Supplementary materials for "A Study on Truncated Newton Methods for Linear Classification"

Leonardo Galli LEONARDO.GALLI@UNIFI.IT
Department of Information Engineering
University of Florence
Via Santa Marta 3, Firenze, Italia

Chih-Jen Lin CJLIN@CSIE.NTU.EDU.TW
Department of Computer Science
National Taiwan University
Taipei, Taiwan, 106

We report here some equations from the original paper that might be useful in the rest of the supplementary materials file. In Table I we collected the notations we used in this paper.

The **linear classification problem**:

$$\min_{\boldsymbol{w}} f(\boldsymbol{w}) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum\nolimits_{i=1}^{l} \xi(y_i\boldsymbol{w}^T\boldsymbol{x}_i), \tag{i}$$

where $(y_i, \boldsymbol{x}_i)$, $i = 1, ..., l$ are the training data, $y_i = \pm 1$ is the label, $\boldsymbol{x}_i \in \mathbb{R}^n$ is a feature vector, $\boldsymbol{w}^T\boldsymbol{w}/2$ is the regularization term, $C > 0$ is a regularization parameter and $\xi(y_i\boldsymbol{w}^T\boldsymbol{x}_i)$ is a generic LC^1 (continuously differentiable with locally Lipschitz continuous gradient) convex loss function. With $\boldsymbol{w}^T\boldsymbol{w}$, f is a LC^1 strongly convex function and the minimum $\boldsymbol{w}^*$ of f exists and is unique.
The following two **losses** respectively correspond to logistic regression (C^2, i.e. twice continuously differentiable) and l2-loss linear SVM (LC^1)

$$\begin{aligned} \xi_{\mathrm{LR}}(y\boldsymbol{w}^T\boldsymbol{x}) &= \log\big(1 + \exp\big(-y\boldsymbol{w}^T\boldsymbol{x}\big)\big) \\ \xi_{\mathrm{L2}}(y\boldsymbol{w}^T\boldsymbol{x}) &= (\max(0, 1 - y\boldsymbol{w}^T\boldsymbol{x}))^2. \end{aligned} \tag{ii}$$

The **gradient** and the **Hessian** of the objective function f:

$$\begin{aligned} \boldsymbol{g} &:= \nabla f(\boldsymbol{w}) = \boldsymbol{w} + C\sum\nolimits_{i=1}^{l} \xi'(y_i\boldsymbol{w}^T\boldsymbol{x}_i)y_i\boldsymbol{x}_i, \\ H &:= \nabla^2 f(\boldsymbol{w}) = I + CX^TDX, \end{aligned} \tag{iii}$$

where I is the identity matrix, $X = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_l]^T$ is the data matrix and D is a diagonal matrix with $D_{ii} = \xi''(y_i\boldsymbol{w}^T\boldsymbol{x}_i)$. The **Newton equation**:

$$\nabla^2 f(\boldsymbol{w}_k)\boldsymbol{s} = -\nabla f(\boldsymbol{w}_k). \tag{iv}$$

Generic **truncation criterion**:

$$ratio(\boldsymbol{s}_k^j) \leq \eta_k, \tag{v}$$

Armijo condition:

$$f(\boldsymbol{w}_k + \omega_k\boldsymbol{s}_k) \leq f(\boldsymbol{w}_k) + \gamma\omega_k\boldsymbol{g}_k^T\boldsymbol{s}_k, \quad \text{with } 0 < \gamma < 1. \tag{vi}$$

Algorithm I: Conjugate Gradient

Input: $\boldsymbol{s}_k^1 = \mathbf{0} \in \mathbb{R}^n$, $\boldsymbol{d}_k^1 = -\nabla Q_1 = -\boldsymbol{g}_k$

for *j=1, 2, ...* **do**
- $\alpha_k^j = \frac{\|\nabla Q_j\|^2}{\boldsymbol{d}_k^{j\,T} H_k \boldsymbol{d}_k^j}$
- $\boldsymbol{s}_k^{j+1} = \boldsymbol{s}_k^j + \alpha_k^j \boldsymbol{d}_k^j$
- $\nabla Q_{j+1} = \nabla Q_j + \alpha_k^j H_k \boldsymbol{d}_k^j$
- **if** (v) *is satified* **then**
  - $\boldsymbol{s}_k = \boldsymbol{s}_k^{j+1}$
  - **return** $\boldsymbol{s}_k$
- $\beta_j = \frac{\|\nabla Q_{j+1}\|^2}{\|\nabla Q_j\|^2}$
- $\boldsymbol{d}_k^{j+1} = -\nabla Q_{j+1} + \beta_j \boldsymbol{d}_k^j$

Wolfe condition:

$$\nabla f(\boldsymbol{w}_k + \omega_k \boldsymbol{s}_k)^T \boldsymbol{s}_k \geq \gamma \boldsymbol{g}_k{}^T \boldsymbol{s}_k, \quad \text{with } 0 < \gamma < 1. \tag{vii}$$

Level set:

$$\mathcal{L}_1 := \{\boldsymbol{w} \in \mathbb{R}^n : f(\boldsymbol{w}) \leq f(\boldsymbol{w}_1)\}. \tag{viii}$$

This lemma can be obtained directly from the structure of the losses and it will be needed in Section I.

Lemma I *The Hessian matrix* (iii) *is bounded, i.e. there exist two constants* $M_1, M_2 > 0$ *such that*

$$M_1 \geq \|H_k\| \geq M_2 \ \ \forall k. \tag{ix}$$

Proof By the definition of the losses functions (ii) we have that $0 \leq D_{ii} \leq 2$ (see Appendix A for details). Thus from (iii) we have that $\forall k$ H_k is bounded. Furthermore, with the matrix I in (iii), H_k is bounded above zero. ■

I. Global and Local Convergence of TNCG

In this section we relax f to be any $f \in \mathrm{LC}^1$.

I.1 Global Convergence by Treating TNCG as a Common-Directions Algorithm

In Algorithm I we detail the instructions of the conjugate gradient method for solving a generic quadratic strongly convex function of the shape $Q(\boldsymbol{s}) = \boldsymbol{g}_k^T \boldsymbol{s} + \frac{1}{2}\boldsymbol{s}^T H_k \boldsymbol{s}$. We define $\nabla Q_j := \nabla Q(\boldsymbol{s}_k^j)$, where $\nabla Q(\boldsymbol{s}) = \boldsymbol{g}_k + H_k \boldsymbol{s}$.

Lemma II *Let* $\boldsymbol{s}_k$ *be the direction returned by Algorithm I and assume that it terminates after* m_k *iterations. Then* $\boldsymbol{s}_k = \sum_{j=1}^{m_k} \alpha_k^j \boldsymbol{d}_k^j$, *where* $P_k = \{\boldsymbol{d}_k^1, \dots \boldsymbol{d}_k^{m_k}\}$ *are conjugate directions*

Algorithm II: Truncated Newton Conjugate Gradient

Input: $\boldsymbol{w}_1 \in \mathbb{R}^n$ starting point

for $k = 1, 2, \ldots$ **do**

- compute the direction $\boldsymbol{s}_k$ by approximately solving (iv) with a CG method, until (v) is satisfied
- compute a step length ω_k
- $\boldsymbol{w}_{k+1} = \boldsymbol{w}_k + \omega_k \boldsymbol{s}_k$

with respect to H_k *and* $\{\alpha_k^1, \ldots, \alpha_k^{m_k}\}$ *are scalars defined to minimize*

$$\begin{aligned} &\min_{\boldsymbol{\alpha}_k} \boldsymbol{g}_k^T \boldsymbol{s}_k + \frac{1}{2} \boldsymbol{s}_k^T H_k \boldsymbol{s}_k \\ &s.t.\ \boldsymbol{s}_k = P_k \boldsymbol{\alpha}_k, \end{aligned} \tag{x}$$

where $P_k := [\boldsymbol{d}_k^1 | \ldots | \boldsymbol{d}_k^m] \in \mathbb{R}^{n \times m}$ *and* $\boldsymbol{g}_k \in \{\boldsymbol{d}_k^1, \ldots, \boldsymbol{d}_k^m\}$.

Proof From instructions of Algorithm I we directly get that

$$\boldsymbol{s}_k = \boldsymbol{s}_k^{m_k+1} = \sum_{j=1}^{m_k} \alpha_k^j \boldsymbol{d}_k^j.$$

In addition, from Theorem 5.2 of Nocedal and Wright (1999) and the fact that $\boldsymbol{s}_k^1 = \boldsymbol{0}$ we have that $\boldsymbol{s}_k$ is the minimizer of (x) on the span of the conjugate directions $\{\boldsymbol{d}_k^1, \ldots \boldsymbol{d}_k^{m_k}\}$. This concludes the proof. ■

We now point out that in the framework (Lee et al., 2017), the total number of common-directions m is assumed to be fixed through the whole optimization procedure. Since the termination criterion (v) is employed in Algorithm I, in the TNCG case this number m_k depends on the iteration k. On the other hand, it is possible to see that in the proofs of Lee et al. (2017), m is never really required to be fixed. In fact, as long as m is bounded, the problem (x) is well defined and all the properties required for global convergence follow from the fact that $\boldsymbol{w}_k$ is its solution, together with the fact that $\boldsymbol{g}_k \in \{\boldsymbol{d}_k^1, \ldots \boldsymbol{d}_k^{m_k}\}$. Note that problem (x) is highly dependent on k, and the dimensionality of $\boldsymbol{\alpha}_k$ can be seen as a detail of the internal procedure for obtaining $\boldsymbol{w}_k$. For these reasons, we can relax the assumption that m is a fixed constant, and we instead require that it is bounded. In the case of Algorithm I, Theorem 5.1 from Nocedal and Wright (1999) ensures that the procedure terminates within n iterations, so we have that m_k is always bounded by n.

I.2 Global Convergence Following a More Classical Path

We first need to prove that the direction $\boldsymbol{s}_k$ obtained by Algorithm I is able to satisfy (xi) and (xii). In this section we relax the assumption that H_k employed in $Q(\boldsymbol{s})$ of Algorithm I is the Hessian (or the generalized Hessian) of f. In fact, it is interesting to point out that for obtaining global convergence of $\{\boldsymbol{w}_k\}$, H_k can be a generic positive definite matrix. Indeed in general, second-order information of f are not needed to obtain global convergence to a stationary point.

Proposition III *Let $f \in LC^1$. In addition, assume that H_k is positive definite and $\forall k$ H_k is bounded as in* (ix). *Let $\boldsymbol{s}_k$ be generated by Algorithm I. Then there exist two constants $a_1 > 0$ and $a_2 > 0$ such that*

$$\nabla f(\boldsymbol{w}_k)^T \boldsymbol{s}_k \leq -a_1 \|\nabla f(\boldsymbol{w}_k)\|^2, \tag{xi}$$

$$\|\boldsymbol{s}_k\| \leq a_2 \|\nabla f(\boldsymbol{w}_k)\|. \tag{xii}$$

Proof If we pre-multiply

$$\nabla Q_j = \nabla Q_1 + \sum_{i=1}^{j-1} \alpha_i H_k \boldsymbol{d}_k^i$$

by $\boldsymbol{d}_k^j$ we obtain

$$\boldsymbol{d}_k^{j\,T} \nabla Q_j = \boldsymbol{d}_k^{j\,T} \nabla Q_1, \tag{xiii}$$

since $\boldsymbol{d}_k^j$ are all mutually conjugated. From Theorem 5.1 of Hestenes and Stiefel (1952) we have that

$$\|\nabla Q_j\|^2 = -\nabla Q_i{}^T \boldsymbol{d}_k^j \quad \forall 1 \leq i \leq j, \tag{xiv}$$

which together with (xiii) let us obtain that

$$\alpha^j = \frac{\|\nabla Q_j\|^2}{\boldsymbol{d}_k^{j\,T} H_k \boldsymbol{d}_k^j} = -\frac{\nabla Q_j{}^T \boldsymbol{d}_k^j}{\boldsymbol{d}_k^{j\,T} H_k \boldsymbol{d}_k^j} = -\frac{\nabla Q_1{}^T \boldsymbol{d}_k^j}{\boldsymbol{d}_k^{j\,T} H_k \boldsymbol{d}_k^j}. \tag{xv}$$

From (xv) we get that

$$\boldsymbol{s}_k = \boldsymbol{s}_k^{j+1} = \sum_{i=1}^{j} \alpha^i \boldsymbol{d}_k^i = -\sum_{i=1}^{j} \frac{\nabla Q_1{}^T \boldsymbol{d}_k^i}{\boldsymbol{d}_k^{i\,T} H_k \boldsymbol{d}_k^i} \boldsymbol{d}_k^i. \tag{xvi}$$

Thus,

$$\begin{aligned} \nabla f(\boldsymbol{w}_k)^T \boldsymbol{s}_k &= -\sum_{i=1}^{j} \frac{\left(\nabla f(\boldsymbol{w}_k)^T \boldsymbol{d}_k^i\right)^2}{\boldsymbol{d}_k^{i\,T} H_k \boldsymbol{d}_k^i} \\ &\leq -\frac{\left(\nabla f(\boldsymbol{w}_k)^T \nabla f(\boldsymbol{w}_k)\right)^2}{\boldsymbol{d}_k^{1\,T} H_k \boldsymbol{d}_k^1}, \end{aligned}$$

where the equality follows from the fact that $\nabla Q_1 = \nabla f(\boldsymbol{w}_k)$ and the inequality follows from the fact that H_k is positive definite and from the fact that in the algorithm initialization we have $\boldsymbol{d}_k^1 = \nabla f(\boldsymbol{w}_k)$. Thus, $\boldsymbol{s}_k$ is a descent direction and we get

$$|\nabla f(\boldsymbol{w}_k)^T \boldsymbol{s}_k| \geq \frac{\|\nabla f(\boldsymbol{w}_k)\|^4}{\|\nabla f(\boldsymbol{w}_k)\|^2 \|H_k\|} \geq \frac{1}{M_1} \|\nabla f(\boldsymbol{w}_k)\|^2,$$

where the last inequality follows from upper-boundedness of H with the constant $M_1 > 0$. This proves that (xi) is satisfied with $a_1 = \frac{1}{M_1}$. Now to prove (xii) we first observe that H_k is also lower bounded, i.e.

$$\boldsymbol{d}_k^{j\,T} H_k \boldsymbol{d}_k^j \geq M_2 \boldsymbol{d}_k^{j\,T} I \boldsymbol{d}_k^j \geq M_2 \|\boldsymbol{d}_k^j\|^2, \tag{xvii}$$

where I is the identity matrix. Thus, from (xvi) and (xvii) we can obtain

$$\begin{aligned}
\|\boldsymbol{s}_k\| = \|\boldsymbol{s}_k^{j+1}\| &= \sum_{i=1}^{j} \left| \frac{\nabla Q_1{}^T \boldsymbol{d}_k^i}{\boldsymbol{d}_k^{i\,T} H_k \boldsymbol{d}_k^i} \right| \|\boldsymbol{d}_k^i\| \\
&\leq \sum_{i=1}^{j} \frac{\|\boldsymbol{d}_k^i\|^2 \|\nabla Q_1\|}{|\boldsymbol{d}_k^{i\,T} H_k \boldsymbol{d}_k^i|} \\
&= \sum_{i=1}^{j} \frac{\|\boldsymbol{d}_k^i\|^2}{\left|\boldsymbol{d}_k^{i\,T} H_k \boldsymbol{d}_k^i\right|} \|\nabla f(\boldsymbol{w}_k)\| \\
&\leq \frac{j}{M_2} \|\nabla f(\boldsymbol{w}_k)\| \\
&\leq \frac{n}{M_2} \|\nabla f(\boldsymbol{w}_k)\|,
\end{aligned}$$

where the last inequality follows from the fact that Algorithm I always terminates in a number of steps that is bounded by n (see Theorem 5.1 of Nocedal and Wright (1999)). Thus, (xii) is proved to be satisfied with $a_2 = \frac{n}{M_2}$. ■

From Proposition III we have that $\boldsymbol{s}_k$ is a descent direction and this means that the Armijo line search procedure terminates finitely (see for instance Nocedal and Wright (1999) for details). Then we obtain global convergence by exploiting Armijo line search properties. In the following theorem we relax the assumption that $\boldsymbol{s}_k$ is obtained by Algorithm I. In fact, the Theorem IV is still valid for any procedure that yields a direction $\boldsymbol{s}_k$ that satisfies (xi) and (xii).

Theorem IV *Let $f \in LC^1$ and let $\mathcal{L}_1$ defined in* (viii) *be compact. Let $\{\boldsymbol{w}_k\}$ be a sequence generated by Algorithm II, where at each step k the direction in Step 3 is obtained by any procedure that yields a $\boldsymbol{s}_k$ that satisfies* (xi) *and* (xii)*. Then,*

$$\lim_{k\to\infty} \|\nabla f(\boldsymbol{w}_k)\| = 0. \tag{xviii}$$

Proof From the Armijo line search instruction we get that

$$f(\boldsymbol{w}_{k+1}) \leq f(\boldsymbol{w}_k) + \gamma \omega_k \nabla f(\boldsymbol{w}_k)^T \boldsymbol{s}_k, \tag{xix}$$

and from this we obtain that

$$\begin{aligned} f(\boldsymbol{w}_1) - f(\boldsymbol{w}_k) &= \sum_{i=1}^{k} f(\boldsymbol{w}_i) - f(\boldsymbol{w}_{i+1}) \\ &\geq -\gamma \sum_{i=1}^{k} \omega_i \nabla f(\boldsymbol{w}_i)^T \boldsymbol{s}_i \\ &\geq \gamma \sum_{i=1}^{k} \omega_i a_1 \|\nabla f(\boldsymbol{w}_i)\|^2, \end{aligned} \tag{xx}$$

where the last inequality follows from (xi). By contradiction, there exists a subsequence $K \subset \{1, 2 \dots\}$ for which

$$\|\nabla f(\boldsymbol{w}_k)\| \geq \epsilon > 0, \quad \forall k \in K. \tag{xxi}$$

From Armijo line search properties we get that $\boldsymbol{w}_k \in \mathcal{L}_1$ and thus the left part of inequality (xx) is bounded from above, which means that also the right part of (xx) is bounded from above. Note that the right part of (xx) is also positive, thus,

$$\lim_{k \to \infty} \gamma \sum_{i=1}^{k} \omega_i a_1 \|\nabla f(\boldsymbol{w}_i)\| < \infty.$$

Since $\|\nabla f(\boldsymbol{w}_k)\|$ is bounded from below because of (xxi), we get that

$$\lim_{\substack{k \to \infty \\ k \in K}} \sum_{i=1}^{k} \omega_i < \infty,$$

which, in turn, brings to

$$\lim_{\substack{k \to \infty \\ k \in K}} \omega_k = 0. \tag{xxii}$$

Because of (xxii), it is not possible to have an infinite subsequence of K in which Armijo always terminates with $\omega_k = 1$. Thus, from Armijo line search properties there exists a $\hat{k}$ for which $\forall k > \hat{k}, k \in K$ Armijo always terminates with a ω_k such that

$$f(\boldsymbol{w}_k + \frac{\omega_k}{\delta} \boldsymbol{s}_k) > f(\boldsymbol{w}_k) + \gamma \frac{\omega_k}{\delta} \nabla f(\boldsymbol{w}_k)^T \boldsymbol{s}_k.$$

From the Mean Value Theorem, applied on the above inequality, we obtain that

$$\nabla f(\boldsymbol{z}_k)^T \boldsymbol{s}_k > \gamma \nabla f(\boldsymbol{w}_k)^T \boldsymbol{s}_k, \tag{xxiii}$$

where $\boldsymbol{z}_k = \boldsymbol{w}_k + \theta_k \frac{\omega_k}{\delta} \boldsymbol{s}_k$ and $\theta_k \in [0, 1]$. Since $\{\boldsymbol{w}_k\}_K \in \mathcal{L}_1$, from K we can extract another infinite subsequence (redefined K) for which

$$\lim_{\substack{k \to \infty \\ k \in K}} \boldsymbol{w}_k = \hat{\boldsymbol{w}}. \tag{xxiv}$$

Now, again from the fact that $\{\boldsymbol{w}_k\}_K \in \mathcal{L}_1$, we have that there exists a constant $M > 0$ such that $\|\nabla f(\boldsymbol{w}_k)\| < M \ \ \forall \boldsymbol{w}_k \in \mathcal{L}_1$ and thus, together with (xii) we get

$$\|\boldsymbol{s}_k\| \leq a_2 \|\nabla f(\boldsymbol{w}_k)\| \leq a_2 M. \tag{xxv}$$

Thus, from (xxiv), (xxii) and (xxv) we get that

$$\lim_{\substack{k\to\infty \\ k\in K}} \boldsymbol{z}_k = \lim_{\substack{k\to\infty \\ k\in K}} \boldsymbol{w}_k + \theta_k \frac{\omega_k \boldsymbol{s}_k}{\delta} = \hat{\boldsymbol{w}}. \tag{xxvi}$$

From (xxv) we have that $\{\boldsymbol{s}_k\}_K$ is limited, so from K we can extract another infinite subsequence (redefined K) for which

$$\lim_{\substack{k\to\infty \\ k\in K}} \boldsymbol{s}_k = \hat{\boldsymbol{s}}. \tag{xxvii}$$

Moreover, from (xi) and (xxi) we have

$$\nabla f(\hat{\boldsymbol{w}})^T \hat{\boldsymbol{s}} \leq -a_1 \|\nabla f(\hat{\boldsymbol{w}})\|^2 \leq -a_1 \epsilon^2 < 0. \tag{xxviii}$$

Finally, from (xxiii), (xxvi) and (xxvii) we get that

$$\gamma \nabla f(\hat{\boldsymbol{w}})^T \hat{\boldsymbol{s}} = \lim_{\substack{k\to\infty \\ k\in K}} \gamma \nabla f(\boldsymbol{w}_k)^T \boldsymbol{s}_k \leq \lim_{\substack{k\to\infty \\ k\in K}} \nabla f(\boldsymbol{z}_k)^T \boldsymbol{s}_k = \nabla f(\hat{\boldsymbol{w}})^T \hat{\boldsymbol{s}},$$

which is absurd, because from (xxviii) we have $\nabla f(\hat{\boldsymbol{w}})^T \hat{\boldsymbol{s}} < 0$ and $\gamma < 1$. ■

I.3 Q-Superlinear Local Convergence

In this subsection we are going to show details on how to obtain Q-Superlinear local convergence for Algorithm II. We study local convergence by assuming that the *ratio* employed in the stopping rule (v) is (xl). Note that if f is the function defined in (i), the strong convexity implies BD-regularity, and, in turn, BD-regularity[1] implies the following property (see Lemma 2.6 of Qi (1993)).

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

$$\|H^{-1}\| \leq M \quad \forall H \in \partial_B \nabla f(\boldsymbol{w}) \ \ \forall \boldsymbol{w} \in N(\bar{\boldsymbol{w}}, \delta). \tag{xxix}$$

From semismoothness we obtain the following (see Proposition 1 of Pang and Qi (1993)).

Lemma VI *If ∇f is semismooth at $\boldsymbol{w} \in \mathbb{R}^n$, then*

$$\lim_{\substack{\boldsymbol{s}\to 0 \\ H\in\partial\nabla f(\boldsymbol{w}+\boldsymbol{s})}} \frac{\|\nabla f(\boldsymbol{w}+\boldsymbol{s}) - \nabla f(\boldsymbol{w}) - H\boldsymbol{s}\|}{\|\boldsymbol{s}\|} = 0. \tag{xxx}$$

1. Note that in Qi (1993) this property is called strong BD-regularity, but later on it was simply called BD-regularity, even by the author of Qi (1993), for instance in Martínez and Qi (1995).

In addition, if ∇f is semismooth, also a second order Taylor expansion is available.

Proposition VII *If ∇f is semismooth on $\mathbb{R}^n$, then*

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

where $H \in \partial f(\boldsymbol{z})$ for some $\boldsymbol{z} \in (\boldsymbol{u}, \boldsymbol{w})$.

The following proposition is an extension of Lemma VI (see Proposition 2.4 from Facchinei (1995)).

Proposition VIII *Assume ∇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 $\{\boldsymbol{w}_k\}$ converging to a point $\boldsymbol{w}^*$ and for every sequence $\{H_k\}$ such that $H_k \in \partial\nabla f(\boldsymbol{w}^* + \theta_k(\boldsymbol{w}_k - \boldsymbol{w}^*))$, we have*

$$\nabla f(\boldsymbol{w}^*) = \nabla f(\boldsymbol{w}_k) - H_k(\boldsymbol{w}_k - \boldsymbol{w}^*) + o(\|\boldsymbol{w}_k - \boldsymbol{w}^*\|).$$

In the proof of Lemma below (called Lemma 1 in the main paper), we first provide the result for a generic *forcing sequence* to clarify that when (xliii) is employed and $f \in \mathrm{LC}^1$ Q-L convergence is still an open question. If we instead directly assume $\eta_k \to 0$, the following proof would be more similar to the one given in Theorem 5.3 of Qi and Sun (2006).

Lemma IX *Let ∇f be semismooth and BD-regular. Let $\{\boldsymbol{w}_k\}$ be a generic sequence convergent to a critical point $\boldsymbol{w}^*$. Further at each $\boldsymbol{w}_k$ we generate a $\boldsymbol{s}_k$ by solving the Newton linear equation* (iv) *to satisfy the condition* (v) *where the ratio employed is* (xl)*. Then there exist $\delta > 0, M > 0$ such that for any $\boldsymbol{w}_k \in N(\boldsymbol{w}^*, \delta)$ we have*

$$\|\boldsymbol{w}_k + \boldsymbol{s}_k - \boldsymbol{w}^*\| \leq o(\|\boldsymbol{w}_k - \boldsymbol{w}^*\|) + M\eta_k\|\boldsymbol{w}_k - \boldsymbol{w}^*\|. \tag{xxxi}$$

Further if $\eta_k \to 0$ in generating $\boldsymbol{s}_k$, then we have

$$\lim_{k\to\infty} \frac{\|\boldsymbol{w}_k + \boldsymbol{s}_k - \boldsymbol{w}^*\|}{\|\boldsymbol{w}_k - \boldsymbol{w}^*\|} = 0. \tag{xxxii}$$

Proof Since ∇f is semismooth, by Lemma V we obtain that there exist constants $M_1, \delta_1 > 0$ such that if $\|\boldsymbol{w} - \boldsymbol{w}^*\| \leq \delta_1$ and $H \in \partial_B \nabla f(\boldsymbol{w})$ then H is nonsingular and

$$\|H^{-1}\| \leq M_1. \tag{xxxiii}$$

From semismoothness of ∇f we also have local Lipschitz continuity of ∇f. This, together with stationarity of $\boldsymbol{w}^*$ means that there exist two constants L, $\delta \in (0, \delta_1)$ such that

$$\|\nabla f(\boldsymbol{w})\| \leq L\|\boldsymbol{w} - \boldsymbol{w}^*\| \quad \forall \boldsymbol{w} \in N(\boldsymbol{w}^*, \delta). \tag{xxxiv}$$

By (xxxiv), (xxxiii), Lemma VI, (xl) and the stationarity of $\boldsymbol{w}^*$, for any $\boldsymbol{w}_k \in N(\boldsymbol{w}^*, \delta)$ we have:

$$\begin{aligned}
\|\boldsymbol{w}_k + \boldsymbol{s}_k - \boldsymbol{w}^*\| &= \|\boldsymbol{w}_k - \boldsymbol{w}^* - (H_k)^{-1}\nabla f(\boldsymbol{w}_k) + (H_k^{-1})(H_k\boldsymbol{s}_k + \nabla f(\boldsymbol{w}_k))\| \\
&\leq \|(H_k)^{-1}\| \cdot \big(\|H_k\boldsymbol{s}_k + \nabla f(\boldsymbol{w}_k)\| + \|\nabla f(\boldsymbol{w}_k) - \nabla f(\boldsymbol{w}^*) - H_k(\boldsymbol{w}_k - \boldsymbol{w}^*)\|\big) \\
&\leq o(\|\boldsymbol{w}_k - \boldsymbol{w}^*\|) + \eta_k M_1\|\nabla f(\boldsymbol{w}_k)\| \\
&\leq o(\|\boldsymbol{w}_k - \boldsymbol{w}^*\|) + \eta_k M\|\boldsymbol{w}_k - \boldsymbol{w}^*\|,
\end{aligned}$$

which proves (xxxi) with $M := M_1 L$. Now, if $\eta_k \to 0$, from (xxxi), we also get (xxxii). ■

The following is called Theorem 2 in the main paper.

Theorem X *Let f be strongly convex and ∇f be semismooth. Let $\mathcal{L}_1$ defined in* (viii) *be compact. Let $\{\boldsymbol{w}_k\}$ be generated by Algorithm II with $\gamma \in (0, \frac{1}{2})$ and $\eta_k \to 0$. Then:*
(1) there exists $\hat{k}$ such that (vi) *is satisfied with $\omega_k = 1 \forall k \geq \hat{k}$.*
(2) $\{\boldsymbol{w}_k\}$ is Q-SL convergent to a critical point $\boldsymbol{w}^*$.

Proof Let $\{\boldsymbol{w}_k\}$ be a sequence generated by Algorithm II in which $\gamma \in (0, \frac{1}{2})$, $\eta_k \to 0$. Since ∇f is semismooth we have that $f \in \mathrm{LC}^1$, and since $\mathcal{L}_1$ is compact we also have that the (generalized) Hessian is bounded. Furthermore, since H_k employed in Algorithm II is the actual (generalized) Hessian of f and f is strongly convex, we have that H_k is positive definite and bounded for all $\boldsymbol{w} \in \mathcal{L}_1$. Thus, all the assumptions of Proposition III of the supplementary are satisfied and we obtain (xi) and (xii). Now, from Theorem IV of the supplementary we obtain $\lim_{k\to\infty} \|\nabla f(\boldsymbol{w}_k)\| = 0$, which together with the strict convexity of f implies that $\{\boldsymbol{w}_k\}$ is convergent to a critical point $\boldsymbol{w}^*$. This (together with the fact that strong convexity of f implies BD-regularity of ∇f) means that we can apply Lemma IX on $\{\boldsymbol{w}_k\}$, since it is a sequence converging to a critical point $\boldsymbol{w}^*$ in which $\boldsymbol{s}_k$ is generated by solving the Newton linear equation (iv) to satisfy the condition (v) where the *ratio* employed is (xl).

Since $\eta_k \to 0$, we can now apply (xxxii), on both

$$\frac{\|\boldsymbol{w}_k + \boldsymbol{s}_k - \boldsymbol{w}^*\|}{\|\boldsymbol{w}_k - \boldsymbol{w}^*\|} \geq \frac{\|\boldsymbol{w}_k - \boldsymbol{w}^*\|}{\|\boldsymbol{w}_k - \boldsymbol{w}^*\|} - \frac{\|\boldsymbol{s}_k\|}{\|\boldsymbol{w}_k - \boldsymbol{w}^*\|} \geq 1 - \frac{\|\boldsymbol{s}_k\|}{\|\boldsymbol{w}_k - \boldsymbol{w}^*\|}$$

and

$$\frac{\|\boldsymbol{w}_k + \boldsymbol{s}_k - \boldsymbol{w}^*\|}{\|\boldsymbol{w}_k - \boldsymbol{w}^*\|} \geq \frac{\|\boldsymbol{s}_k\|}{\|\boldsymbol{w}_k - \boldsymbol{w}^*\|} - \frac{\|\boldsymbol{w}_k - \boldsymbol{w}^*\|}{\|\boldsymbol{w}_k - \boldsymbol{w}^*\|} \geq \frac{\|\boldsymbol{s}_k\|}{\|\boldsymbol{w}_k - \boldsymbol{w}^*\|} - 1$$

and obtain

$$\lim_{k\to\infty} \frac{\|\boldsymbol{s}_k\|}{\|\boldsymbol{w}_k - \boldsymbol{w}^*\|} = 1. \tag{xxxv}$$

From this and (xxxii) we can obtain

$$\lim_{k\to\infty} \frac{\|\boldsymbol{w}_k + \boldsymbol{s}_k - \boldsymbol{w}^*\|}{\|\boldsymbol{s}_k\|} = \lim_{k\to\infty} \frac{\|\boldsymbol{w}_k + \boldsymbol{s}_k - \boldsymbol{w}^*\|}{\|\boldsymbol{w}_k - \boldsymbol{w}^*\|} \cdot \frac{\|\boldsymbol{w}_k - \boldsymbol{w}^*\|}{\|\boldsymbol{s}_k\|} = 0,$$

which leads to

$$\|\boldsymbol{w}_k + \boldsymbol{s}_k - \boldsymbol{w}^*\| = o(\|\boldsymbol{s}_k\|). \tag{xxxvi}$$

Since ∇f is semismooth, we get that H is locally upper-bounded in $N(\boldsymbol{w}^*, \delta)$ for any δ. That is, there exists a $M > 0$ such that

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

From (xxxii) and the fact that $\{\boldsymbol{w}_k\} \to \boldsymbol{w}^*$, there exist a $\delta > 0$ and a $k(\delta)$ such that

$$\boldsymbol{w}_k \in N(\boldsymbol{w}^*, \delta) \text{ and } \boldsymbol{w}_k + \boldsymbol{s}_k \in N(\boldsymbol{w}^*, \delta) \; \forall k \geq k(\delta). \tag{xxxviii}$$

By Proposition VII and since $\nabla f(\boldsymbol{w}^*) = \boldsymbol{0}$, we can write

$$f(\boldsymbol{w}_k + \boldsymbol{s}_k) = f(\boldsymbol{w}^*) + \frac{1}{2}(\boldsymbol{w}_k + \boldsymbol{s}_k - \boldsymbol{w}^*)^T H_k (\boldsymbol{w}_k + \boldsymbol{s}_k - \boldsymbol{w}^*)$$
$$f(\boldsymbol{w}_k) = f(\boldsymbol{w}^*) + \frac{1}{2}(\boldsymbol{w}_k - \boldsymbol{w}^*)^T R_k (\boldsymbol{w}_k - \boldsymbol{w}^*),$$

where $H_k \in \partial\nabla f(\boldsymbol{z}_k)$ for some $\boldsymbol{z}_k \in (\boldsymbol{w}^*, \boldsymbol{w}_k + \boldsymbol{s}_k)$ and $R_k \in \partial\nabla f(\boldsymbol{u}_k)$ for some $\boldsymbol{u}_k \in (\boldsymbol{w}^*, \boldsymbol{w}_k)$. Subtracting the two above equalities we have

$$f(\boldsymbol{w}_k + \boldsymbol{s}_k) - f(\boldsymbol{w}_k) = -\frac{1}{2}(\boldsymbol{w}_k - \boldsymbol{w}^*)R_k(\boldsymbol{w}_k - \boldsymbol{w}^*) + o(\|\boldsymbol{s}_k\|^2),$$

where the last term follows from (xxxvi), (xxxvii) and (xxxviii). By subtracting $\frac{1}{2}\nabla f(\boldsymbol{w}_k)^T\boldsymbol{s}_k$, we get

$$\begin{aligned}
&f(\boldsymbol{w}_k + \boldsymbol{s}_k) - f(\boldsymbol{w}_k) - \frac{1}{2}\nabla f(\boldsymbol{w}_k)^T\boldsymbol{s}_k \\
&= -\frac{1}{2}(\boldsymbol{w}_k - \boldsymbol{w}^*)^T R_k(\boldsymbol{w}_k - \boldsymbol{w}^*) - \frac{1}{2}\nabla f(\boldsymbol{w}_k)^T\boldsymbol{s}_k + o(\|\boldsymbol{s}_k\|^2) \\
&= -\frac{1}{2}(\boldsymbol{w}_k + \boldsymbol{s}_k - \boldsymbol{w}^*)^T R_k(\boldsymbol{w}_k - \boldsymbol{w}^*) + \frac{1}{2}\boldsymbol{s}_k^T R_k(\boldsymbol{w}_k - \boldsymbol{w}^*) - \frac{1}{2}\nabla f(\boldsymbol{w}_k)^T\boldsymbol{s}_k + o(\|\boldsymbol{s}_k\|^2) \\
&= -\frac{1}{2}\boldsymbol{s}_k^T\left(\nabla f(\boldsymbol{w}_k) - R_k(\boldsymbol{w}_k - \boldsymbol{w}^*)\right) + o(\|\boldsymbol{s}_k\|^2) \\
&= -\frac{1}{2}\boldsymbol{s}_k^T\left(\nabla f(\boldsymbol{w}_k) - \nabla f(\boldsymbol{w}^*) - R_k(\boldsymbol{w}_k - \boldsymbol{w}^*)\right) + o(\|\boldsymbol{s}_k\|^2) \\
&= o(\|\boldsymbol{s}_k\|^2),
\end{aligned}$$

where the third equality follows from (xxxv), (xxxvi), and (xxxvii), while the last equality follows from Proposition VIII and again (xxxv). Thus, from the last equation we have

$$\begin{aligned}
f(\boldsymbol{w}_k + \boldsymbol{s}_k) - f(\boldsymbol{w}_k) &= \frac{1}{2}\nabla f(\boldsymbol{w}_k)^T\boldsymbol{s}_k + o(\|\boldsymbol{s}_k\|^2) + \gamma\nabla f(\boldsymbol{w}_k)^T\boldsymbol{s}_k - \gamma\nabla f(\boldsymbol{w}_k)^T\boldsymbol{s}_k \\
&\leq \gamma\nabla f(\boldsymbol{w}_k)^T\boldsymbol{s}_k + o(\|\boldsymbol{s}_k\|^2) - \left(\frac{1}{2} - \gamma\right) a_1\|\nabla f(\boldsymbol{w}_k)\|^2 \qquad \text{(xxxix)} \\
&\leq \gamma\nabla f(\boldsymbol{w}_k)^T\boldsymbol{s}_k + o(\|\boldsymbol{s}_k\|^2) - \left(\frac{1}{2} - \gamma\right) \frac{a_1}{{a_2}^2}\|\boldsymbol{s}_k\|^2,
\end{aligned}$$

where the first inequality follows from (xi), while the second inequality from (xii). The inequality (xxxix) proves that there exists a $\hat{k}$ for which (vi) is satisfied with $\omega_k = 1, \forall\ k > \hat{k}$ and $\gamma \in (0, \frac{1}{2})$. This means that $\forall k > \hat{k}$ we can replace $\boldsymbol{w}_k + \boldsymbol{s}_k$ with $\boldsymbol{w}_{k+1}$ in the limit (xxxii) and obtain that $\{\boldsymbol{w}_k\}$ Q-superlinearly converges to $\boldsymbol{w}^*$. ■

II. Details on Truncation Criteria in TNCG

We first recall all the *ratio*:

- residual:
$$ratio = \frac{\|\boldsymbol{g}_k + H_k\boldsymbol{s}_k^j\|}{\|\boldsymbol{g}_k\|}. \qquad \text{(xl)}$$
- residual$_{l1}$:
$$ratio = \frac{\|\boldsymbol{g}_k + H_k\boldsymbol{s}_k^j\|_1}{\|\boldsymbol{g}_k\|_1}. \qquad \text{(xli)}$$

Algorithm III: PCG for solving (xlvi). Assume M has been factorized to EE^T.

Given $\eta_k < 1, \Delta_k > 0$, let $\hat{\boldsymbol{s}} = \boldsymbol{0}, \hat{\boldsymbol{r}} = \hat{\boldsymbol{d}} = -E^{-1}\boldsymbol{g}_k, \gamma = \hat{\boldsymbol{r}}^T\hat{\boldsymbol{r}}$
while True **do**
  $\hat{\boldsymbol{v}} \leftarrow E^{-1}(H_k(E^{-T}\hat{\boldsymbol{d}}))$,
  $\alpha \leftarrow \|\hat{\boldsymbol{r}}\|^2/(\hat{\boldsymbol{d}}^T\hat{\boldsymbol{v}})$
  $\hat{\boldsymbol{s}} \leftarrow \hat{\boldsymbol{s}} + \alpha\hat{\boldsymbol{d}}$
  $\hat{\boldsymbol{r}} \leftarrow \hat{\boldsymbol{r}} - \alpha\hat{\boldsymbol{v}}$
  if $\|\hat{\boldsymbol{r}}\| < \eta_k\|\hat{\boldsymbol{g}}_k\|$ **then**
    return $\boldsymbol{s}_k = E^{-T}\hat{\boldsymbol{s}}$
  $\gamma^{\text{new}} \leftarrow \hat{\boldsymbol{r}}^T\hat{\boldsymbol{r}}$
  $\beta \leftarrow \gamma^{\text{new}}/\gamma$
  $\hat{\boldsymbol{d}} \leftarrow \hat{\boldsymbol{r}} + \beta\hat{\boldsymbol{d}}$
  $\gamma \leftarrow \gamma^{\text{new}}$

- quadratic:

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

Now we recall all the forcing sequences:

- constant:

$$\eta_k = c_0 \qquad \text{with } c_0 \in (0,1). \tag{xliii}$$

- adaptive:

$$\eta_k = \min\{c_1; c_2\|\boldsymbol{g}_k\|^{c_3}\}, \quad \text{with } c_1 \in (0,1),\ c_2 > 0,\ c_3 \in (0,1]. \tag{xliv}$$

- adaptive$_{l1}$:

$$\eta_k = \min\{c_1; c_2\|\boldsymbol{g}_k\|_1^{c_3}\}, \quad \text{with } c_1 \in (0,1),\ c_2 > 0,\ c_3 \in (0,1]. \tag{xlv}$$

III. Preconditioning

In this section we will show that Algorithm 2 from the original paper is equivalent to applying Algorithm III and then obtaining $\boldsymbol{s}$ from $\boldsymbol{s} = E^{-1}\hat{\boldsymbol{s}}$. For sake of simplicity in this section and in Algorithm III we will not use the iteration counter j and k will only be reported on H_k and $\boldsymbol{g}_k$. Let us write the instructions of the original CG method (Algorithm I) applied to the system

$$E^{-1}H_kE^{-T}\hat{\boldsymbol{s}} = -E^{-1}\boldsymbol{g}_k. \tag{xlvi}$$

At the internal generic CG step we have

$$\hat{\boldsymbol{s}} = \hat{\boldsymbol{s}} + \alpha\hat{\boldsymbol{d}}, \tag{xlvii}$$

where

$$\alpha = \frac{\|\hat{\boldsymbol{r}}\|^2}{\hat{\boldsymbol{d}}^T E^{-1}H_kE^{-T}\hat{\boldsymbol{d}}}$$

with

$$\hat{\boldsymbol{r}} = -E^{-1}H_k E^{-T}\hat{\boldsymbol{s}} - E^{-1}\boldsymbol{g}_k = E^{-1}\left(-H_k E^{-T}\hat{\boldsymbol{s}} - \boldsymbol{g}_k\right). \tag{xlviii}$$

Pre-multiplying (xlvii) by E^{-1}, using the transformation $\boldsymbol{s} = E^{-1}\hat{\boldsymbol{s}}$ and the definition

$$\boldsymbol{d} := E^{-1}\hat{\boldsymbol{d}} \tag{xlix}$$

we get

$$\boldsymbol{s} = \boldsymbol{s} + \alpha\boldsymbol{d}.$$

Again using the transformation $\boldsymbol{s} = E^{-1}\hat{\boldsymbol{s}}$, (xlviii) can be re-written to get

$$\hat{\boldsymbol{r}} = E^{-1}\left(-H_k\boldsymbol{s} - \boldsymbol{g}_k\right) = E^{-1}\boldsymbol{r}, \tag{l}$$

which together with (xlix) brings to

$$\alpha = \frac{\boldsymbol{r}^T M^{-1}\boldsymbol{r}}{\boldsymbol{d}^T\boldsymbol{v}}, \quad \text{where } \boldsymbol{v} = H_k\boldsymbol{d}.$$

Moreover we have that

$$\hat{\boldsymbol{d}} = \hat{\boldsymbol{r}} + \beta\hat{\boldsymbol{d}}, \tag{li}$$

where

$$\beta = \frac{\hat{\boldsymbol{r}}_{\text{new}}^T\hat{\boldsymbol{r}}_{\text{new}}}{\hat{\boldsymbol{r}}^T\hat{\boldsymbol{r}}} = \frac{\boldsymbol{r}_{\text{new}}^T M^{-1}\boldsymbol{r}_{\text{new}}}{\boldsymbol{r}^T M^{-1}\boldsymbol{r}}.$$

Pre-multiplying (li) by E^{-1}, from (l) and (xlix) we get

$$\boldsymbol{d} = \boldsymbol{r} + \beta\boldsymbol{d}.$$

Finally, defining the vector

$$\boldsymbol{z} := M^{-1}\boldsymbol{r} \quad \text{and the scalar } \gamma := \boldsymbol{r}^T\boldsymbol{z},$$

α and β can be re-written as

$$\alpha = \frac{\boldsymbol{r}^T\boldsymbol{z}}{\boldsymbol{d}^T\boldsymbol{v}} \quad \text{and} \quad \beta = \frac{\gamma^{\text{new}}}{\gamma}.$$

IV. Complete Numerical Results

For experiments we use two-class classification problems shown in Table II. Except for `yahookr` and `yahoojp`, others are available from LIBSVM Data Sets (2007). Table II shows the statistics of each data set used in our experiments. For a fair evaluation, all different settings are implemented based on the LIBLINEAR package[2]. To simulate the practical use we conduct a five-fold cross-validation to select the regularization parameter C_{Best} that

2. Exceptions are for Scikit and ScikitArmijo of Figure 1 from the original paper, where Scikit-learn is used. In particular, to train a `LogisticRegression` model, the user should select the `newton-cg` option for the parameter `solver`. By default Scikit-learn solves a slightly different optimization problem with a bias term. To solve (i), the bias term is thus ignored by setting the option `fit_intercept` to `False`.

achieves the best validation accuracy (see Table II for the resulting C). Then in experiments we consider $C = C_{\text{Best}} \times \{1, 100\}$.

In figures below we show the total number of CG steps needed to solve the training problem (i). On the y axis we report the relative reduction of the function value, computed by

$$\frac{f(\boldsymbol{w}_k) - f(\boldsymbol{w}^*)}{f(\boldsymbol{w}^*)},$$

where $\boldsymbol{w}^*$ is the optimal solution[3] of (i). The four horizontal lines in figures below indicate places where the following stopping condition is met respectively with $\epsilon = \{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$

$$\|\boldsymbol{g}_k\| \leq \epsilon \frac{\min\{\#\text{pos}, \#\text{neg}\}}{l} \cdot \|\boldsymbol{g}_1\|, \tag{lii}$$

where #pos, #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 (lii) 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. For the Armijo line search, we follow Hsia et al. (2017) to have $\delta = 0.5$ for deciding the step size $\alpha \in \{1, \delta, \delta^2, \dots\}$ and $\gamma = 0.01$ in the sufficient decrease condition (vi).

IV.1 Selection of the Forcing Sequence in Truncation Criteria

See complete results in Figures i and ii.

IV.2 Selection of the Constant Threshold in the Inner Stopping Condition

In this section, results in Figures iii and iv are obtained by comparing

- rescons: *ratio*= (xl), η_k= (xliii), $c_0 = 0.1$;
- rescons05: *ratio*= (xl), η_k= (xliii), $c_0 = 0.5$;
- rescons09: *ratio*= (xl), η_k= (xliii), $c_0 = 0.9$;
- quadcons: *ratio*= (xlii), η_k= (xliii), $c_0 = 0.1$;
- quadcons05: *ratio*= (xlii), η_k= (xliii), $c_0 = 0.5$;
- quadcons09: *ratio*= (xlii), η_k= (xliii), $c_0 = 0.9$.

By using the same color (red) for all the settings with a residual *ratio* (xl) and one other (blue) for all those with a quadratic *ratio* (xlii), in Figures iii and iv we want to observe the robustness of the two *ratios* w.r.t. a modification of the constant threshold c_0:

- When $C = C_{\text{Best}}$, there is no evidence of one implementation that is more robust than the other.

3. In practice $\boldsymbol{w}^*$ is not available, so we run enough iterations to get a good approximate solution.

- When $C = 100C_{\text{Best}}$, the quadratic *ratio* is more sensible on `covtype`, `rcv1` and also on `url`, while the residual one is more sensible on `yahoojp`, `yahookr`, `kdda`, `kddb`, `kdd12` and also on `real-sim`.

The value $C = 100C_{\text{Best}}$ leads to a more difficult optimization problem and it is thus reasonable that the modification of the constant threshold c_0 has here a stronger effect on the convergence speed. From the observations above we can conclude that the quadratic *ratio* is more robust than the residual one, especially on difficult optimization problems.

IV.3 Comparison Between Adaptive and Constant Forcing Sequences

In this section, results in Figures v and vi are obtained by comparing

- rescons05: *ratio*= (xl), η_k= (xliii), $c_0 = 0.5$;
- resada: *ratio*= (xl), η_k= (xliv);
- quadcons05: *ratio*= (xlii), η_k= (xliii), $c_0 = 0.5$;
- quadada: *ratio*= (xlii), η_k= (xliv).

From Figures v and vi we can observe that the constant *forcing sequence* and the adaptive one are (almost) always leading to an identical convergence speed. This means that the constant $c_1 = 0.5$ is (almost) always smaller than $c_2\|\nabla f(\boldsymbol{w}_k)\|^{c_3}$, with $c_2 = 1$ and $c_3 = 0.5$. In fact, this result depends also on the choice of c_1, c_2 and c_3, since, for instance, with a bigger initial threshold $c_1 = 0.9$ this overlapping effect would not be so common. On the other side, the setting $(c_1 = 0.5, c_2 = 1, c_3 = 0.5)$ is the one used in Scikit-learn and suggested also in Nocedal and Wright (1999). Moreover, experiments with many other settings of c_1, c_2 and c_3 have been carried out and there is no evidence of configurations that work consistently better than the standard one.

IV.4 Comparison Between l2-Norm and l1-Norm and with TR

See complete results in Figures vii and viii. On the dataset `covtype` with $C = 100C_{\text{Best}}$ both the setting quadada and quadada_l1 are facing slow convergence issues because of the threshold c_1. Comparing with resada it seems that in early iterations the Newton equation needed to be solved more accurately. Since in Section IV.2 it is not easy to investigate the choice of the threshold c_1 (because of the single color blue) we decided to deepen the analysis on quadada in Section IV.9.

IV.5 Analysis via Conditions for Global Convergence

In the convergence proof from Section I.2 one of the key properties for proving global convergence is obtaining (xi) and (xii). The following combination of the two requirements is called angle condition

$$-\boldsymbol{g}_k^T \boldsymbol{s}_k \geq c\|\boldsymbol{g}_k\|\|\boldsymbol{s}_k\|. \tag{liii}$$

In fact, most of the line-search based optimization methods (see Section 3.2 from Nocedal and Wright (1999)) requires the direction $\boldsymbol{s}_k$ to satisfy (liii) to ensure global convergence[4].

4. Note anyway that global convergence is always guaranteed for Algorithm II as proved in Section I.2, so the discussion in this section is only pointing out an undesired issue that might come out in practice.

This suggests us the idea of analyzing different inner stopping criteria on this value

$$\cos(-\boldsymbol{g}_k, \boldsymbol{s}_k) = \frac{-\boldsymbol{g}_k^T \boldsymbol{s}_k}{\|\boldsymbol{g}_k\| \|\boldsymbol{s}_k\|}. \tag{liv}$$

The term (liv) is the cosine of the angle between the direction $\boldsymbol{s}_k$ and the anti-gradient $-\boldsymbol{g}_k$. Inequality (liii) 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 $\boldsymbol{s}_k$ cannot be completely orthogonal to the anti-gradient. Otherwise, $\boldsymbol{s}_k$ 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 (liii) more easily satisfied. Detailed experiments in Figures ix and x confirm this result.

The condition (liii) is also useful to investigate the relationship between the regularization parameter C and the difficulty of the optimization problem (i). Many past experiments have shown that when C is large, the optimization problem is more difficult and slow convergence more frequently occurs Hsia et al. (2017, 2018). In particular, when the Hessian of the objective function is positive definite and we approximate $\boldsymbol{s}_k \approx {H_k}^{-1} \boldsymbol{g}_k$ we have

$$\begin{aligned} \boldsymbol{g}_k^T \boldsymbol{s}_k &\approx -\boldsymbol{g}_k^T {H_k}^{-1} \boldsymbol{g}_k \leq -\frac{\|\boldsymbol{g}_k\|^2}{\lambda_{\max}(H_k)} \\ \|\boldsymbol{s}_k\| &\approx \|H_k^{-1} \boldsymbol{g}_k\| \leq \frac{\|\boldsymbol{g}_k\|}{\lambda_{\min}(H_k)}. \end{aligned} \tag{lv}$$

Now, from (lv) and (ix) we have

$$-\boldsymbol{g}_k^T \boldsymbol{s}_k \geq \frac{M_2}{M_1} \|\boldsymbol{g}_k\| \|\boldsymbol{s}_k\|. \tag{lvi}$$

This means that when the regularization parameter C is large (e.g., $C = 100 C_{\text{Best}}$), (lvi) shows that the lower bound of the cosine value in (liv) we can derive is smaller. Thus the angle condition (liii) is harder to be satisfied. Note that equation (liii) 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 (liii), while what we hope is that the left-hand side of (liii) is large so that it can be easily satisfied. Unfortunately (lvi) shows that for large C, the left-hand side of (liii) 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 $-\boldsymbol{g}_k$ and the resulting direction $\boldsymbol{s}_k$. In Figures ix and x we compare

- quadada: *ratio*= (xlii), η_k= (xliv);
- rescons09: *ratio*= (xlii), η_k= (xliii) $c_0 = 0.9$;
- resada: *ratio*= (xl), η_k= (xliv).

Cosines have been calculated until the outer termination criterion (lii) is satisfied with $\epsilon = 10^{-4}$. In Figures ix and x, 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 ix-x and i-ii 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, Figures i and ii have shown that it behaves worse in terms of convergence speed.

  Long continuous regions of low cosine in Figures ix-x might correspond to regions of slow convergence in Figures i-ii (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 (i) as C changes.

- In Figures i-ii, quadada converges much faster than resada. From Figures ix-x we observe that quadada's cosine is generally higher than resada's. This indicates that in the early Newton iterations, the resulting direction $\boldsymbol{s}_k$ 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.

IV.6 Preconditioning: Effect of Preconditioning on Residual and Quadratic Ratio

See complete results in Figures xi and xii.

IV.7 Preconditioning: Comparison with Trust Region and Other Rules

See complete results in Figures xiii and xiv.

IV.8 Additional Results in the Preconditioned Case: Comparison Between Adaptive and Constant Forcing Sequences

In this section, results in Figures xv and xvi are obtained by comparing

- rescons05_p: *ratio*= (xl), η_k= (xliii), $c_0 = 0.5$, preconditioned;
- resada_p: *ratio*= (xl), η_k= (xliv), preconditioned;
- quadcons05_p: *ratio*= (xlii), η_k= (xliii), $c_0 = 0.5$, preconditioned;
- quadada_p: *ratio*= (xlii), η_k= (xliv), preconditioned.

Also in this case, like in the non-preconditioned one of Section IV.3, the constant *forcing sequence* and the adaptive one are (almost) always leading to an identical convergence speed.

IV.9 Additional Results in the Preconditioned Case: Selection of the c_1 Threshold in the Quadratic Adaptive Rule

In this section, results in Figures xvii and xviii are obtained by comparing

- quadada01: *ratio*= (xlii), η_k= (xliv), $c_1 = 0.1, c_2 = 1, c_3 = 0.5$;
- quadada01_p: *ratio*= (xlii), η_k= (xliv) $c_1 = 0.1, c_2 = 1, c_3 = 0.5$, preconditioned;
- quadada: *ratio*= (xlii), η_k= (xliv), $c_1 = 0.5, c_2 = 1, c_3 = 0.5$;
- quadada_p: *ratio*= (xlii), η_k= (xliv), $c_1 = 0.5, c_2 = 1, c_3 = 0.5$ preconditioned.

From Figures xvii and xviii we can make the following observations:

- Using a smaller c_1 threshold ($c_1 = 0.1$) is solving the issues encountered by quadada on `covtype` with $C = 100C_{\text{Best}}$ in Figure viii.
- Both the settings quadada01 and quadada01_p are generally slower than quadada and quadada_p.
- The slow convergence issue encountered by quadada on `covtype` with $C = 100C_{\text{Best}}$ in Figure viii is also solved by the use of preconditioning (quadada_p).

IV.10 Additional Results in the Preconditioned Case: Comparison Against a Residual Constant Rule with Higher Constant Threshold

In this section, results in Figures xix and xx are obtained by comparing

- rescons09: *ratio*= (xl), η_k= (xliii), $c_0 = 0.9$;
- rescons09_p: *ratio*= (xl), (xliii), $c_0 = 0.9$, preconditioned;
- quadada: *ratio*= (xlii), η_k= (xliv);
- quadada_p: *ratio*= (xlii), η_k= (xliv), preconditioned.

From Figures xix and xx we can make the following observations:

- The settings rescons09_p and quadada_p are often obtaining a very similar speed of convergence.
- There are differences on $C = C_{\text{Best}}$ in which rescons09_p is faster than quadada_p (`rcv1`, `real-sim`) and quadada_p is faster than rescons09_p (`covtype` and `criteo` on the fourth line), but the detachment is less than 20 CG steps.
- By looking at `yahookr` with $C = C_{\text{Best}}$, we can see that rescons09_p is noticeably faster than quadada_p (at the third line is around twice faster, by 150 CG steps). The opposite situation is instead happening on `yahookr` with $C = 100C_{\text{Best}}$, where being twice faster on the third line means saving around 750 CG steps.

IV.11 Additional Results in the Preconditioned Case: Comparison Between l2-Norm and l1-Norm

In this section, results in Figures xxi and xxii are obtained by comparing

- resada_l1: *ratio*= (xli), η_k= (xlv);
- rescons_l1_p: *ratio*= (xli), (xlv), preconditioned;
- resada: *ratio*= (xl), η_k= (xliv);
- resada_p: *ratio*= (xl), η_k= (xliv), preconditioned.

From Figures xxi and xxii we can notice that rescons_l1_p is often slower than resada_p, so also in the preconditioned case there is no reason to employ L1-norm instead of L2-norm.

IV.12 Results for L2-loss SVM

In this section, results in Figures xxiii and xxiv are obtained by employing the ξ_{L2} loss as defined in (ii) and we compare:

- tr_rescons: *ratio*= (xl), η_k= (xliii), $c_0 = 0.1$, Trust Region instead of line search;
- rescons09: *ratio*= (xl), η_k= (xliii), $c_0 = 0.9$;
- resada: *ratio*= (xl), η_k= (xliv);
- quadada: *ratio*= (xlii), η_k= (xliv).

IV.13 Results for L2-loss SVM in the Preconditioned Case

In this section, results in Figures xxv and xxvi are obtained by employing the ξ_{L2} loss as defined in (ii) and we compare:

- tr_rescons_p: *ratio*= (xl), η_k= (xliii), $c_0 = 0.1$, Trust Region instead of line search, preconditioned;
- rescons09_p: *ratio*= (xl), η_k= (xliii), $c_0 = 0.9$, preconditioned;
- resada_p: *ratio*= (xl), η_k= (xliv), preconditioned;
- quadada_p: *ratio*= (xlii), η_k= (xliv), preconditioned.

IV.14 Comparison with Other State-of-the-Art Software

In this section we are comparing the proposed TNCG implementation with the two state-of-the-art software SAG (Schmidt et al., 2017) and SAGA (Defazio et al., 2014). A handy implementation of the two methods is provided by Scikit-learn. All the settings are the same described in Footnote 1 of Section IV, except for the fact that for training a `LogisticRegression` model with SAG or SAGA the user should select the `sag` or `saga` option for the parameter `solver`. Few lines of python/cython code have been added in the Scikit implementation of SAG/SAGA to compute the function value and measure the CPU

time (with the function `time.time()` of the library `time`). This additional computation has not been considered in the total time employed. On the other side, in the case of the LIBLINEAR implementation the time has been measured with the function `clock()` of the library `time.h`. For both the implementations the starting time has been taken immediately before the main optimization cycle. Experiments are conducted on a machine with an AMD EPYC 7401 CPU and 128 GB of memory. Results in Figures xxvii and xxviii are obtained by comparing quadada_p (*ratio*= (xlii), η_k= (xliv), preconditioned), SAG and SAGA.

Appendix A. Gradient and Hessian Details of the Losses from (ii)

Let $X = [\boldsymbol{x}_1, \dots, \boldsymbol{x}_l]^T$ be the data matrix.

A.1 Logistic Loss

In this section we compute the gradient and the Hessian of f when the loss employed is ξ_{LR}. Let $\sigma(t) := (1 + e^{-t})^{-1}$.
The gradient of f is

$$\nabla f(\boldsymbol{w}) = \boldsymbol{w} + C \sum_{i=1}^{l} (\sigma(y_i \boldsymbol{w}^T \boldsymbol{x}_i) - 1) y_i \boldsymbol{x}_i$$

The Hessian of f is

$$\nabla^2 f(\boldsymbol{w}) = I + CX^T DX$$

with D a diagonal matrix such that $D_{ii}(\boldsymbol{w}) = \sigma(y_i \boldsymbol{w}^T \boldsymbol{x}_i)(1 - \sigma(y_i \boldsymbol{w}^T \boldsymbol{x}_i))$.

A.2 L2-loss SVM

In this section we compute the gradient and the Hessian of f when the loss employed is ξ_{L2}. Let $\boldsymbol{y} = [y_1, \dots, y_l]^T$. The gradient of f is

$$\nabla f(\boldsymbol{w}) = \boldsymbol{w} + 2C {X_{\mathcal{I},:}}^T \left(X_{\mathcal{I},:} \boldsymbol{w} - \boldsymbol{y}_{\mathcal{I}} \right),$$

where $\mathcal{I} = \left\{ i \in \{1, \dots l\} | 1 - y_i \boldsymbol{w}^T \boldsymbol{x}_i > 0 \right\}$.
Now we define

$$\phi(t) = \max(0, 1 - t)^2, \qquad \phi'(t) = \begin{cases} -2(1-t) & \text{if } 1 - t > 0, \\ 0 & \text{otherwise,} \end{cases}$$

$$\partial_B \phi'(t) = \begin{cases} 2 & \text{if } 1 - t > 0, \\ 0 & \text{if } 1 - t < 0, \\ \{0, 2\} & \text{if } 1 - t = 0, \end{cases}$$

$$\partial \phi'(t) = \text{conv}\{\partial_B \phi'(t)\} = \begin{cases} 2 & \text{if } 1 - t > 0, \\ 0 & \text{if } 1 - t < 0, \\ \mu \text{ with } \mu \in [0, 2] & \text{if } 1 - t = 0, \end{cases}$$

where $\text{conv}\{\cdot\}$ is the convex hull of a set. Therefore, the generalized Hessian of f is

$$\partial \nabla f = I + CX^T DX, \quad \text{with } D \text{ a diagonal matrix such that } D_{ii}(\boldsymbol{w}) = \partial \phi'(y_i \boldsymbol{w}^T \boldsymbol{x}_i).$$

An easy way to calculate $\partial\nabla f$ in practice is to replace the original $D_{ii}(\boldsymbol{w})$ with the following

$$\bar{D}_{ii}(\boldsymbol{w}) = \begin{cases} 2, & \text{if } 1 - y_i \boldsymbol{w}^T \boldsymbol{x}_i > 0, \\ 0, & \text{otherwise.} \end{cases}$$

A.3 Cost of Armijo, Wolfe and Exact Line Searches

Here we recall the cost of Armijo (Section 2.1 of Hsia et al. (2017)) and exact line search (Section 3.3 of Hsia et al. (2017)), and we show how to obtain the same save of operations even for Wolfe line search.

- Exact requires $\mathcal{O}(l)$ operations for each trial ω,
- Armijo requires $\mathcal{O}(l)$ operations for each trial ω,
- Wolfe requires $\mathcal{O}(l)$ operations for each trial ω.

To show the trick in computing the Wolfe condition, we first recall the same trick for Armijo (vi). For computing f at each ω, we can store each $\boldsymbol{x}_i^T \boldsymbol{w}_k$, $\boldsymbol{x}_i^T \boldsymbol{s}_k$ $\boldsymbol{w}_k^T \boldsymbol{s}_k$, ${\boldsymbol{w}_k}^T \boldsymbol{w}_k$ and ${\boldsymbol{s}_k}^T \boldsymbol{s}_k$ respectively in these scalars $x_{\bar{i}}^T w, x_{\bar{i}}^T s, \bar{w}^T s, \bar{w}^T w$ and $\bar{s}^T s$ and compute f at $(\boldsymbol{w}_k + \omega \boldsymbol{s}_k)$ in the following way

$$\begin{aligned} f(\boldsymbol{w}_k + \omega \boldsymbol{s}_k) &= \frac{1}{2}(\boldsymbol{w}_k + \omega \boldsymbol{s}_k)^T (\boldsymbol{w}_k + \omega \boldsymbol{s}_k) + C \sum_{i=1}^{l} \xi(y_i (\boldsymbol{w}_k + \omega \boldsymbol{s}_k)^T \boldsymbol{x}_i) \\ &= \frac{1}{2}(\bar{w}^T w + 2\omega \bar{w}^T s + \omega^2 \bar{s}^T s) + C \sum_{i=1}^{l} \xi(y_i (x_{\bar{i}}^T w + \omega x_{\bar{i}}^T s)). \end{aligned}$$

Thus the cost of Armijo line search is $\mathcal{O}(l)\times$ (line search steps). The cost of Wolfe condition (vii) is also $\mathcal{O}(l)\times$ (line search steps), because we don't have to compute $\nabla f(\boldsymbol{w}_k + \omega \boldsymbol{s}_k)$ at each ω, but we can directly compute $\nabla f(\boldsymbol{w}_k + \omega \boldsymbol{s}_k)^T \boldsymbol{s}_k$. Thus, again by storing everything as above, we have:

$$\begin{aligned} \nabla f(\boldsymbol{w}_k + \omega \boldsymbol{s}_k)^T \boldsymbol{s}_k &= \boldsymbol{w}_k^T \boldsymbol{s}_k + \omega \boldsymbol{s}_k^T \boldsymbol{s}_k + C \sum_{i=1}^{l} \xi'(y_i (\boldsymbol{w}_k + \omega \boldsymbol{s}_k)^T \boldsymbol{x}_i) \cdot y_i \cdot \boldsymbol{x}_i^T \boldsymbol{s}_k \\ &= \bar{w}^T w + \omega \bar{s}^T s + C \sum_{i=1}^{l} \xi'(y_i (x_{\bar{i}}^T w + \omega x_{\bar{i}}^T s)) \cdot y_i \cdot x_{\bar{i}}^T s. \end{aligned}$$

Symbol	Meaning
n	number of instances
l	number of features
k	outer iteration index (TN)
j	inner step index (CG)
i	index of the dataset instance
$\boldsymbol{w}$	vector of weights ($\in \mathbb{R}^n$)
f	objective function
$\boldsymbol{g}$ or ∇f	gradient of f ($\in \mathbb{R}^n$)
H or $\nabla^2 f$	Hessian of f ($\in \mathbb{R}^{(n\times n)}$)
y	label
$\boldsymbol{x}$	feature vector ($\in \mathbb{R}^l$)
ξ	loss function
X	matrix vector ($\in \mathbb{R}^{(l\times n)}$)
$\boldsymbol{s}$	TN direction ($\in \mathbb{R}^n$)
C	regularization parameter
$Q_k(\cdot)$	quadratic approximation of f
I	identity matrix
η	forcing sequence
ω	step length
γ	constant of Armijo line search
$\mathcal{L}$	level set
$\boldsymbol{d}$	conjugate direction ($\in \mathbb{R}^n$)
B	a positive definite matrix ($\in \mathbb{R}^{(n\times n)}$)
M_1, M_2	bounding constants
$\boldsymbol{r}$	Newton equation residual ($\in \mathbb{R}^n$)
$\partial_B \nabla f$	B-subdifferential of ∇f
$\partial \nabla f$	generalized Hessian of f
$\|\cdot\|$	l2-norm
$\|\cdot\|_1$	l1-norm
c_0, c_1, c_2, c_3	constants in the forcing sequences
ϵ	constant of the outer stopping condition
M	preconditioning matrix ($\in \mathbb{R}^{(n\times n)}$)
E	a factor matrix of $M = EE^T$ ($\in \mathbb{R}^{(n\times n)}$)
$\hat{\boldsymbol{s}}$	preconditioned TN direction ($\in \mathbb{R}^n$)
$\hat{\boldsymbol{r}}$	preconditioned residual ($\in \mathbb{R}^n$)
$\hat{\boldsymbol{g}}$	preconditioned gradient ($\in \mathbb{R}^n$)
$\hat{H}$	preconditioned Hessian ($\in \mathbb{R}^{(n\times n)}$)
$\|\cdot\|_A$	norm induced by the matrix A
$\lambda_{\min}(\cdot), \lambda_{\max}(\cdot)$	min and max eigenvalue
C_{Best}	C achieving the best validation accuracy
$N(\boldsymbol{w}, \delta)$	neighbor of $\boldsymbol{w}$ with radius δ

Table I: Notation table.

Data sets	#instances	#features	density	$\log_2(C_{\text{Best}})$	
				LR	L2
news20	19,996	1,355,191	0.0336%	9	3
w8a	49,749	300	3.8834%	8	2
covtype	581,012	54	0.22 %	-23	-26
rcv1	20,242	47,236	0.1568%	6	0
url	2,396,130	3,231,962	0.0036%	-7	-10
real-sim	72,309	20,958	0.2448%	3	-1
yahoojp	176,203	832,026	0.0160%	3	-1
yahookr	460,554	3,052,939	0.0111%	6	1
kdd2010a	8,407,752	20,216,831	0.0001%	-3	-5
kdd2010b	19,264,097	29,890,095	0.0001%	-1	-4
HIGGS	11,000,000	28	92.1057%	-6	-12
webspam	350,000	16,609,143	0.0220%	2	-3
criteo	45,840,617	1,000,000	0.0039%	-15	-12
kdd2012	149,639,105	54,686,452	0.00002%	-4	-11

Table II: Data statistics. The density is the average number of non-zeros per instance. C_{Best} is the regularization parameter selected by cross validation.

References

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

Francisco Facchinei. Minimization of SC1 functions and the Maratos effect. *Operations Research Letters*, 17(3):131–138, 1995.

Magnus Rudolph Hestenes and Eduard Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49(1):409–436, 1952.

Chih-Yang Hsia, Ya Zhu, and Chih-Jen Lin. A study on trust region update rules in Newton methods for large-scale linear classification. In *Proceedings of the Asian Conference on Machine Learning (ACML)*, 2017. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/newtron/newtron.pdf`.

Chih-Yang Hsia, Wei-Lin Chiang, and Chih-Jen Lin. Preconditioned conjugate gradient methods in truncated Newton frameworks for large-scale linear classification. In *Proceedings of the Asian Conference on Machine Learning (ACML)*, 2018. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/tron_pcg/precondition.pdf`.

Ching-Pei Lee, Po-Wei Wang, Weizhu Chen, and Chih-Jen Lin. Limited-memory common-directions method for distributed optimization and its application on empirical risk minimization. In *Proceedings of SIAM International Conference on Data Mining (SDM)*, 2017. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/l-commdir/l-commdir.pdf`.

LIBSVM Data Sets, 2007. `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets`.

José Mario Martínez and Liqun Qi. Inexact newton methods for solving nonsmooth equations. *Journal of Computational and Applied Mathematics*, 60(1-2):127–145, 1995.

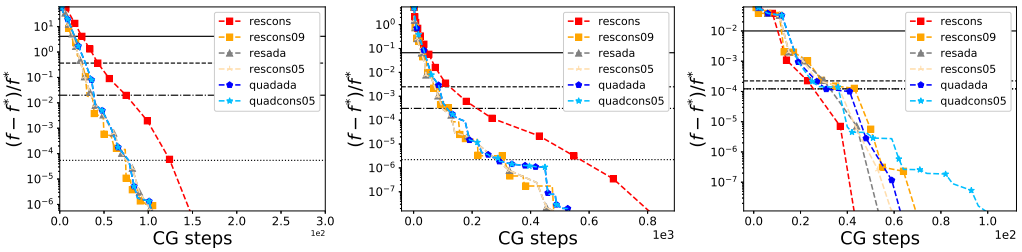
Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer-Verlag, New York, NY, 1999.

Jong-Shi Pang and Liqun Qi. Nonsmooth equations: Motivation and algorithms. *SIAM Journal on Optimization*, 3:443–465, 1993.

Houduo Qi and Defeng Sun. A quadratically convergent Newton method for computing the nearest correlation matrix. *SIAM Journal on Matrix Analysis and Applications*, 28(2): 360–385, 2006.

Liqun Qi. Convergence analysis of some algorithms for solving nonsmooth equations. *Mathematics of Operations Research*, 18(1):227–244, 1993.

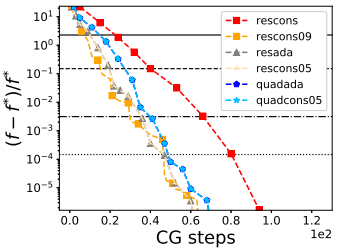
Mark Schmidt, Nicolas Le Roux, and Francis Bach. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 162(1-2):83–112, 2017.



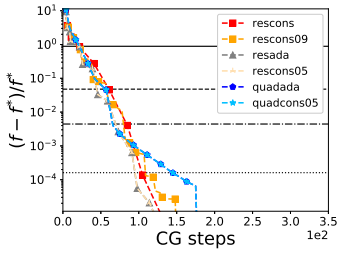
(a) `new20`

(b) `w8a`

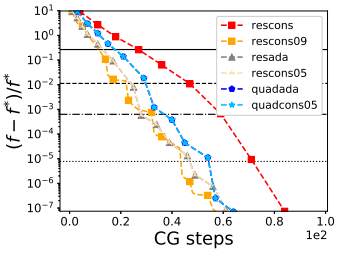
(c) `covtype`



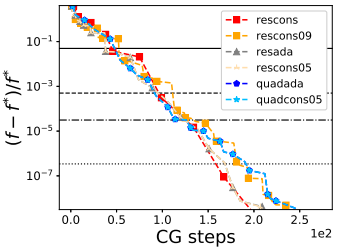
(d) `rcv1`



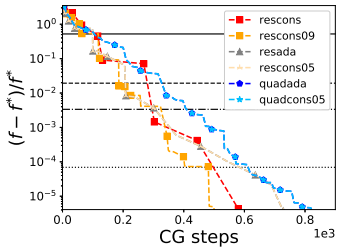
(e) `url`



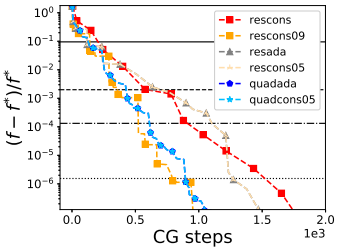
(f) `real-sim`



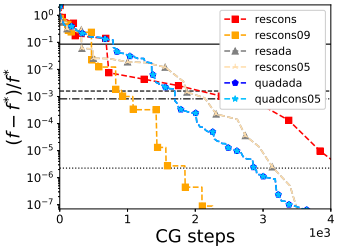
(g) `yahoojp`



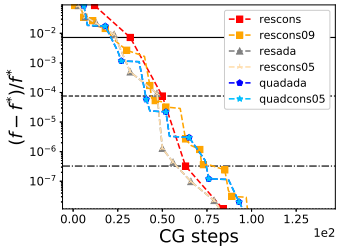
(h) `yahookr`



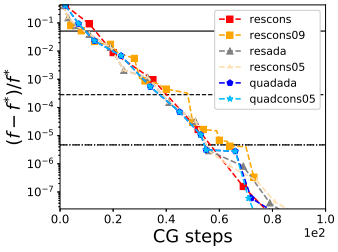
(i) `kdd2010a`



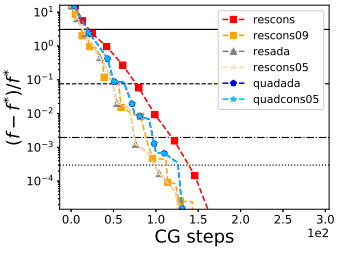
(j) `kdd2010b`



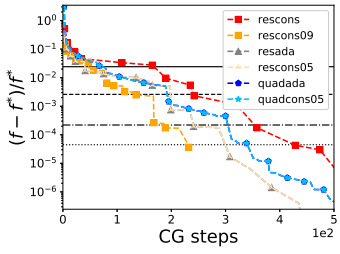
(k) `HIGGS`



(l) `criteo`

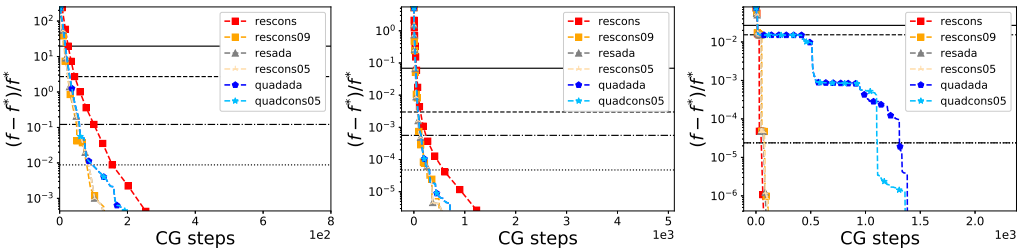


(m) `webspam`



(n) `kdd2012`

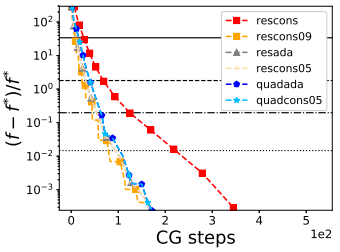
Figure i: A comparison of various forcing sequences in the inner stopping condition. We show the convergence of a truncated Newton method for logistic regression. The x-axis shows the cumulative number of CG steps. Loss=LR and $C = C_{\text{Best}}$ in each sub-figure.



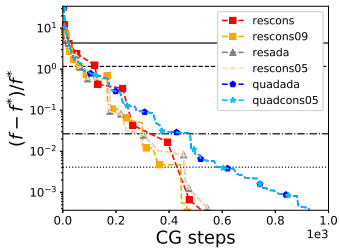
(a) `new20`

(b) `w8a`

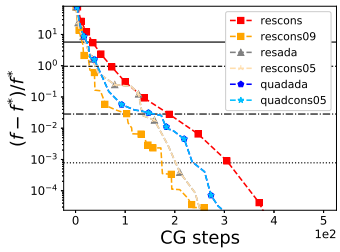
(c) `covtype`



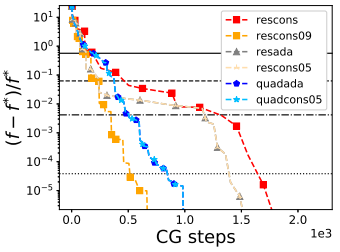
(d) `rcv1`



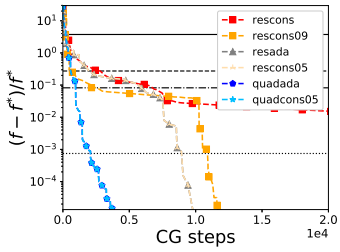
(e) `url`



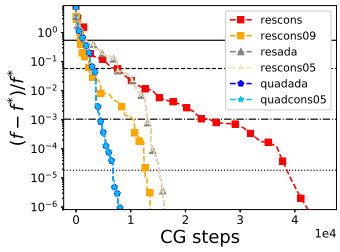
(f) `real-sim`



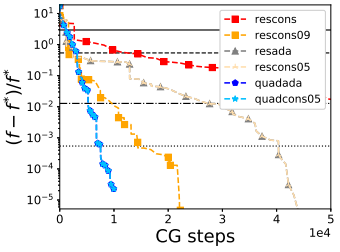
(g) `yahoojp`



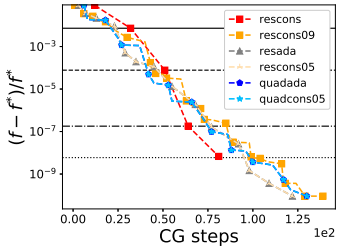
(h) `yahookr`



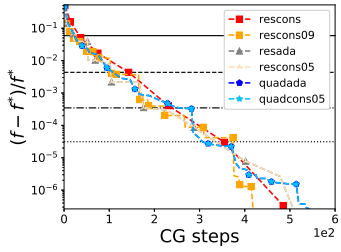
(i) `kdd2010a`



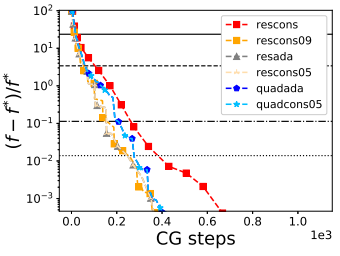
(j) `kdd2010b`



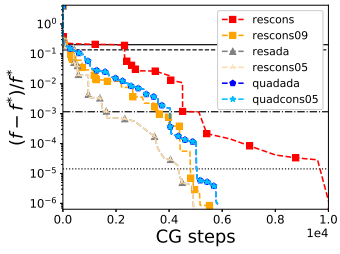
(k) `HIGGS`



(l) `criteo`

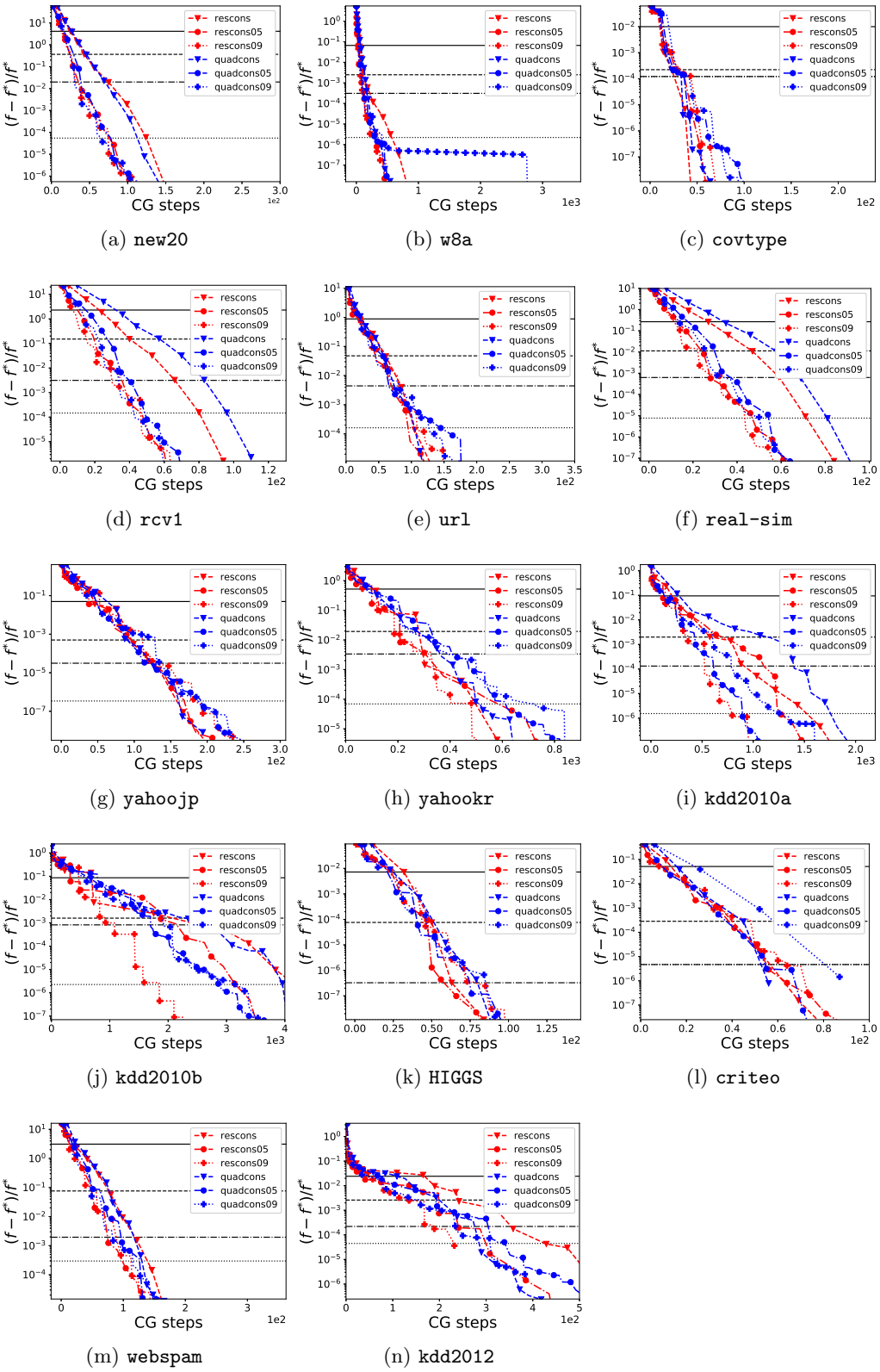


(m) `webspam`



(n) `kdd2012`

Figure ii: A comparison of various forcing sequences in the inner stopping condition. We show the convergence of a truncated Newton method for logistic regression. The x-axis shows the cumulative number of CG steps. Loss=LR and $C = 100C_{\text{Best}}$ in each sub-figure.



(a) `new20` (b) `w8a` (c) `covtype`

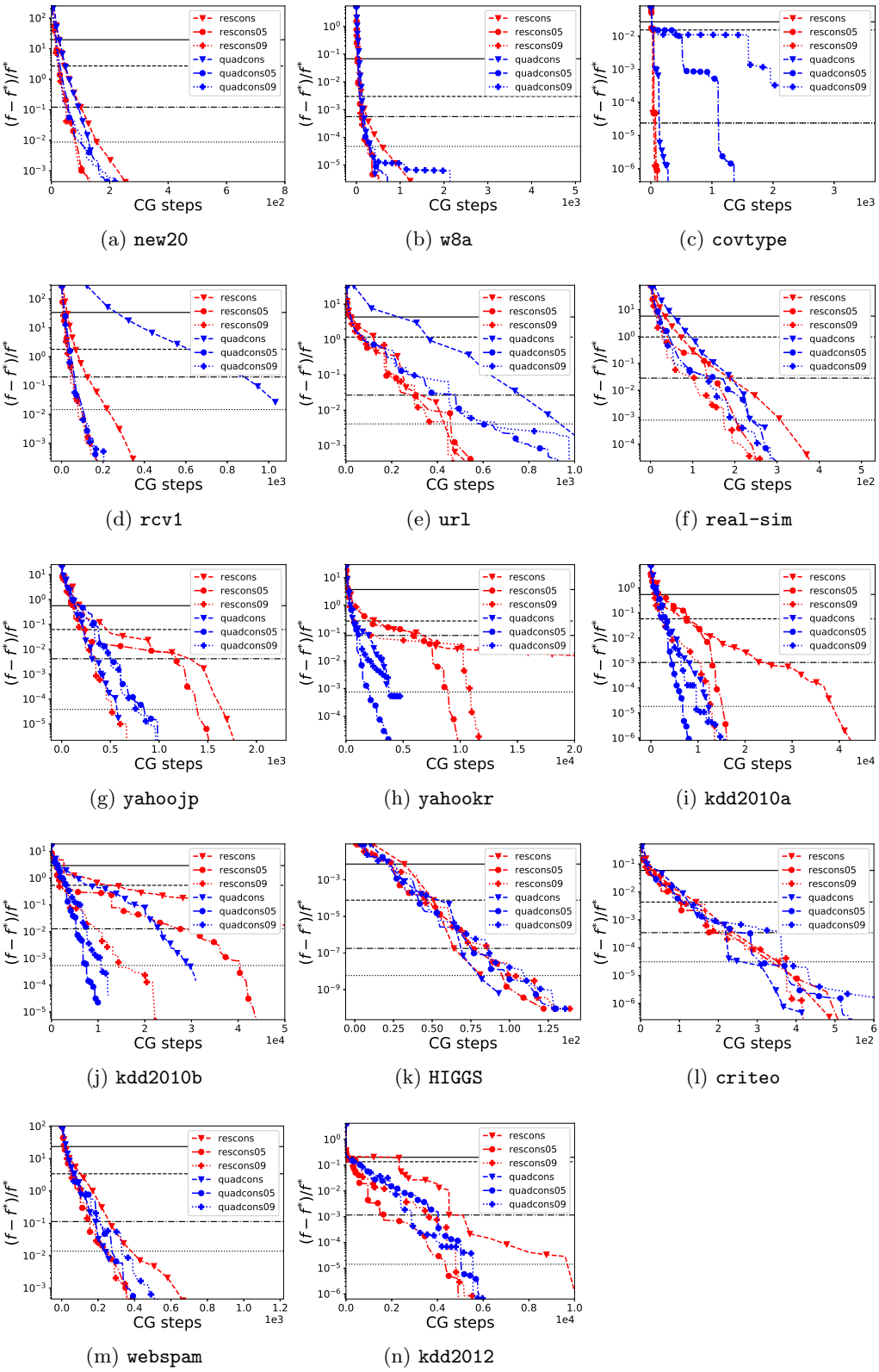
(d) `rcv1` (e) `url` (f) `real-sim`

(g) `yahoojp` (h) `yahookr` (i) `kdd2010a`

(j) `kdd2010b` (k) `HIGGS` (l) `criteo`

(m) `webspam` (n) `kdd2012`

Figure iii: A comparison of different constant thresholds in the inner stopping condition. We show the convergence of a truncated Newton method for logistic regression. The x-axis shows the cumulative number of CG steps. Loss=LR and $C = C_{\text{Best}}$ in each sub-figure.



(a) `new20` (b) `w8a` (c) `covtype`

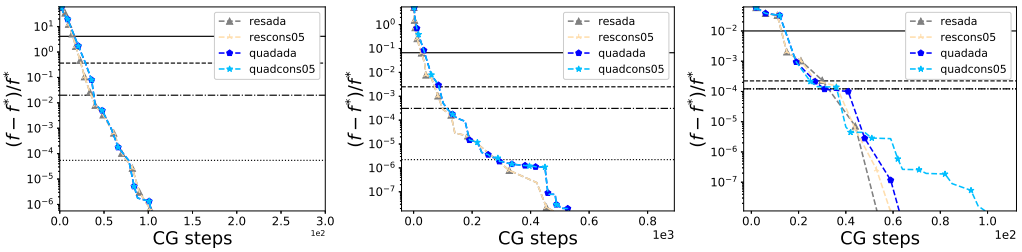
(d) `rcv1` (e) `url` (f) `real-sim`

(g) `yahoojp` (h) `yahookr` (i) `kdd2010a`

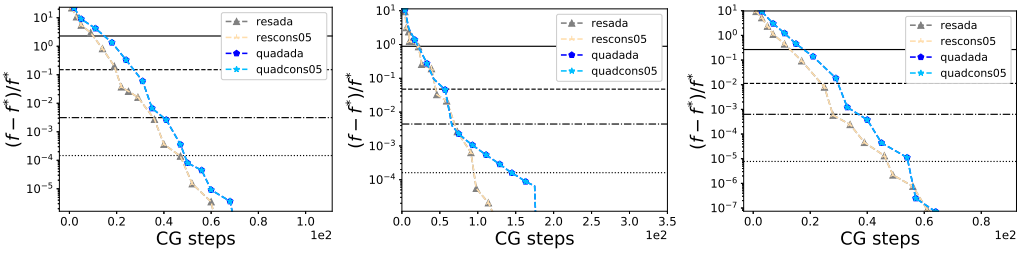
(j) `kdd2010b` (k) `HIGGS` (l) `criteo`

(m) `webspam` (n) `kdd2012`

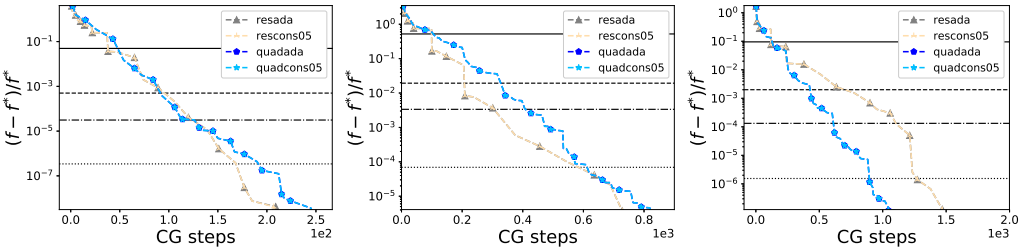
Figure iv: A comparison of different constant thresholds in the inner stopping condition. We show the convergence of a truncated Newton method for logistic regression. The x-axis shows the cumulative number of CG steps. Loss=LR and $C = 100C_{\text{Best}}$ in each sub-figure.



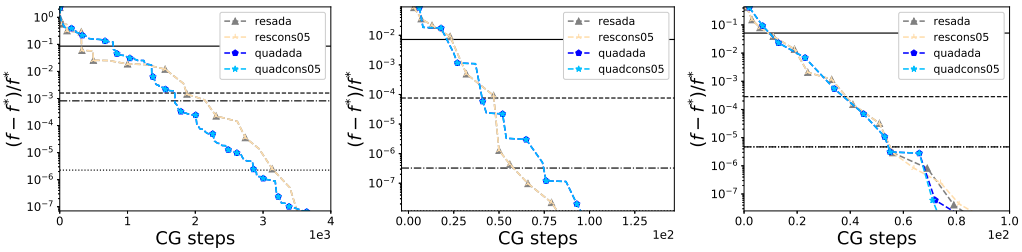
(a) `new20` (b) `w8a` (c) `covtype`



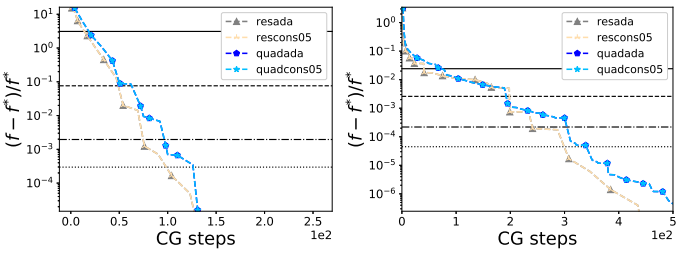
(d) `rcv1` (e) `url` (f) `real-sim`



(g) `yahoojp` (h) `yahookr` (i) `kdd2010a`

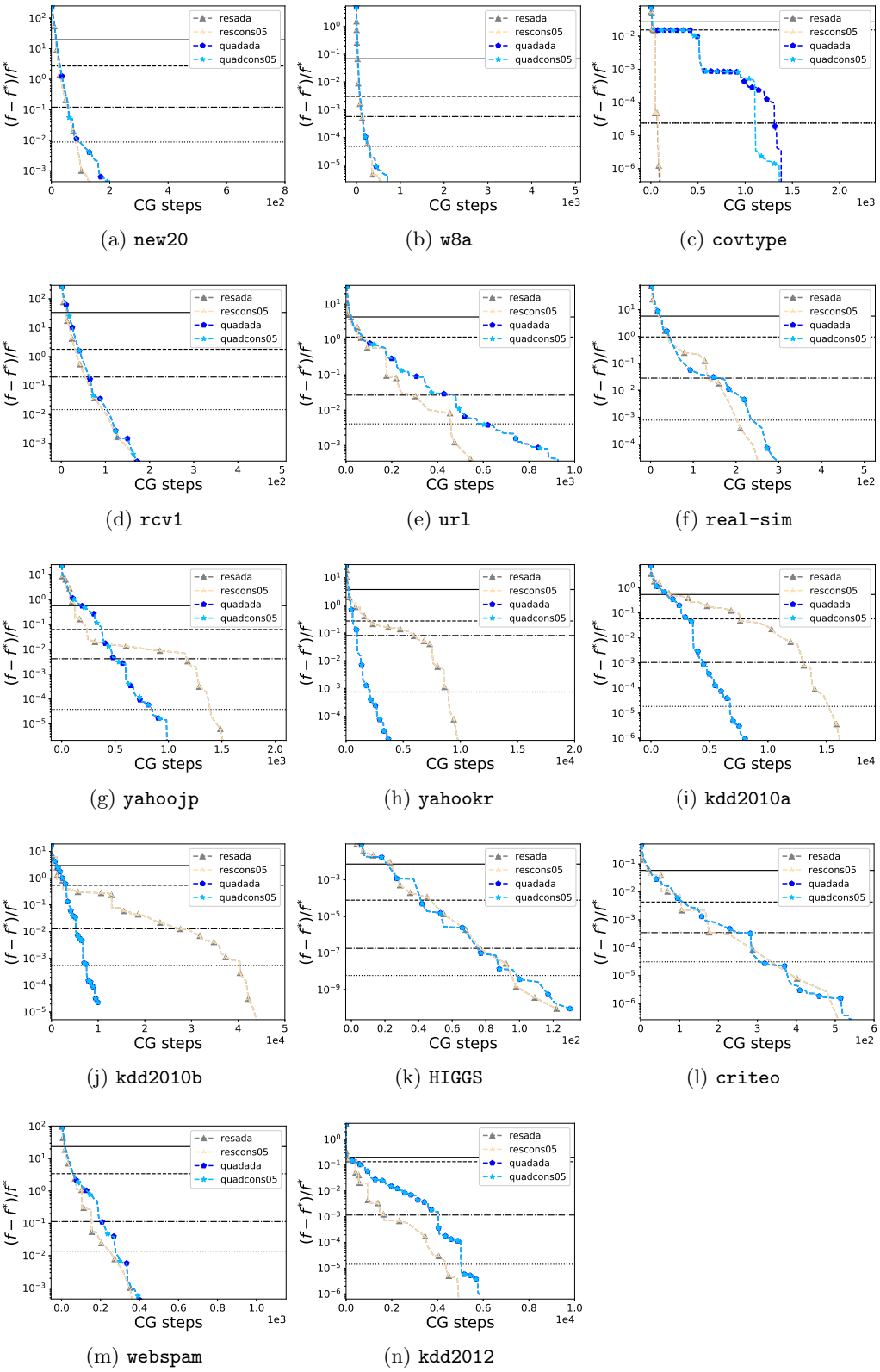


(j) `kdd2010b` (k) `HIGGS` (l) `criteo`



(m) `webspam` (n) `kdd2012`

Figure v: A comparison between adaptive and constant forcing sequence. We show the convergence of a truncated Newton method for logistic regression. The x-axis shows the cumulative number of CG steps. Loss=LR and $C = C_{\text{Best}}$ in each sub-figure.



(a) `new20` (b) `w8a` (c) `covtype`

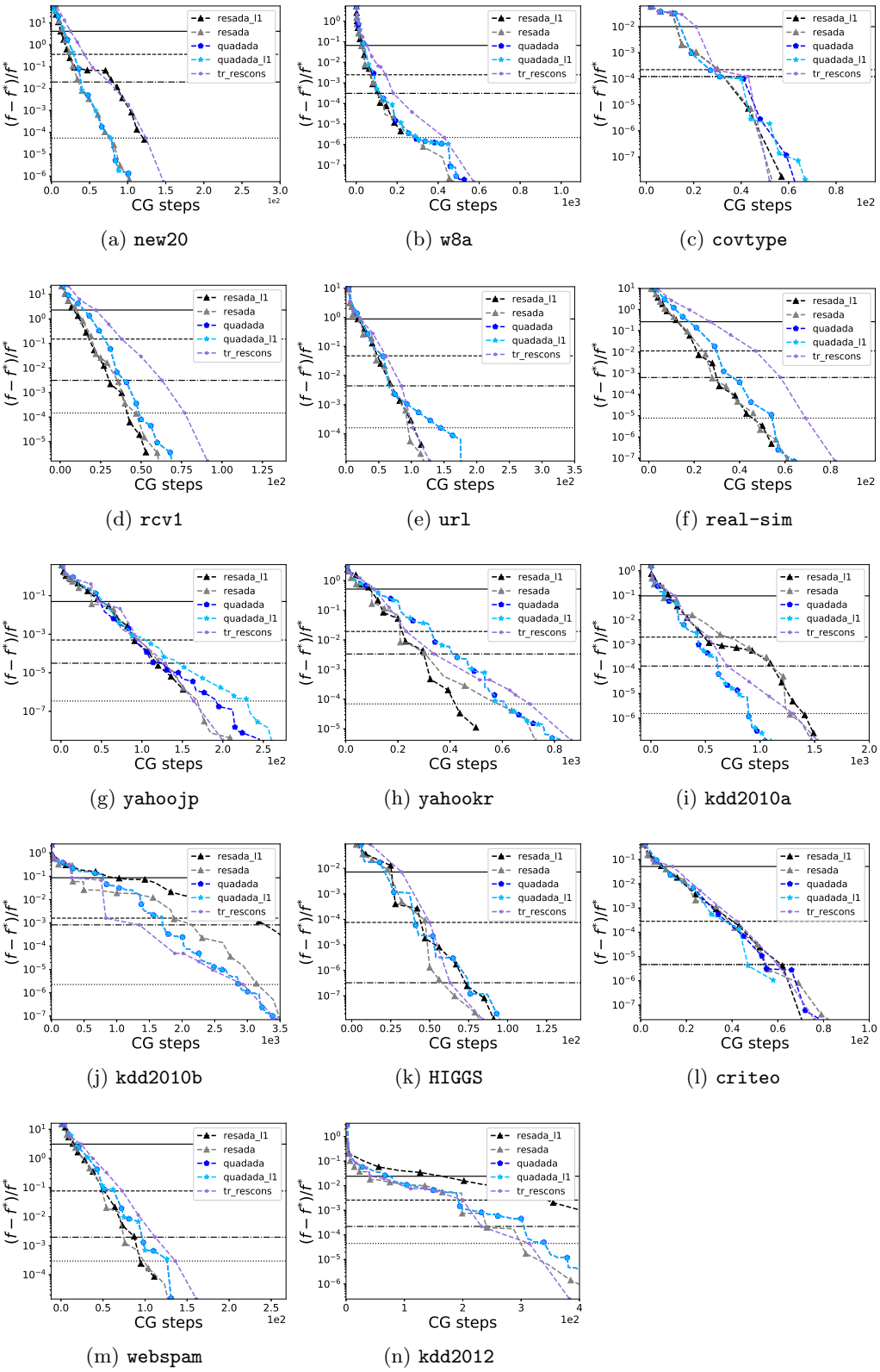
(d) `rcv1` (e) `url` (f) `real-sim`

(g) `yahoojp` (h) `yahookr` (i) `kdd2010a`

(j) `kdd2010b` (k) `HIGGS` (l) `criteo`

(m) `webspam` (n) `kdd2012`

Figure vi: A comparison between adaptive and constant forcing sequence. We show the convergence of a truncated Newton method for logistic regression. The x-axis shows the cumulative number of CG steps. Loss=LR and $C = 100C_{\text{Best}}$ in each sub-figure.



(a) `new20` (b) `w8a` (c) `covtype`

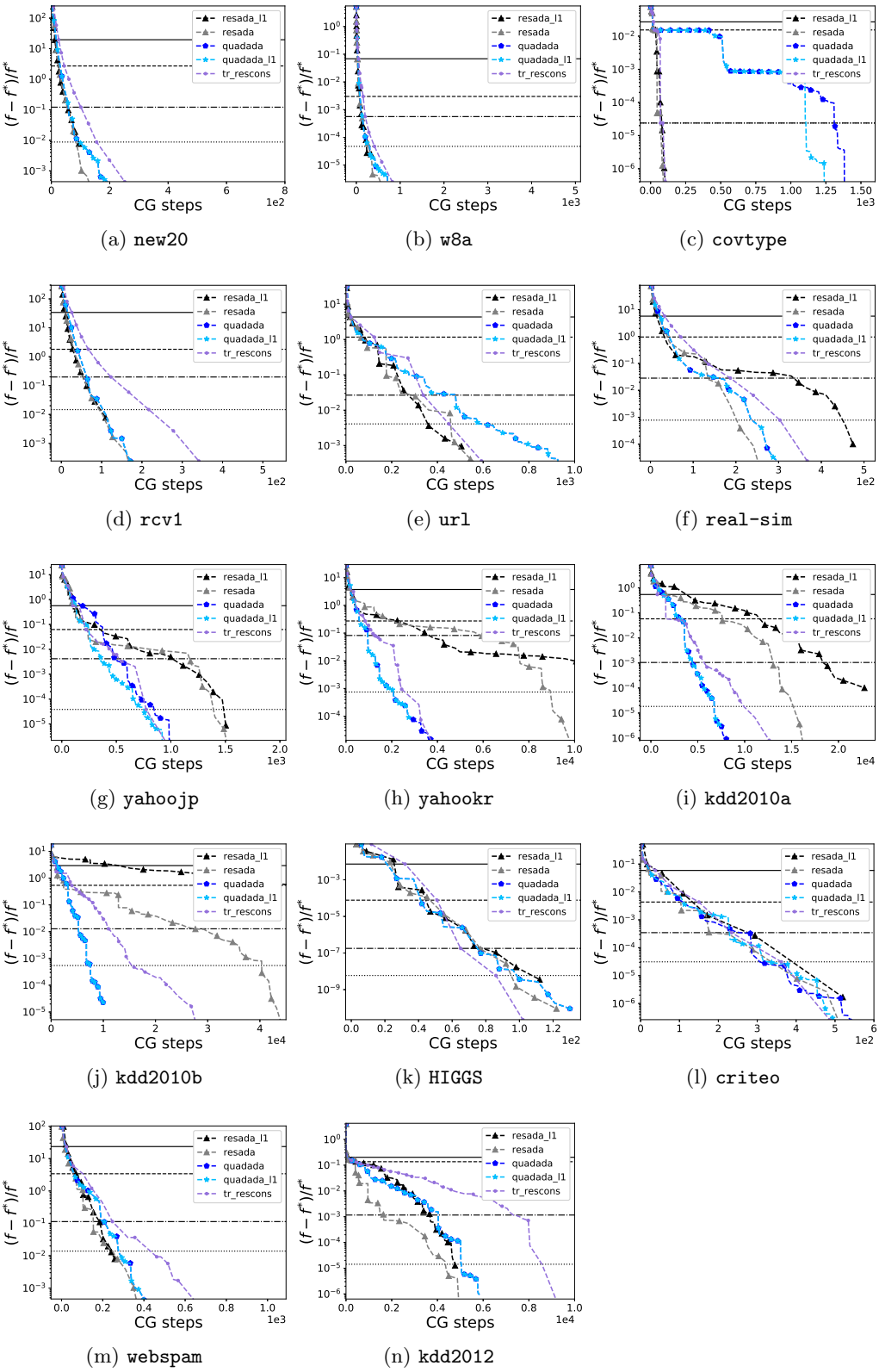
(d) `rcv1` (e) `url` (f) `real-sim`

(g) `yahoojp` (h) `yahookr` (i) `kdd2010a`

(j) `kdd2010b` (k) `HIGGS` (l) `criteo`

(m) `webspam` (n) `kdd2012`

Figure vii: A comparison between l2-norm and l1-norm and with the trust-region approach. We show the convergence of a truncated Newton method for logistic regression. The x-axis shows the cumulative number of CG steps. Loss=LR and $C = C_{\text{Best}}$ in each sub-figure.



(a) `new20` (b) `w8a` (c) `covtype`

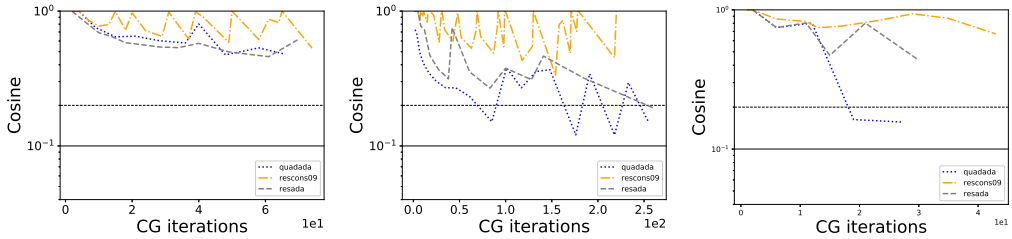
(d) `rcv1` (e) `url` (f) `real-sim`

(g) `yahoojp` (h) `yahookr` (i) `kdd2010a`

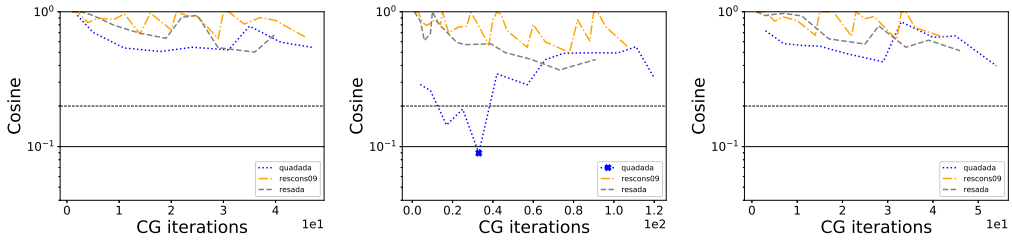
(j) `kdd2010b` (k) `HIGGS` (l) `criteo`

(m) `webspam` (n) `kdd2012`

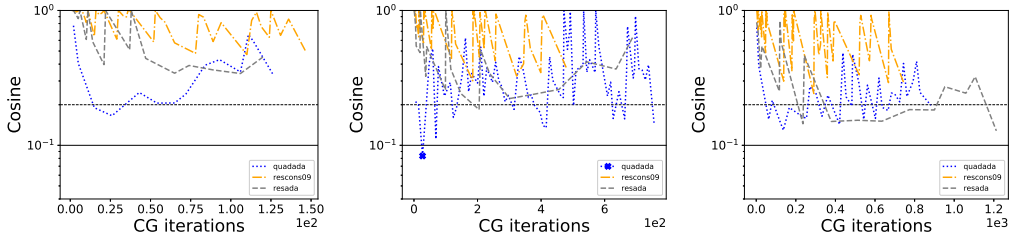
Figure viii: A comparison between l2-norm and l1-norm and with the trust-region approach. We show the convergence of a truncated Newton method for logistic regression. The x-axis shows the cumulative number of CG steps. Loss=LR and $C = 100C_{\text{Best}}$ in each sub-figure.



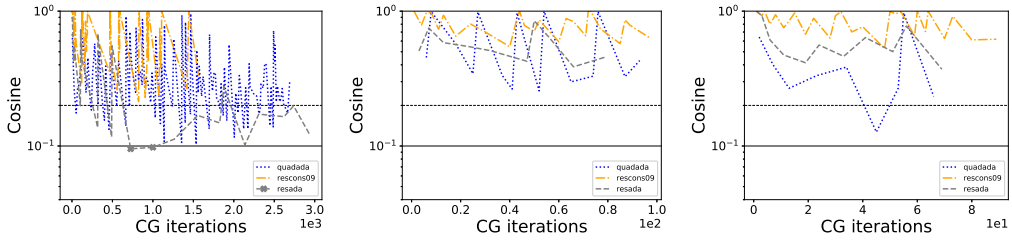
(a) `new20` (b) `w8a` (c) `covtype`



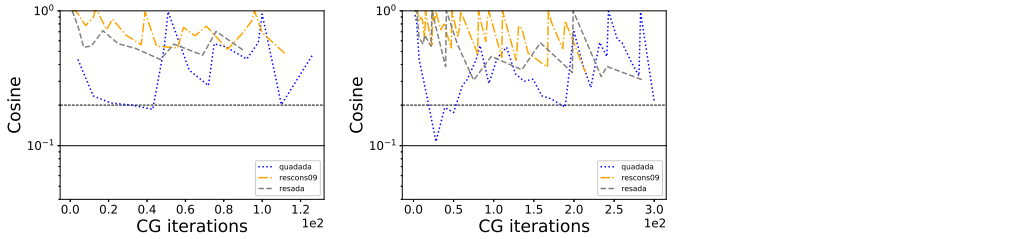
(d) `rcv1` (e) `url` (f) `real-sim`



(g) `yahoojp` (h) `yahookr` (i) `kdd2010a`

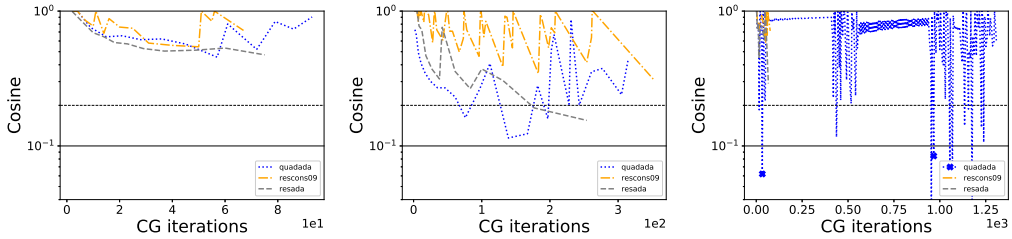


(j) `kdd2010b` (k) `HIGGS` (l) `criteo`

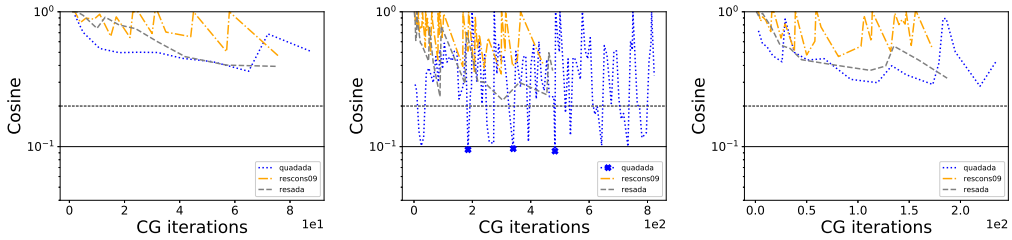


(m) `webspam` (n) `kdd2012`

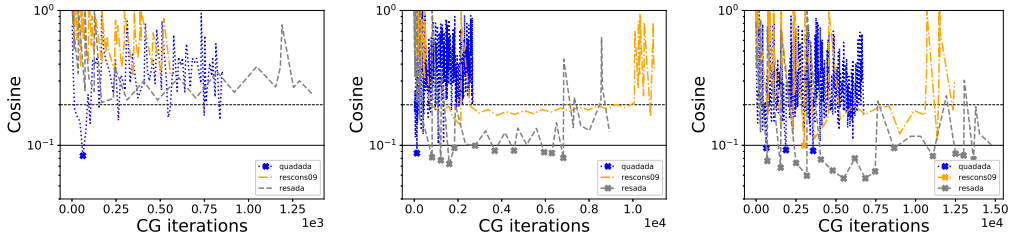
Figure ix: Cosines between the anti-gradient $-\nabla f(\boldsymbol{w}_k)$ and the resulting direction $\boldsymbol{s}_k$, using different truncation rules. Steps in which the cosine is below the 0.1 threshold are marked with a ✖. The x-axis shows the cumulative number of CG steps. Loss=LR and $C = C_{\text{Best}}$ in each sub-figure.



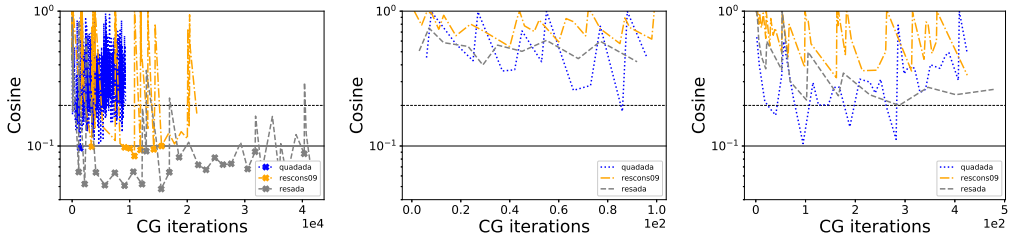
(a) `new20` (b) `w8a` (c) `covtype`



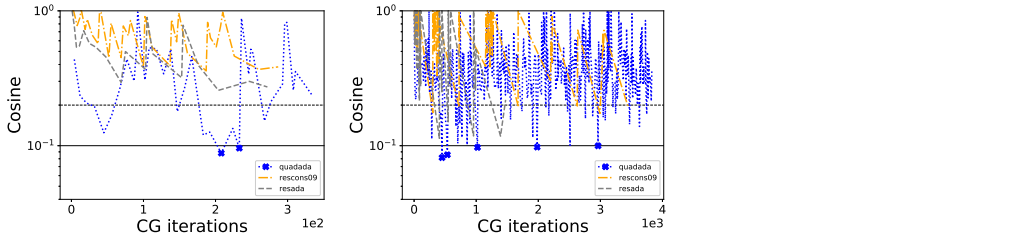
(d) `rcv1` (e) `url` (f) `real-sim`



(g) `yahoojp` (h) `yahookr` (i) `kdd2010a`

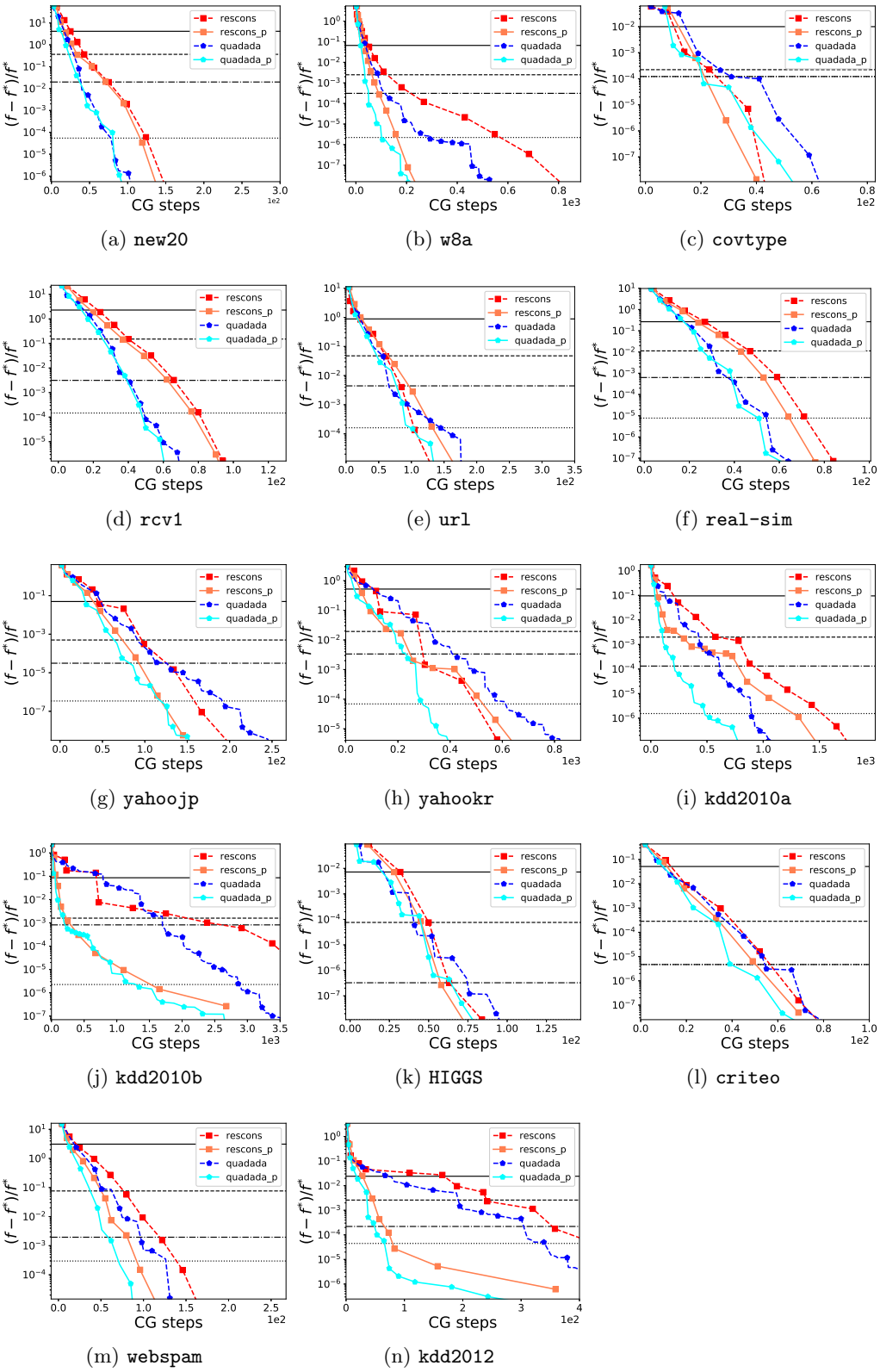


(j) `kdd2010b` (k) `HIGGS` (l) `criteo`



(m) `webspam` (n) `kdd2012`

Figure x: Cosines between the anti-gradient $-\nabla f(\boldsymbol{w}_k)$ and the resulting direction $\boldsymbol{s}_k$, using different truncation rules. Steps in which the cosine is below the 0.1 threshold are marked with a ✖. The x-axis shows the cumulative number of CG steps. Loss=LR and $C = 100C_{\text{Best}}$ in each sub-figure.



(a) `new20` (b) `w8a` (c) `covtype`

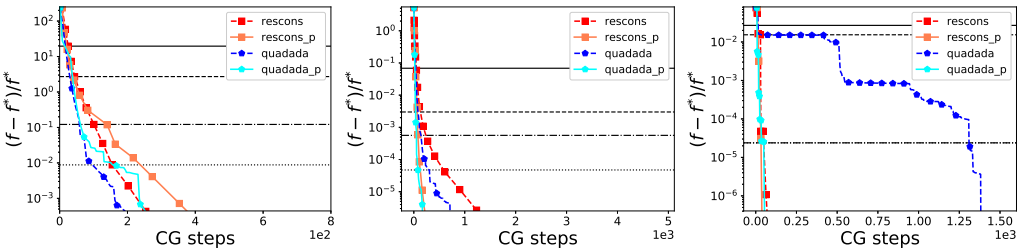
(d) `rcv1` (e) `url` (f) `real-sim`

(g) `yahoojp` (h) `yahookr` (i) `kdd2010a`

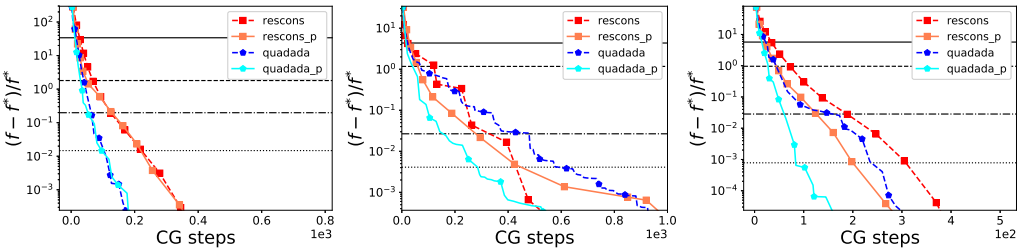
(j) `kdd2010b` (k) `HIGGS` (l) `criteo`

(m) `webspam` (n) `kdd2012`

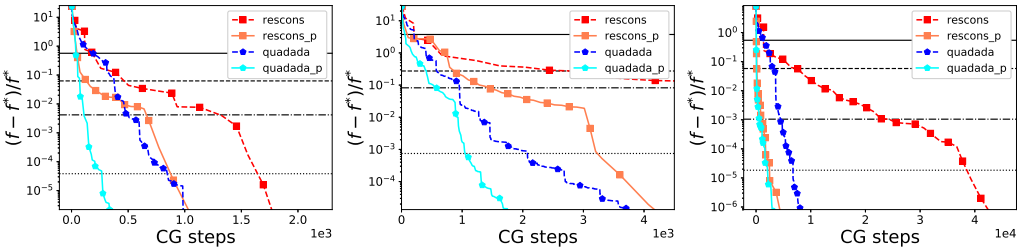
Figure xi: A comparison between adaptive and constant forcing sequences in the preconditioned case. We show the convergence of a truncated Newton method for logistic regression. The x-axis shows the cumulative number of CG steps. Loss=LR and $C = 100C_{\text{Best}}$ in each sub-figure. Loss=LR and $C = C_{\text{Best}}$.



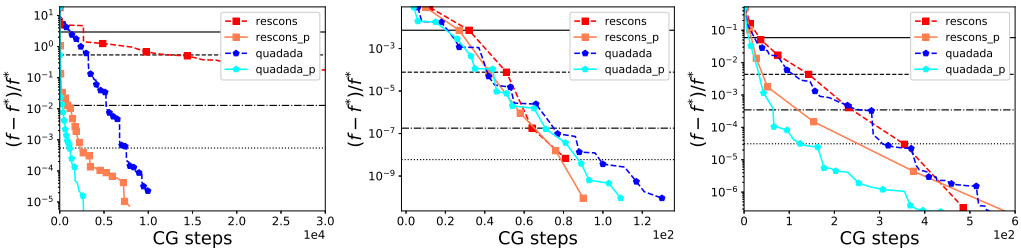
(a) `new20` (b) `w8a` (c) `covtype`



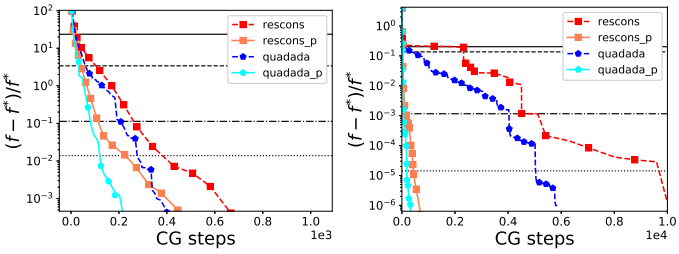
(d) `rcv1` (e) `url` (f) `real-sim`



(g) `yahoojp` (h) `yahookr` (i) `kdd2010a`

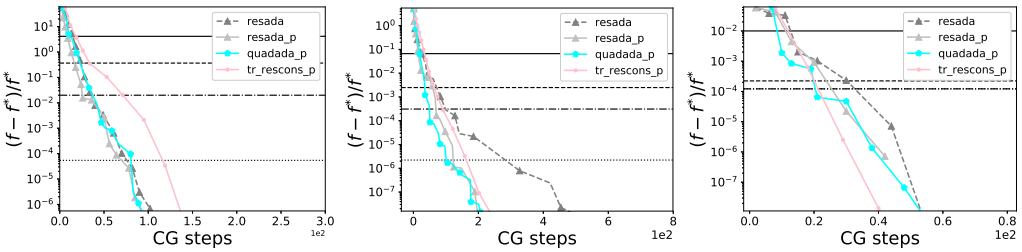


(j) `kdd2010b` (k) `HIGGS` (l) `criteo`

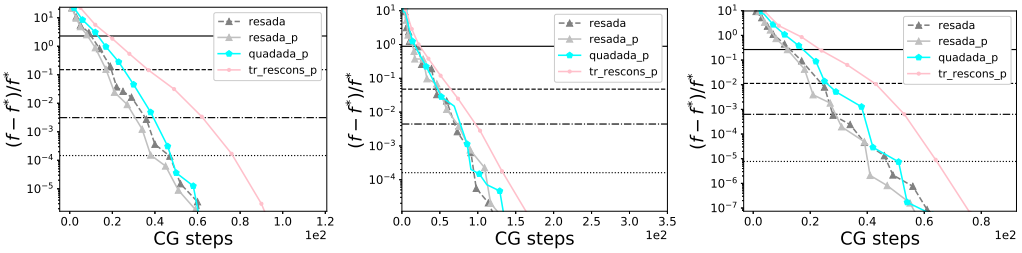


(m) `webspam` (n) `kdd2012`

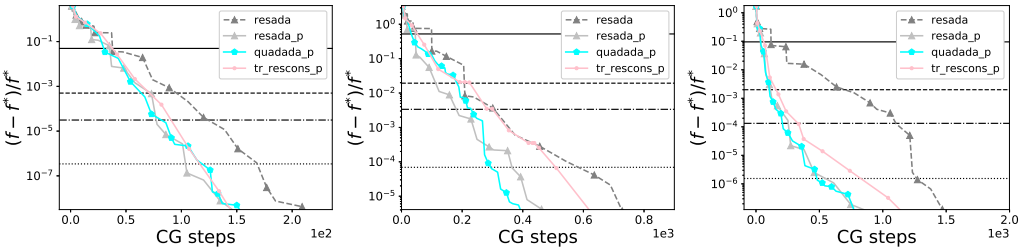
Figure xii: A comparison between adaptive and constant forcing sequences in the preconditioned case. We show the convergence of a truncated Newton method for logistic regression. The x-axis shows the cumulative number of CG steps. Loss=LR and $C = 100C_{\text{Best}}$ in each sub-figure.



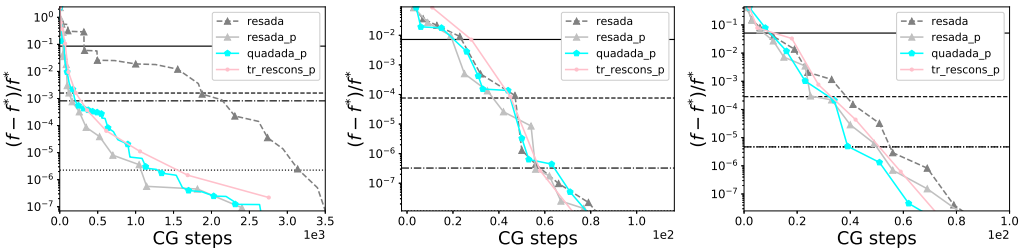
(a) `new20` (b) `w8a` (c) `covtype`



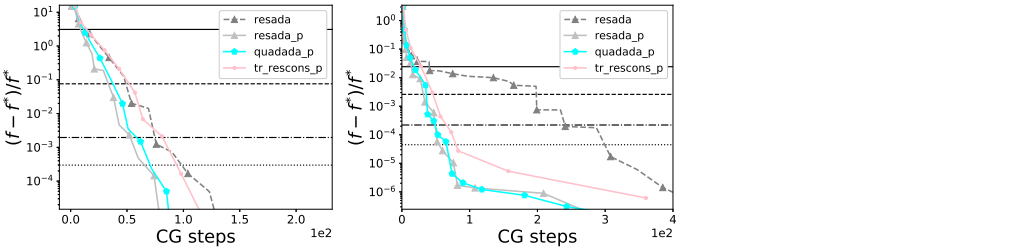
(d) `rcv1` (e) `url` (f) `real-sim`



(g) `yahoojp` (h) `yahookr` (i) `kdd2010a`

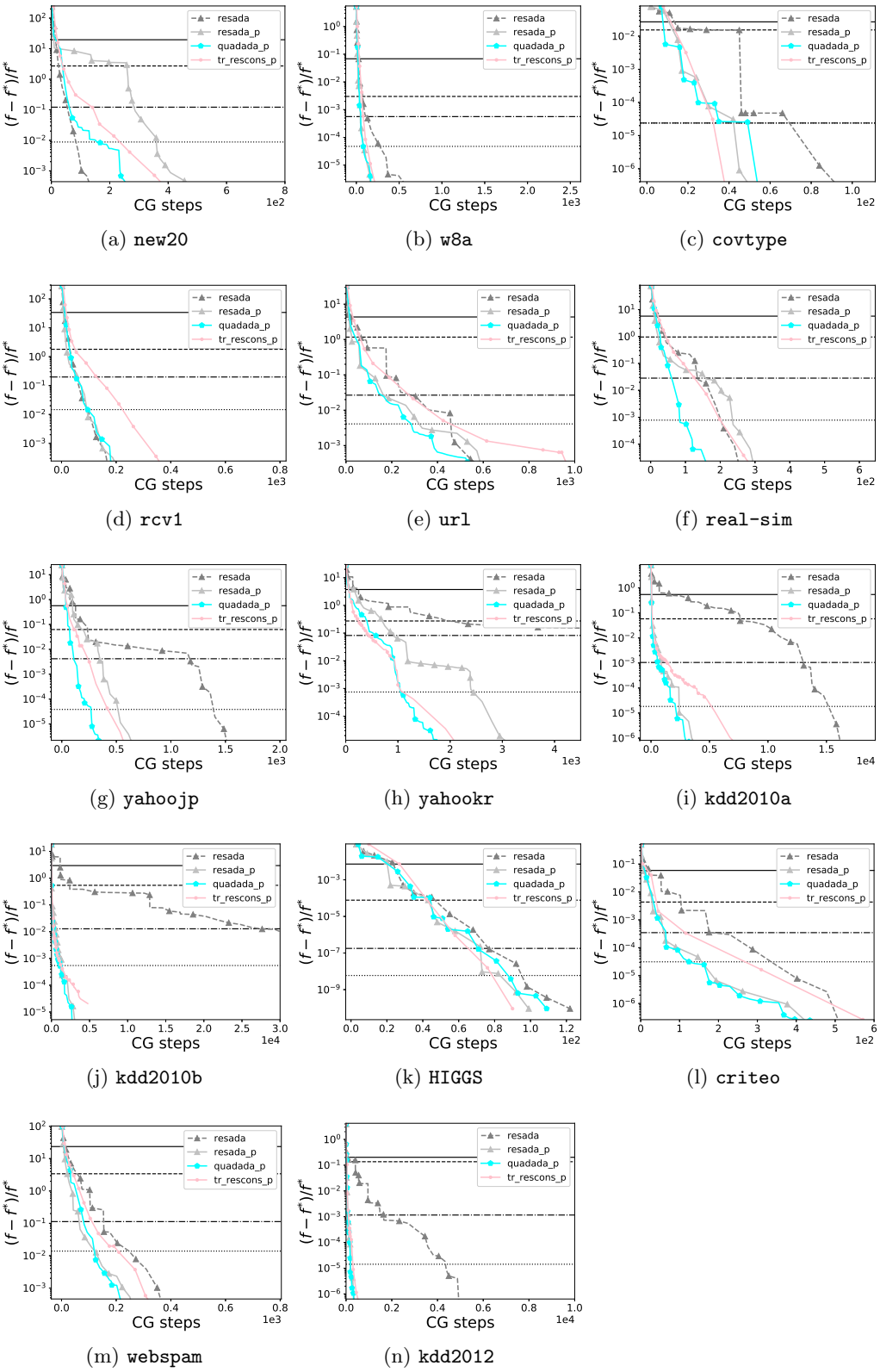


(j) `kdd2010b` (k) `HIGGS` (l) `criteo`



(m) `webspam` (n) `kdd2012`

Figure xiii: A comparison between adaptive rules and the trust region approach in the preconditioned case. We show the convergence of a truncated Newton method for logistic regression. The x-axis shows the cumulative number of CG steps. Loss=LR and $C = C_{\text{Best}}$ in each sub-figure.



(a) `new20` (b) `w8a` (c) `covtype`

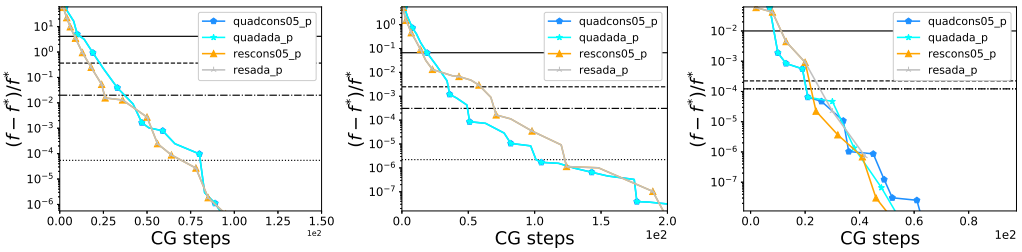
(d) `rcv1` (e) `url` (f) `real-sim`

(g) `yahoojp` (h) `yahookr` (i) `kdd2010a`

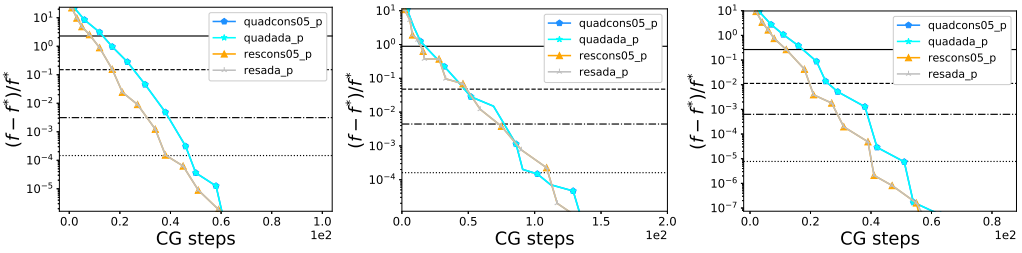
(j) `kdd2010b` (k) `HIGGS` (l) `criteo`

(m) `webspam` (n) `kdd2012`

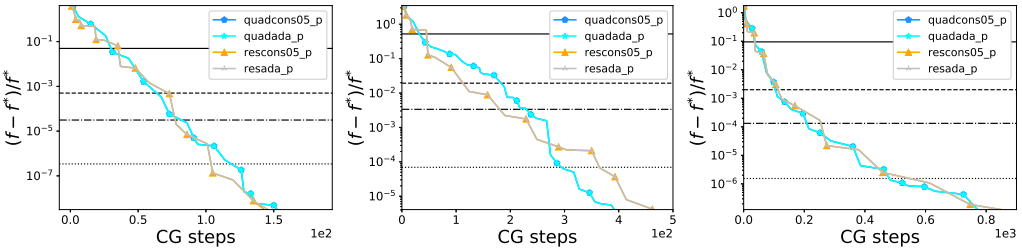
Figure xiv: A comparison between adaptive rules and the trust region approach in the preconditioned case. We show the convergence of a truncated Newton method for logistic regression. The x-axis shows the cumulative number of CG steps. Loss=LR and $C = 100C_{\text{Best}}$ in each sub-figure.



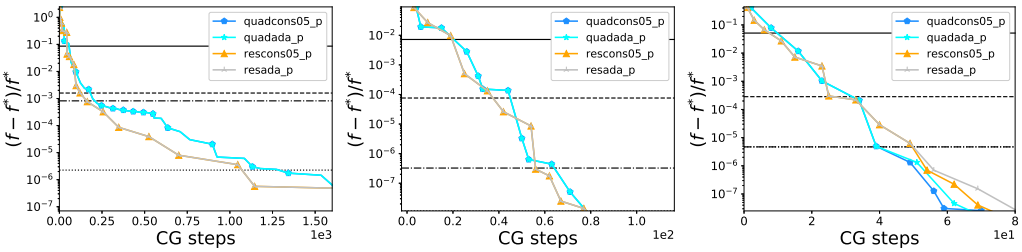
(a) `new20` (b) `w8a` (c) `covtype`



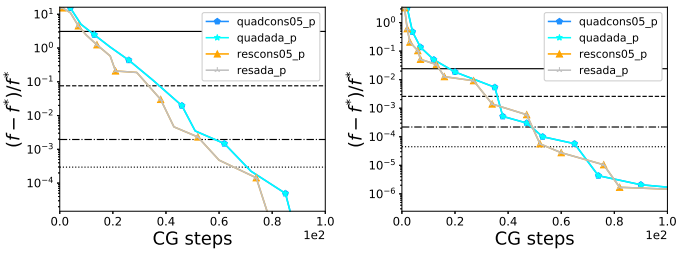
(d) `rcv1` (e) `url` (f) `real-sim`



(g) `yahoojp` (h) `yahookr` (i) `kdd2010a`

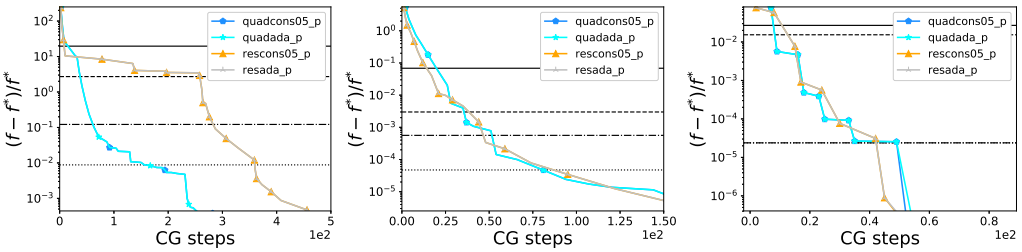


(j) `kdd2010b` (k) `HIGGS` (l) `criteo`

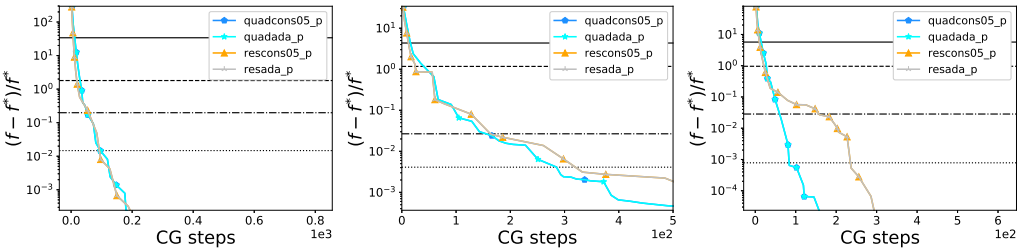


(m) `webspam` (n) `kdd2012`

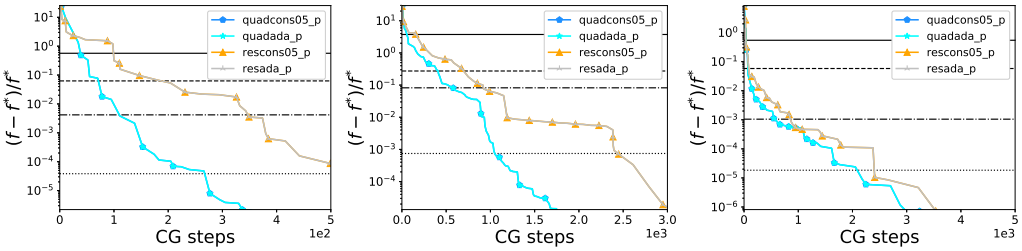
Figure xv: A comparison between adaptive and constant forcing sequence in the preconditioned case. We show the convergence of a truncated Newton method for logistic regression. The x-axis shows the cumulative number of CG steps. Loss=LR and $C = C_{\text{Best}}$ in each sub-figure.



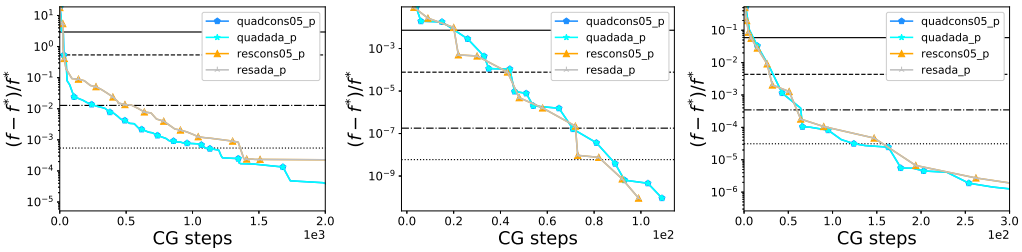
(a) `new20` (b) `w8a` (c) `covtype`



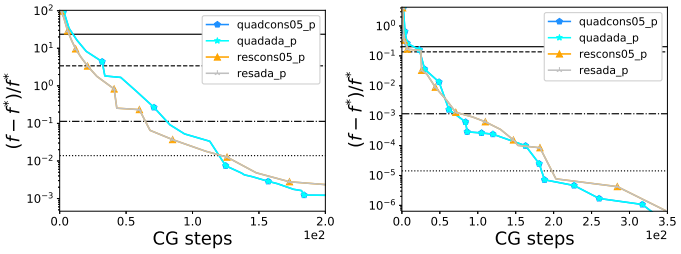
(d) `rcv1` (e) `url` (f) `real-sim`



(g) `yahoojp` (h) `yahookr` (i) `kdd2010a`

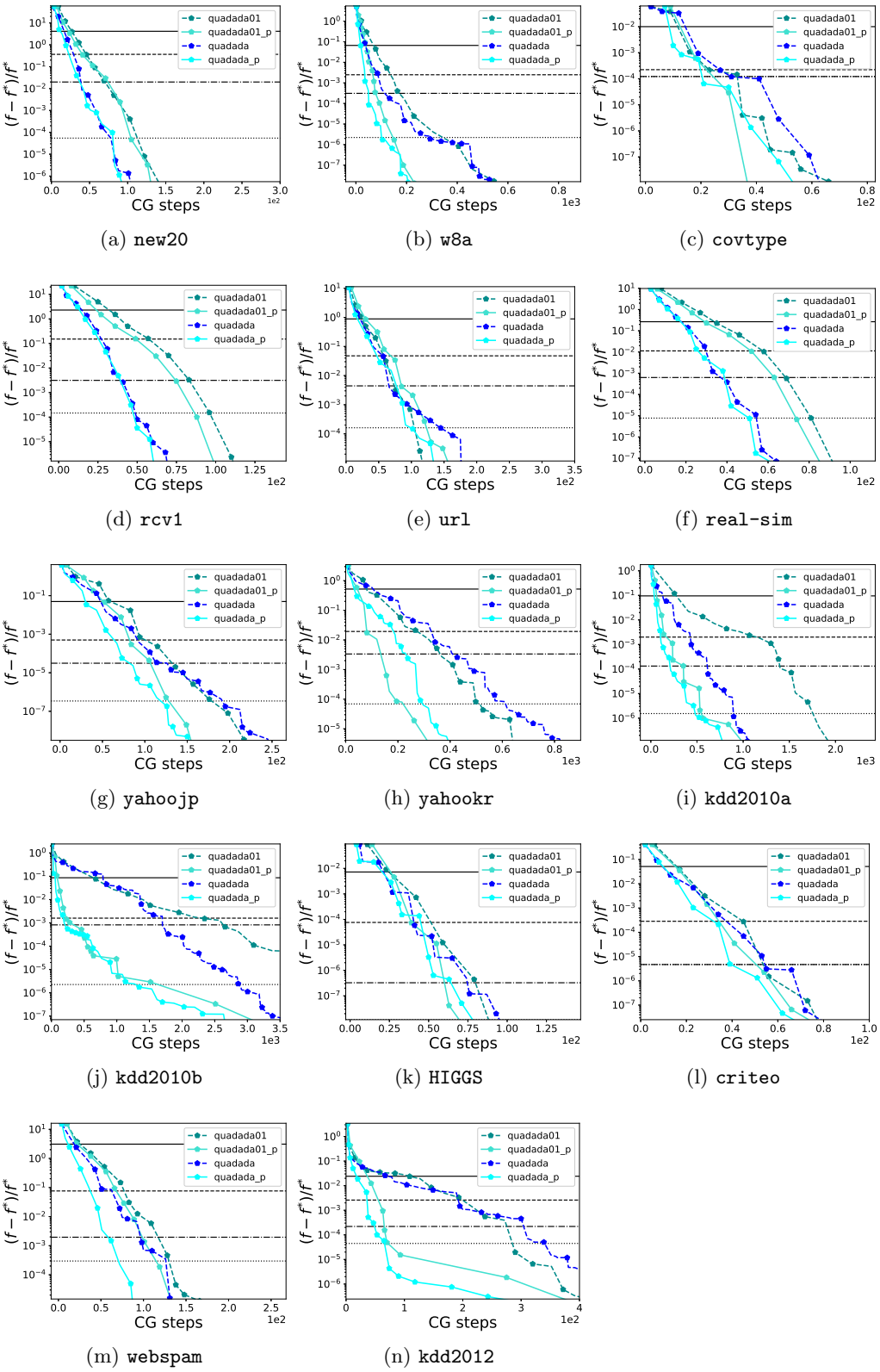


(j) `kdd2010b` (k) `HIGGS` (l) `criteo`



(m) `webspam` (n) `kdd2012`

Figure xvi: A comparison between adaptive and constant forcing sequence in the preconditioned case. We show the convergence of a truncated Newton method for logistic regression. The x-axis shows the cumulative number of CG steps. Loss=LR and $C = 100C_{\text{Best}}$ in each sub-figure.



(a) `new20` (b) `w8a` (c) `covtype`

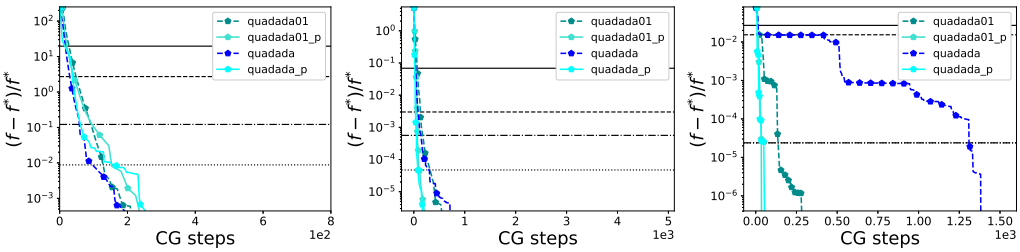
(d) `rcv1` (e) `url` (f) `real-sim`

(g) `yahoojp` (h) `yahookr` (i) `kdd2010a`

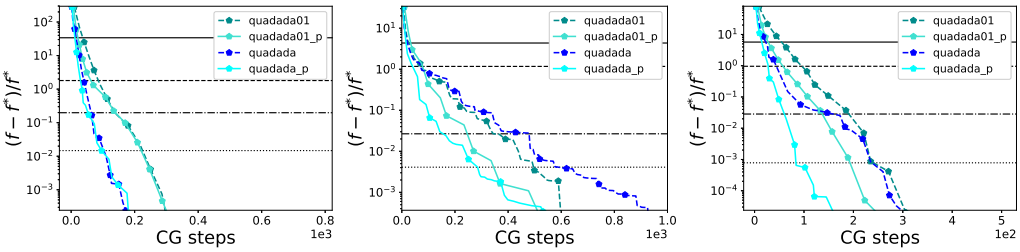
(j) `kdd2010b` (k) `HIGGS` (l) `criteo`

(m) `webspam` (n) `kdd2012`

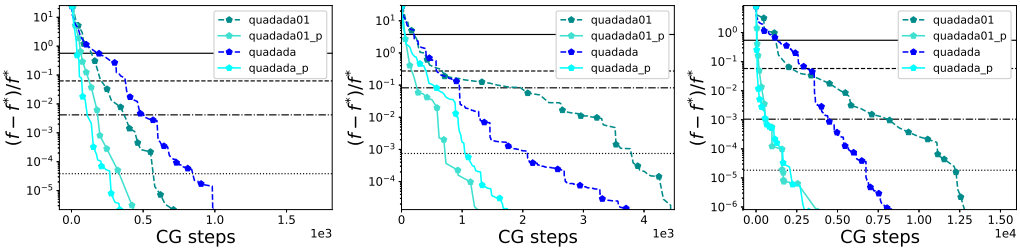
Figure xvii: A comparison of different early thresholds in the quadratic adaptive inner stopping condition. We show the convergence of a truncated Newton method for logistic regression. The x-axis shows the cumulative number of CG steps. Loss=LR and $C = C_{\text{Best}}$ in each sub-figure.



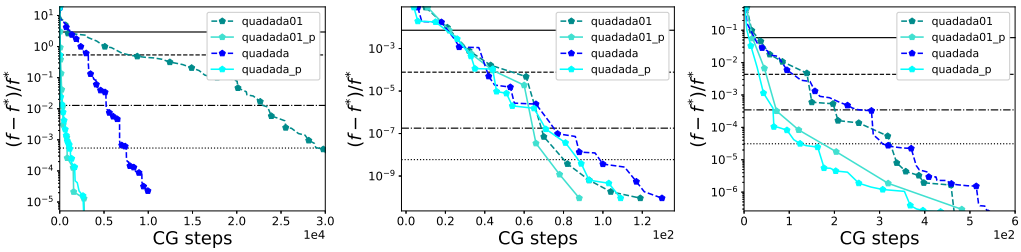
(a) `new20` (b) `w8a` (c) `covtype`



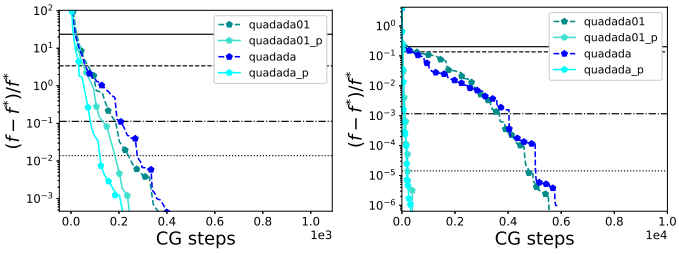
(d) `rcv1` (e) `url` (f) `real-sim`



(g) `yahoojp` (h) `yahookr` (i) `kdd2010a`

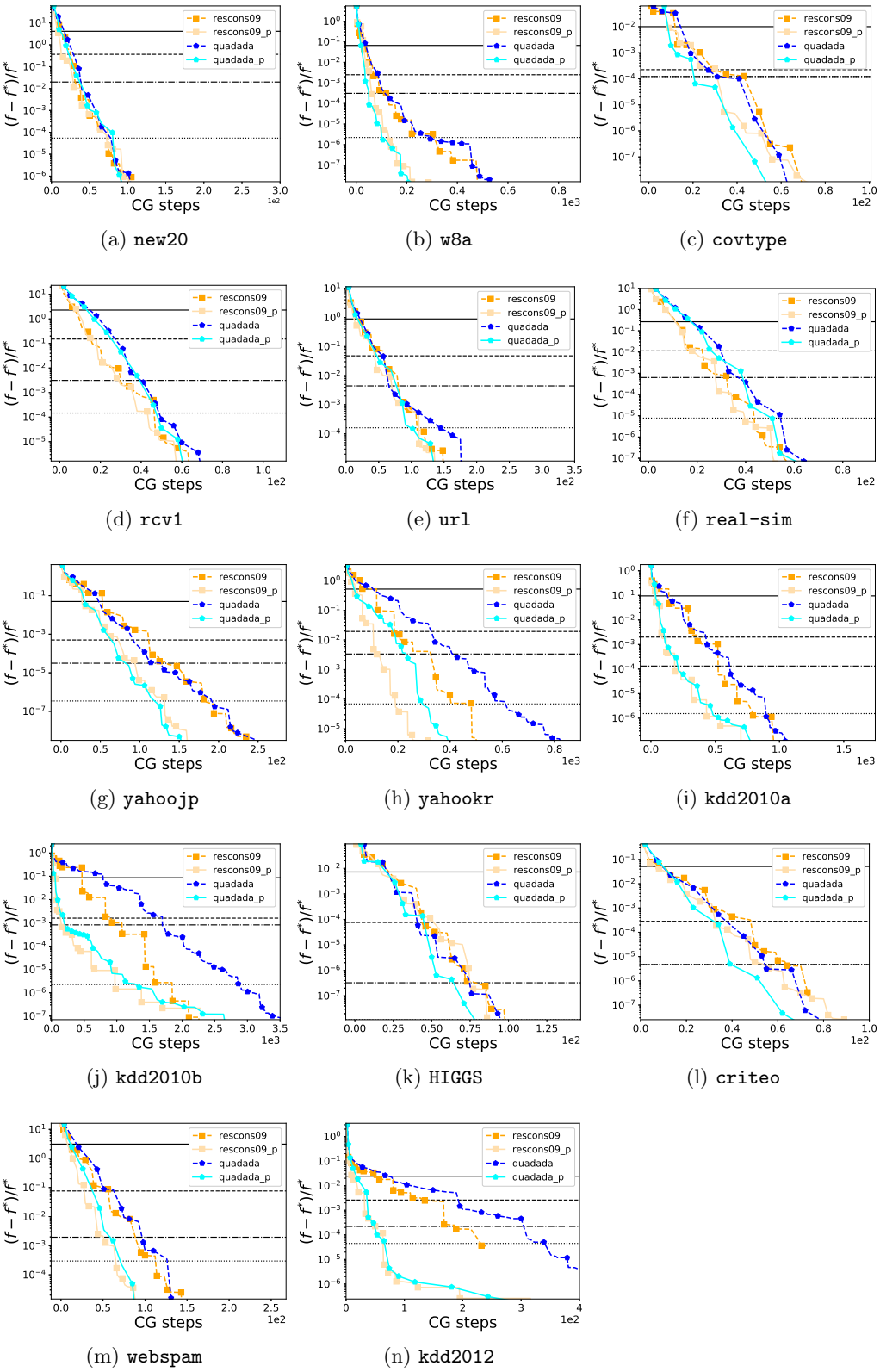


(j) `kdd2010b` (k) `HIGGS` (l) `criteo`



(m) `webspam` (n) `kdd2012`

Figure xviii: A comparison of different early thresholds in the quadratic adaptive inner stopping condition. We show the convergence of a truncated Newton method for logistic regression. The x-axis shows the cumulative number of CG steps. Loss=LR and $C = 100C_{\text{Best}}$ in each sub-figure.



(a) `new20` (b) `w8a` (c) `covtype`

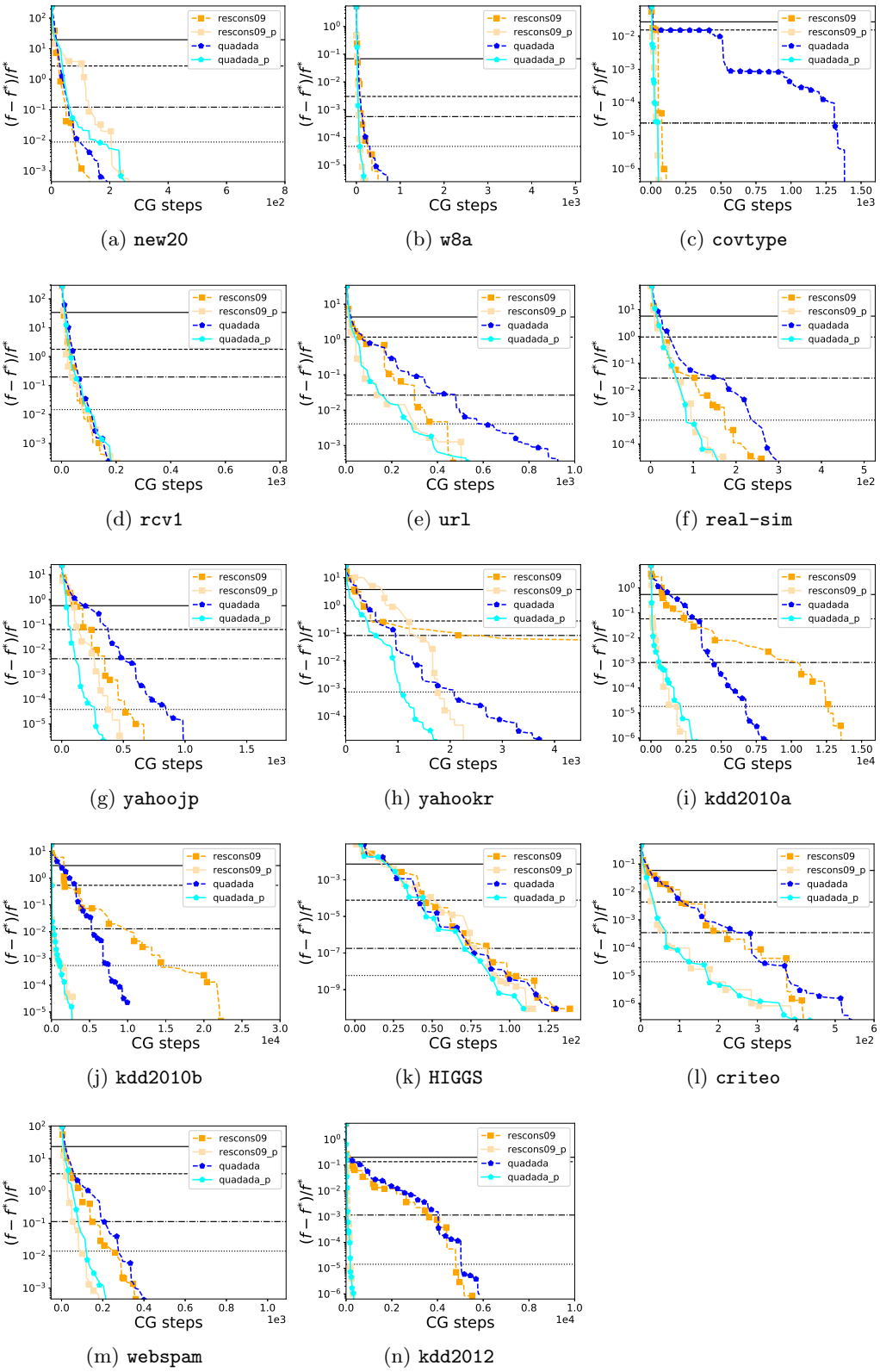
(d) `rcv1` (e) `url` (f) `real-sim`

(g) `yahoojp` (h) `yahookr` (i) `kdd2010a`

(j) `kdd2010b` (k) `HIGGS` (l) `criteo`

(m) `webspam` (n) `kdd2012`

Figure xix: Comparison against a residual constant rule with higher threshold in the preconditioned case. We show the convergence of a truncated Newton method for logistic regression. The x-axis shows the cumulative number of CG steps. Loss=LR and $C = C_{\text{Best}}$ in each sub-figure.



(a) `new20` (b) `w8a` (c) `covtype`

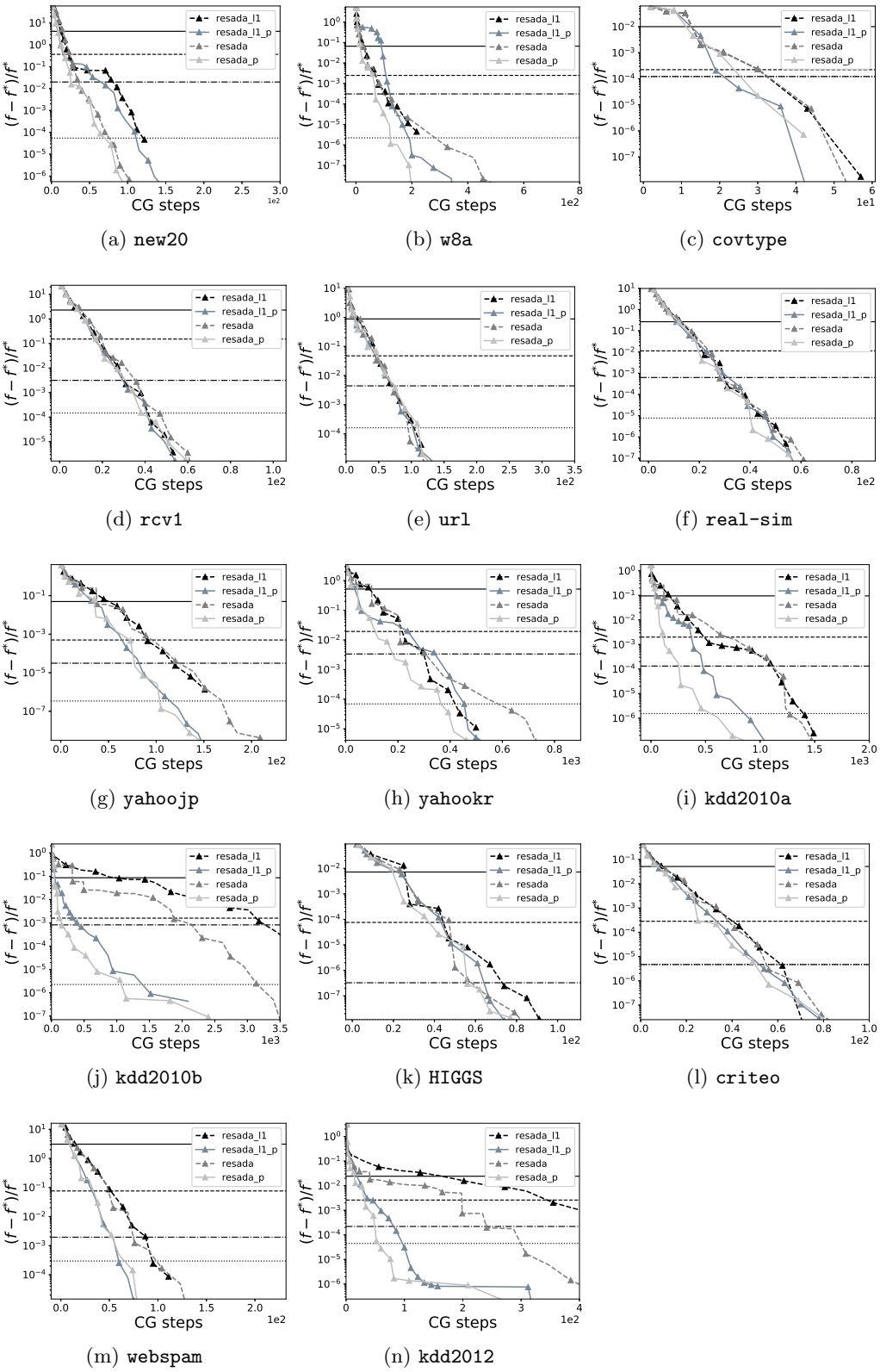
(d) `rcv1` (e) `url` (f) `real-sim`

(g) `yahoojp` (h) `yahookr` (i) `kdd2010a`

(j) `kdd2010b` (k) `HIGGS` (l) `criteo`

(m) `webspam` (n) `kdd2012`

Figure xx: Comparison against a residual constant rule with higher threshold in the preconditioned case. We show the convergence of a truncated Newton method for logistic regression. The x-axis shows the cumulative number of CG steps. Loss=LR and $C = 100C_{\text{Best}}$ in each sub-figure.



(a) `new20` (b) `w8a` (c) `covtype`

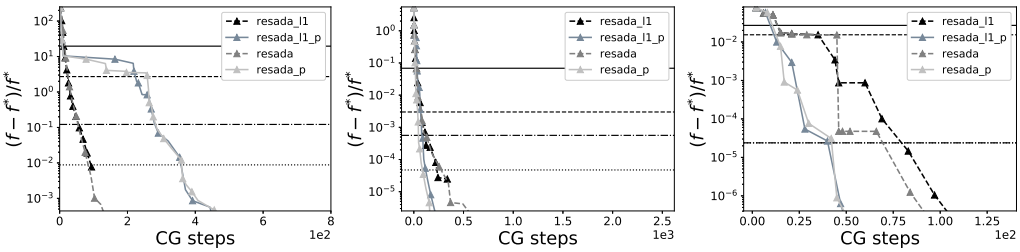
(d) `rcv1` (e) `url` (f) `real-sim`

(g) `yahoojp` (h) `yahookr` (i) `kdd2010a`

(j) `kdd2010b` (k) `HIGGS` (l) `criteo`

(m) `webspam` (n) `kdd2012`

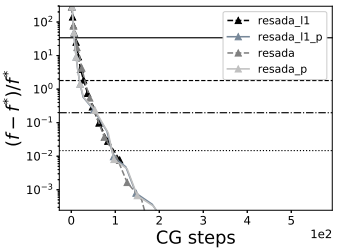
Figure xxi: An investigation on the robustness of residual adaptive rules in the preconditioned case. We show the convergence of a truncated Newton method for logistic regression. The x-axis shows the cumulative number of CG steps. Loss=LR and $C = C_{\text{Best}}$ in each sub-figure.



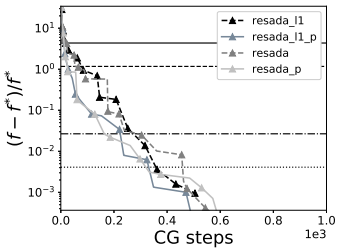
(a) `new20`

(b) `w8a`

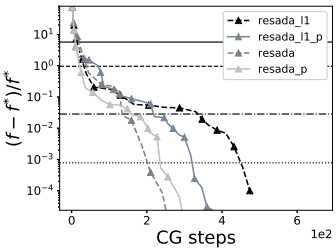
(c) `covtype`



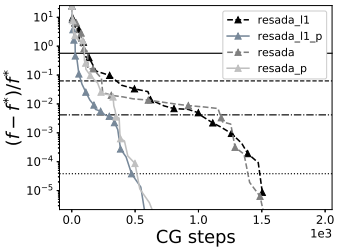
(d) `rcv1`



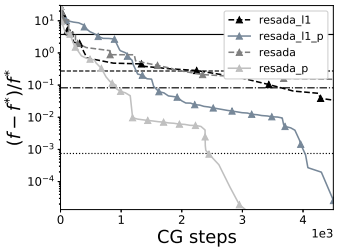
(e) `url`



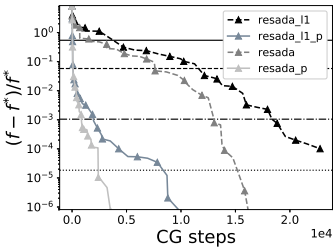
(f) `real-sim`



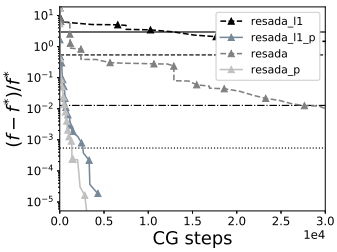
(g) `yahoojp`



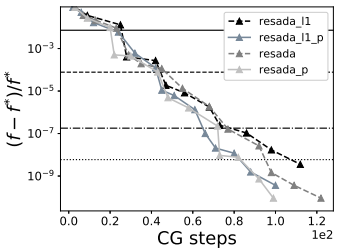
(h) `yahookr`



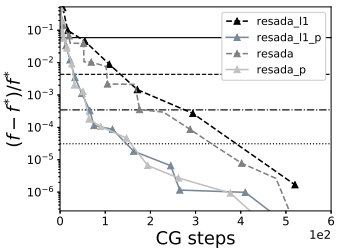
(i) `kdd2010a`



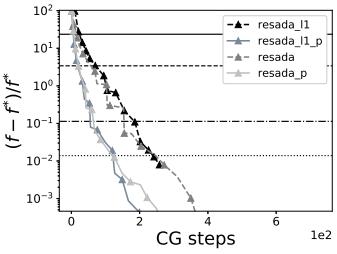
(j) `kdd2010b`



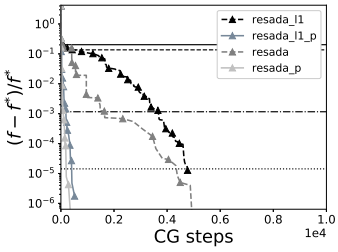
(k) `HIGGS`



(l) `criteo`

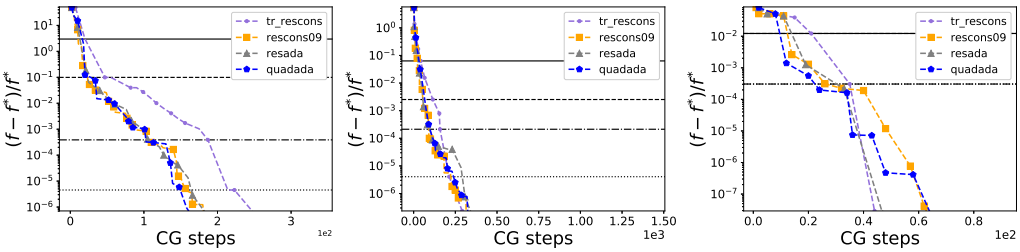


(m) `webspam`



(n) `kdd2012`

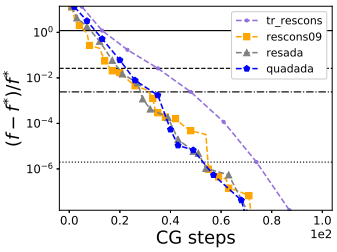
Figure xxii: An investigation on the robustness of residual adaptive rules in the preconditioned case. We show the convergence of a truncated Newton method for logistic regression. The x-axis shows the cumulative number of CG steps. Loss=LR and $C = 100C_{\text{Best}}$ in each sub-figure.



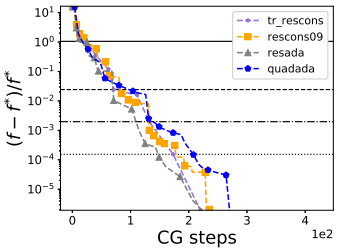
(a) `new20`

(b) `w8a`

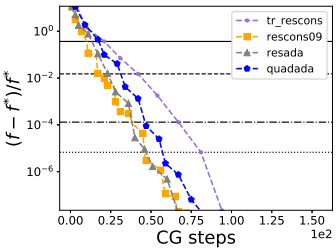
(c) `covtype`



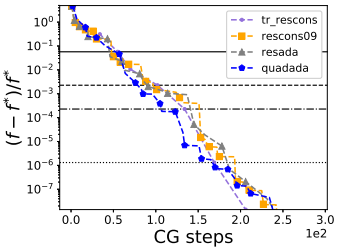
(d) `rcv1`



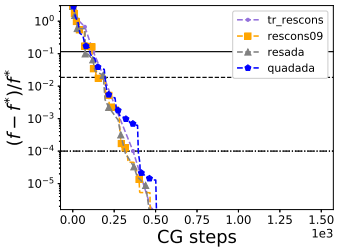
(e) `url`



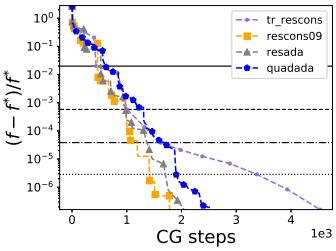
(f) `real-sim`



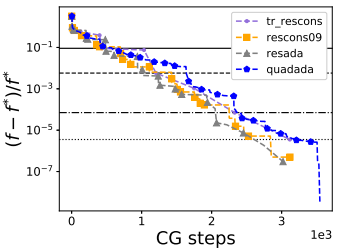
(g) `yahoojp`



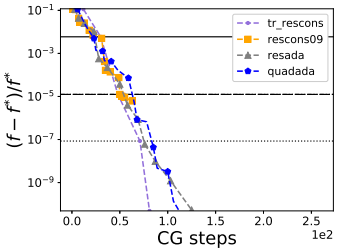
(h) `yahookr`



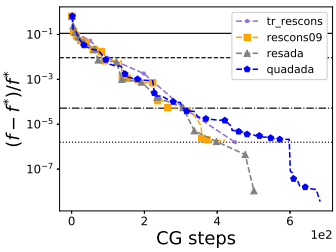
(i) `kdd2010a`



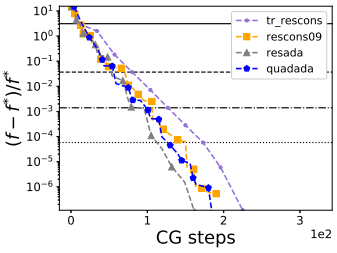
(j) `kdd2010b`



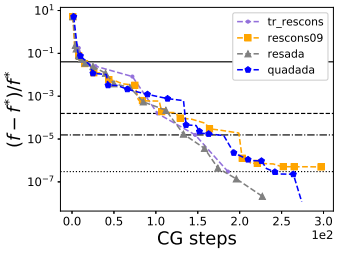
(k) `HIGGS`



(l) `criteo`

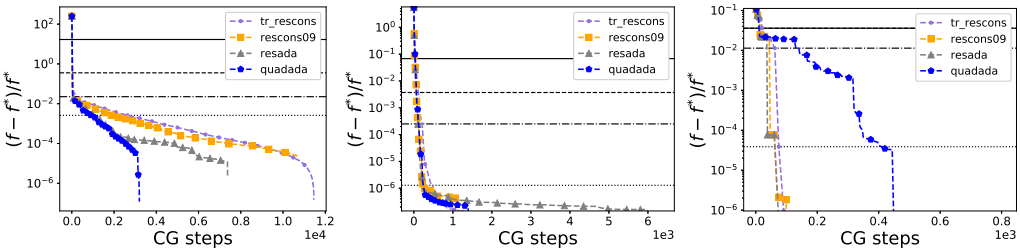


(m) `webspam`

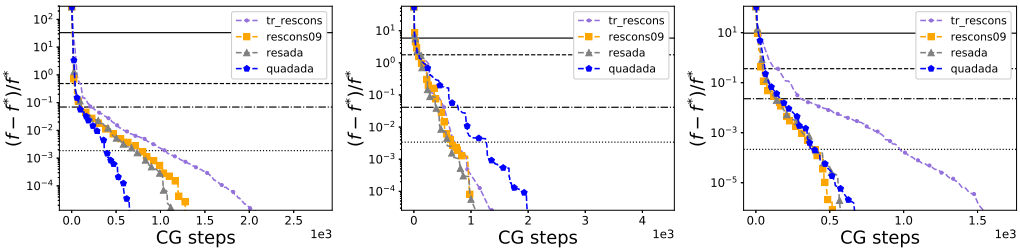


(n) `kdd2012`

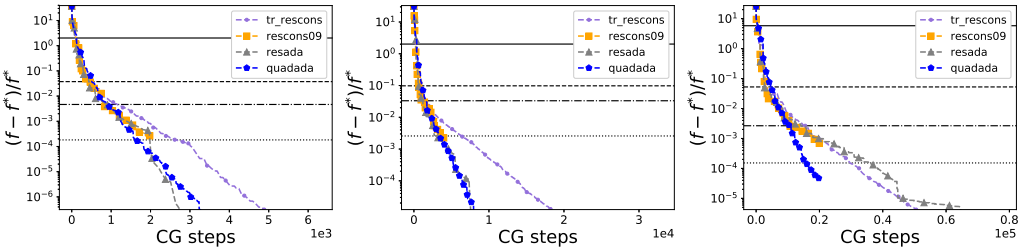
Figure xxiii: A summary comparison between various forcing sequences. We show the convergence of a truncated Newton method for linear SVM. The x-axis shows the cumulative number of CG steps. Loss=L2 and $C = C_{\text{Best}}$ in each sub-figure.



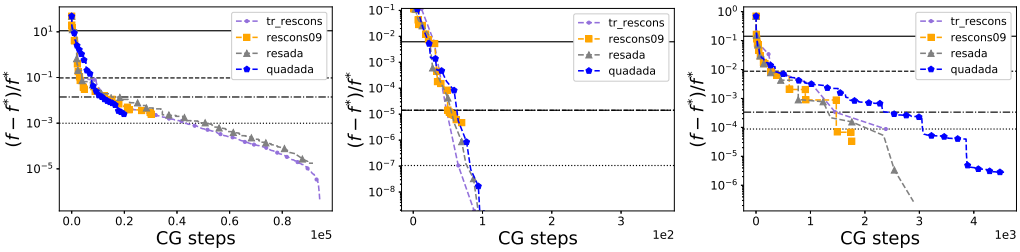
(a) `new20` (b) `w8a` (c) `covtype`



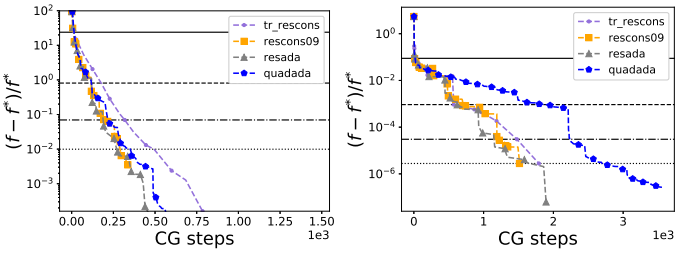
(d) `rcv1` (e) `url` (f) `real-sim`



(g) `yahoojp` (h) `yahookr` (i) `kdd2010a`

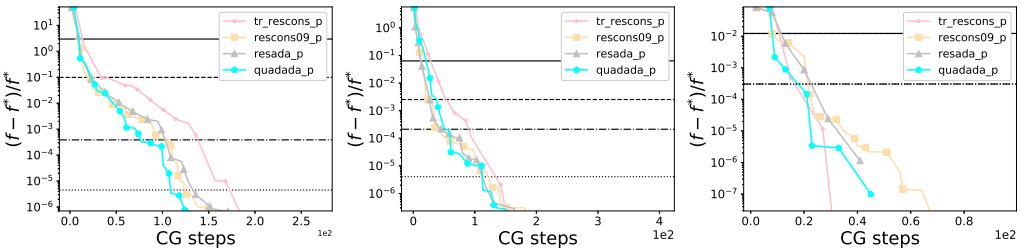


(j) `kdd2010b` (k) `HIGGS` (l) `criteo`

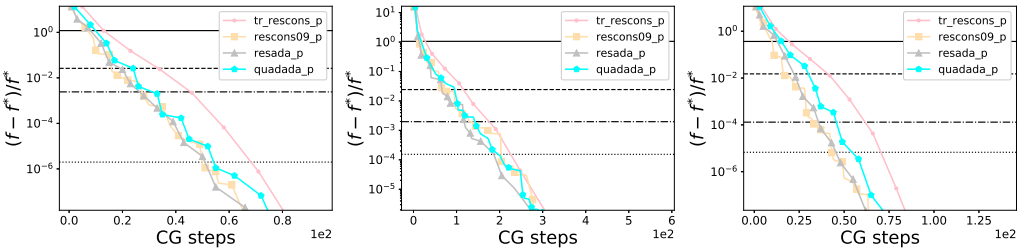


(m) `webspam` (n) `kdd2012`

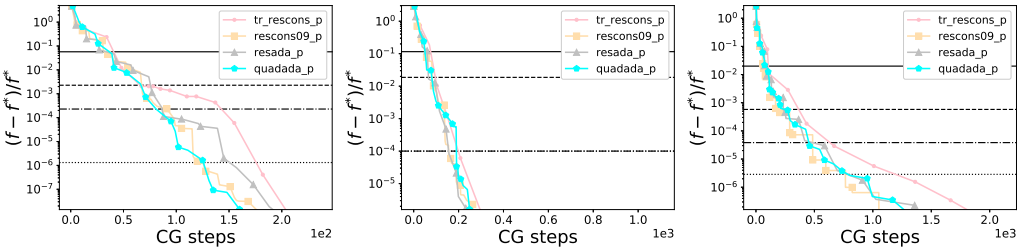
Figure xxiv: A summary comparison between various forcing sequences. We show the convergence of a truncated Newton method for linear SVM. The x-axis shows the cumulative number of CG steps. Loss=L2 and $C = 100C_{\text{Best}}$ in each sub-figure.



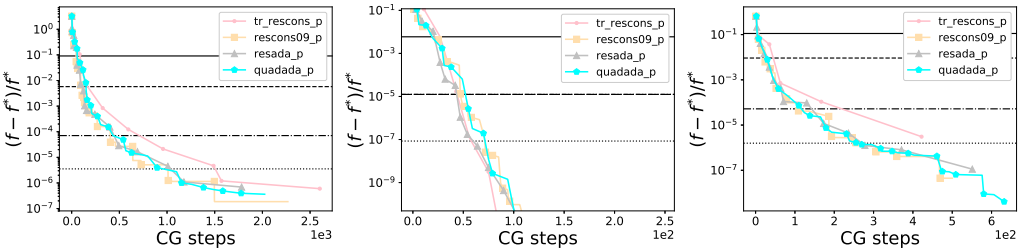
(a) `new20` (b) `w8a` (c) `covtype`



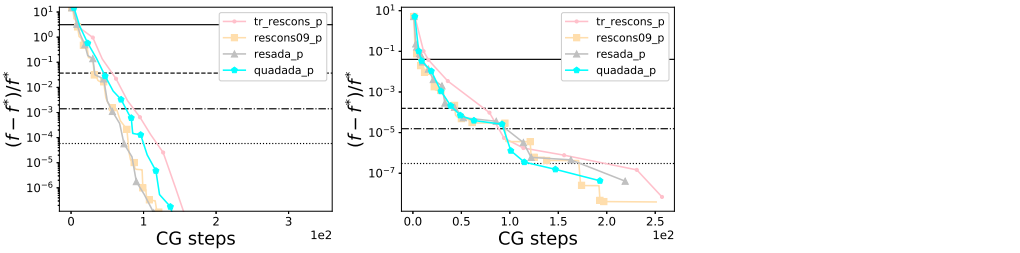
(d) `rcv1` (e) `url` (f) `real-sim`



(g) `yahoojp` (h) `yahookr` (i) `kdd2010a`

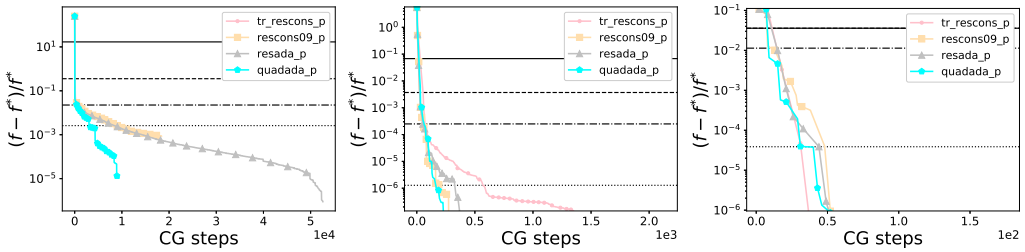


(j) `kdd2010b` (k) `HIGGS` (l) `criteo`



(m) `webspam` (n) `kdd2012`

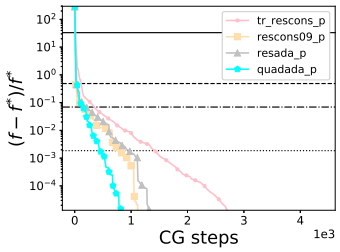
Figure xxv: A summary comparison between various forcing sequences in the preconditioned case. We show the convergence of a truncated Newton method for linear SVM. The x-axis shows the cumulative number of CG steps. Loss=L2 and $C = C_{\text{Best}}$ in each sub-figure.



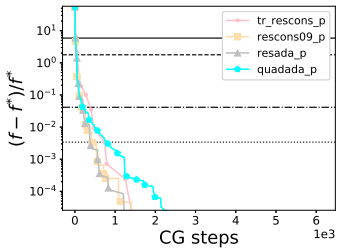
(a) `new20`

(b) `w8a`

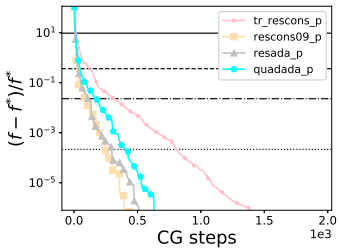
(c) `covtype`



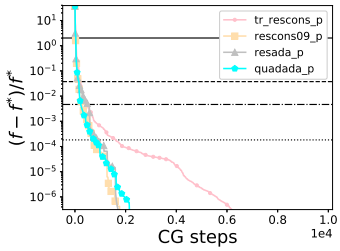
(d) `rcv1`



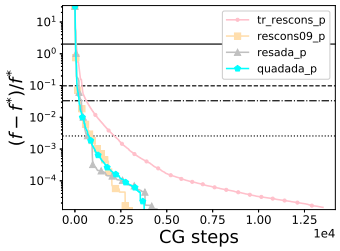
(e) `url`



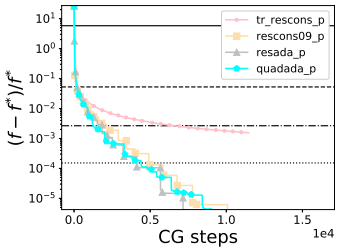
(f) `real-sim`



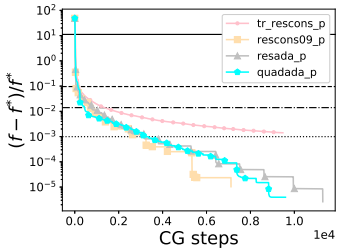
(g) `yahoojp`



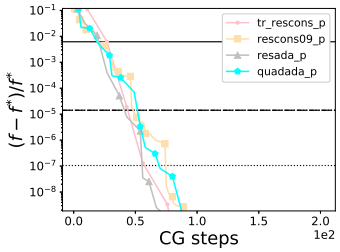
(h) `yahookr`



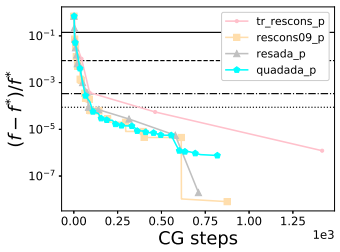
(i) `kdd2010a`



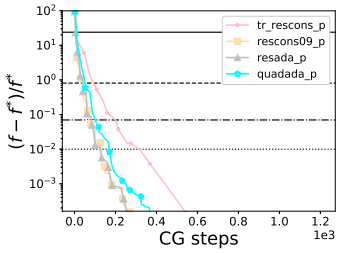
(j) `kdd2010b`



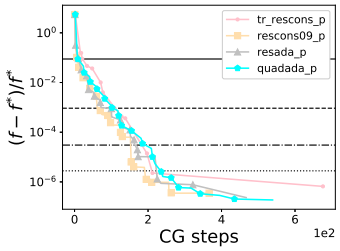
(k) `HIGGS`



(l) `criteo`

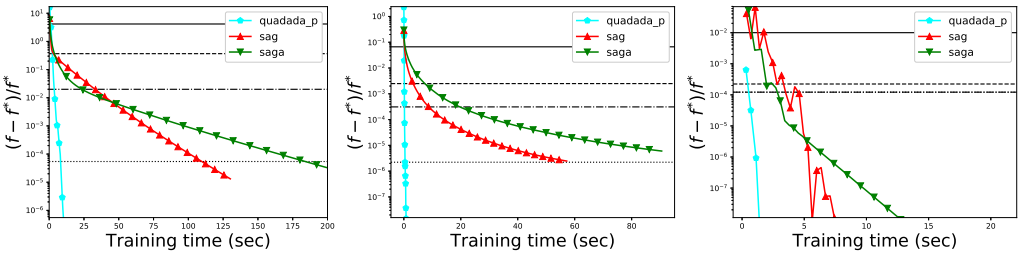


(m) `webspam`

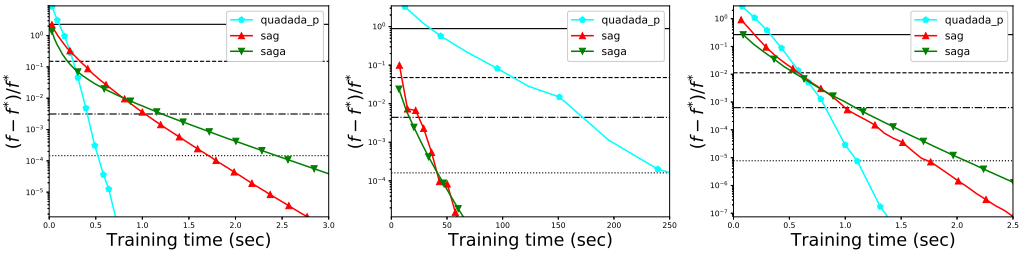


(n) `kdd2012`

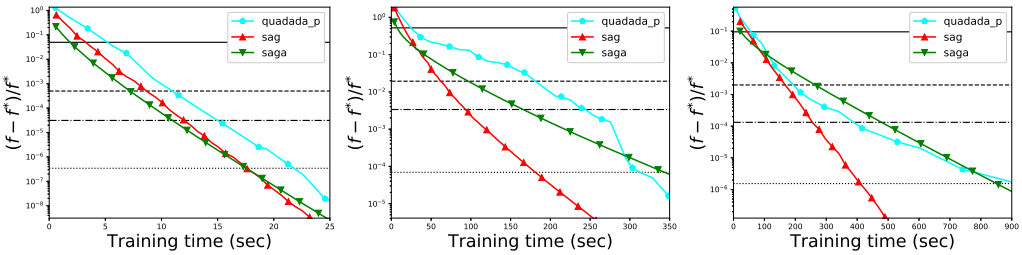
Figure xxvi: A summary comparison between various forcing sequences in the preconditioned case. We show the convergence of a truncated Newton method for linear SVM. The x-axis shows the cumulative number of CG steps. Loss=L2 and $C = 100C_{\text{Best}}$ in each sub-figure.



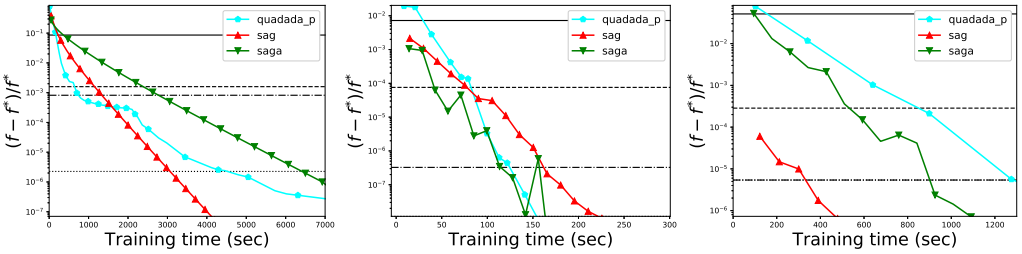
(a) `new20` (b) `w8a` (c) `covtype`



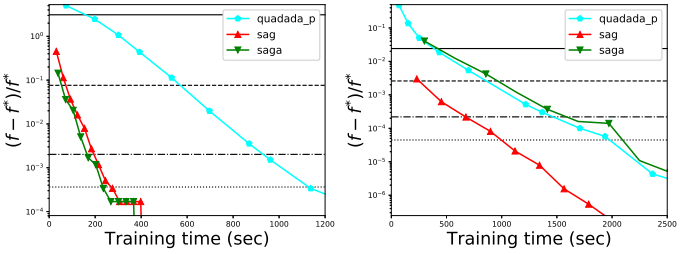
(d) `rcv1` (e) `url` (f) `real-sim`



(g) `yahoojp` (h) `yahookr` (i) `kdd2010a`

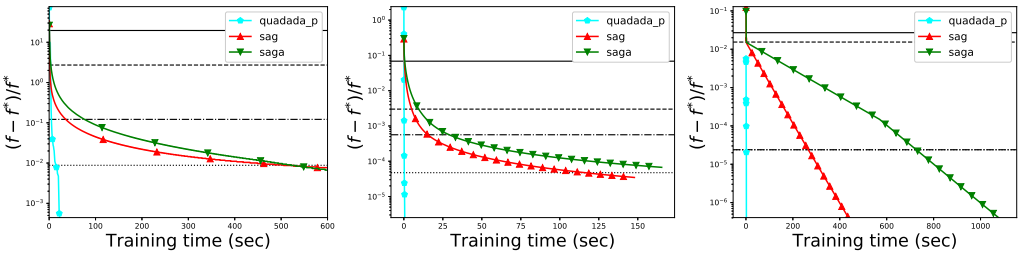


(j) `kdd2010b` (k) `HIGGS` (l) `criteo`

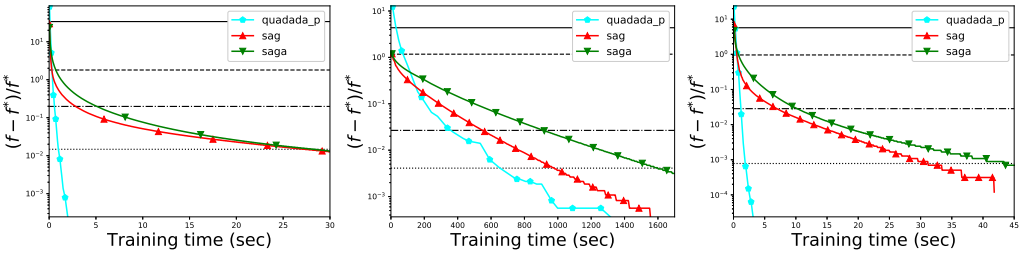


(m) `webspam` (n) `kdd2012`

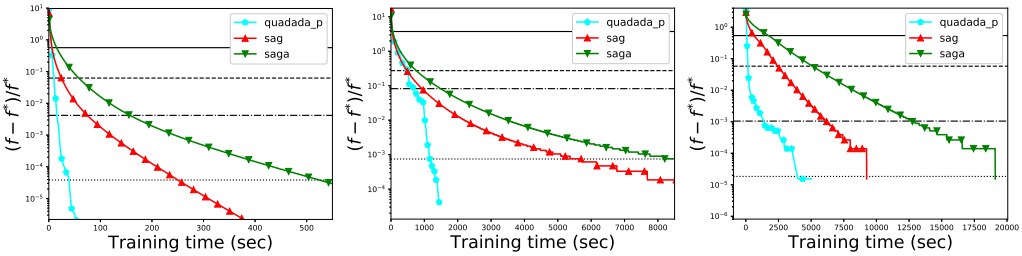
Figure xxvii: A comparison between the proposed implementation of the TNCG and the two state-of-the-art first order methods SAG and SAGA.The x-axis is the cumulative time in seconds. Loss=LR and $C = C_{\text{Best}}$ in each sub-figure.



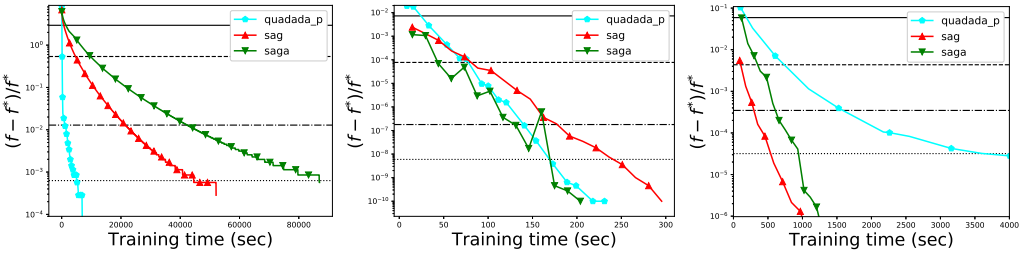
(a) `new20` (b) `w8a` (c) `covtype`



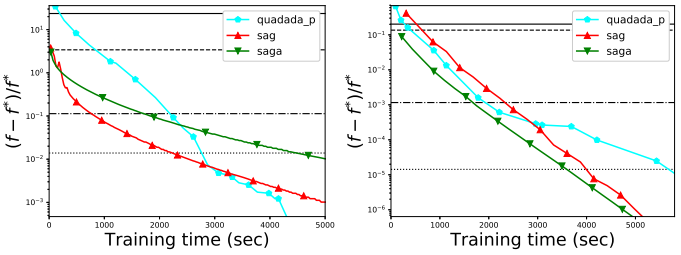
(d) `rcv1` (e) `url` (f) `real-sim`



(g) `yahoojp` (h) `yahookr` (i) `kdd2010a`



(j) `kdd2010b` (k) `HIGGS` (l) `criteo`



(m) `webspam` (n) `kdd2012`

Figure xxviii: A comparison between the proposed implementation of the TNCG and the two state-of-the-art first order methods SAG and SAGA.The x-axis is the cumulative time in seconds. Loss=LR and $C = 100C_{\text{Best}}$ in each sub-figure.