International Journal of Education, Management Sciences and Professional Studies (IJEMPS)
Vol. 3 Issue 2, 2026 PRINT *ISSN:3115-5065* || E-ISSN:3115-5545
URL: https://journals.iempsglobal.org/index.php/IJEMPS

A META-LEARNING SCALING STRATEGY FOR LIMITED-MEMORY QUASI-NEWTON METHODS IN NONLINEAR OPTIMIZATION

[1]Adejoke Olumide DELE-ROTIMI
dele-rotimi.adejoke@bouesti.edu.ng (0000-0002-0120-8343)

&

[2]Temitayo Emmanuel OLAOSEBIKAN
olaosebikan.temitayo@bouesti.edu.ng (0000-0001-7625-9420)

[1&2]Department of Mathematical Sciences,
Bamidele Olumilua University of Education, Science and Technology, Ikere Ekiti

Abstract

Limited-memory quasi-Newton methods are among the principal numerical methods for smooth large-scale nonlinear optimization because they exploit curvature information without explicitly storing or factorizing a dense Hessian approximation. A persistent design choice in L-BFGS is the scalar used to initialize the implicit inverse-Hessian approximation at each iteration. This paper develops a structure-preserving meta-learning strategy in which this scalar is multiplied by a learned, strictly positive and bounded correction determined by inexpensive optimization-state statistics. The resulting method, denoted ML-QN, retains the standard L-BFGS curvature pairs, two-loop recursion, line-search mechanism, and limited-memory structure. The correction is represented by five scalar weights and is trained by a natural-evolution-strategies-style black-box procedure on a small set of training problems. The mathematical analysis establishes that a positive scaling of the initial inverse-Hessian approximation preserves positive definiteness; together with the usual curvature condition $s_k^T y_k > 0$*, this preserves the descent-direction mechanism of BFGS/L-BFGS. The numerical study uses nine held-out problems, five independent initializations per problem, memory length* $m = 10$*, and a primary budget of 3,000 iterations for L-BFGS and ML-QN. Across 45 paired problem-seed comparisons, ML-QN reduces iterations to convergence by a mean of 6.57% and function evaluations by approximately 5.1%; the Wilcoxon signed-rank tests give* $p = 1.5 \times 10^{-6}$ *for iterations and* $p = 1.22 \times 10^{-4}$ *for function evaluations. The wall-clock difference is not statistically significant (*$p = .524$*), with an approximately 3% average increase associated with the additional scalar feature calculations. The largest iteration reduction, 34.83%, occurs for the extended Powell singular problem. These results indicate that meta-learning can be used as a narrowly targeted scaling mechanism within a classical quasi-Newton framework without changing its memory order or basic curvature-update structure.*

Keywords: limited-memory quasi-Newton methods; L-BFGS; nonlinear optimization; Hessian scaling; adaptive scaling; meta-learning; numerical optimization; convergence analysis.

Introduction

Nonlinear optimization provides a mathematical framework for a broad class of problems in scientific computing, engineering design, statistics, machine learning, control, economics, and operations research. A large class of these problems can be written as minimize $f(x)$, $x \in \mathbb{R}^n$, where $f: \mathbb{R}^n \to \mathbb{R}$ is sufficiently smooth. When n is large, the explicit construction and factorization of a dense Hessian matrix can be prohibitively expensive. Quasi-Newton methods address this difficulty by constructing curvature information from successive iterates and gradients rather than forming the exact Hessian. The BFGS method, developed independently in closely related forms by Broyden (1970), Fletcher (1970), Goldfarb (1970), and Shanno (1970), is a central member of the quasi-Newton family. It is numerically stable and has a very effective "self-correcting properties" accounts for its superior performance in practice (Aderibigbe, Adebayo and Dele-Rotimi, 2015). Its local convergence properties were subsequently developed by Broyden, Dennis, and Moré (1973) and Dennis and Moré (1977). For large-scale problems, the limited-memory formulation avoids storing an $n \times n$ matrix. Instead, L-BFGS stores a fixed number m of curvature pairs and applies the inverse-Hessian approximation through a two-loop recursion (Nocedal, 1980; Liu & Nocedal, 1989; Nocedal & Wright, 2006). Although the main L-BFGS recursion is mathematically well established, several implementation choices remain heuristic. One such choice is the scalar γ_k used to initialize the implicit inverse-Hessian approximation. The classical rule is curvature based and is effective over a wide range of problems, but a fixed rule cannot explicitly condition its scaling on the evolving optimization trajectory. Earlier work on self-scaling variable-metric methods and conditioning provides mathematical motivation for adaptive scaling (Oren & Luenberger, 1974; Shanno & Phua, 1978), while modern learning-to-optimize research suggests that algorithmic components can be adapted from experience across related tasks (Andrychowicz et al., 2016; Li & Malik, 2017; Hospedales et al., 2022). The present work combines these two lines of reasoning in a deliberately restricted way. Rather than replacing the quasi-Newton update by a learned optimizer, only the scalar initialization of the inverse-Hessian approximation is modified. A five-dimensional feature vector summarizes recent progress, curvature, iteration progress, and memory occupancy. A five-parameter linear map is then exponentiated and clipped, producing a positive bounded multiplier. This design permits the numerical benefits of adaptation to be examined while preserving the central mathematical structure of L-BFGS.

The central numerical issue is the choice of the initial inverse-Hessian scaling in L-BFGS. The classical scalar depends primarily on the most recent curvature pair. Consequently, the same functional rule is applied regardless of whether the current trajectory is well conditioned,

strongly anisotropic, nonconvex, or approaching a singular solution. The problem addressed here is therefore whether a low-dimensional, learned correction to this scalar can improve convergence efficiency without changing the curvature-pair mechanism, memory requirement, or positive-definiteness properties of L-BFGS. The objective is to formulate, analyze, and numerically evaluate an adaptive scaling strategy for L-BFGS in which the classical scaling scalar is multiplied by a meta-learned positive correction. Particular attention is given to positive definiteness, computational complexity, convergence behaviour, and performance on ill-conditioned and nonconvex problems.

1. How can a meta-learned scaling correction be incorporated into L-BFGS while preserving its limited-memory structure?
2. Can the correction weights learned on a small distribution of optimization problems transfer to held-out problems?
3. Does adaptive scaling reduce the number of iterations required for convergence?
4. Does it reduce the number of function evaluations required by the line-search procedure?
5. Does the reduction in iterations translate into lower wall-clock computation time at the problem sizes considered?

H01: The meta-learned scaling strategy does not significantly reduce iterations to convergence relative to classical L-BFGS.

H02: The meta-learned scaling strategy does not significantly reduce cumulative function evaluations relative to classical L-BFGS.

H03: The meta-learned scaling strategy does not significantly reduce wall-clock computation time relative to classical L-BFGS.

Literature Review

State Features

The five components of φ_k are constructed from quantities already available during an L-BFGS iteration:

Feature	Definition / interpretation
φ_1	Constant bias term: 1.
φ_2	Progress signal based on tanh of the current-to-previous gradient-norm ratio minus one.
φ_3	Curvature signal based on tanh of one quarter of the logarithm of a curvature ratio derived from the most recent step.
φ_4	Logarithmic iteration count normalized by the logarithm of problem dimension.
φ_5	Current curvature-pair memory occupancy divided by the maximum memory length m.

The feature construction is intentionally low dimensional. It uses trajectory statistics rather than the full gradient or Hessian and therefore does not alter the asymptotic order of the L-BFGS computation.

Positive Definiteness and Descent

The adaptive multiplier is strictly positive. Hence, if $\gamma_k > 0$, then $\tilde{\gamma}_k > 0$ and $H_k^{(0)} = \tilde{\gamma}_k I$ is symmetric positive definite. If accepted curvature pairs satisfy $s_k^T y_k > 0$, the BFGS update preserves positive definiteness. Thus the learned component modifies the scale of the initial metric but does not introduce an indefinite initial matrix.

Computational Complexity

The standard L-BFGS two-loop recursion requires O(mn) arithmetic operations per iteration and O(mn) storage. The feature map uses a fixed number of scalar operations independent of n. Therefore

$$T_{L-BFGS}(n,m) = O(mn),$$

$$T_{ML-QN}(n,m) = O(mn) + O(1) = O(mn).$$

The method consequently retains the linear dependence on problem dimension that motivates limited-memory quasi-Newton methods.

Proposed Adaptive ML-QN Method

Algorithmic Structure

The method is designed so that the learned component is isolated from the curvature-pair generation and update. At iteration k, the current gradient is computed, the direction is obtained from the standard two-loop recursion, and the line search determines α_k. The only learned quantity is the scalar used in the initial inverse-Hessian approximation.

6. Initialize x_0, memory length m, and learned parameter vector ω.
7. Evaluate $g_k = \nabla f(x_k)$.
8. If a previous accepted curvature pair exists, compute the classical scale γ_k; otherwise use the initial scale specified by the implementation.
9. Construct φ_k from the current optimization trajectory.
10. Compute $m_k = \exp(\text{clip}(\omega^T \varphi_k, -4, 4))$ and set $\tilde{\gamma}_k = m_k \gamma_k$.
11. Apply the standard L-BFGS two-loop recursion using $H_k^{(0)} = \tilde{\gamma}_k I$ to obtain p_k.
12. Perform the same backtracking line search and curvature check used by the classical implementation.
13. Set $x_{k+1} = x_k + \alpha_k p_k$ and form s_k and y_k.
14. Accept the curvature pair when the curvature condition is satisfied; otherwise skip it.
15. Repeat until the convergence criterion is reached or the iteration budget is exhausted.

Meta-Training Procedure

The meta-objective is not treated as differentiable because small changes in ω can alter the accepted curvature pairs and consequently change the complete subsequent trajectory. The study therefore uses a natural-evolution-strategies-style black-box search. At each generation, twelve antithetic Gaussian perturbations of the current mean parameter vector are generated with standard deviation 0.30. Each candidate is evaluated on five training problems and three random seeds per problem. For a candidate ω, the training objective is the mean base-10 logarithm of the final gradient norm after a fixed 150-iteration budget, with a small penalty related to iteration count for runs that converge within the budget. The twelve objective values are rank transformed, and a fitness-weighted average perturbation is used to update the parameter vector with learning rate 0.15. Training proceeds for more than 40 generations. The final learned vector reported in the source study is

$$\omega = [0.216, 0.186, -0.176, -0.173, 0.401]^{T}.$$

The five entries correspond respectively to the bias, progress, curvature, normalized iteration, and memory-occupancy features. The training meta-objective decreased from −4.637 for the classical scaling (ω = 0) to −5.092 after meta-training.

Convergence Proposition

Proposition 1. *Let* $\gamma_k > 0$ and suppose the learned multiplier satisfies $0 < m_{min} \leq m_k \leq m_{max} < \infty$. Then the adaptive initialization $H_k^{(0)} = m_k \gamma_k I$ is symmetric positive definite. If each accepted curvature pair satisfies $y_k^T s_k > 0$, every subsequent BFGS inverse-Hessian approximation generated from that initialization is symmetric positive definite.

Proof. Since $\gamma_k > 0$ and $m_k \geq m_{min} > 0$, the product $m_k \gamma_k$ is strictly positive. Therefore, for any nonzero vector $v \in \mathbb{R}^n$,

$$v^T H_k^{(0)} v = m_k y_k \|v\|^2 > 0.$$

Hence $H_k^{(0)}$ is symmetric positive definite. For an accepted curvature pair, $y_k^T s_k > 0$ implies $\rho_k = 1/(y_k^T s_k) > 0$. The BFGS update is the sum of a congruence transformation of an SPD matrix and the positive semidefinite matrix $\rho_k s_k s_k^T$. Therefore H_{k+1} remains symmetric positive definite. By induction, the property holds for every subsequent accepted update. Consequently, the search direction $p_k = -H_k g_k$ satisfies $g_k^T p_k = -g_k^T H_k g_k < 0$ whenever $g_k \neq 0$, so p_k is a descent direction.

Implication for Local Convergence

The proposition establishes preservation of the positive-definite mechanism, not an unconditional new superlinear-convergence theorem. Under the standard smoothness, line-search, boundedness, and curvature assumptions used in the local convergence theory of BFGS/L-BFGS, a bounded positive scaling correction leaves the algorithm within the same class of positive-definite quasi-Newton iterations. Thus the proposed modification is structure

preserving. A full claim of superlinear convergence additionally requires the standard assumptions ensuring the relevant Dennis–Moré-type condition; those assumptions are not independently re-established by the numerical experiments in this study.

Implementation and Correctness Validation

The classical and ML-QN methods were implemented from scratch in Python using NumPy. They share a single two-loop-recursion L-BFGS routine, a backtracking line search enforcing an Armijo sufficient-decrease condition together with a weak curvature check, a memory length of m = 10, and identical convergence testing. The two implementations differ only in the scaling function supplied to the common routine. The implementation was checked against the independently engineered SciPy L-BFGS-B implementation on a moderately ill-conditioned quadratic and the extended Rosenbrock function.

Held-Out Test Problems

Problem	Family	N	Basis
Diagonal quadratic, $\kappa = 50$	Convex quadratic	1,000	Constructed
Diagonal quadratic, $\kappa = 500$	Convex quadratic	1,000	Constructed
Diagonal quadratic, $\kappa = 5{,}000$	Convex quadratic	1,000	Constructed
Diagonal quadratic, $\kappa = 50{,}000$	Convex quadratic	1,000	Constructed
Extended Rosenbrock	Nonconvex	1,000	Moré et al. (1981), problem 21
Extended Powell singular	Nonconvex; singular at solution	1,000	Moré et al. (1981), problem 13
Generalized Trigonometric	Nonconvex	200	Moré et al. (1981), problem 26
Logistic regression, Breast Cancer Wisconsin	Convex, real data	30	Pedregosa et al. (2011)
Logistic regression, Digits 3 vs. 8	Convex, real data	64	Pedregosa et al. (2011)

Experimental Protocol and Statistical Analysis

Each problem was solved from five independent random initializations (seeds 0–4), using small random perturbations of the standard starting point. L-BFGS and ML-QN used m = 10 and the same convergence criterion based on gradient norm. The primary iteration budget was 3,000 for the quasi-Newton methods and 15,000 for gradient descent. A secondary 1,500-iteration budget was used to examine constrained-budget convergence reliability. The recorded measures were iterations to convergence, cumulative function evaluations including line-search trial points, final objective value, final gradient norm, wall-clock time, and convergence status. Classical L-BFGS and ML-QN were paired at the problem-seed level. The three formal

hypotheses were assessed with two-sided Wilcoxon signed-rank tests on paired differences, using $\alpha = .05$. The aggregate tests were restricted to the 45 pairs for which both methods converged within the primary budget.

Methodology

Mathematical Preliminaries

Unconstrained Nonlinear Optimization

Let f be continuously differentiable and let x_k denote the kth iterate. Define the gradient by $g_k = \nabla f(x_k)$. A line-search quasi-Newton method generates

$$x_{k+1} = x_k + \alpha_k p_k, \quad p_k = -H_k g_k,$$

where H_k is an approximation to the inverse Hessian and $\alpha_k > 0$ is determined by a line-search procedure. The curvature pair associated with the step is

$$s_k = x_{k+1} - x_k, \quad y_k = g_{k+1} - g_k.$$

For a convex objective and a suitable line search, the curvature condition $s_k^T y_k > 0$ is the key condition used by BFGS to preserve positive definiteness.

BFGS and L-BFGS Updates

The inverse-Hessian BFGS update can be written as

$$H_{k+1} = (I - \rho_k s_k y_k^T) H_k (I - \rho_k y_k s_k^T) + \rho_k s_k s_k^T, \quad \rho_k = 1/(y_k^T s_k).$$

If H_k is symmetric positive definite and $y_k^T s_k > 0$, then H_{k+1} is also symmetric positive definite. L-BFGS obtains the action $H_k g_k$ without explicitly storing H_k. With memory length m, only the most recent m curvature pairs are retained, giving $O(mn)$ arithmetic and storage proportional to mn rather than n^2 (Liu & Nocedal, 1989; Nocedal & Wright, 2006).

Classical Hessian Scaling

At each iteration, the two-loop recursion is initialized with a multiple of the identity matrix:

$$H_k^{(0)} = \gamma_k I.$$

The classical L-BFGS scaling used in the study is the curvature-based quantity

$$\gamma_k = (s_{k-1}^T y_{k-1})/(y_{k-1}^T y_{k-1}),$$

whenever the preceding curvature pair is available and accepted. This choice provides a scalar estimate of the inverse curvature along the latest displacement.

Adaptive Scaling Formulation

The proposed method retains the classical L-BFGS recursion and replaces only γ_k by

$$\tilde{\gamma}_k = m_k \gamma_k,$$

where m_k is a learned positive multiplier. Let $\omega \in \mathbb{R}^5$ be the learned parameter vector and let $\varphi_k \in \mathbb{R}^5$ denote the feature vector. The multiplier is defined by

$$m_k = exp(clip(\omega^T \varphi_k, -4, 4)).$$

where $\text{clip}(z, -4,4)$ restricts z to the interval $[-4,4]$. $z = \omega^T \varphi_k$

Consequently, $e^{-4} \leq m_k \leq e^4$. The learned scaling is therefore bounded away from zero and infinity and cannot change the sign of γ_k.

Numerical Results

Overall Convergence Performance

Both L-BFGS variants converged on all 45 problem-seed combinations within the primary budget. Gradient descent converged only on the two best-conditioned diagonal quadratics and the two logistic-regression tasks under its larger budget; it did not converge within the specified budget on the two most ill-conditioned quadratics or the nonconvex test functions.

Problem	Method	Iterations M (SD)	Function evals M	Time (ms) M	Rate
Diag. quad., κ=50	Gradient descent	358.4 (1.67)	461.2	5.47	5/5
Diag. quad., κ=50	L-BFGS	74.2 (0.45)	82.2	6.08	5/5
Diag. quad., κ=50	ML-QN	72.2 (1.10)	80.0	5.94	5/5
Diag. quad., κ=500	Gradient descent	3,470.4 (33.52)	4,488.4	68.09	5/5
Diag. quad., κ=500	L-BFGS	239.4 (1.14)	252.2	18.87	5/5
Diag. quad., κ=500	ML-QN	232.2 (2.28)	235.8	21.86	5/5
Diag. quad., κ=5,000	Gradient descent	n/c	n/c	n/c	0/5
Diag. quad., κ=5,000	L-BFGS	773.2 (11.03)	800.0	61.35	5/5
Diag. quad., κ=5,000	ML-QN	748.4 (4.51)	759.0	63.06	5/5
Diag. quad., κ=50,000	Gradient descent	n/c	n/c	n/c	0/5
Diag. quad., κ=50,000	L-BFGS	2,491.0 (23.95)	2,560.6	202.05	5/5
Diag. quad., κ=50,000	ML-QN	2,436.6 (25.59)	2,456.4	217.84	5/5
Ext. Rosenbrock	Gradient descent	n/c	n/c	n/c	0/5
Ext. Rosenbrock	L-BFGS	342.2 (77.75)	392.6	36.41	5/5

Ext. Rosenbrock	ML-QN	304.0 (67.78)	337.2	34.94	5/5
Ext. Powell singular	Gradient descent	n/c	n/c	n/c	0/5
Ext. Powell singular	L-BFGS	1,375.8 (245.23)	1,459.0	302.72	5/5
Ext. Powell singular	ML-QN	876.4 (103.58)	956.2	207.73	5/5
Trigonometric	Gradient descent	223.0 (0.00)	256.0	8.24	5/5
Trigonometric	L-BFGS	56.0 (0.00)	63.0	5.22	5/5
Trigonometric	ML-QN	52.0 (0.00)	66.0	5.59	5/5
Logreg. breast cancer	Gradient descent	92.8 (0.45)	452.4	17.82	5/5
Logreg. breast cancer	L-BFGS	22.0 (1.00)	22.8	1.90	5/5
Logreg. breast cancer	ML-QN	22.4 (0.55)	25.6	2.30	5/5
Logreg. digits 3 vs. 8	Gradient descent	93.2 (0.45)	437.2	15.26	5/5
Logreg. digits 3 vs. 8	L-BFGS	24.4 (0.89)	25.6	1.96	5/5
Logreg. digits 3 vs. 8	ML-QN	23.2 (0.84)	24.2	1.94	5/5

Meta-Training Outcome

Over more than 40 generations, the mean training meta-objective improved from −4.637 under the classical scaling to −5.092 using the learned parameter vector. The positive bias and memory-occupancy coefficients indicate that the learned map tended to increase scaling as the memory buffer filled, whereas the negative curvature coefficient provided a compensating reduction when the most recent curvature signal was large.

Iterations to Convergence

Across the 45 paired comparisons, the two-sided Wilcoxon signed-rank test for iterations gave $W = 75.0$ and $p = 1.5 \times 10^{-6}$. The mean reduction in iterations was 6.57%. Thus the data show a statistically significant paired difference in iteration counts between ML-QN and classical L-BFGS over the evaluated suite.

Function Evaluations

For cumulative function evaluations, the Wilcoxon signed-rank statistic was $W = 177.5$ with $p = 1.22 \times 10^{-4}$. The mean reduction was approximately 5.1%. Because function evaluations include trial points generated by the common backtracking line search, the reduction broadly follows the reduction in iterations.

Wall-Clock Time and Constrained-Budget Reliability

The wall-clock comparison did not produce a statistically significant difference: $W = 460.0$ and $p = .524$. Across the paired runs, ML-QN was associated with an approximately 3% increase in wall-clock time. The most direct interpretation is implementation-level overhead: the learned feature map requires additional scalar calculations, including logarithmic and hyperbolic-tangent evaluations, and these operations are not always negligible at $n \leq 1{,}000$. Under the secondary 1,500-iteration budget, the extended Powell singular problem produced a descriptive reliability difference. Classical L-BFGS converged on four of five initializations (80%), whereas ML-QN converged on all five (100%). Because the comparison contains only five initializations, this reliability result is reported descriptively rather than treated as a formal hypothesis test.

Problem-Level Iteration Reduction

Problem	% reduction in iterations
Diagonal quadratic, κ=50	2.70
Diagonal quadratic, κ=500	3.00
Diagonal quadratic, κ=5,000	3.19
Diagonal quadratic, κ=50,000	2.17
Extended Rosenbrock	3.19
Extended Powell singular	34.83
Generalized Trigonometric	7.14
Logistic regression, breast cancer	−1.99
Logistic regression, digits 3 vs. 8	4.90
Overall mean, 45 paired runs	6.57

Positive values indicate fewer iterations for ML-QN. The largest improvement occurs on the extended Powell singular problem, whose Hessian is singular at the solution. The breast-cancer logistic-regression problem is the only case with a small negative mean change, indicating that the learned correction need not improve every easy, well-conditioned problem.

Summary of Hypothesis Tests

Hypothesis	Test	Statistic	p-value	Decision
H01: iterations to convergence	Wilcoxon signed-rank	$W = 75.0$	1.5×10^{-6}	Reject H01
H02: function evaluations	Wilcoxon signed-rank	$W = 177.5$	1.22×10^{-4}	Reject H02
H03: wall-clock time	Wilcoxon signed-rank	$W = 460.0$	.524	Fail to reject H03

Note. All tests are two-sided, $n = 45$ paired problem-seed comparisons, $\alpha = .05$.

Discussion

Interpretation from a Numerical Optimization Perspective

The principal numerical result is that a very small modification of the L-BFGS metric initialization can alter convergence efficiency while leaving the curvature-pair recursion unchanged. This is important mathematically because the proposed method does not attempt to learn a replacement for the quasi-Newton approximation. Instead, it learns a scalar adjustment to an already curvature-informed quantity. The concentration of the largest gain on the extended Powell singular problem is consistent with the intended role of adaptive scaling. On relatively regular quadratics, the classical curvature estimate already provides a useful metric scale, leaving limited scope for correction. By contrast, strongly ill-conditioned and nonconvex trajectories can produce rapidly changing curvature information. The learned map conditions the scalar on progress and curvature statistics and therefore has more opportunity to modify the initial metric scale in such cases. The results also demonstrate that iteration efficiency and computational efficiency are not identical. The significant reductions in iterations and function evaluations did not produce a statistically significant reduction in wall-clock time. The feature calculations introduce a fixed per-iteration overhead, and the reported NumPy implementation was not optimized as a production numerical library.

Relation to Learning-to-Optimize

The method occupies a deliberately narrow position within the learning-to-optimize literature. Learned optimizers such as those of Andrychowicz et al. (2016), Li and Malik (2017), Chen et al. (2017), and Wichrowska et al. (2017) learn substantially more of the update rule, often using high-dimensional recurrent parameterizations. ML-QN instead learns five scalar weights and retains the deterministic quasi-Newton machinery. This restriction reduces the flexibility of the learned component but improves interpretability and makes the mathematical effect of the modification easier to isolate.

Preservation of the Quasi-Newton Structure

The proposed correction does not alter s_k or y_k, does not alter the acceptance rule for curvature pairs, does not change the two-loop recursion, and does not increase the memory length. Its role is therefore analogous to a learned scaling heuristic rather than a learned optimizer in the broader model-replacement sense. This distinction is central to the mathematical interpretation of the results.

Limitations

First, the maximum dimension in the study is 1,000, while the two real-data logistic-regression problems use only 30 and 64 features. The evidence therefore concerns medium-to-large numerical optimization problems rather than modern deep-learning models with millions or billions of variables. Second, meta-training used only five training problems and three random seeds per problem, with a single evolution-strategies search run. A variance estimate over independent meta-training runs is consequently unavailable. Third, the feature and parameter

spaces were deliberately restricted to five features and five weights. Alternative features, nonlinear maps, or adaptive memory-length policies were not compared. Fourth, the aggregate Wilcoxon tests combine heterogeneous problems whose iteration counts differ substantially. Although rank-based testing reduces sensitivity to raw magnitude, problem-specific reductions remain important for interpretation. Finally, wall-clock measurements were obtained from a single, unoptimized NumPy implementation and should not be interpreted as production-level software benchmarks.

Conclusion

This paper reformulated the original learning-oriented study as a mathematical and numerical optimization investigation. The resulting method, ML-QN, introduces a meta-learned positive bounded multiplier into the classical L-BFGS scaling rule while leaving the quasi-Newton curvature pairs, two-loop recursion, line search, and limited-memory structure unchanged. The mathematical analysis shows that the learned multiplier preserves positive definiteness of the initial inverse-Hessian approximation and, under the usual curvature condition, preserves the positive-definite BFGS update mechanism and descent direction. The numerical evidence shows a mean 6.57% reduction in iterations and approximately 5.1% reduction in function evaluations across 45 paired problem-seed comparisons, with statistically significant paired differences for both measures. The wall-clock difference is not statistically significant, and the learned method incurs an approximately 3% average time increase in the reported implementation. The results therefore support a restrained conclusion: a low-dimensional learned correction can be used to adapt a classical quasi-Newton scaling heuristic while retaining the mathematical and computational structure that makes L-BFGS attractive for large-scale nonlinear optimization. The largest observed improvement occurs on the extended Powell singular problem, whereas the easiest logistic-regression problem shows a small regression. The method should consequently be viewed as a structure-preserving adaptive scaling strategy rather than a universal replacement for classical quasi-Newton optimization.

Recommendations for Further Research

1. Evaluate the scaling strategy at substantially larger dimensions to determine how the fixed feature- evaluation overhead behaves relative to the O(mn) L-BFGS computation.
2. Repeat meta-training with multiple random seeds and a broader, more diverse training distribution to quantify transfer variability.
3. Investigate additional curvature and trajectory features, while retaining positivity and boundedness constraints on the learned multiplier.
4. Study learned adaptation of other quasi-Newton design parameters, including memory length and curvature-pair damping or acceptance thresholds.
5. Extend the method to stochastic optimization, where gradient noise changes the reliability of curvature pairs.
6. Compare black-box meta-training with differentiable unrolled optimization and other meta-optimization strategies.

7. Develop formal convergence results for adaptive bounded scaling under explicit Dennis–Moré-type assumptions and, where appropriate, stochastic curvature conditions.
8. Benchmark the method in optimized numerical libraries and report performance profiles in addition to aggregate paired tests.

References

Andrychowicz, M., Denil, M., Gómez, S., Hoffman, M. W., Pfau, D., Schaul, T., Shillingford, B., & de Freitas, N. (2016). Learning to learn by gradient descent. *Advances in Neural Information Processing Systems, 29, 3981–3989.*

Aderibigbe, F. M., Adebayo, K. J., & Dele-Rotimi, A. O. (2015). On quasi-Newton method for solving unconstrained optimization problems. *American Journal of Applied Mathematics, 3(2), 47–50.*

Broyden, C. G. (1970). The convergence of a class of double-rank minimization algorithms 1. General considerations. *IMA Journal of Applied Mathematics, 6(1), 76–90.*

Broyden, C. G., Dennis, J. E., & Moré, J. J. (1973). On the local and superlinear convergence of quasi-Newton methods. IMA Journal of Applied Mathematics, 12(3), 223–245.

Chen, Y., Hoffman, M. W., Colmenarejo, S. G., Denil, M., Lillicrap, T. P., Botvinick, M., & de Freitas, N. (2017). Learning to learn without gradient descent by *gradient descent. Proceedings of the 34th International Conference on Machine Learning, 70, 748–756.*

Dennis, J. E., & Moré, J. J. (1977). Quasi-Newton methods, motivation and theory. *SIAM Review, 19(1), 46–89.*

Fletcher, R. (1970). A new approach to variable metric algorithms. The Computer Journal, 13(3), 317–322.

Hospedales, T. M., Antoniou, A., Micaelli, P., & Storkey, A. J. (2022). Meta-learning in neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 44(9), 5149–5169.*

Li, K., & Malik, J. (2017). Learning to optimize. Proceedings of the 5th International Conference on Learning Representations.

Liu, D. C., & Nocedal, J. (1989). On the limited memory BFGS method for large scale optimization. *Mathematical Programming, 45(1–3), 503–528.*

Moré, J. J., Garbow, B. S., & Hillstrom, K. E. (1981). Testing unconstrained optimization software. *ACM Transactions on Mathematical Software, 7(1), 17–41.*

Nocedal, J. (1980). Updating quasi-Newton matrices with limited storage. *Mathematics of Computation, 35(151), 773–782.*

Nocedal, J., & Wright, S. J. (2006). Numerical optimization (2nd ed.). Springer.

Oren, S. S., & Luenberger, D. G. (1974). Self-scaling variable metric (SSVM) algorithms I: Criteria and sufficient conditions for scaling a class of algorithms. *Management Science, 20(5), 845–862.*

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research, 12, 2825–2830.*

Powell, M. J. D. (1978). Algorithms for nonlinear constraints that use Lagrangian functions. *Mathematical Programming, 14(1), 224–248.*

Shanno, D. F. (1970). Conditioning of quasi-Newton methods for function minimization. *Mathematics of Computation, 24(111), 647–656.*

Shanno, D. F., & Phua, K. H. (1978). Matrix conditioning and nonlinear optimization. *Mathematical Programming, 14(1), 149–160.*

Wichrowska, O., Maheswaranathan, N., Hoffman, M. W., Colmenarejo, S. G., Denil, M., de Freitas, N., & Sohl-Dickstein, J. (2017). Learned optimizers that scale and generalize. Proceedings of the 34th International Conference on Machine Learning, 70.