its main advantage over traditional Newton–CG. For a self-concordant function f, the algorithm outputs, with probability $1-\delta$, $y \in \mathbf{dom}(f)$ such that either $\lambda_f(y) \le \epsilon_1$ and $\lambda_f(x_0) \ge \sqrt{\frac{1-\alpha_\star}{1+\alpha_\star}}\epsilon_0$, or $y = x_0$ and $\lambda_f(x_0) \le \epsilon_0$. A more precise convergence theorem, together with its proof, is deferred to Theorem A.10 in Appendix A.3. This algorithm may be of independent interest in convex optimization.

Algorithm 3 A Newton-CG Method for Self-Concordant Functions

```
1: **Input:** $\kappa \ge 0$, $\epsilon_0 > 0$, $\epsilon_1 > 0$, $\beta > 1$, $\delta \in [0,1)$.
2: **Initialization:** $x_0 \in \mathbf{dom}(f)$.
3: $\beta_0 \leftarrow$ sqrt-cond$(\nabla^2 f(x_0), 0.5\delta, \beta)$, flag $\leftarrow$ True.
4: for $k = 0,1,2,\dots$ do
5:     $g_k \leftarrow \nabla f(x_k)$, $h_k \leftarrow$ CG-Inverse$(\nabla^2 f(x_k), g_k, \beta_k, \alpha_\star)$.
6:     $\rho_k \leftarrow h_k^\top g_k$, $\delta_k \leftarrow \nabla^2 f(x_k)[h_k, h_k]$.
7:     return $x_k$ if $\rho_k \le (1-\alpha_\star)\epsilon_1^2$ or ($k = 0$ and $\rho_0 \le (1-\alpha_\star)\epsilon_0^2$).
8:     if $\rho_k > \frac{R_1^2}{\kappa^2}$ then
9:         $t_k \leftarrow \frac{\rho_k}{\delta_k + \kappa\rho_k\sqrt{\delta_k}}$, $B_k \leftarrow (1 + \frac{\sqrt{1+\alpha_\star}}{1-\alpha_\star}\kappa\sqrt{\rho_k})^2$, $\beta_{k+1} \leftarrow B_k\beta_k$.
10:    else
11:        if flag then
12:            $\beta^\star \leftarrow$ sqrt-cond$(\nabla^2 f(x_k), 0.5\delta, \beta_k)$, $\beta_{\text{local}} \leftarrow C_3^4\beta^\star$, flag $\leftarrow$ False.
13:        end if
14:        $t_k \leftarrow \frac{\rho_k}{\delta_K + 2\kappa\rho_k\sqrt{\delta_k}}$, $\beta_{k+1} \leftarrow \beta_{\text{local}}$.
15:    end if
16:    $x_{k+1} \leftarrow x_k - t_k h_k$.
17: end for
18: function sqrt-cond$(H, \delta, \beta)$:
19:     Use the Lanczos algorithm to obtain $\lambda_1, \lambda_2$ such that $|\lambda_1 - \lambda_{\max}(H)| \le \frac{1}{3}\lambda_{\max}(H)$ and $|\lambda_2 - \lambda_{\min}(H)| \le \frac{1}{3}\lambda_{\min}(H)$. return $\sqrt{2\lambda_1/\lambda_2}$.
20: function CG-Inverse$(H, g, \beta, \alpha)$:
21:     Perform $\min\left\{n, \left\lfloor \log_{\frac{\beta-1}{\beta+1}} 0.5\alpha \right\rfloor + 1\right\}$ CG iterations to compute $H^{-1}g$ start from 0.
```

We are now ready to introduce Algorithm 4 (IPPM). At each iterate z_j, IPPM computes an inexact proximal point z_{j+1} by approximately solving the subproblem $\min f_j$ to a prescribed accuracy using

Algorithm 3. The convergence of IPPM is established in the Theorem A.3, whose proof is deferred to the end of Appendix A.3. This result relies on Assumption A.2, which requires that all proximal subproblems are sufficiently well conditioned.

Algorithm 4 Inexact Proximal Point Method (IPPM)

```
1: Input: $\kappa \geq 0$, $\mu > 0$, $\epsilon > 0$, $\beta > 1$, $\delta \in [0, 1)$, $\Delta > 0$.
2: Initialization: $z_0 \in \mathbf{dom}(f)$.
3: $K \leftarrow \frac{8\mu\Delta}{(\mu-\ell)\epsilon^2}$, $\delta' \leftarrow (K+2)^{-1}\delta$, $\beta' \leftarrow C_3^2\beta$, $\epsilon' \leftarrow \left(\sqrt{\frac{1-\alpha_\star}{1+\alpha_\star}} - 0.5\right)\epsilon$
4: for $j = 0, 1, 2, \ldots$ do
5:     Set $f_j : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ with $f_j(y) := f(y) + \frac{1}{2}\mu\|y - z_j\|^2$.
6:     $z_{j+1} \leftarrow$ The output of Algorithm 3 with objective function $f_j$, input $\kappa \leftarrow \kappa$, $\epsilon_0 \leftarrow \epsilon$, $\epsilon_1 \leftarrow \epsilon'$, $\beta \leftarrow \beta'$,
       $\delta \leftarrow \delta'$, and initialization $x_0 \leftarrow z_j$.
7:     return $z_j$ if $z_{j+1} = z_j$.
8: end for
```

Assumption A.2. *In IPPM, there exists $B > 0$ such that for all $j \geq 0$, we have* $\mathrm{cond}(\nabla^2 f(z_0) + \mu I) \leq B^2$ *and* $\mathrm{cond}(\nabla^2 f_j(\arg\min f_j) + \mu I) \leq B^2$.

Theorem A.3. *Assume that f is (κ, ℓ)-weakly self-concordant, lower bounded, and that Assumption A.2 holds. Let $\mu > \ell$, $\beta \geq B$, $\Delta \geq f(z_0) - \inf_y f(y)$, and $\epsilon' \leq \frac{R_2}{\kappa}$. Then, with probability at least $1 - \delta$, IPPM terminates in at most $\left\lfloor \frac{8\mu(f(z_0) - \inf_y f(y))}{(\mu-\ell)\epsilon^2} \right\rfloor + 2$ iterations, and its output y satisfies $\nu_{f, \frac{1}{2}\mu\|\cdot\|^2}(y) \leq \epsilon$. Moreover, the total number of HVP computations performed by IPPM is $O(\epsilon^{-2}(\log(\epsilon^{-1}) + \log(\delta^{-1})))$.*

A.3 Convergence Analysis

In this section, we prove the convergence of Algorithms 3 and 4. To do so, we first present some technical lemmas. Lemma A.4 analyzes the function sqrt-cond in Algorithm 3 by a classical result [31] on the Lanczos method. Lemma A.5 analyzes the function CG-Inverse in Algorithm 3 using [48, Theorem 5.5]. Lemma A.6 describes the relationship between the condition numbers of the Hessians at two nearby points for self-concordant functions.

Lemma A.4. *Let $A \succ 0$. Given $\delta \in [0, 1)$ and $\beta \geq \sqrt{\mathrm{cond}(A)}$, sqrt-cond$(A, \delta, \beta)$ outputs a number in $[\sqrt{\mathrm{cond}(A)}, 2\sqrt{\mathrm{cond}(A)}]$ with probability at least $1 - \delta$. The total number of matrix-vector multiplications with A required by the method is at most $C_L \min\{n, \beta\log(n\delta^{-2})\}$, where $C_L > 0$ is a constant.*

Proof. Let $\epsilon' = \frac{1}{10\beta^2}$. It follows from [31, Theorem 4.2] that, with probability at least $\frac{1}{2}\delta$, the Lanczos method outputs λ_1 such that $|\lambda_1 - \lambda_{\max}(A)| \leq \frac{1}{3}\lambda_{\max}(A)$ in $O(\min\{n, \log(n\delta^{-2})\})$ matrix-vector multiplications. With probability at least $1 - \frac{1}{2}\delta$, the Lanczos method outputs λ' such that $\left|\lambda' - \lambda_{\max}\left(\frac{2\lambda_1}{1-\epsilon'}I - A\right)\right| \leq \epsilon' \cdot \lambda_{\max}\left(\frac{2\lambda_1}{1-\epsilon'}I - A\right) \leq \frac{1}{3}\lambda_{\min}(A)$ in $O(\min\{n, \beta\log(n\delta^{-2})\})$ matrix-vector multiplications. Letting $\lambda_2 = \frac{2\lambda_2}{1-\epsilon'} - \lambda'$, then we have $|\lambda_2 - \lambda_{\min}(A)| \leq \frac{1}{3}\lambda_{\min}(A)$. Thus, letting $X^2 = 2\lambda_1/\lambda_2$ suffices. □

Lemma A.5 (Theorem 5.5 of [48])**.** *Given a matrix $H \succ 0$ and $\beta \geq \sqrt{\mathrm{cond}(H)}$, CG-Inverse$(H, g, \beta, \alpha)$ returns $h \in \mathbb{R}^n$ such that $\|h - H^{-1}g\|_H \leq \alpha\|g\|_{H^{-1}}$. The algorithm requires at most $C_{\mathrm{CG}}\min\{n, \beta\log(\alpha^{-1})\}$ matrix-vector multiplications with H, where $C_{\mathrm{CG}} > 0$ is a constant. Moreover, we have $(1 - \alpha)\|g\|_{H^{-1}} \leq \|h\|_H \leq (1 + \alpha)\|g\|_{H^{-1}}$ and $(1 - \alpha)g^\top H^{-1}g \leq h^\top g \leq (1 + \alpha)g^\top H^{-1}g$.*

Lemma A.6. *Let f be κ-self-concordant with global minimizer $x_f^\star$. Let $x, y \in$ **dom**(f), $R \in (0, 0.5)$, and $\kappa > 0$ such that $\lambda_f(x) \le \frac{R}{\kappa}$ and $\lambda_f(y) \le \frac{R}{\kappa}$. Then* $\operatorname{cond}(\nabla^2 f(x)) \le \left(\frac{R^{-1}-1}{R^{-1}-2}\right)^4 \operatorname{cond}(\nabla^2 f(x_f^\star)) \le \left(\frac{R^{-1}-1}{R^{-1}-2}\right)^8 \operatorname{cond}(\nabla^2 f(y))$.

Proof. [45, Theorem 5.1.13] implies that $x_f^\star$ exists. It follows from [45, Theorem 5.1.7, Theorem 5.2.1] that $(1-\kappa r)^2 \nabla^2 f(x) \preceq \nabla^2 f(x_f^\star) \preceq \frac{1}{(1-\kappa r)^2} \nabla^2 f(x)$, where $r = \|x - x_f^\star\|_{\nabla^2 f(x)} \le \frac{1}{(R^{-1}-1)\kappa}$. The desired result follows from direct calculation. $\square$

We are ready to analyze Algorithm 3. Convergence of the algorithm consists of two stages. In the first stage, it converges to a neighborhood of the global minimizer (see Lemma A.7). In the second stage, it converges Q-linearly to the global minimizer (see Lemma A.8).

Lemma A.7. *Suppose f is κ-self-concordant. At any given iteration of Algorithm 3, if $\beta_k^2 \ge$* $\operatorname{cond}(\nabla^2 f(x_k))$ *and $\rho_k > \frac{R_1^2}{\kappa^2}$, then*

$$f(x_k) - f(x_{k+1}) \ge \kappa^{-2}\, \omega\left(\tfrac{\sqrt{1-\alpha_\star}}{1+\alpha_\star}\, \kappa\sqrt{\rho_k}\right) \ge \kappa^{-2}\omega(R_3)$$

and $\sqrt{\frac{\operatorname{cond}(\nabla^2 f(x_{k+1}))}{\operatorname{cond}(\nabla^2 f(x_k))}} \le B_k \le \exp\left(C_1 \kappa^2 (f(x_k) - f(x_{k+1}))\right)$.

Proof. It follows from Lemma A.5 that $\frac{\rho_k}{\delta_k} \in [\frac{1-\alpha_\star}{(1+\alpha_\star)^2}, \frac{1+\alpha_\star}{(1-\alpha_\star)^2}]$. Thus, Proposition 2.6 implies that $f(x_k) - f(x_{k+1}) \ge \kappa^{-2}\omega\left(\frac{\kappa\rho_k}{\sqrt{\delta_k}}\right) \ge \kappa^{-2}\omega\left(\frac{\sqrt{1-\alpha_\star}}{1+\alpha_\star}\kappa\sqrt{\rho_k}\right) \ge \kappa^{-2}\omega(R_3)$. Let $r_k = \|x_{k+1} - x_k\|_{\nabla^2 f(x_k)} = \frac{\rho_k\sqrt{\delta_k}}{\delta_k + \kappa\rho_k\sqrt{\delta_k}}$. Hence, we have $(1-\kappa r_k)^{-1} \le 1 + \frac{\sqrt{1+\alpha_\star}}{1-\alpha_\star}\kappa\sqrt{\rho_k}$. It follows from [45, Theorem 5.1.7] that $\sqrt{\frac{\operatorname{cond}(\nabla^2 f(x_{k+1}))}{\operatorname{cond}(\nabla^2 f(x_k))}} \le (1-\kappa r_k)^{-2} \le B_k$. Moreover, we have

$$\frac{\log B_k}{\kappa^2(f(x_k) - f(x_{k+1}))} \le \frac{2\log\left(1 + \frac{\sqrt{1+\alpha_\star}}{1-\alpha_\star}\kappa\sqrt{\rho_k}\right)}{\omega\left(\frac{\sqrt{1-\alpha_\star}}{1+\alpha_\star}\kappa\sqrt{\rho_k}\right)} \le \frac{2\log\left(1 + \frac{\sqrt{1+\alpha_\star}}{1-\alpha_\star}R_1\right)}{\omega\left(\frac{\sqrt{1-\alpha_\star}}{1+\alpha_\star}R_1\right)} \le C_1,$$

where the second inequality holds by the monotonicity of $\frac{\log\left(1+\frac{\sqrt{1+\alpha_\star}}{1-\alpha_\star}t\right)}{\omega\left(\frac{\sqrt{1-\alpha_\star}}{1+\alpha_\star}t\right)}$ for $t > R_1$. This completes the proof. $\square$

Lemma A.8. *Suppose f is κ-self-concordant. At any given iteration of Algorithm 3, if $\beta_k^2 \ge$* cond $(\nabla^2 f(x_k))$ *and $\rho_k \le \frac{R_1^2}{\kappa^2}$, then $(1+\alpha_\star)\lambda_f(x_{k+1})^2 \le C_2\rho_k$.*

Proof. If $\rho_k \le \frac{R_1^2}{\kappa^2}$, then we have $t_k = \frac{\rho_k}{\delta_K + 2\kappa\rho_k\sqrt{\delta_k}}$. Letting $\lambda_k = \lambda_f(x_k)$, $H_k = \nabla^2 f(x_k)$, $G_k = \int_0^1 \nabla^2 f(x_k + \tau(x_{k+1} - x_k))d\tau$, $\zeta_k = \kappa\sqrt{\delta_k}$, we have $\nabla f(x_{k+1}) = G_k(x_{k+1} - x_k) + \nabla f(x_k) = -t_k G_k h_k + g_k$. It follows from [45, Corollary 5.1.5] that $(1 - t_k\zeta_k + \frac{1}{3}t_k^2\zeta_k^2)H_k \preceq G_k \preceq \frac{1}{1-t_k\zeta_k}H_k$. Moreover, [45, Lemma 5.1.7] implies that $H_{k+1}^{-1} \preceq \frac{1}{(1-t_k\zeta_k)^2}H_k^{-1}$. Thus, we have $\lambda_f(x_{k+1})^2 \le \frac{1}{(1-t_k\zeta_k)^2}\|\nabla f(x_{k+1})\|_{H_k^{-1}}^2$, where

$$\|\nabla f(x_{k+1})\|_{H_k^{-1}}^2 = \lambda_k^2 - 2t_k h_k^\top G_k h_k + t_k^2 h_k^\top G_k H_k^{-1} G_k h_k - 2t_k h_k^\top G_k(H_k^{-1}g_k - h_k).$$

Using the Cauchy–Schwarz inequality and Lemma A.5, we have

$$|2h_k^\top G_k(H_k^{-1}g_k - h_k)| \le \alpha_\star\|G_k h_k\|_{H_k^{-1}}^2 + \alpha_\star^{-1}\|H_k^{-1}g_k - h_k\|_{H_k}^2.$$

Thus, we have $\lambda_{k+1}^2 \le \lambda_k^2 g(t_k, \zeta_k)$, where

$$g(t,\zeta) = \frac{1 - 2t(1-\alpha_\star)^2(1 - t\zeta + \frac{1}{3}t^2\zeta^2)}{(1-t\zeta)^2} + \frac{t^2(1+\alpha_\star)^2}{(1-t\zeta)^4} + \frac{\alpha_\star t}{(1-t\zeta)^2}\left(\frac{(1+\alpha_\star)^2}{(1-t\zeta)^2} + 1\right).$$

Lemma A.5 implies $\zeta_k \in [0, \frac{1+\alpha_\star}{\sqrt{1-\alpha_\star}} R_1]$. For any $\zeta_l < \zeta_u$, we define $t_u = \frac{1}{\frac{(1-\alpha_\star)^2}{1+\alpha_\star} + 3\zeta_l}$ and $t_l = \frac{1}{\frac{(1+\alpha_\star)^2}{1-\alpha_\star} + 3\zeta_u}$. Since $t_k = \frac{1}{\frac{\delta_k}{\rho_k} + 3\zeta_k}$ and $\frac{\delta_k}{\rho_k} \in [\frac{(1-\alpha_\star)^2}{1+\alpha_\star}, \frac{(1+\alpha_\star)^2}{1-\alpha_\star}]$, we have

$$g(t_k, \zeta_k) \le \frac{1-2t_l(1-\alpha_\star)^2(1-t_u\zeta_u)}{(1-t_l\zeta_l)^2} + \frac{t_u^2(1+\alpha_\star)^2}{(1-t_u\zeta_u)^4} + \frac{\alpha_\star t_u}{(1-t_u\zeta_u)^2}\left(\frac{(1+\alpha_\star)^2}{(1-t_u\zeta_u)^2} + 1\right)$$

for any $\zeta_k \in [\zeta_l, \zeta_u]$. Dividing $[0, \frac{1+\alpha_\star}{\sqrt{1-\alpha_\star}} R_1]$ into 1000 intervals equally, we can verify numerically that $g(t_k, \zeta_k) \le 0.945$. The desired result follows from $\lambda_k^2 \le \rho_k(1-\alpha_\star)^{-1}$. $\square$

In Algorithm 3, we define $k^\star = \min\{k : \rho_k \le R_1^2 \kappa^{-2}\}$ and the events

$$\begin{aligned}
A_1 &= \left\{\mathrm{cond}(\nabla^2 f(x_0)) \le \beta_0^2 \le 4\mathrm{cond}(\nabla^2 f(x_0))\right\}, \\
A_2 &= \left\{\mathrm{cond}(\nabla^2 f(x_{k^\star})) \le (\beta^\star)^2 \le 4\mathrm{cond}(\nabla^2 f(x_{k^\star}))\right\}, \\
A_3 &= \left\{\beta_k^2 \ge \mathrm{cond}(\nabla^2 f(x_k)),\ \forall k \ge 0\right\}.
\end{aligned}$$

Lemma A.9 shows that $A_1 \cap A_2 \cap A_3$ happens with high probability. Hence, using two calls of sqrt-cond, Algorithm 3 gets good estimates of Hessian condition numbers.

If $A_1 \cap A_2 \cap A_3$ holds, then for all $0 \le k \le k^\star$, using Lemma A.7, we have

$$\beta_k = \beta_0 B_0 B_1 \cdots B_{k-1} \le 2\sqrt{\mathrm{cond}(\nabla^2 f(x_0))} \exp(C_1 \kappa^2 (f(x_0) - f(x_k))). \tag{21}$$

Moreover, for all $k \ge k^\star + 1$, by Lemma A.6, we have

$$\beta_k = C_3^4 \beta^\star \le 2C_3^4 \sqrt{\mathrm{cond}(\nabla^2 f(x_{k^\star}))} \le 2C_3^6 \sqrt{\mathrm{cond}(\nabla^2 f(\arg\min f))}. \tag{22}$$

Lemma A.9. *Suppose that f is κ-self-concordant and $\beta^2 \ge \mathrm{cond}(\nabla^2 f(x_0))$, then $\mathbb{P}(A_1 \cap A_2 \cap A_3) \ge 1 - \delta$.*

Proof. By Lemma A.4, we have $\mathbb{P}(A_1) \ge 1 - \frac{1}{2}\delta$. Therefore, it suffices to show $\mathbb{P}(A_2 | A_1) \ge 1 - \frac{1}{2}\delta$ and $A_1 \cap A_2 \subseteq A_3$. To show $\mathbb{P}(A_2|A_1) \ge 1 - \frac{1}{2}\delta$, we observe from Lemma A.7 that $\beta_0^2 \ge \mathrm{cond}(\nabla^2 f(x_0))$ implies $\beta_{k^\star}^2 \ge \mathrm{cond}(\nabla^2 f(x_{k^\star}))$. Thus, Lemma A.4 implies $\mathbb{P}(A_2|A_1) \ge 1 - \frac{1}{2}\delta$. It remains to show $A_1 \cap A_2 \subseteq A_3$. Suppose that $A_1 \cap A_2$ holds, then we have $\mathrm{cond}(\nabla^2 f(x_l)) \le \beta_l^2$ for $0 \le l \le k^\star$ by Lemma A.7. Moreover, Lemma A.8 yields $\rho_l \le \frac{R_1^2}{\kappa^2}$ for all $l \ge k^\star + 1$. Therefore, using Lemma A.6, we have $\mathrm{cond}(\nabla^2 f(x_l)) \le C_3^8 \mathrm{cond}(\nabla^2 f(x_{k^\star})) \le C_3^8 (\beta^\star)^2 = \beta_{\mathrm{local}}^2 = \beta_l^2$. This completes the proof. $\square$

We present the convergence theorem for Algorithm 3.

Theorem A.10. *Suppose that f is κ-self-concordant, bounded below, and that $\beta^2 \ge \mathrm{cond}(\nabla^2 f(x_0))$. Then, with probability at least $1 - \delta$, the following statements hold:*

1. *Algorithm 3 terminates and returns a point y. If $y \ne x_0$, then $\lambda_f(y) \le \epsilon_1$ and $\lambda_f(x_0) \ge \sqrt{\frac{1-\alpha_\star}{1+\alpha_\star}}\epsilon_0$; if $y = x_0$, then $\lambda_f(x_0) \le \epsilon_0$. Let $\Delta_f = \kappa^2(f(x_0) - f(y))$, $K_1 = \frac{\Delta_f}{\omega(R_3)}$, and $K_2 = 2\log_{C_2^{-1}}\left(\frac{R_2}{\kappa\epsilon_1}\right)$. Then Algorithm 3 terminates within $2 + \lfloor K_1 \rfloor + \max(\lfloor K_2 \rfloor, 0)$ iterations.*

2. *Define $B = \max\left(\sqrt{\mathrm{cond}(\nabla^2 f(x_0))}, \sqrt{\mathrm{cond}(\nabla^2 f(\arg\min_y f(y)))}\right)$, and*

$$\begin{aligned}
\Lambda_1 &= C_{\mathrm{CG}} \min\left\{n,\ 2Be^{C_1\Delta_f}\log(\alpha_\star^{-1})\right\}, \\
\Lambda_2 &= C_{\mathrm{CG}} \min\left\{n,\ 2BC_3^6 \log(\alpha_\star^{-1})\right\}, \\
\Lambda_3 &= C_L \left(\min\left\{n,\ \beta\log(n\delta^{-2})\right\} + \min\left\{n,\ 2Be^{C_1\Delta_f}\log(n\delta^{-2})\right\}\right).
\end{aligned}$$

The total number of HVP computations performed by Algorithm 3 is at most $(\Lambda_1 + 1)(K_1 + 1) + (\Lambda_2 + 1)(K_2 + 1) + \Lambda_3 = O(\log(\epsilon_1^{-1}) + \log(\delta^{-1}))$.

Proof. By Lemma A.9, it suffices to prove all statements hold deterministically assuming that the event $A_1 \cap A_2 \cap A_3$ holds. For all $k = 0, \ldots, k^\star - 1$, we have $\rho_k > \frac{R_1^2}{\kappa^2}$. Therefore, it follows from Lemma A.7 that $f(x_k) - f(x_{k+1}) \geq \kappa^{-2}\omega(R_3)$. Summing this inequality over $k = 0, 1, \ldots, k^\star - 1$ yields $k^\star \kappa^{-2}\omega(R_3) \leq f(x_0) - f(x_{k^\star})$. Thus, $k^\star$ must be finite. Lemma A.8 implies that we have $\rho_{k+1} \leq C_2 \rho_k \leq \frac{R_1^2}{\kappa^2}$ for all $k \geq k^\star$. Thus, we have $(1-\alpha_\star)\epsilon_1^2 \leq \rho_k \leq C_2^{k-k^\star}\rho_{k^\star} \leq C_2^{k-k^\star}\frac{R_1^2}{\kappa^2}$. Therefore, the algorithm terminates at some $k \leq 1 + \lfloor K_1 \rfloor + \max(\lfloor K_2 \rfloor, 0)$.

Suppose that Algorithm 3 returns $x_M \neq x_0$, then we have $\rho_0 \geq (1-\alpha_\star)\epsilon_0^2$. It follows from Lemma A.5 that $\lambda_f(x_0) \geq \sqrt{\frac{1-\alpha_\star}{1+\alpha_\star}}\epsilon_0$. Moreover, since Algorithm 3 terminates at x_M, we have $\rho_M \leq (1-\alpha_\star)\epsilon_1^2$, and thus $\lambda_f(x_M) \leq \epsilon_1$. Suppose that Algorithm 3 returns x_0, then we have $\lambda_f(x_0) \leq \epsilon_0$ by Lemma A.5.

Now, we can estimate the total number of HVPs. By Lemma A.5, the total number of HVPs called by CG-Inverse is bounded by $\sum_{k=0}^{k^\star} C_{\text{CG}} \min\{n, \beta_k \log(\alpha_\star^{-1})\} + \sum_{k=1}^{\lfloor K_2 \rfloor + 1} C_{\text{CG}} \min\{n, \beta_{k+k^\star}\log(\alpha_\star^{-1})\}$. Moreover, we call sqrt-cond at the zeroth iteration and the $k^\star$-th iteration. Hence, the total number of HVP computations called by sqrt-cond is at most Λ_3, by Lemma A.4 and Eq. (22). One HVP is computed in each iteration additionally. Putting these pieces together, the desired result follows from Eq. (21) and Eq. (22). □

We are ready to prove the convergence of IPPM, which shares a similar spirit with the methods from [8].

Proof of Theorem A.3. With probability at least $1 - (K+2)\delta' \geq 1 - \delta$, the results of Theorem A.10 hold for the first $\lfloor K \rfloor + 2$ calls of Algorithm 3. Hence, if we are able to prove that IPPM must terminate in $\lfloor K \rfloor + 2$ iterations under the assumption that the results of Theorem A.10 hold for all calls of Algorithm 3, then indeed IPPM terminates in $\lfloor K \rfloor + 2$ iterations with probability at least $1 - \delta$.

It follows from Proposition 2.6, Lemma A.7, and Lemma A.8 that $f(z_{j+1}) + \frac{1}{2}\mu\|z_{j+1} - z_j\|^2 = f_j(z_{j+1}) \leq f_j(z_j) = f(z_j)$. If we have $z_j \neq z_{j+1}$ for $j = 0, 1, \ldots, N-1$, then summing this inequality over $j = 0, 1, \ldots, N-1$ yields

$$\tfrac{1}{2}\mu \sum_{j=0}^{N-1} \|z_{j+1} - z_j\|^2 \leq f(z_0) - f(z_N) \leq f(z_0) - \inf_y f(y). \tag{23}$$

Let $H_j = \nabla^2 f(z_j) + \mu I$. By the fact that $\lambda_{\max}(H_j^{-1}) = (\lambda_{\min}(H_j))^{-1} \leq \frac{1}{\mu - \ell}$ and the definition of f_j, we have $\|z_{j+1} - z_j\|^2 \geq (\mu - l)(z_{j+1} - z_j)^\top H_{j+1}^{-1}(z_{j+1} - z_j) = \frac{\mu - l}{\mu^2}\|\nabla f(z_{j+1}) - \nabla f_j(z_{j+1})\|_{H_{j+1}^{-1}}^2$. Thus, we have $\|z_{j+1} - z_j\|^2 \geq \frac{\mu-l}{\mu^2}(\|\nabla f(z_{j+1})\|_{H_{j+1}^{-1}} - \|\nabla f_j(z_{j+1})\|_{H_{j+1}^{-1}})^2 = \frac{\mu-l}{\mu^2}(\lambda_{f_{j+1}}(z_{j+1}) - \lambda_{f_j}(z_{j+1}))^2 \geq \frac{\mu-l}{\mu^2}(\sqrt{\frac{1-\alpha_\star}{1+\alpha_\star}}\epsilon - \epsilon')^2 = \frac{\mu-l}{4\mu^2}\epsilon^2$, where the last inequality follows from Theorem A.10. Therefore, we have $\|z_{j+1} - z_j\|^2 \geq \frac{\mu-l}{4\mu^2}\epsilon^2$ and thus $N \leq \frac{8\mu(f(z_0) - \inf f)}{(\mu-\ell)\epsilon^2}$ by Eq. (23).

Suppose that IPPM terminates at $z_N = z_{N+1}$, then we have $N \leq \frac{8\mu(f(z_0)-\inf f)}{(\mu-\ell)\epsilon^2}$. This implies that IPPM terminates in $\lfloor K \rfloor + 2$ iterations. It follows from Theorem A.10 that $\lambda_{f_{N+1}}(z_{N+1}) = \nu_{f, \frac{1}{2}\mu\|\cdot\|^2}(z_{N+1}) \leq \epsilon$.

Let $\Lambda_1 = C_{\text{CG}} \min\left\{n,\, 2Be^{C_1\Delta_f}\log(\alpha_\star^{-1})\right\}$, $\Lambda_2 = C_{\text{CG}}\min\{n,\, 2BC_3^6\log(\alpha_\star^{-1})\}$, and $\Lambda_3 = C_L(\min\{n,\, \beta'\log(n\delta'^{-2})\} + \min\{n,\, 2Be^{C_1\Delta_f}\log(n\delta'^{-2})\})$. For $j \leq N$, we define $\Delta_f^{(j)} = f_j(z_j) - f_j(z_{j+1})$ and $K_1^{(j)} = \frac{\Delta_f^{(j)}}{\omega(R_3)}$. Moreover, we define $\Delta_f^{(N+1)} = 0$ and $K_2 = 2\log_{C_2^{-1}}\frac{R_2}{\kappa\epsilon'}$. Theorem A.10 yields that the total number of HVP computations by IPPM is bounded by $\sum_{j=0}^{N+1}(1 + K_1^{(j)})(1 + \Lambda_1) + (1 + K_2)(1 + \Lambda_2) + \Lambda_3 \leq \frac{1+\Lambda_1}{\omega(R_3)}\sum_{j=0}^{N+1}\Delta_f^{(j)} + (1 + \Lambda_2)(N+2)2\log_{C_2^{-1}}\frac{R_2}{\kappa\epsilon'} + (N+2)(2 + \Lambda_1 + \Lambda_2 + \Lambda_3)$. The desired result follows from $\sum_{j=0}^{N+1}\Delta_f^{(j)} = \sum_{j=0}^{N+1}(f_j(z_j) - f_j(z_{j+1})) = \sum_{j=0}^{N+1}(f(z_j) - f(z_{j+1}) - \frac{1}{2}\mu\|z_{j+1} - z_j\|^2) \leq f(z_0) - \inf f$. □

References

[1] N. Abreu, N. Vyas, S. Kakade, and D. Morwani. The potential of second-order optimization for LLMs: A study with full Gauss-Newton. *ArXiv Preprint: 2510.09378*, 2025.

[2] P.-A. Absil, R. Mahony, and B. Andrews. Convergence of the iterates of descent methods for analytic cost functions. *SIAM Journal on Optimization*, 16(2):531–547, 2005.

[3] N. Agarwal, Z. Allen-Zhu, B. Bullins, E. Hazan, and T. Ma. Finding approximate local minima faster than gradient descent. In *STOC*, pages 1195–1199, 2017.

[4] F. Bach. Self-concordant analysis for logistic regression. *Electronic Journal of Statistics*, 4:384–414, 2010.

[5] F. Bach. Adaptivity of averaged stochastic gradient descent to local strong convexity for logistic regression. *The Journal of Machine Learning Research*, 15(1):595–627, 2014.

[6] T. J. Boerner, S. Deems, T. R. Furlani, S. L. Knuth, and J. Towns. Access: Advancing innovation: NSF's advanced cyberinfrastructure coordination ecosystem: Services & support. In *Practice and Experience in Advanced Research Computing 2023: Computing for the Common Good*, pages 173–176. Association for Computing Machinery (ACM), 2023.

[7] A. Carderera, M. Besançon, and S. Pokutta. Simple steps are all you need: Frank-Wolfe and generalized self-concordant functions. In *NeurIPS*, pages 5390–5401, 2021.

[8] Y. Carmon, J. C. Duchi, O. Hinder, and A. Sidford. Accelerated methods for nonconvex optimization. *SIAM Journal on Optimization*, 28(2):1751–1772, 2018.

[9] C. Cartis, N. I. M. Gould, and P. L. Toint. On the complexity of steepest descent, Newton's and regularized Newton's methods for nonconvex unconstrained optimization problems. *SIAM Journal on Optimization*, 20(6):2833–2852, 2010.

[10] C. Cartis, N. I. M. Gould, and P. L. Toint. Adaptive cubic regularisation methods for unconstrained optimization. Part I: motivation, convergence and numerical results. *Mathematical Programming*, 127(2):245–295, 2011.

[11] C. Cartis, N. I. M. Gould, and P. L. Toint. Adaptive cubic regularisation methods for unconstrained optimization. Part II: worst-case function-and derivative-evaluation complexity. *Mathematical Programming*, 130(2):295–319, 2011.

[12] C. Cartis, N. I. M. Gould, and P. L. Toint. *Evaluation Complexity of Algorithms for Nonconvex Optimization: Theory, Computation and Perspectives*. SIAM, 2022.

[13] A. R. Conn, N. I. M. Gould, and P. L. Toint. *Trust Region Methods*. SIAM, 2000.

[14] F. E. Curtis, Z. Lubberts, and D. P. Robinson. Concise complexity analyses for trust region methods. *Optimization Letters*, 12(8):1713–1724, 2018.

[15] F. E. Curtis, D. P. Robinson, C. W. Royer, and S. J. Wright. Trust-region Newton-CG with strong second-order complexity guarantees for nonconvex optimization. *SIAM Journal on Optimization*, 31(1):518–544, 2021.

[16] F. E. Curtis, D. P. Robinson, and M. Samadi. A trust region algorithm with a worst-case iteration complexity of $\mathcal{O}(\epsilon^{-3/2})$ for nonconvex optimization. *Mathematical Programming*, 162:1–32, 2017.

[17] F. E. Curtis and Q. Wang. Worst-case complexity of trace with inexact subproblem solutions for nonconvex smooth optimization. *SIAM Journal on Optimization*, 33(3):2191–2221, 2023.

[18] D. Davis and D. Drusvyatskiy. Stochastic model-based minimization of weakly convex functions. *SIAM Journal on Optimization*, 29(1):207–239, 2019.

[19] N. Doikov. Minimizing quasi-self-concordant functions by gradient regularization of Newton method. *Mathematical Programming*, pages 1–39, 2025.

[20] N. Doikov and Y. Nesterov. Gradient regularization of Newton method with Bregman distances. *Mathematical Programming*, 204(1):1–25, 2024.

[21] J.-P. Dussault, T. Migot, and D. Orban. Scalable adaptive cubic regularization methods. *Mathematical Programming*, 207(1):191–225, 2024.

[22] P. Dvurechensky, P. Ostroukhov, K. Safin, S. Shtern, and M. Staudigl. Self-concordant analysis of Frank-Wolfe algorithms. In *ICML*, pages 2814–2824. PMLR, 2020.

[23] P. Dvurechensky, K. Safin, S. Shtern, and M. Staudigl. Generalized self-concordant analysis of Frank-Wolfe algorithms. *Mathematical Programming*, 198(1):255–323, 2023.

[24] W. Gao and D. Goldfarb. Quasi-Newton methods: Superlinear convergence without line searches for self-concordant functions. *Optimization Methods and Software*, 34(1):194–217, 2019.

[25] D. Goldfarb. The use of negative curvature in minimization algorithms. Technical report, Cornell University, 1980.

[26] S. Gratton, S. Jerad, and P. L. Toint. Yet another fast variant of Newton's method for nonconvex optimization. *IMA Journal of Numerical Analysis*, 45(2):971–1008, 2025.

[27] S. Hanzely, D. Kamzolov, D. Pasechnyuk, A. Gasnikov, P. Richtárik, and M. Takáč. A damped Newton method achieves global $o(1/k^2)$ and local quadratic convergence rate. In *NeurIPS*, pages 25320–25334, 2022.

[28] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[29] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[30] A. Krizhevsky. Learning multiple layers of features from tiny images. *Master's Thesis, University of Toronto*, 2009.

[31] J. Kuczyński and H. Woźniakowski. Estimating the largest eigenvalue by the power and Lanczos algorithms with a random start. *SIAM Journal on Matrix Analysis and Applications*, 13(4):1094–1122, 1992.

[32] D. Lee and S. Seung. Algorithms for non-negative matrix factorization. In *NeurIPS*, pages 535–541, 2000.

[33] J. Lee, S.-Y. Yun, and K.-S. Jun. Improved regret bounds of (multinomial) logistic bandits via regret-to-confidence-set conversion. In *AISTATS*, pages 4474–4482. PMLR, 2024.

[34] S. Łojasiewicz. Une propriété topologique des sous-ensembles analytiques réels. *Les équations aux dérivées partielles*, 117(87-89), 1963.

[35] Z. Lu. Randomized block proximal damped Newton method for composite self-concordant minimization. *SIAM Journal on Optimization*, 27(3):1910–1942, 2017.

[36] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Supervised dictionary learning. In *NeurIPS*, pages 1033–1040, 2008.

[37] J. Martens. New insights and perspectives on the natural gradient method. *The Journal of Machine Learning Research*, 21(146):1–76, 2020.

[38] J. Martens and R. Grosse. Optimizing neural networks with Kronecker-factored approximate curvature. In *ICML*, pages 2408–2417. PMLR, 2015.

[39] J. Martens and I. Sutskever. Training deep and recurrent networks with Hessian-free optimization. In *Neural Networks: Tricks of the Trade: Second Edition*, pages 479–535. Springer, 2012.

[40] G. P. McCormick. A modification of Armijo's step-size rule for negative curvature. *Mathematical Programming*, 13(1):111–115, 1977.

[41] K. Mishchenko. Regularized Newton method with global $\boldsymbol{\mathcal{O}(1/k^2)}$ convergence. *SIAM Journal on Optimization*, 33(3):1440–1462, 2023.

[42] J. J. Moré and D. C. Sorensen. On the use of directions of negative curvature in a modified Newton method. *Mathematical Programming*, 16(1):1–20, 1979.

[43] J.-J. Moreau. Proximité et dualité dans un espace hilbertien. *Bulletin de la Société mathématique de France*, 93:273–299, 1965.

[44] Y. Nesterov. Accelerating the cubic regularization of Newton's method on convex problems. *Mathematical Programming*, 112(1):159–181, 2008.

[45] Y. Nesterov. *Lectures on Convex Optimization*, volume 137. Springer, 2018.

[46] Y. Nesterov and A. Nemirovskii. *Interior-Point Polynomial Algorithms in Convex Programming.* SIAM, 1994.

[47] Y. Nesterov and B. T. Polyak. Cubic regularization of Newton method and its global performance. *Mathematical Programming*, 108(1):177–205, 2006.

[48] J. Nocedal and S. J. Wright. *Numerical Optimization.* Springer, 2006.

[49] R. Pascanu and Y. Bengio. Revisiting natural gradient for deep networks. In *ICLR*, 2014.

[50] B. T. Polyak. Gradient methods for solving equations and inequalities. *USSR Computational Mathematics and Mathematical Physics*, 4(6):17–32, 1964.

[51] R. T. Rockafellar. Monotone operators and the proximal point algorithm. *SIAM journal on control and optimization*, 14(5):877–898, 1976.

[52] A. Rodomanov and Y. Nesterov. Greedy quasi-Newton methods with explicit superlinear convergence. *SIAM Journal on Optimization*, 31(1):785–811, 2021.

[53] C. W. Royer, M. O'Neill, and S. J. Wright. A Newton-CG algorithm with complexity guarantees for smooth unconstrained optimization. *Mathematical Programming*, 180(1):451–488, 2020.

[54] C. W. Royer and S. J. Wright. Complexity analysis of second-order line-search algorithms for smooth nonconvex optimization. *SIAM Journal on Optimization*, 28(2):1448–1477, 2018.

[55] M. Schmidt, G. Fung, and R. Rosales. Fast optimization methods for l1 regularization: A comparative study and two new approaches. In *ECML*, pages 286–297. Springer, 2007.

[56] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[57] S. Strande, H. Cai, M. Tatineni, W. Pfeiffer, C. Irving, A. Majumdar, D. Mishin, R. Sinkovits, M. Norman, N. Wolter, et al. Expanse: Computing without boundaries: Architecture, deployment, and early operations experiences of a supercomputer designed for the rapid evolution in science and engineering. In *Practice and Experience in Advanced Research Computing 2021: Evolution Across All Dimensions*, pages 1–4. Association for Computing Machinery (ACM), 2021.

[58] J. Sun, Q. Qu, and J. Wright. A geometric analysis of phase retrieval. *Foundations of Computational Mathematics*, 18(5):1131–1198, 2018.

[59] T. Tang, K.-C. Toh, N. Xiao, and Y. Ye. A Riemannian dimension-reduced second-order method with application in sensor network localization. *SIAM Journal on Scientific Computing*, 46(3):A2025–A2046, 2024.

[60] M. J. Todd. *Minimum-Volume Ellipsoids: Theory and Algorithms*. SIAM, 2016.

[61] Q. Tran-Dinh, A. Kyrillidis, and V. Cevher. Composite self-concordant minimization. *The Journal of Machine Learning Research*, 16(1):371–416, 2015.

[62] Q. Tran-Dinh, T. Sun, and S. Lu. Self-concordant inclusions: A unified framework for path-following generalized Newton-type algorithms. *Mathematical Programming*, 177(1):173–223, 2019.

[63] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, et al. SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature Methods*, 17(3):261–272, 2020.

[64] S. Zagoruyko and N. Komodakis. Wide residual networks. In *BMVC*, pages 87–1. British Machine Vision Association, 2016.

[65] G. Zhang, C. Wang, B. Xu, and R. Grosse. Three mechanisms of weight decay regularization. In *ICLR*, 2019.

[66] Y. Zhang, C. Chen, T. Ding, Z. Li, R. Sun, and Z.-Q. Luo. Why transformers need ADAM: A Hessian perspective. In *NeurIPS*, pages 131786–131823, 2024.

[67] Y. Zhang and X. Lin. DISCO: Distributed optimization for self-concordant empirical loss. In *ICML*, pages 362–370. PMLR, 2015.

[68] Y.-J. Zhang and M. Sugiyama. Online (multinomial) logistic bandit: Improved regret and constant computation cost. In *NeurIPS*, pages 29741–29782, 2023.