A Study on Truncated Newton Methods for Linear Classification

Leonardo Galli, Chih-Jen Lin

Abstract—Truncated Newton (TN) methods have been a useful technique for large-scale optimization. Instead of obtaining the full Newton direction, a truncated method approximately solves the Newton equation with an inner Conjugate Gradient (CG) procedure (TNCG for the whole method). These methods have been employed to efficiently solve linear classification problems. But even in this deeply studied field, various theoretical and numerical aspects were not completely explored. The first contribution of this work is to comprehensively study the global and local convergence when TNCG is applied to linear classification. Because of the lack of twice differentiability under some losses, many past works cannot be applied here. We prove various missing pieces of theory from scratch and clarify many proper references. The second contribution is to study the termination of the CG method. For the first time when TNCG is applied to linear classification, we show that the inner stopping condition strongly affects the convergence speed. We propose adaptive stopping criteria that are combined with that of preconditioning. We discuss how convergence theory is affected by preconditioning and finally propose a new state-of-the-art preconditioned TNCG.

Index Terms—Truncated Newton, Conjugate gradient, Linear classification, Truncation criteria, Preconditioning.

I. INTRODUCTION

In this work we focus on the problem of estimating the model parameter \( w \) of a linear classifier. In particular, two widely used models are logistic regression and linear Support Vector Machines (SVM). The problem of training both these models might be written as follows

\[
\min_w f(w) = \frac{1}{2} w^T w + C \sum_{i=1}^l \xi_i (y_i w^T x_i),
\]

(1)

where \((y_i, x_i), i = 1, ..., l\) are the training data, \(y_i = \pm 1\) is the label, \( x_i \in \mathbb{R}^n \) is a feature vector, \( w^T w / 2 \) is the regularization term, \( C > 0 \) is a regularization parameter and \( \xi_i (y_i w^T x_i) \) is any LC\(^1\) (continuously differentiable with locally Lipschitz continuous gradient) convex loss function. With \( w^T w, f \) is a LC\(^1\) strongly convex function and the minimum \( w^* \) of \( f \) exists and is unique. The following two losses respectively correspond to logistic regression (C\(^2\), i.e. twice continuously differentiable) and \( l_2 \)-loss linear SVM (LC\(^1\)),

\[
\begin{align*}
\xi_{LR}(y w^T x) &= \log(1 + \exp(-y w^T x)) \\
\xi_{L2}(y w^T x) &= (\max(0, 1 - y w^T x))^2.
\end{align*}
\]

(2)

In this work we focus on the Truncated Newton (TN) method for solving large scale optimization problems that might be written as in (1). In this field, solving the Newton equation to obtain a direction is often very challenging because of the dimensionality of the system. A truncated method approximately solves the Newton equation with an internal iterative procedure specific for a linear system of equations. In [1] and [2] they employ a Conjugate Gradient (CG) method [3] to avoid the storage of the Hessian in solving the Newton equation. The resulting method is thus called TNCG.

Our main concern is the convergence of TNCG, both from the numerical and theoretical point of view. For this reason, we brought into question various aspects of TNCG methods for (1) that till now were taken for granted. The first contribution is to comprehensively study the global and local theoretical convergence of TNCG. In particular, when \( f \notin C^2 \) and for \( \xi_{L2} \), we find out that most of the proofs are not at hand and some of them are not even existing. We thus obtained global and local Q-SuperLinear (Q-SL) convergence. Moreover, we proved that the TNCG is a special case of the general common-directions framework from [4], so some nice theoretical properties follow.

The second finding was that the choice of the CG inner termination (or truncation) criterion has never been addressed for linear classification problems. In fact, we show that the convergence speed of some widely used machine learning software can be improved by very simple modifications on this stopping rule. Through experimental and empirical illustrations we thus identify that an adaptive criterion based on checking the quadratic model of the current Newton iteration is robust and effective for large scale linear classification. Finally, to obtain local Q-SL convergence for all the different truncation criteria, various results have been proved.

The third contribution is that of combining the study on truncation criteria with that on preconditioning. It is well known that for ill-conditioned linear systems the Preconditioned CG (PCG) can be helpful to improve the rate of convergence of the original CG. The idea is to pre-multiply the linear system by a preconditioner matrix that will improve its condition. Let us call TNPCG the complete method. In this work, we first discuss how the convergence proofs are affected by the use of preconditioners. Then, we integrate our numerical analysis on truncation criteria with that conducted in [5] on preconditioning. They find out that a mixed approach between the identity matrix and a diagonal preconditioner was able to improve convergence speed in the majority of the data sets employed. Thanks to this new extensive numerical analysis on the combination between the truncation criteria and the preconditioning, we are able to propose a highly robust and effective TNPCG method for linear classification.

This paper is organized as follows. In Section II we go...
over past works from the literature that have some similarities with ours. In Section III, we review TNCG for large scale linear classification. Section IV gives a detailed analysis on the theoretical aspects related to TNCG, including both global and local convergence. In Section V we first discuss the importance of having a robust inner stopping criterion. Then we investigate some criteria and prove their theoretical local convergence. In Section VI we first show how to apply preconditioning in our case, which approach has been employed, and how to obtain local and global convergence in the preconditioned case. In Section VII we conduct extensive experiments on termination criteria, while in Section VIII we combine them with the use of preconditioning. Conclusions are given in Section IX. Proofs of some main theorems are in the appendix, while the rest of the proofs and of the experiments are enclosed in the supplementary materials available at https://www.csie.ntu.edu.tw/~cjlin/papers/tncg/. Programs used for experiments are available at the same page, while the proposed method has been incorporated into the software LIBLINEAR (version 2.40 and after).

II. RELATED WORKS

TNCG is a classical optimization method so the convergence theory has been deeply investigated. One would expect that, when TNCG is applied to linear classification, all the theorems are easily accessible. We instead found out that this is not the case. Theoretical properties (especially the fast local convergence ones) may either be partially covered in some works or be disjointedly presented in various paper fragments. The original Q-SL convergence result for TN methods was given in [6]. In this result, $f$ is assumed to be $C^2$, which is true in the case of the logistic loss, but not true for the $l2$ loss (2). This loss is only $LC^1$, so the Hessian of $f$ does not exist and we should instead refer to the generalized Hessian $\partial N \ f$ in the sense of Clarke [7] (see Section IV-B for details). Nonetheless, in some studies on TNCG for linear classification (for instance [2]), theory has been studied for the $\ell_{1,2}$ loss, but was ignored for the $\ell_{1,2}$ loss. In the field of linear classification, many following works focused on various aspects of the TNCG (e.g. hyperparameter selection [8], globalization techniques [9], preconditioning [5]), but never on the fast convergence result for not twice continuous differentiable losses. See [10] for a detailed survey on the topic.

The first Q-SL convergence proof for TN methods in the case of a nonsmooth system of equations was given in [11]. Such a result is useful in our case since the arising Newton equation is also a nonsmooth system, but the convergence in [11] is not covering the whole theory since their theorems are only given for a non-globalized TN method (i.e. the TNCG without line search, see (8) and the discussion above it). A fast convergence result for a globalized TNCG is instead given in Theorem 5.3 of [12]. This theorem is applied on functions with globally Lipschitz gradient, but here we develop theory to cover the more general situation of functions whose gradient is only locally Lipschitz. Besides this main distinction, [12] differs from us in the following aspects. They need to prove the regularity of the generalized Hessian, while we employ the more standard and general BD-regularity (see Lemma 1 of Appendix A). Next, in our local convergence theory, we provide a rigorous treatment by proving first the Q-SL result for a generic converging sequence (see Lemma 3 of Appendix A), while they directly assume that such a sequence is generated by the globalized TNCG. To conclude this section we can also mention the recent paper [13] on linear SVM, even if the theory therein is not very clear. In their Theorem 1 they cite Theorem 3.2 of [14], that is the first fast convergence result for a (non-truncated) semismooth Newton method, but the truncated situation is not discussed.

III. THE TNCG FOR LINEAR CLASSIFICATION

At the current iterate $w_k$, where $k$ is the iteration index, a Newton method finds an update direction by minimizing the following second-order approximation

$$Q_k(s) = \nabla f(w_k)^T s + \frac{1}{2} s^T \nabla^2 f(w_k) s. \quad (3)$$

As $f \in LC^1$, $\nabla^2 f$ is the generalized Hessian; see Section IV-B. Because of the convexity of (1), this minimization is equivalent to solving the Newton equation

$$\nabla^2 f(w_k) s = -\nabla f(w_k). \quad (4)$$

The gradient and the Hessian of $f(w)$ are

$$g := \nabla f(w) = \nabla f(w) + C \sum_{i=1}^l \xi_i (y_i w^T x_i) y_i x_i,$$

$$H := \nabla^2 f(w) = I + C X^T D X,$$

where $I$ is the identity matrix, $X = [x_1, \ldots, x_l]^T$ is the data matrix and $D$ is a diagonal matrix with $D_{ii} = \xi_i (y_i w^T x_i)$.

The linear system (4) is difficult to solve because of the possible high dimensionality of $w_k$. Thus the CG method is applied to avoid the explicit forming of the Hessian and employing instead the following Hessian-vector products

$$H s = (I + C X^T D X) s = s + C X^T (D(X s)). \quad (6)$$

Therefore, each Newton iteration (called an outer iteration from now on) involves an inner iterative procedure of some CG steps, each of which conducts a Hessian-vector product.

Unfortunately, accurately solving the Newton equation (4) may expensively require many CG steps. As we will show later, CG steps are the bottleneck of the Newton method to solve (1). To reduce the number of CG steps, TN methods are applied to solve the Newton equation approximately. This approximation is controlled by the CG inner termination criterion. We will show that this choice has a direct and great impact on the convergence speed. Assume $s_k^1, s_k^2, \ldots$ are inner CG iterates. CG is stopped at a step $j$ whenever $s_k^j$ satisfies a truncation rule as

$$\text{ratio}(s_k^j) \leq \eta_k, \quad (7)$$

where the left side is the actual condition to be checked, usually a ratio between two terms, and $\eta_k \in (0, 1)$ is the forcing sequence. The resulting $s_k^j$ is then called $s_k$.

To ensure the convergence of TNCG, by following most existing optimization methods, a globalization procedure is needed. Usually this means to find a suitable step size $\omega_k$.
so that \( \omega_k s_k \) is used to update the current iterate \( w_k \). Two major globalization techniques are line search and Trust-Region (TR). While TR may be more stable, it is more sophisticated (see final discussion in Section VIII). Now assume we therefore consider the easiest line search, Armijo (8). It finds the largest \( \omega_k \in \{1, \delta, \delta^2, \ldots\} \) with \( \delta \in (0, 1) \) such that the function value is sufficiently decreased, satisfying the following condition, with \( \gamma \in (0, 1) \),

\[
f(w_k + \omega_k s_k) \leq f(w_k) + \gamma \omega_k g_k^T s_k. \tag{8}
\]

Our implementation of TNCG is given in Algorithm 1.

\begin{algorithm}
\caption{Truncated Newton Conjugate Gradient}
\begin{algorithmic}[1]
\State \textbf{Input:} \( w_1 \in \mathbb{R}^n \) starting point
\For {\( k = 1, 2, \ldots \)}
\State compute the direction \( s_k \) by approximately solving (4) with a CG method, until (7) is satisfied
\State compute a step length \( \omega_k \) by an Armijo line search technique (8)
\State \( w_{k+1} = w_k + \omega_k s_k \)
\EndFor
\end{algorithmic}
\end{algorithm}

We now discuss the complexity of the whole procedure. From (6) the cost per outer iteration is roughly

\[
O(nl) \times (\text{#CG steps} + 2) + \text{cost of deciding the step size}, \tag{9}
\]

where \( O(nl) \) is the cost associated with each evaluation of function, gradient or Hessian-vector product. If \( f \) is sparse, the term \( O(nl) \) above might be replaced by the number of non-zero elements (\( \text{#nnz} \)) in the matrix \( X \). In most optimization methods the cost of deciding the step size is relatively smaller, so the complexity is proportional to the total number of CG steps. In fact, it has been shown in Section 2.1 of [9] (see also Section A.3 of the supplementary) that Armijo requires \( O(l) \) operations for each new \( \omega \). This means that the cost of deciding the step size is not the computational bottleneck.

IV. GLOBAL AND LOCAL CONVERGENCE OF TNCG

In this section we relax \( f \) to be any \( f \in \mathcal{L}^1 \), rather than the particular form in (1). Assumptions needed on \( f \) will be clarified in each theorem statement.

A. Global Convergence by Treating TNCG as a Common-Directions Algorithm

At the current iteration \( k \), a common-directions algorithm computes \( s_k \) by combining \( m \) different directions \( \{d_k^1, \ldots, d_k^m\} \) to minimize the quadratic approximation of \( f \). In the field of empirical risk minimization, in [4] they developed a framework that provides global and local convergence results for common-directions methods that satisfy Assumption 1. Here we will show for the first time that even the TNCG method can be seen as a special common-directions algorithm.

Assumption 1. At each iteration \( k \) a common-directions algorithm computes a direction \( s_k \) such that

\[
\min_{\alpha_k} g_k^T s_k + \frac{1}{2} s_k^T B_k s_k \quad \text{s.t. } s_k = P_k \alpha_k, \tag{10}
\]

where \( P_k := [d_k^1] \ldots [d_k^m] \in \mathbb{R}^{n \times m} \) and \( g_k \in \{d_k^1, \ldots, d_k^m\} \). The iterate update is \( w_{k+1} = w_k + \omega_k s_k \), where \( \omega_k \) is the step-size computed by Armijo (8). In addition, \( B_k \) is a positive definite matrix and is bounded, i.e. there exist two constants \( M_1, M_2 > 0 \) such that \( M_1 \geq \|B_k\| \geq M_2 \forall k \).

Note that their result is rather general since \( B_k \) does not need to be the Hessian, but can be any bounded positive definite matrix. To obtain the global convergence result, the gradient must be one of the common-directions. In Theorem 3.2 and Corollary 3.1 of [4] they show the following theorem.

**Theorem 1.** Let \( f \in \mathcal{L}^1 \) with a positive Lipschitz constant and \( \{w_k\} \) be generated by a method that satisfies Assumption 1. Assume that the following level set

\[
L_1 := \{w \in \mathbb{R}^n : f(w) \leq f(w_1)\} \tag{11}
\]

is compact. Then the minimum of the norm of gradients of the iterates vanishes at an \( O(1/e) \) rate, i.e.

\[
\min_{0 \leq j \leq k} \|g_j\| = \mathcal{O}(1/\sqrt{k+1}) \text{ and }
\]

\[
\lim_{k \to \infty} \|g_k\| = 0. \tag{12}
\]

In addition, if \( f \) is strongly convex the function values linearly converge, i.e. it takes \( \mathcal{O}(\log(1/e)) \) to get an \( \epsilon \)-accurate solution satisfying \( f(w_k) - f(w^*) \leq \epsilon \).

Note that the framework from [4] assumes that \( m \) is fixed, but all their results are still valid for any bounded \( m \) (see supplementary for a discussion). Since CG method terminates in a number of steps bounded by the dimensionality of the linear system (4), the choice of the termination criteria (7) will not affect results contained in this subsection.

Now, to show that Algorithm 1 satisfies Assumption 1 we first need to recall that the CG method is designed to minimize a strictly convex quadratic function, as in (10). In particular, this is obtained by combining a set of directions \( \{d_k^1, \ldots, d_k^m\} \) that are conjugate with respect to \( B_k \), i.e. for any \( d_k^i, d_k^j \) we have \( d_k^i^T B_k d_k^j = 0 \) (see Lemma II of the supplementary and [15]). Once \( s_k \) is obtained, Algorithm 1 is then employing an Armijo line search along \( s_k \). Now to satisfy the rest of the assumptions on \( f \) and \( B_k \) we refer to \( f \) as defined in (1).

- In [4], they assume \( f \) to have globally Lipschitz gradient, as in the case of \( f \) defined in (1). For \( f \in \mathcal{L}^1 \), this assumption is not needed if we have assumed that the level set (11) is compact. In this case, \( \nabla f \) is globally Lipschitz on \( L_1 \). In [4] they also assume that \( f \) is bounded from below. We have this from a compact \( L_3 \).

- Since in Algorithm 1 we approximately solve (4) employing the Hessian (or the generalized one) from (5), for the two losses in (2), we have that \( H_k \) is bounded (see Lemma I of supplementary). This also imply that the minimum and maximum eigenvalue of \( H_k \) (respectively \( \lambda_{\min}(H_k) \) and \( \lambda_{\max}(H_k) \)) are bounded. In particular, from (5) we have that there exist two constants, \( M_1, M_2 > 0 \), such that

\[
M_2 = 1 \leq \lambda_{\min}(H_k) \leq \lambda_{\max}(H_k) \leq 1 + C \lambda_{\max}(X^T DX) = M_1. \tag{13}
\]

- From CG method’s details we have that \( d_k^1 = -g_k \).
• $f$ in (1) is strongly convex by definition. Further, with the
regularization term $w^Tw$ and $\xi(\cdot) \in \text{LC}^1$, $f$ in (1) is in \text{LC}^1
with a positive Lipschitz constant.
Thus, Assumption 1 and the assumptions of Theorem 1 are
satisfied, which means that Algorithm 1 is globally convergent
and the sequence \{\$f(w_k)\$\} is linear convergent to $f(w^\ast)$.
Global convergence of Algorithm 1 can also be proved by
following a more classical way (see Proposition III and
Theorem IV from supplementary). First we need to prove that
CG method obtains a direction $s_k$ for which there exist two
constants $a_1 > 0$ and $a_2 > 0$ such that
$$g_k^Ts_k \leq -a_1\|g_k\|^2$$
and $\|s_k\| \leq a_2\|g_k\|$.
(14)
Then, from Armijo line search properties we can obtain (12).

B. Q-Superlinear Local Convergence

Since $\nabla f$ is locally Lipschitz, from Rademacher’s theorem
it is differentiable almost everywhere [7]. We indicate by $D_{\nabla f}$
the set of points in which $\nabla f$ is differentiable. For any $w$
we define the B-subdifferential of $\nabla f$ at $w$ as the nonempty
compact set
$$\partial_B\nabla f(w) := \{H \in \mathbb{R}^{n \times n} : H = \lim_{j \to \infty} \nabla^2 f(j)\}.$$ 

Then the Clarke generalized differential $\partial \nabla f$ (or equivalently
the generalized Hessian) at $w$ is the convex hull of $\partial_B \nabla f$.
We say that $\nabla f$ is semismooth at $w$ if the limit
$$\lim_{s_j \to s, t_j \to 0^+} H_j s_j \exists \forall H_j \in \partial \nabla f(w + t_j s_j).$$

If $\nabla f$ is semismooth it can be proved that the above limit is
equal to the directional derivative of $\nabla f$ in the
direction $s$. Semismooth functions are a particular class of
locally Lipschitz continuous function for which generalized
differentials define a legitimate Newton approximation scheme
(see Chapter 7 of [16]). The class of semismooth functions is
very broad, in particular $\nabla f$ is semismooth for both losses
defined in (2) because they both have piecewise-smooth gradient.
In addition, we say that $\nabla f$ is BD-regular at $w$
if all elements in $\partial \nabla f(w)$ are nonsingular. Note that strong
convexity implies BD-regularity.

In this subsection, we study local convergence by assuming
that the ratio employed in the stopping rule (7) is
$$\text{ratio} = \frac{\|g_k + H_k s_k\|}{\|g_k\|} = \|r\|$$
where $r := -g_k - H_k s_k$ (15)
is the residual maintained though the CG procedure (indices $j$
and $k$ omitted) and $H_k \in \partial \nabla f(w_k)$ is the generalized Hessian
considered in Algorithm 1. This condition of checking the ratio
between the residual and the right-hand side of the Newton
equation is probably the most commonly used inner stopping
criterion [15]. In Section V we will show all the other criteria
and how to obtain the same local results.

Now to obtain Q-SL local convergence, we first need to
prove that the non-globalized TNCG is able to achieve Q-SL
local convergence (see Theorem 3 of [11]).

Proposition 1. Let $\nabla f$ be semismooth and BD-regular. Let
\{\$w_k\$\} be any sequence generated by Algorithm 1 without line
search (i.e. $w_{k+1} = w_k + s_k$) and via the truncation rule of
using (15). If \{\$w_k\$\} converges to a critical point $w^\ast$, then we have both:
(1) there exist two constants $\eta > 0$ and $\delta > 0$ such that if
$\|w_1 - w^\ast\| \leq \delta$ and $\eta_k \leq \eta$ \forall $k$, then \{\$w_k\$\} is Q-Linear (Q-L)
convergent to $w^\ast$.
(2) moreover, if $\eta_k \to 0$, then \{\$w_k\$\} is Q-SL convergent to $w^\ast$.

Once we prove this, we can exploit a very general result
for semismooth functions (see Theorem 3.2 of [17]). It ensures
that any line search based algorithm will eventually accept
the initial step-size $\omega_k = 1$ if the same algorithm without
the line search step achieves Q-SL convergence, as just obtained
from Proposition 1.

Theorem 2. Let $f$ be strongly convex and $\nabla f$ be semismooth.
Let $\mathcal{L}_1$ defined in (11) be compact. Let \{\$w_k\$\} be generated by
Algorithm 1 with $\gamma \in (0, \frac{1}{2})$ and $\eta_k \to 0$. Then:
(1) there exists $k$ such that (8) is satisfied with $\omega_k = 1/k \geq k$.
(2) \{\$w_k\$\} is Q-SL convergent to a critical point $w^\ast$.

Proofs of theorems in this section are in Appendix A. This
is the place where it is possible to notice the differences (e.g.
local Lipschitz gradient) between our theory and that of [12].

V. TRUNCATION CRITERIA IN TNCG

Truncation (or CG inner stopping) criteria as (7) are ex-
plored here. In this paper, for the first time on linear classification
problems, a study on truncation criteria is performed.

A. The Importance of Truncation Criteria

By a simple example of checking the settings in some
popular software, we point out that without a careful choice
of the inner stopping criterion, the convergence of the TNCG
method can be very slow. In Figure 1 we show the total
number of CG steps needed to solve the training problem
(1) on the dataset kdd2010b. Note that as showed in (9)
the running time is always proportional to the total number
of CG steps. On the $y$ axis we report the relative reduction of the function value, computed by $\frac{f(\mathbf{w}_k) - f(\mathbf{w}^*)}{f(\mathbf{w}^*)}$, where $\mathbf{w}^*$ is the optimal solution of (1). The hyper-parameter $C'$ used is equal to $100C_{\text{best}}$, where $C_{\text{best}}$ is the value that yields the best accuracy on the validation set (see Section VII for details). In the Figure we compare

- **Scikit**: This is the TNCG method implemented in the package Scikit-learn [18], version 0.22. The CG truncation criterion is a 11-norm variant of (15) with

$$ \text{ratio} = \frac{\|g_k + H_k s_k\|_1}{\|g_k\|_1} $$

(16)

where $H_k \in \partial \nabla f(\mathbf{w}_k)$ and $\eta_k = \min\{0.5; \|g_k\|_1^{0.5}\}$. They require Armijo and Wolfe conditions.

- **ScikitArmijo**: Same implementation as Scikit above, but using only the Armijo condition.

- **Standard**: This is the TNCG method implemented in the package LIBLINEAR [19]. They consider the TR technique, but we replace it with an Armijo line search. The criterion used there is (15) with $\eta_k = 0.1$.

- **Standard09**: Even for the simple Standard setting above, it is unclear what the threshold $\eta_k$ should be. We arbitrary change it from 0.1 to 0.9 and see if the TNCG method performs similarly or not.

The four horizontal lines in Figure 1 indicate places where the following stopping condition is met respectively with $\epsilon = \{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$

$$ \|g_k\| \leq \epsilon \min\{\#\text{pos}, \#\text{neg}\} \cdot \|g_k\|, $$

(17)

where $\#\text{pos}, \#\text{neg}$ are the numbers of positive- and negative-labeled instances respectively, and $\epsilon > 0$ is the thresholding constant that controls the precision of the training procedure. Note that (17) with $\epsilon = 10^{-2}$ is the outer stopping condition employed in LIBLINEAR. Thus the behavior before $10^{-1}$ and after $10^{-4}$ is not crucial, since the training would be stopped too early or too late. From Figure 1 we can observe:

- By changing only the inner stopping criterion, the overall convergence can be dramatically different. We see that Scikit and Standard require more than four times many CG steps as Standard09 to reach the third horizontal line, while ScikitArmijo does not even reach it.

- From Standard to Standard09, we see that even just the simple change of a constant in the truncation criteria might yield remarkable improvements in the convergence speed.

- In Figure 1, every mark indicates 5 outer Newton iterations. Besides Standard09, all others waste many CG steps in each early iteration. Apparently, the inner stopping criteria may be too strict in the early stage of the optimization process. We conclude that without a careful choice of the inner stopping criterion, the TNCG method may converge slowly.

### B. Details on Truncation Criteria

We investigate truncation criteria by combining various ratios and forcing sequences. We check three different ratio:

1. residual: this is the ratio used in (15), which evaluates the norm of the residual of equation (4) w.r.t. the norm of the gradient. The setting of comparing with the norm of the gradient $\|g_k\|$ addresses a well-known issue of Newton methods: in early iterations the quadratic model is not a good local approximation of the function, so the Newton direction might not be consistently better than a simpler gradient-based direction. For this reason, over-solving (4) to better approximate the Newton direction might result in a waste of resources. Because $\|g_k\|$ is not close to zero at early iterations, the relative setting in (15) avoids a too large ratio. Then the CG procedure can stop without accurately solving the Newton equation.

2. residual11: the choice employed in Scikit-learn; see (16).

3. quadratic: in [20] it was first introduced

$$ \text{ratio} = \frac{(Q_j - Q_{j-1})}{Q_j / \lambda}, $$

(18)

where $Q_j := Q(s^*_k)$ and $Q_{j-1} := Q(s^{*-1}_k)$, and $Q(\cdot)$ is defined in (3). In (18) the reduction in the quadratic model at the current CG step $(Q_j - Q_{j-1})$ is compared with the average reduction per CG step $(Q_j / \lambda)$. The rationale of using (18) rather than (15) is that $Q_j$ reflects what we are minimizing in applying a CG method. Note in addition that computing the quadratic ratio does not require any expensive extra computations, since from (15) it can be computed by

$$ Q(s_k^*) = -\frac{1}{2} s_k^T (r - g_k) = \frac{1}{2} s_k^T H_k s_k + s_k^T g_k $$

(19)

that just requires one subtraction and one inner product between vectors. Thus, its cost is roughly $O(n)$, while the bottleneck in each CG step is still $O(nl)$ required by each Hessian-vector product.

Regarding $\eta_k$, many different families of forcing sequences have been proposed [6], [21], [22], [23], [24]. In this work we mainly investigate the following three rules:

1. constant: $\eta_k = c_0$ with $c_0 \in (0, 1)$.

This is the $\eta_k$ employed in LIBLINEAR in [9] with $c_0 = 0.1$. Selecting a suitable constant $c_0$ is never easy as indicated from the example in Section V-A.

2. adaptive: to improve the constant setting (20), the following adaptive one was proposed in [6],

$$ \eta_k = \min\{c_1; c_2 \|g_k\|^{c_3}\}, $$

(21)

where $c_1 \in (0, 1)$, $c_2 > 0$, $c_3 \in (0, 1]$. This is the $\eta_k$
TABLE I: Local convergence results of different truncation rules. Q-SL = Q-SuperLinear. F-L = Linear convergence of the Function values.

<table>
<thead>
<tr>
<th>ratio</th>
<th>ηk</th>
<th>(21)</th>
<th>(22)</th>
<th>(20)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(15)</td>
<td>Q-SL</td>
<td>Q-SL</td>
<td>F-L</td>
<td></td>
</tr>
<tr>
<td>(16)</td>
<td>Q-SL</td>
<td>Q-SL</td>
<td>F-L</td>
<td></td>
</tr>
<tr>
<td>(18)</td>
<td>Q-SL</td>
<td>Q-SL</td>
<td>F-L</td>
<td></td>
</tr>
</tbody>
</table>

employed in Scikit-learn with \(c_1 = 0.5, c_2 = 1, c_3 = 0.5\). It differs from (21) only in the use of the 11-norm. Beyond the above rules, more complex forcing sequences like those proposed in [21], [23], [24] are available for general optimization problems. However, from some preliminary results we conduct, they are not performing consistently better than the adaptive families (21) and (22). The reason may be that the adaptive setting in (21), as it was designed, has distinguished well the global phase (i.e. early iterations) from the local phase (i.e. final iterations). Because (1) is convex, a good distinction between the two phases is often enough to achieve an efficient TNCG implementation. Moreover some of the proposed choices from the literature are very highly parametrized and might often lead to over-solve (4).

C. Theoretical Properties

In Table I we present the local convergence results of inner stopping criteria discussed in Section V-B. The first \(2 \times 2\) square is pretty easy to fill, since (21) and (22) imply \(\eta_k \to 0\) and Theorem 2 then leads to the Q-SL convergence. For criteria involving 11-norm we need

\[
\|H_k s_k^j + g_k\|_2 \leq \|H_k s_k^j + g_k\|_1 \leq \eta_k\|g_k\|_1 \leq \sqrt{\eta_k}\|g_k\|_2,
\]

where \(H_k \in \partial f(w_k)\).

If the quadratic ratio (18) is considered, filling the first two entries of the last row in Table I is instead not straightforward. From some private communication with the author of [25], the result may have been given there, but that technical report is no longer available. Many steps of the proof may be found in [26], but not some of the most crucial ones. With the help of Dr. Nash we are able to prove Theorem 3, whose details are in Appendix B.

**Theorem 3.** If we employ (18) as the truncation criteria of the CG procedure to solve \(H_k s = g_k\), with \(H_k\) a symmetric positive definite matrix, then we get \(\|H_k s_k^j + g_k\|_2 \leq N_k\|g_k\|\|\frac{g_k}{\|g_k\|}\|\|s_k^j\|_1\|

where \(N_k = \sqrt{\frac{K_k\|g_k\|\|g_k\|}{1 - K_k}}\), \(K_k = \left(\frac{\lambda_{\max}(H_k) - \lambda_{\min}(H_k)}{\lambda_{\min}(H_k) - \lambda_{\min}(H_k)}\right)^2\).

Note that in our case \(H_k \in \partial f(w_k)\) and since \(f\) is strongly convex \(H_k\) is symmetric positive definite. Moreover, from (13) and the fact that \(H_k\) is bounded we get that also \(N_k\), \(\forall k\) is bounded. Thus if \(\eta_k \to 0\) by (21) or (22), then \(N_k\sqrt{\eta_k} \to 0\) and the Q-SL convergence follows from Theorem 2.

For filling the last column of Table I we note that Q-L convergence is not proved in Section IV-B. In fact, Proposition 1 does not ensure Q-L convergence for any \(\eta < 1\), but it only proves that there exists a \(\eta < 1\) for which this is obtained. Moreover, in [11] they also provide a counterexample in which a Q-L sequence does not converge when (15) is applied with a generic \(\eta < 1\). To the best of our knowledge, if no additional modification on Algorithm 1 is added and \(\eta_k\) is a generic constant (less than 1), Q-L convergence of TNCG for linear classification is still an open question. On the other hand, from the newly discovered connection between TNCG and the framework in [4] we get from Theorem 1 the local linear convergence of the function value (F-L).

VI. PRECONDITIONING IN TNCG

**Algorithm 2:** PCG for solving (24). Assume \(M\) has not been factorized to \(EE^T\).

1. Let \(s = 0, r = -g_k, d = z = M^{-1}r, \gamma = r^Tz\)
2. for \(j = 1, 2, \ldots \) do
3. \(v \leftarrow H_k d\)
4. \(\alpha \leftarrow \frac{r^Tv}{\gamma^Tv}\)
5. \(s \leftarrow s + \alpha d\)
6. \(r \leftarrow r - \alpha v\)
7. \(z \leftarrow M^{-1}r\)
8. if (7) holds with ratio \(\eta_k\) then
9. \(\text{return } s_k = s\)
10. \(\gamma_{\text{new}} \leftarrow \gamma_{\text{old}} + \delta_d\)
11. \(\beta \leftarrow \gamma_{\text{new}} / \gamma\)
12. \(d \leftarrow z + \beta d\)
13. \(\gamma \leftarrow \gamma_{\text{new}}\)

To reduce the total amount of inner CG steps needed to solve the system (4) we can apply a preconditioner matrix and solve instead the equivalent system

\[
E^{-1}H_kE^{-T}s = -E^{-1}g_k,
\]

where \(E\) is a symmetric nonsingular matrix. Once the preconditioned solution \(s_k\) is obtained, we can get the original \(s_k\) by employing \(s_k = E^{-1}s_k\). The idea behind the preconditioning techniques [27] is that of considering a matrix \(M = EE^T \approx H_k\) to obtain a condition number of \(E^{-1}H_kE^{-T}\) to be as close as possible to 1. When the approximation is good, this technique would lead to a new system (24) that is easier to be solved than the original non-preconditioned one (4).

To solve (24) one could either use the original CG procedure (see Algorithm III of the supplementary) on it or, as showed in [28], avoid the factorization of \(M = EE^T\) and use instead Algorithm 2, which has iterates \(s_k^j\) instead of \(s_k^j\). Note that Algorithm 2 is still solving the preconditioned system (24) even if it operates mostly with non-preconditioned variables. In [5] it has been showed that when the factorization is actually available, Algorithm 2 is still preferable, because, even if the two algorithms have basically the same cost, Algorithm 2 may be numerically more stable. For sake of simplicity in Algorithm 2 and in this section, we will not use the iteration counter \(j\) and \(k\) will only be reported on \(H_k\) and \(g_k\). Now, from the properties of the CG procedures (see Section III of
we have that \( \hat{r} := E^{-1}r, \hat{g}_k := E^{-1}g_k \) and \( z := M^{-1}r \). Thus
\[
\begin{align*}
\|\hat{r}\|_2 &= \|r\|_{M^{-1}} = \sqrt{r^T M^{-1} r} = \sqrt{r^T z}, \\
\|\hat{g}_k\|_2 &= \|g_k\|_{M^{-1}} = \sqrt{g_k^T M^{-1} g_k},
\end{align*}
\]
where given a vector \( a \) and a matrix \( A \) we have \( \|a\|_A := \sqrt{a^T A a} \). Then the classical ratio (15) employed in the stopping criterion of Algorithm 2 becomes
\[
\frac{\|\hat{r}\|}{\|\hat{g}_k\|} = \frac{\|r\|_{M^{-1}}}{\|g_k\|_{M^{-1}}} = \frac{\sqrt{r^T M^{-1} r}}{\sqrt{g_k^T M^{-1} g_k}} = \frac{\sqrt{r^T z}}{\sqrt{g_k^T M^{-1} g_k}}.
\]
(25)
Note that also the other ratios introduced in Section V-B can be employed in the preconditioned case. In particular, the quadratic ratio (18) can be computed in the same way by (19) for both the preconditioned and non-preconditioned cases. In fact, since Algorithm 2 is working with non-preconditioned variables like \( s, r \) and \( g_k \), we still have \( r = -H_k s - g_k \).

A. The Preconditioning by [5]

In this work we will apply the preconditioning proposed in [5]. We first recall that a diagonal preconditioner can be obtained by extracting all the diagonal elements in the Hessian
\[
M = \text{diag}(H_k), \text{where } M_{ij} = \begin{cases} (H_k)_{ij}, & \text{if } i = j, \\ 0, & \text{otherwise}. \end{cases}
\]
Once the above diagonal preconditioner is built, the original idea from [5] was that of solving in parallel the preconditioned system and the original one at each Newton iteration, and use the direction \( s \) of the procedure that terminated first. As they extensively studied both theoretically and numerically, a robust single thread alternative was instead that of combining the above diagonal preconditioner \( \hat{M} \) with the identity matrix
\[
M = \alpha \times \hat{M} + (1 - \alpha) \times I,
\]
(26)
where \( \alpha \in (0, 1) \) is the scalar that weights the combination.

We will now show that the cost of using a diagonal preconditioner at each Newton iteration is irrelevant if compared with (9). From (5) we get that
\[
(H_k)_{jj} = 1 + C \sum_i D_{ij} x_i^2,
\]
(27)
which means that constructing the above diagonal preconditioner costs \( O(nl) \). From Algorithm 2 we can see that at each CG step we additionally need to compute \( M^{-1}r \), which cost \( O(n) \). Thus, the extra cost of using the diagonal preconditioner is simply \( O(n) \times (#\text{CG steps}) + O(nl) \).

B. Global and Local Convergence in the Preconditioned Case

In this section we assume that the minimum and maximum eigenvalue of \( M^{-1} \) (respectively \( \lambda_{\min}(M^{-1}) \) and \( \lambda_{\max}(M^{-1}) \)) are bounded. This means that also \( \lambda_{\min}(M) \) and \( \lambda_{\max}(M) \) are bounded. In addition we remind that given a vector \( a \) and a matrix \( A \) we have
\[
\lambda_{\min}(A)\|a\|_2 \leq \|a\|_A \leq \lambda_{\max}(A)\|a\|_2.
\]
(28)
Note that the factorization \( M = EE^T \) is only required for the convergence analysis in this section, but never in practice.

We first point out that the global convergence result is still valid even if in Algorithm 1 instead of solving the original Newton equation (4) we solve the preconditioned one (24). In fact, the two systems are equivalent, e.g. \( s \) is the solution of (24) if and only if \( s = E^{-1} \hat{s} \) is the solution of (4). In particular, applying Algorithm 2 is equivalent to employing Algorithm III of the supplementary and then obtaining \( s \) from \( s = E^{-1} \hat{s} \) (see Section III of the supplementary or [28]). This means that we can apply Proposition III of the supplementary on the preconditioned system (24) and obtain
\[
\hat{g}_k^T \hat{s} \leq -a_1\|\hat{g}_k\|^2 \quad \text{and} \quad \|\hat{s}\| \leq a_2\|\hat{g}_k\|.
\]
(29)

From (29), (28), and with definitions of \( \hat{g}_k \) and \( \hat{s} \), we get both
\[
g_k^T s \leq a_1\|g_k\|^2 = -a_1\|E^{-1}g_k\|^2 \leq -a_1\|E^{-1}M^{-1} g_k\|^2
\]
(30)
and
\[
\lambda_{\min}(M)\|s\|_2 \leq \|g_k\|_{M^{-1}} \leq a_2\|g_k\|_2.
\]
(31)
Now, from the fact that \( \lambda_{\min}(M), \lambda_{\max}(M), \lambda_{\min}(M^{-1}) \) and \( \lambda_{\min}(M^{-1}) \) are bounded and together with (30) and (31) we get that there exist two new positive constants \( \hat{a}_1, \hat{a}_2 \) such that
\[
g_k^T s \leq -\hat{a}_1\|g_k\|^2 \quad \text{and} \quad \|s\| \leq \hat{a}_2\|g_k\|.
\]
This means that we can apply Theorem IV of the supplementary to prove global convergence.

Next we discuss local convergence results. Note that even if Algorithm 2 is employing non-preconditioned variables like \( s \) and \( r \), the ratio (25) is focusing on the norm of the residual of the preconditioned system \( \hat{r} \) and on the norm of \( \hat{g}_k \). For this reason, to ensure that local Q-SL convergence results are not harmed by the application of the preconditioning, we must connect the ratio (25) to the one in (15). From (28) we have
\[
\lambda_{\min}(M^{-1})\|r\|_2 \leq \|r\|_{M^{-1}} \leq \lambda_{\max}(M^{-1})\|r\|_2
\]
(32)
and
\[
\lambda_{\min}(M^{-1})\|g_k\|_2 \leq \|g_k\|_{M^{-1}} \leq \lambda_{\max}(M^{-1})\|g_k\|_2,
\]
which from we get that
\[
\|r\|_2 \leq \frac{1}{\lambda_{\min}(M^{-1})}\|r\|_{M^{-1}} \leq \frac{\eta_k}{\lambda_{\min}(M^{-1})}\|g_k\|_{M^{-1}}
\]
(33)
\[
\leq \eta_k\frac{\lambda_{\max}(M^{-1})}{\lambda_{\min}(M^{-1})}\|g_k\|_2.
\]
Thus, from (33), boundness of \( \lambda_{\min}(M^{-1}) \) and \( \lambda_{\min}(M^{-1}) \) and the norm equivalence results (32) and (23), one could either use (15), (16) or (25) in combination with (21) or (22) and still obtain Q-SL convergence.

Now, to prove that Q-SL convergence can be ensured also if the quadratic ratio (18) is employed, we need to notice that
\[
\hat{Q}_k(\hat{s}) := (E^{-1}g_k)^T \hat{s} + \frac{1}{2} \hat{s}^T (E^{-1}H_k E^{-T}) \hat{s}
\]
(34)
\[
= g_k^T s + \frac{1}{2} s^T H_k s = Q_k(s).
\]
This means that we can repeat the same proof of Theorem 3 for the system (24) and obtain
\[
\|r\|_{M^{-1}} \leq \sqrt{\eta_k} \hat{N}_k\|g_k\|_{M^{-1}}.
\]
(34)
where \( \tilde{N}_k = \sqrt{K_k \| H_k \| \| H_k^{-1} \|} \), \( \tilde{K}_k = \left( \frac{\lambda_{\max} - \lambda_{\min}}{\lambda_{\max} + \lambda_{\min}} \right)^2 \), and \( \lambda_{\max} \) and \( \lambda_{\min} \) are respectively the highest and lowest eigenvalues of \( H_k = E^1 H_k E^T \). Note that thanks to the fact that \( \lambda_{\min}(M^{-1}) \) and \( \lambda_{\max}(M^{-1}) \) are bounded, we also get that \( \lambda_{\min} \) and \( \lambda_{\max} \) are bounded. Together with (34) and (33) this ensures that if we use (18) in combination with (21) or (22) we can still obtain Q-SL convergence. Finally, in our case we can ensure that \( \lambda_{\min}(M^{-1}) \) and \( \lambda_{\max}(M^{-1}) \) are bounded thanks to the preconditioner defined in (26) and (27).

VII. NUMERICAL ANALYSIS ON TRUNCATION CRITERIA

We conduct experiments to analyze different inner stopping conditions. We focus on applying logistic regression on the three sets kdd2010a, kdd2010b and yahookr, while leave data statistics, detailed settings and complete results including those of l2-loss linear SVM in Section IV of the supplementary materials. For a fair evaluation, all different settings are implemented based on the LIBLINEAR package. To simulate the practical use we conduct a five-fold cross-validation to select the regularization parameter \( C_{\text{Best}} \) that achieves the best validation accuracy. See the selected values in Table I of the supplementary materials. Then in experiments we consider \( C = C_{\text{Best}} \times \{1, 100\} \). All other settings are the same as those used in Section V-A.

A. Selection of the Forcing Sequence in Truncation Criteria

In this section, we will first focus on the effect of using various forcing sequences on both the residual and the quadratic ratio. We propose a comparison between various constant thresholds in the forcing sequence (20) and the adaptive one (21). In fact, in Figure 1 we showed that even changing one single constant in the truncation rule might have a great impact on the speed of convergence. Moreover, we want to evaluate the effect of employing the adaptive forcing sequence instead of the constant one. Unless differently specified, the parameter setting used for both (21) and (22) is \( (c_1 = 0.5, c_2 = 1, c_3 = 0.5) \), since this is the one used in Scikit-learn and suggested also in [15]2. Note that we will use the standard l2-norm, but in the next section we will analyze the effect of changing the norm. In Sections IV.1-IV.3 of the supplementary materials are reported the isolated analysis on each different aspect (e.g. constant forcing sequence against adaptive one).

In Figure 2 we compare
- rescons: \( \text{ratio} = (15), \ \eta_k = (20) \ \ c_0 = 0.1 \) (in Figure 1 it was called Standard);
- rescons05: \( \text{ratio} = (15), \ \eta_k = (20) \ \ c_0 = 0.5 \);
- rescons09: \( \text{ratio} = (15), \ \eta_k = (20) \ \ c_0 = 0.9 \) (in Figure 1 it was called Standard(09));
- resada: \( \text{ratio} = (15), \ \eta_k = (21) \);
- quadcons05: \( \text{ratio} = (18), \ \eta_k = (20), \ \ c_0 = 0.5 \);
- quadada: \( \text{ratio} = (18), \ \eta_k = (21) \).

From Figure 2 we can make the following observations:

1. The setting rescons09 is faster than rescons in the large majority of the cases (the same can be observed also for the quadratic ratio; see Section IV.2 of the supplementary). This result seems to indicate that over-solving the Newton equation (4) by a small constant threshold has often a negative effect on the speed of convergence.
2. The setting rescons09 is faster than rescons05, while the constant \( c_0 = 0.9 \) in the case of the quadratic ratio is not improving results w.r.t. the setting quadcons05 (see Section IV.2 of the supplementary). This is probably caused by the fact that the quadratic ratio was originally designed to fight over-solving, being often smaller than the residual one. The quadratic ratio seems to be less influenced by changes in the constant threshold \( c_0 \).
3. The settings resada and quadada are respectively identical to rescons05 and quadcons05 in the large majority of the cases (see Section IV.3 of the supplementary). This means that the initial threshold \( c_1 = 0.5 \) is (almost) always smaller than \( c_2 \| \nabla f(w_k) \| c_3 \).
4. The setting quadada is overall performing better than resada (see also supplementary materials). The difference is remarkable in the case \( C = 100 C_{\text{Best}} \). In fact if we check

![Figure 2: A comparison of various forcing sequences in the inner stopping condition.](image-url)
the third horizontal line, quadada is always four times faster than resada. This gives another evidence of the fact that the quadratic ratio is well suited for fighting over-solving.

- Next we compare the two best settings rescons09 and quadada. By checking the third horizontal line we see that in the configuration \( C = 100C_{\text{Best}} \), on both yahookr and kdd2010a, quadada is more than two times faster than rescons09. The opposite situation is instead happening when \( C = C_{\text{Best}} \), on yahookr and kdd2010b, although rescons09 is no more than two times faster.

While results here seem to indicate that a larger constant \( c_0 \) is often better, interestingly for the l2-loss SVM the opposite occurs on some data sets; see Section IV.10 from supplementary. In fact, we decided to maintain the use of a safeguard for avoiding a possible under-solving issue in the later stage of the optimization procedure. We thus preferred the use of the adaptive forcing sequence in place of the constant one, even if results are in most of the case identical.

Finally, because the configuration of \( C = 100C_{\text{Best}} \) leads to a more difficult optimization problem with the total amount of CG steps 10 times more than that of \( C = C_{\text{Best}} \), it seems that quadada is overall a more effective choice.

### B. Comparison Between l2-Norm and l1-Norm and with TR

In Figure 3 we now address the effect of switching between l2-norm and l1-norm on resada. The setting quadada should be very little influenced by this modification since given a generic vector \( v \) we have \( \|v\|_2 \leq \|v\|_1 \) and, as showed above, \( c_1 = 0.5 \) is already (almost) always smaller than \( c_2 \|\nabla f(w_k)\|_2 \). Moreover we will also check the TR implementation of LIBLINEAR. In Figure 3 we compare

- resada\_l1: \( \text{ratio=} (16) \), \( \eta_k= (22) \). This setting is exactly the same as the approach ScikitArmijo in Section V-A. It is a simplification of the setting used in Scikit-learn without considering the Wolfe condition;
- resada: \( \text{ratio=} (15) \), \( \eta_k= (21) \);
- quadada\_l1: \( \text{ratio=} (18) \), \( \eta_k= (22) \);
- quadada: \( \text{ratio=} (18) \), \( \eta_k= (21) \);
- tr\_rescons: LIBLINEAR employs a TR globalization method (instead of the line search); see [9].

From Figure 3 we can make the following observations:

- resada is overall performing better than resada\_l1 (see also supplementary materials). This finding naturally suggests to question why in the software Scikit-learn [18] it is used the l1-norm instead of the l2-norm. On the other side, as we have seen in Section VII-A, resada is slower than quadada in most situations.

- As expected, quadada\_l1 and quadada have exactly the same speed of convergence in the large majority of the cases.

- The setting quadada is generally slightly faster than tr\_rescons, even if their speed of convergence is similar. In fact they both exploit the information obtained from the quadratic model. The setting tr\_rescons uses the information to adjust the size of the trust region, while quadada uses it to terminate the CG procedure. What we have achieved here is that with a suitable inner stopping criterion, the simpler line search setting becomes comparable or even faster than the more complicated TR approach. On the other hand, resada and resada\_l1 do not use the information from the quadratic model, and their convergence is slower.

We conclude that the truncation criteria that exploit a quadratic model are faster than those that do not use such information.

### C. Analysis via Conditions for Global Convergence

In the convergence proof from Section IV-A above or Section L2 of the supplementary one of the key properties for proving global convergence is obtaining (14). The following combination of the two requirements is called angle condition

\[ -g_k^T s_k \geq \epsilon \|g_k\| \|s_k\|. \tag{35} \]

In fact, most of the line-search based optimization methods (see Section 3.2 from [15]) requires the direction \( s_k \) to satisfy (35) to ensure global convergence\(^3\). This suggests us the idea of analyzing different inner stopping criteria on this value

\[ \cos(-g_k, s_k) = \frac{-g_k^T s_k}{\|g_k\| \|s_k\|}. \tag{36} \]

\(^3\)Note anyway that global convergence is always guaranteed for Algorithm 1 as proved in Section IV-A and Section L2 of the supplementary, so the discussion in this section is only pointing out an undesired issue that might come out in practice.
The term (36) is the cosine of the angle between the direction \( s_k \) and the anti-gradient \(-g_k\). Inequality (35) is indeed called angle condition because it requires the cosine of the above angle to be greater or equal to a positive \( c \), with \( c < 1 \). This means that the new direction \( s_k \) cannot be completely orthogonal to the anti-gradient. Otherwise, \( s_k \) is not a descent direction and we cannot use it to decrease the function value. We suspect that a good inner stopping condition should make (35) more easily satisfied. Detailed experiments in Figure 4 confirm this result.

The condition (35) is also useful to investigate the relationship between the regularization parameter \( C \) and the difficulty of the optimization problem (1). Many past experiments have shown that when \( C \) is large, the optimization problem is more difficult and slow convergence more frequently occurs [9], [5]. In particular, when the Hessian of the objective function is positive definite and we approximate \( s_k \approx H_k^{-1}g_k \) we have

\[
\begin{align*}
g_k^T s_k &\approx - g_k^T H_k^{-1} g_k \leq - \frac{\|g_k\|^2}{\lambda_{\text{max}}(H_k)} \\
\|s_k\| &\approx \| H_k^{-1} g_k \| \leq \frac{\|g_k\|}{\lambda_{\text{min}}(H_k)}.
\end{align*}
\]

(37)

Now, from (37) and (13) we have

\[
- g_k^T s_k \geq \frac{M_2}{M_1} \| g_k \| \| s_k \|.
\]

(38)

This means that when the regularization parameter \( C \) is large (e.g., \( C = 100C_{\text{Best}} \)), (38) shows that the lower bound of the cosine value in (36) we can derive is smaller. Thus the angle condition (35) is harder to be satisfied. Note that equation (35) is a property needed in the convergence proof, which requires the existence of a constant \( c \). In fact, there is no control on the right-hand side of (35), while what we hope is that the left-hand side of (35) is large so that it can be easily satisfied. Unfortunately (38) shows that for large \( C \), the left-hand side of (35) tends to be not that large. In conclusion, even if theoretically convergence is not an issue, the above discussion still gives us a hint on what might actually happen numerically when \( C \) is large.

We now show experiments in which we investigate the cosine between the anti-gradient \(-g_k\) and the resulting direction \( s_k \). In Figure 4 we compare

- quadada: ratio= (18), \( \eta_k= (21) \);
- rescons09: ratio= (18), \( \eta_k= (20) \) \( c_0 = 0.9 \);
- resada: ratio= (15), \( \eta_k= (21) \).

Cosines have been calculated till the outer termination criterion (17) is satisfied with \( \epsilon = 10^{-4} \). In Figure 4, we present the cosine value at each Newton iteration. Further, two horizontal lines corresponding to cosine = 0.1 and cosine = 0.2 have been drawn. From the comparison between Figures 4 and 2 we can make the following observations:

- Algorithms having low cosine values (for example below 0.1) are more likely to converge slower to the solution. In particular, resada has a cosine more frequently below 0.1 when \( C = 100C_{\text{Best}} \). For such configurations, Figure 2 has shown that it behaves worse in terms of convergence speed. Long continuous regions of low cosine in Figure 4 might correspond to regions of slow convergence in Figure 2 (see for example rescons09 on yahookr and kdd2010a when \( C = 100C_{\text{Best}} \), below the 0.2 threshold).
- The cosine is often smaller when \( C \) is higher. This result confirms the theoretical analysis detailed in this section on the difficulty of solving (1) as \( C \) changes.
- In Figure 2, quadada converges much faster than resada. From Figure 4 we observe that quadada’s cosine is generally higher than resada’s. This indicates that in the early Newton iterations, the resulting direction \( s_k \) of quadada is closer to the anti-gradient. We have mentioned above in this section that this is a desired property. Since early CG iterates are very close to the anti-gradient, the inner stopping condition in quadada effectively stops the CG procedure early to have this property.

We conclude that in early outer iterations a suitable criterion should avoid over-solving to obtain directions having a higher cosine with respect to the anti-gradient.

VIII. NUMERICAL ANALYSIS ON PRECONDITIONING AND TRUNCATION CRITERIA

In this section we combine the study on truncation criteria with that of preconditioning. We focus on applying logistic re-
gression on the four sets yahoo\textsubscript{r}, kdd2010a, kdd2010b and news20 while complete results, including those of l2-loss linear SVM, can be found in Sections IV.5, IV.6 and IV.11 of the supplementary. The experimental settings are the same as those in Section VII. In the previous section we pointed out that quadada is the most robust and efficient truncation rule for linear classification. In this section we will thus apply the quadada truncation criterion with the preconditioning from [5] to understand if such a combined implementation maintains the best performances.

A. Comparison Against Rules with Constant Forcing Sequences

Our first comparison is to check both quadada and rescons by applying the preconditioner from [5]. Recall that slow convergence occurred for rescons in Figure 1 (there called Standard) so subsequently we developed quadada. It is essential to check the situation after preconditioning. Note that rescons is the truncation rule employed in LIBLINEAR, even if in LIBLINEAR they employ a TR technique instead of a line search. In Figure 5 we compare

- rescons: \( \text{ratio}= (15), \eta_k = (20) \ c_0 = 0.1; \)
- rescons\_p: \( \text{ratio}= (15), \eta_k = (20) \ c_0 = 0.1, \text{preconditioned}; \)
- quadada: \( \text{ratio}= (18), \eta_k = (21); \)
- quadada\_p: \( \text{ratio}= (18), \eta_k = (21), \text{preconditioned}. \)

From Figure 5 we can make the following observations:

- The settings rescons\_p and quadada\_p are respectively faster than rescons and quadada in the large majority of the cases (see also supplementary). This result confirms that the preconditioner suggested in [5] is improving speed of convergence for most of the ill-conditioned linear systems.
- On news20 with \( C = 100C\text{Best} \) the non-preconditioned versions are slightly faster than the preconditioned ones. In fact, as detailed in [5], designing a preconditioner matrix that improves convergence in every case is very difficult. Nonetheless, the difference is not remarkable and this is one of the very few cases in which this situation is encountered.
- The setting quadada\_p is generally faster than rescons\_p (see also supplementary). This result seems to show that even in the preconditioned case the quadratic adaptive termination rule is obtaining better performances.

B. Comparison of Rules with Adaptive Forcing Sequences and Comparison with Trust Region

We now apply the preconditioning on resada, the l2-norm version of the truncation rule implemented in the package Scikit-learn. Moreover we will also check the TR implementation of LIBLINEAR. In Figure 6 we compare

- resada: \( \text{ratio}= (15), \eta_k = (21); \)
- resada\_p: \( \text{ratio}= (15), \eta_k = (21), \text{preconditioned}; \)
- quadada\_p: \( \text{ratio}= (18), \eta_k = (21), \text{preconditioned}; \)
- tr_resada\_p: \( \text{ratio}= (15), \eta_k = (20), c_0 = 0.1, \text{preconditioned}; \)

LIBLINEAR employs a TR globalization method (instead of the line search); see [5]. This is the latest LIBLINEAR version 2.30.

From Figures 5 and Figure 6 we can observe:

- Exactly as in Figure 5, also in Figure 6 the preconditioned version (resada\_p) is faster than the non-preconditioned one (resada) in the large majority of the cases (see also supplementary).
- In Figure 6 by checking news20 with \( C = 100C\text{Best} \) we can see that resada is remarkably faster than resada\_p. Instead, by comparing quadada and quadada\_p in Figure 5, the convergence speed is very similar till the third horizontal line, and even after that quadada\_p is not much slower than quadada.
- The settings quadada\_p and resada\_p often have a very similar convergence speed (see also supplementary). Nonethe-
From Figure 5 and Figure 6 we conclude that the slow convergence issues pointed out in Figure 1 for both LIBLINEAR and Scikit-learn can either be solved by employing the quadratic adaptive termination rule (as resulting from the analysis of Section VII) or by applying the preconditioner designed in [5]. Moreover, the combination of the two modifications is more efficient and robust than each of the two alone. Finally, the line search version quadada is also generally faster than tr_rescons_p implemented in the latest LIBLINEAR 2.30.

Apart from the numerical evidences just showed, we also prefer quadada_p instead of tr_rescons_p because the line search is conceptually simpler than the TR. In fact, while the TR is both deciding the step size and the direction by solving a constrained quadratic problem, all the line search based algorithms first determine the direction and then the step size. Such a separation of concern is a desirable feature, because once the direction is obtained, the line search simply needs to address a one-dimension problem to choose a scalar value. This means that if we implement a backtracking line search (e.g. Armijo as in quadada), whenever a failure is encountered (i.e. the sufficient decrease condition is not satisfied), we only need to try a new step size, while the direction remains untouched. On the contrary in the case of TR algorithms, whenever a failure is encountered (i.e. there is not a good agreement between the actual and the predicted reduction), the TR radius needs to be updated and, thus, the resulting direction might be different from the previous one. In addition, the TR update is generally more complicated than the step size update. Finally, even if the procedure for computing the direction is the same in both versions of the TNCG algorithm (i.e. an internal CG procedure like Algorithm 2), in the case of the TR there is also the need of addressing the situation in which the norm of the direction is reaching the TR boundary.

As a final remark we want to point out that in the preconditioned case the choice of the truncation rule does not cause the same dramatic difference in the speed of convergence as in the non-preconditioned case. Nonetheless, as showed in Figures 5 and 6 (see also supplementary) quadada_p is generally more robust and effective than the other truncation rules. Moreover, the quadratic ratio has an additional advantage on the residual-based ratios: it does not rely on norms. As showed in Section VI, this makes it norm- and preconditioner-independent.

IX. Conclusions

In this paper we focused on both theoretical and numerical convergence of TNCG for linear classification. We first proved global and local convergence, finding out that literature was surprisingly wanting or unclear, especially in the case of not twice differentiable losses. We filled some gaps, enlightened some open questions and discovered a new connection between TNCG and common-directions algorithms. From the algorithmic and numerical point of view, for the first time in the field of linear classification, we show the importance of selecting an appropriate truncation rule. We found out that various machine learning software are affected by slow convergence issues and we identify the circumstances that are causing them. We propose a truncation rule based on the quadratic model that is able to avoid this circumstances and demonstrate its effectiveness and robustness for linear classification.
Finally we combined the study on truncation rules with that on preconditioning. We first discussed how to obtain the same global and local convergence results also in the preconditioned case. Second, we integrated our investigation on truncation rule with the use of preconditioning. Thus we proposed a new preconditioned TNCG algorithm that is able to improve the state of the art in the field of linear classification.

Appendix A

Proof of Proposition 1 and Theorem 2

In this subsection we are going to show details on how to obtain Q-Superlinear local convergence for Algorithm 1. We study local convergence by assuming that the ratio employed in the stopping rule (7) is (15). Note that if \( f \) is the function defined in (1), the strong convexity implies BD-regularity, and, in turn, BD-regularity\(^2\) implies the following property (see Lemma 2.6 of [29]).

**Lemma 1.** If \( \nabla f \) is BD-regular at a generic point \( w \) and \( \nabla f \) is locally Lipschitz, then there exist a constant \( \delta > 0 \), a neighborhood \( N(w, \delta) := \{ w \in \mathbb{R}^n : \| w - w_0 \| \leq \delta \} \) of \( w \) and a constant \( M > 0 \) such that \( H \) is nonsingular and

\[
\| H^{-1} \| \leq M \quad \forall H \in \partial_B \nabla f(w) \quad \forall w \in N(w, \delta) .
\]

From semismoothness we obtain the following (see Proposition 1 of [30]).

**Lemma 2.** If \( \nabla f \) is semismooth at \( w \in \mathbb{R}^n \), then

\[
\lim_{s \to 0} \frac{\| \nabla f(w + s) - \nabla f(w) - Hs \|}{\| s \|} = 0 .
\]

In addition, if \( \nabla f \) is semismooth, also a second order Taylor expansion is available.

**Proposition 2.** If \( \nabla f \) is semismooth on \( \mathbb{R}^n \), then

\[
f(u) = f(w) + \nabla f(w)^T (u - w) + \frac{1}{2} (u - w)^T H(u - w) ,
\]

where \( H \in \partial f(z) \) for some \( z \in (w, u) \).

The following proposition is an extension of Lemma 2 (see Proposition 2.4 from [17]).

**Proposition 3.** Assume \( \nabla f \) to be semismooth on \( \mathbb{R}^n \) and let \( \{ \theta_k \} \) be any sequence of numbers such that \( \theta_k \in (0, 1) \) \( \forall k \). Then, for every sequence \( \{ w_k \} \) converging to a point \( w^* \) and for every sequence \( \{ H_k \} \) such that \( H_k \in \partial \nabla f(w_k^{\theta} + \theta_k (w_k - w^*)) \), we have

\[
\nabla f(w^*) = \nabla f(w_k) - H_k (w_k - w^*) + o(\| w_k - w^* \|)
\]

**Lemma 3.** Let \( \nabla f \) be semismooth and BD-regular. Let \( \{ w_k \} \) be a generic sequence convergent to a critical point \( w^* \). Further at each \( w_k \) we generate a \( s_k \) by solving the Newton linear equation (4) to satisfy the condition (7) where the ratio employed is (15). Then there exist \( \delta > 0 \), \( L > 0 \), \( M > 0 \) such that for any \( \epsilon > 0 \) and for any \( w_k \in N(w^*, \delta) \), we have

\[
\| w_k + s_k - w^* \| \leq M(\epsilon + \eta_k L)\| w_k - w^* \| .
\]

Further if \( \eta_k \to 0 \) in generating \( s_k \), then we have

\[
\lim_{k \to \infty} \frac{\| w_k + s_k - w^* \|}{\| w_k - w^* \|} = 0 .
\]

**Proof.** Since \( \nabla f \) is semismooth, by Lemma 1 we obtain that there exist constants \( M, \delta_2 > 0 \) such that if \( \| w - w^* \| \leq \delta_2 \) and \( H \in \partial_B \nabla f(w) \) then \( H \) is nonsingular and

\[
\| H^{-1} \| \leq M .
\]

Since \( \nabla f \) is semismooth we also have that \( \nabla f \) is locally Lipschitz continuous which together with stationarity of \( w^* \) means that there exist two constants \( L, \delta_1 \in (0, \delta_2) \) such that

\[
\| \nabla f(w) \| \leq L\| w - w^* \| \quad \forall w \in N(w^*, \delta_1) .
\]

By Lemma 2, for any given \( \epsilon > 0 \), there is a \( \delta \in (0, \delta_1) \) such that if \( w \in N(w^*, \delta) \) and \( H \in \partial_B \nabla f(w) \), then

\[
\| \nabla f(w) - \nabla f(w^*) - H(w - w^*) \| \leq \epsilon\| w - w^* \| .
\]

By (44), (43), (45), (15) and the stationarity of \( w^* \), for any \( w_k \in N(w^*, \delta) \) we have:

\[
\| w_k + s_k - w^* \| = \| w_k - w^* - (H_k)^{-1} \nabla f(w_k) + (H_k)^{-1} (H_k s_k + \nabla f(w_k)) \|
\]

\[
\leq \| (H_k)^{-1} \| \| (H_k s_k + \nabla f(w_k)) \|
\]

\[
\leq M(\epsilon\| w_k - w^* \| + \eta_k \| \nabla f(w_k) \|)
\]

\[
\leq M(\epsilon + \eta_k L)\| w_k - w^* \| ,
\]

which proves (41). Now, if \( \eta_k \to 0 \), from (41) and arbitrariness of \( \epsilon \) we also get (42).

**Proof of Proposition 1.** Let \( \{ w_k \} \) be a sequence generated by Algorithm 1. By our requirement, \( w_{k+1} = w_k + s_k \) and \( s_k \) is generated by solving the Newton linear equation (4) to satisfy the condition (7) where the ratio employed is (15). If \( \{ w_k \} \) is convergent to a critical point \( w^* \), then \( \{ w_k \} \) is a particular instance of the sequence described in Lemma 3. Thus, we can apply Lemma 3 and obtain that there exist \( \delta > 0 \), \( L > 0 \), \( M > 0 \) such that for any \( \epsilon > 0 \) and for any \( w_k \in N(w^*, \delta) \) we have

\[
\| w_{k+1} - w^* \| \leq M(\epsilon + \eta_k L)\| w_k - w^* \| .
\]

The first equality follows from the fact that in this sequence we can replace \( w_k + s_k \) with \( w_{k+1} \). Note that in Lemma 3 it was not possible to do this replacement because \( \{ w_k \} \) was a generic converging sequence (no process was specified on how to obtain it). (1) Thus, given a \( w_k \in N(w^*, \delta) \), we can set \( \eta := \frac{1}{L M} \) and chose \( \epsilon < \frac{1}{M} - \eta L \). Then from (46) we get that \( w_{k+1} \in N(w^*, \delta) \) and we can repeatedly apply (46). Then \( \{ w_k \} \) is Q-linearly convergent to \( w^* \). (2) If \( \eta_k \to 0 \), we can find a \( \tilde{k} \) such that \( \eta_k < \frac{1}{L M} \forall k > \tilde{k} \). Thus we can repeat a similar argument as above and moreover obtain that \( \{ w_k \} \) converges Q-superlinearly to \( w^* \).

**Proof of Theorem 2.** Let \( \{ w_k \} \) be a sequence generated by Algorithm 1 in which \( \gamma \in (0, \frac{1}{2}) \), \( \eta_k \to 0 \). Since \( \nabla f \) is semismooth we have that \( f \in \mathcal{LC}^1 \), and since \( L_1 \) is compact we also have that the (generalized) Hessian is bounded.
Furthermore, since $H_k$ employed in Algorithm 1 is the actual (generalized) Hessian of $f$ and $f$ is strongly convex, we have that $H_k$ is positive definite and bounded for all $w \in C_1$. Thus, all the assumptions of Proposition III of the supplementary are satisfied and we obtain (14). Now, from Theorem IV of the supplementary we obtain $\lim_{k \to \infty} ||\nabla f(w_k)|| = 0$, which together with the strict convexity of $f$ implies that $\{w_k\}$ is convergent to a critical point $w^*$. This (together with the fact that strong convexity of $f$ implies BD-regularity of $\nabla f$) means that we can apply Lemma 3 on $\{w_k\}$, since it is a sequence converging to a critical point $w^*$ in which $s_k$ is generated by solving the Newton linear equation (4) to satisfy the condition (7) where the ratio employed is (15). Since $\eta_k \to 0$, from Lemma 3 we have (42), which means that for any $\delta > 0$, we can find a $k(\delta)$ such that for any $k \geq k(\delta)$ we have

$$\delta > \left| \frac{\|w_k - w^*\|}{\|w_k - w^*\|} + \frac{s_k}{\|w_k - w^*\|} \right| \geq 1 - \frac{\|s_k\|}{\|w_k - w^*\|}. \tag{47}$$

Thus, from the arbitrariness of $\delta$ we have that

$$\lim_{k \to \infty} \|s_k\| = 0. \tag{48}$$

Now, from (47) we get both, for $\delta < 1$ and $k \geq k(\delta)$,

$$\|w_k + s_k - w^*\| \leq \delta \|w_k - w^*\|, \quad \|w_k - w^*\| \leq \frac{1}{1 - \delta} \|s_k\|.$$

Thus, putting all together we obtain

$$\|w_k + s_k - w^*\| \leq \frac{\delta}{1 - \delta} \|s_k\|,$$

which leads to $\|w_k + s_k - w^*\| = o(\|s_k\|). \tag{49}$

Since $\nabla f$ is semismooth, we get that $H$ is locally upper-bounded in $N(w^*, \delta)$, i.e. there exists an $M > 0$ such that

$$M > \|H\|, \quad \forall H \in \partial \nabla f(w) \quad \forall w \in N(w^*, \delta). \tag{50}$$

By Proposition 2 and since $\nabla f(w^*) = 0$, we can write

$$f(w_k + s_k) = f(w^*) + \frac{1}{2} \langle w_k + s_k - w^*, w_k + s_k - w^* \rangle + o(\|s_k\|^2), \tag{51}$$

where $H_k \in \partial \nabla f(z_k)$ for some $z_k \in (w^*, w_k + s_k)$ and $R_k \in \partial \nabla f(u_k)$ for some $u_k \in (w^*, w_k)$. Subtracting the two above equalities we have

$$f(w_k + s_k) - f(w_k) = -\frac{1}{2} \langle w_k - w^*, R_k(w_k - w^*) + o(\|s_k\|^2),$$

where the last term follows from (49) and (50). By subtracting $\frac{1}{2} \nabla f(w_k)^T s_k$ we get

$$f(w_k + s_k) - f(w_k) - \frac{1}{2} \nabla f(w_k)^T s_k$$

$$= -\frac{1}{2} \langle w_k - w^*, R_k(w_k - w^*) + \frac{1}{2} \nabla f(w_k)^T s_k + o(\|s_k\|^2) \rangle$$

$$= -\frac{1}{2} \langle w_k - s_k, R_k(w_k - w^*) + \frac{1}{2} s_k^T R_k w_k - w^* \rangle$$

$$- \frac{1}{2} \nabla f(w_k)^T s_k + o(\|s_k\|^2)$$

$$= -\frac{1}{2} \nabla f(w_k)^T s_k + o(\|s_k\|^2)$$

$$= o(\|s_k\|^2)$$

where the third equality follows from (48), (49), and (50), while the last equality follows from Proposition 3 and again (48). Thus, from the last equation we have

$$f(w_k + s_k) - f(w_k) = \frac{1}{2} \nabla f(w_k)^T s_k + o(\|s_k\|^2)$$

$$+ \gamma \nabla f(w_k)^T s_k - \gamma \nabla f(w_k)^T s_k$$

$$\leq \frac{1}{2} \nabla f(w_k)^T s_k + o(\|s_k\|^2)$$

$$- \left(1 - \gamma \right) \frac{a_1}{a_2} \nabla f(w_k)^T s_k,$$

where the first inequality follows from the first inequality in (14), while the second inequality from the second inequality in (14). The inequality (51) proves that there exists a $\hat{k}$ for which (8) is satisfied with $\omega_k = 1 \forall k > \hat{k}$ and $\gamma \in (0, \frac{1}{2})$. Finally, let us call again $\{w_k\}$ the sequence that starts with $\hat{k}$ (i.e. $\hat{k} = 0$). This is a new sequence generated by Algorithm 1 without line search ($\omega_k = 1 \forall k > \hat{k}$) that converges to $w^*$. With $\eta_k \to 0$, we can thus apply Proposition 1 and obtain that $\{w_k\}$ Q-superlinearly converges to $w^*$.

\section*{Appendix B}

\textbf{Proof of Theorem 3}

Before giving the proof of Theorem 3 we need to remember that thanks to CG method properties, we have that $Q_j$ is monotonically decreasing and

$$Q_1 = -\frac{1}{2} \frac{g_k^T g_k}{g_k^T H_k g_k} < 0.$$

Thus, we have that $(Q_j - Q_{j+1}) < 0$ and $Q_j < 0$ for every $j \geq 1$. For this reason (18) is equivalent to

$$Q_{j+1} - Q_j \leq \eta_j \cdot \frac{-Q_j}{\lambda_j} \tag{52}.$$ 

We now recall two technical lemmas. In particular, the following result can be found in Lemma 3.1 of [25].

\textbf{Lemma 4.} If $H$ is symmetric and positive-definite then

$$y^T H^2 y \leq \|H\| y^T H y. \tag{53}$$

\textbf{Proof.} We have

$$y^T H^2 y = (H^2 y)^T H (H^2 y) \leq \lambda_{\text{max}}(H) \cdot (H^2 y)^T H y = \|H\| y^T H y. \tag{54}$$

The following result can be found in Theorem 4.3.3 of [25].

\textbf{Lemma 5.} Let $s^*$ be the point that minimizes $Q(s)$. Then, for any $s$ we have

$$\|H_k s + g_k\|^2 \leq 2 \|H_k\| \cdot (Q(s) - Q(s^*)) \tag{55}$$
Proof. From Lemma 4 and \( g_k = -H_k s^* \) we have
\[
\|H_k s + g_k\|^2 = (H_k s + g_k)^T (H_k s + g_k) \\
= (H_k s - H_k s^*)^T (H_k s - H_k s^*) \\
= (s - s^*)^T H_k^2 (s - s^*) \\
\leq \|H_k\| \|s^T H_k s - 2s^T H_k s + s^T H_k s^*\| \\
= \|H_k\| \|(s^T H_k s + 2s^T g_k) - (s^T H_k s^* + 2s^T g_k)\| \\
= 2\|H_k\| \|(Q(s) - Q(s^*))\|
\]
which means that
\[
K_k (Q(s_k^*) - Q(s_k)) \geq Q(s_k^{j+1}) - Q(s_k^*),
\]
where \( s_k^* \) is the point minimizing \( Q(s) \). Thus, moving \( K_k Q(s_k^*) \) on the right side and subtracting \( K_k Q(s_k^{j+1}) \) from both sides of the inequality we get
\[
K_k Q(s_k^*) - K_k Q(s_k^{j+1}) \geq -K_k Q(s_k^{j+1}) + K_k Q(s_k^*) + Q(s_k^{j+1}) - Q(s_k^*) = (1 - K_k) \left( Q(s_k^{j+1}) - Q(s_k^*) \right),
\]
which means that
\[
Q(s_k^{j+1}) - Q(s_k^*) \leq \frac{K_k}{1 - K_k} \left( Q(s_k^*) - Q(s_k^{j+1}) \right). \tag{55}
\]

Now since \( Q(s_k^*) \) is monotonically decreasing as \( j \) increases we get that
\[
-Q(s_k^*) \geq -Q(s_k^*) \quad \forall j. \tag{56}
\]

At the solution \( s_k^* \), from \( s_k^* = -H_k^{-1} g_k \), we have
\[
Q(s_k^*) = s_k^{T} g_k + \frac{1}{2} s_k^T H_k s_k = -\frac{1}{2} g_k^T H_k^{-1} g_k.
\]
Thus, together with (56), we get
\[
-Q(s_k^*) \leq -Q(s_k^*) \leq \frac{1}{2} \|H_k^{-1}\| \|g_k\|^2. \tag{57}
\]
Finally from Lemma 5, (55), (52), (57) we get
\[
\|H_k s_k^* + g_k\|^2 \leq 2\|H_k\| \cdot (Q(s_k^*) - Q(s_k^*)) \\
\leq 2K_k \|H_k\| \left( Q(s_k^{j+1}) - Q(s_k^*) \right) \\
\leq \eta_k \cdot \frac{2K_k \|H_k\|}{1 - K_k} \cdot -Q(s_k^{j+1})j \\
\leq \eta_k \cdot \frac{K_k \|H_k\| \cdot \|H_k^{-1}\| \cdot \|g_k\|^2}{1 - K_k} \\
= \eta_k \cdot N_k^2 \|g_k\|^2,
\]
where \( N_k^2 = \frac{K_k \|H_k\| \cdot \|H_k^{-1}\|}{1 - K_k} \) and the last inequality follows from the fact that \( j \geq 1 \). \( \square \)