Supplementary Materials: Preconditioned Conjugate Gradient Methods in Truncated Newton Frameworks for Large-scale Linear Classification

Chih-Yang Hsia R04922021@NTU.EDU.TW
Department of Computer Science
National Taiwan University

Wei-Lin Chiang R06922166@CSIE.NTU.EDU.TW
Department of Computer Science
National Taiwan University

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

Editors: Jun Zhu and Ichiro Takeuchi

I. Proof of Theorem 1

To prove Theorem 1, we introduce some lemmas first.

Lemma I *If* $A \in \mathbb{R}^{n\times n}$ *is symmetric positive definite, then*

$$\lambda_1(A) = 1/\lambda_n(A^{-1}), \lambda_n(A) = 1/\lambda_1(A^{-1}).$$

Proof A^{-1} exists because A is positive definite. Assume $\lambda > 0$ and $\boldsymbol{v}$ are an eigenvalue and an eigenvector of A, respectively.

$$A\boldsymbol{v} = \lambda\boldsymbol{v} \iff A^{-1}A\boldsymbol{v} = \lambda A^{-1}\boldsymbol{v} \iff A^{-1}\boldsymbol{v} = \lambda^{-1}\boldsymbol{v}.$$

Therefore, if λ is an eigenvalue for A, then λ^{-1} is an eigenvalue for A^{-1}. Since all eigenvalues of A are positive, we have the smallest eigenvalue of A is the reciprocal of the largest eigenvalue of A^{-1} and vice versa. ■

Lemma II *If* $p_1, p_2 \geq 0, q_1, q_2 > 0, \frac{p_1}{q_1} \neq \frac{p_2}{q_2}$ *and* $0 < \alpha < 1$, *then*

$$\min\{\frac{p_1}{q_1}, \frac{p_2}{q_2}\} < \frac{\alpha p_1 + (1-\alpha)p_2}{\alpha q_1 + (1-\alpha)q_2} < \max\{\frac{p_1}{q_1}, \frac{p_2}{q_2}\}.$$

Proof

$$\frac{\alpha p_1 + (1-\alpha)p_2}{\alpha q_1 + (1-\alpha)q_2} - \frac{p_1}{q_1} = \frac{(1-\alpha)q_2}{\alpha q_1 + (1-\alpha)q_2}(\frac{p_2}{q_2} - \frac{p_1}{q_1}), \tag{i}$$

and

$$\frac{\alpha p_1 + (1-\alpha)p_2}{\alpha q_1 + (1-\alpha)q_2} - \frac{p_2}{q_2} = \frac{\alpha q_1}{\alpha q_1 + (1-\alpha)q_2}(\frac{p_1}{q_1} - \frac{p_2}{q_2}). \tag{ii}$$

From assumptions of this lemma,

$$\frac{(1-\alpha)q_2}{\alpha q_1 + (1-\alpha)q_2} > 0, \frac{\alpha q_1}{\alpha q_1 + (1-\alpha)q_2} > 0. \tag{iii}$$

If

$$\frac{p_1}{q_1} - \frac{p_2}{q_2} > 0,$$

then (i)-(iii) imply

$$\frac{p_1}{q_1} > \frac{\alpha p_1 + (1-\alpha)p_2}{\alpha q_1 + (1-\alpha)q_2} > \frac{p_2}{q_2}.$$

Similarly, if

$$\frac{p_1}{q_1} - \frac{p_2}{q_2} < 0,$$

then

$$\frac{p_1}{q_1} < \frac{\alpha p_1 + (1-\alpha)p_2}{\alpha q_1 + (1-\alpha)q_2} < \frac{p_2}{q_2}.$$

Then the proof is complete.

■

Lemma III (Cholesky factorization, Colloray 7.2.9 in Horn and Johnson (2012)) *A is symmetric positive definite if and only if there exists a unique lower triangular matrix L with positive diagonal entries such that*

$$A = LL^T.$$

Note that a triangular matrix with positive diagonal entry is invertible.

Lemma IV *Let A and $\bar{M}$ be any symmetric positive definite matrices. From Lemma III, we can assume that*

$$A = LL^T \text{ and } \bar{M} = \bar{E}\bar{E}^T. \tag{iv}$$

We have that

$$\bar{E}^T A^{-1} \bar{E} \text{ and } L^{-1}\bar{M}L^{-T} \tag{v}$$

share the same set of eigenvalues. Equivalently,

$$\bar{E}^{-1} A \bar{E}^{-T} \text{ and } L^T \bar{M}^{-1} L \tag{vi}$$

share the same set of eigenvalues.

Proof We prove only the case of (v) because (vi) is an equivalent result by swapping A and $\bar{M}$.

Now assume that λ and $\boldsymbol{v}$ are an eigenvalue and an eigenvector of $\bar{E}^T A^{-1} \bar{E}$, respectively. Therefore,

$$\bar{E}^T A^{-1} \bar{E} \boldsymbol{v} = \lambda \boldsymbol{v},$$

By (iv),

$$\begin{aligned} \bar{E}^T L^{-T} L^{-1} \bar{E} \boldsymbol{v} &= \lambda \boldsymbol{v}, \\ L^{-1} \bar{E} \boldsymbol{v} &= \lambda L^T \bar{E}^{-T} \boldsymbol{v}, \\ L^{-1} \bar{E} \bar{E}^T L^{-T} (L^T \bar{E}^{-T} \boldsymbol{v}) &= \lambda (L^T \bar{E}^{-T} \boldsymbol{v}), \\ L^{-1} \bar{M} L^{-T} (L^T \bar{E}^{-T} \boldsymbol{v}) &= \lambda (L^T \bar{E}^{-T} \boldsymbol{v}). \end{aligned}$$

Thus, λ and $L^T \bar{E}^{-T} \boldsymbol{v}$ are eigenvalue and eigenvector of $L^{-1} \bar{M} L^{-T}$, respectively. ■

Now we are ready for the proof of Theorem 1 in main article.

Proof We first denote $A = H_k$ for simplicity. Because M is positive definite, so is $\bar{M}$. From Lemma III, we have

$$A = LL^T, M = EE^T, \bar{M} = \bar{E}\bar{E}^T, \tag{vii}$$

where $L, E, \bar{E}$ are invertible. From Lemma IV,

$$\begin{aligned} &\kappa(\bar{E}^{-1} A \bar{E}^{-T}) = \kappa(L^T \bar{M}^{-1} L) \\ &= \kappa(L^{-1} \bar{M} L^{-T}) = \frac{\lambda_1((1-\alpha)L^{-1}L^{-T} + \alpha L^{-1} M L^{-T})}{\lambda_n((1-\alpha)L^{-1}L^{-T} + \alpha L^{-1} M L^{-T})}. \end{aligned} \tag{viii}$$

By using the property that for symmetric positive definite matrices A and B,

$$\lambda_1(A) = \|A\| \text{ and } \|A + B\| \leq \|A\| + \|B\|,$$

for the numerator of (viii), we have

$$\begin{aligned} &\lambda_1((1-\alpha)L^{-1}L^{-T} + \alpha L^{-1} M L^{-T}) \\ \leq &(1-\alpha)\lambda_1(L^{-1}L^{-T}) + \alpha \lambda_1(L^{-1} M L^{-T}) \\ = &(1-\alpha)\lambda_1(L^{-1}L^{-T}) + \alpha \lambda_1(E^T A^{-1} E) \qquad \text{(by Lemma IV)} \\ = &\frac{1-\alpha}{\lambda_n(A)} + \frac{\alpha}{\lambda_n(E^{-1} A E^{-T})}. \end{aligned} \tag{ix}$$

Note that for the last equality Lemma I is used. Similarly, for the denominator of (viii) we have

$$\begin{aligned} &\lambda_n((1-\alpha)L^{-1}L^{-T} + \alpha L^{-1} M L^{-T}) \\ \geq &(1-\alpha)\lambda_n(L^{-1}L^{-T}) + \alpha \lambda_n(L^{-1} M L^{-T}) \\ = &(1-\alpha)\lambda_n(L^{-1}L^{-T}) + \alpha \lambda_n(E^T A^{-1} E) \qquad \text{(by Lemma IV)} \\ = &\frac{1-\alpha}{\lambda_1(A)} + \frac{\alpha}{\lambda_1(E^{-1} A E^{-T})}. \end{aligned} \tag{x}$$

From (viii), (ix), and (x),

$$\begin{aligned}\kappa(\bar{E}^{-1}A\bar{E}^{-T}) &= \frac{\lambda_1((1-\alpha)L^{-1}L^{-T}+\alpha L^{-1}ML^{-T})}{\lambda_n((1-\alpha)L^{-1}L^{-T}+\alpha L^{-1}ML^{-T})}\\ &\leq \frac{\alpha/\lambda_n(E^{-1}AE^{-T})+(1-\alpha)/\lambda_n(A)}{\alpha/\lambda_1(E^{-1}AE^{-T})+(1-\alpha)/\lambda_1(A)}.\end{aligned}$$

By Lemma II with

$$\begin{aligned}p_1 &= 1/\lambda_n(E^{-1}AE^{-T}), \quad p_2 = 1/\lambda_n(A),\\ q_1 &= 1/\lambda_1(E^{-1}AE^{-T}), \quad q_2 = 1/\lambda_1(A),\end{aligned}$$

and $\frac{p_1}{q_2} \neq \frac{p_2}{q_2}$ from the assumption, we have Theorem 1.

■

II. CG Convergence Properties

The following well-known theorem shows that a smaller condition number of the matrix leads to faster CG convergence.

Theorem V (Theorem 10.2.6 in Golub and Van Loan (1996)) *Consider a linear system $A\boldsymbol{\xi} = \boldsymbol{b}$, where $A \in \mathbb{R}^{n\times n}$ is symmetric positive definite and $\boldsymbol{b} \in \mathbb{R}^n$. If the CG procedure produces iterates $\{\boldsymbol{\xi}_k\}$, then*

$$\|\boldsymbol{\xi}_k - \boldsymbol{\xi}^*\|_A \leq 2 \times (\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1})^k \|\boldsymbol{\xi}_0 - \boldsymbol{\xi}^*\|_A, \tag{xi}$$

where $\boldsymbol{\xi}^*$ is the solution of the linear system, $\|\boldsymbol{v}\|_A = \sqrt{\boldsymbol{v}^T A\boldsymbol{v}}$ for a vector $\boldsymbol{v}$, and $\kappa = \kappa_2(A)$ is the condition number of matrix A.

Because $\boldsymbol{\xi}^*$ is not available in practice, to make a connection to the stopping condition (8) we give Theorem VI.

Theorem VI *Consider a linear system $A\boldsymbol{\xi} = \boldsymbol{b}$, where $A \in \mathbb{R}^{n\times n}$ is symmetric positive definite and $\boldsymbol{b} \in \mathbb{R}^n$. If the CG procedure produces iterates $\{\boldsymbol{\xi}_k\}$, then*

$$\|A(\boldsymbol{\xi}_k - \boldsymbol{\xi}^*)\| \leq 2\sqrt{\kappa}(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1})^k \|A(\boldsymbol{\xi}_0 - \boldsymbol{\xi}^*)\|. \tag{xii}$$

The proof and more discussion are in the following section. Theorem VI shows that $\kappa(A)$ is strongly related to the convergence of the CG method.

II.1. Proof of Theorem VI

Throughout the discussion we assume that $\lambda_i(A)$ denotes the ith largest eigenvalue of matrix A. To prove Theorem VI, we need

Lemma VII *If $A \in \mathbb{R}^{n\times n}$ is symmetric positive definite, then*

$$\frac{\|A\boldsymbol{b}\|_2}{\sqrt{\lambda_1(A)}} \leq \|\boldsymbol{b}\|_A \leq \frac{\|A\boldsymbol{b}\|_2}{\sqrt{\lambda_n(A)}} \tag{xiii}$$

Proof Because $\|\boldsymbol{b}\|_A^2 = \boldsymbol{b}^T A\boldsymbol{b} = (A\boldsymbol{b})^T A^{-1}(A\boldsymbol{b})$ and A^{-1} is also symmetric positive definite, we have

$$\lambda_n(A^{-1})\|A\boldsymbol{b}\|^2 \leq (A\boldsymbol{b})^T A^{-1}(A\boldsymbol{b}) \leq \lambda_1(A^{-1})\|A\boldsymbol{b}\|^2.$$

From Lemma I we have

$$\lambda_n(A^{-1}) = \lambda_1(A), \lambda_1(A^{-1}) = \lambda_n(A),$$

then

$$\frac{\|A\boldsymbol{b}\|^2}{\lambda_1(A)} \leq (A\boldsymbol{b})^T A^{-1}(A\boldsymbol{b}) \leq \frac{\|A\boldsymbol{b}\|^2}{\lambda_n(A)},$$

and the proof is complete. ■

The proof of Theorem VI.

Proof By Theorem V and Lemma VII, we know that

$$\begin{aligned}
&\|A(\boldsymbol{\xi}_k - \boldsymbol{\xi}^*)\| \leq \sqrt{\lambda_1(A)}\|\boldsymbol{\xi}_k - \boldsymbol{\xi}^*\|_A \\
\leq&\sqrt{\lambda_1(A)} \times 2 \times (\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1})^k\|\boldsymbol{\xi}_0 - \boldsymbol{\xi}^*\|_A \\
\leq&\frac{\sqrt{\lambda_1(A)}}{\sqrt{\lambda_n(A)}} \times 2 \times (\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1})^k\|A(\boldsymbol{\xi}_0 - \boldsymbol{\xi}^*)\| \\
=&2\sqrt{\kappa}(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1})^k\|A(\boldsymbol{\xi}_0 - \boldsymbol{\xi}^*)\|
\end{aligned}$$

Then we have (xii). ■

III. PCG without $M = EE^T$

If a factorization of M is not available, we can still apply PCG by calculating the product $M^{-1}\boldsymbol{v}$ between M^{-1} and a vector $\boldsymbol{v}$. We will see such an example in Section 3.2. To use such a preconditioner, it is known (Golub and Van Loan, 1996) that Algorithm I can be performed without a factorization of M. We let

$$\begin{aligned}
\bar{\boldsymbol{s}}_i = E^{-T}\hat{\boldsymbol{s}}, \boldsymbol{d} = E^{-T}\hat{\boldsymbol{d}}, \\
\boldsymbol{r} = E\hat{\boldsymbol{r}}, \boldsymbol{z} = M^{-1}\boldsymbol{r}.
\end{aligned}$$

After doing the substitutions in Algorithm I, E is no longer needed; see details in Algorithm 3. The calculating in Algorithm I involving E^{-1} and E^{-T} is now moved to line 14 in Algorithm 3, where a product between M^{-1} and a vector is needed. However, in checking if

Algorithm I: PCG for solving the transformed trust region sub-problem (14). Assume M has been factorized to EE^T.

```
1  Given $\varepsilon_{\text{CG}} < 1, \Delta_k > 0$, let $\hat{\boldsymbol{s}} = \boldsymbol{0}, \hat{\boldsymbol{g}}_k = E^{-1}\nabla f(\boldsymbol{w}_k), \hat{\boldsymbol{r}} = \hat{\boldsymbol{d}} = -\hat{\boldsymbol{g}}_k$
2  $\texttt{rTr} \leftarrow \hat{\boldsymbol{r}}^T\hat{\boldsymbol{r}}$
3  while True do
4      if $\sqrt{\texttt{rTr}} < \varepsilon_{\text{CG}}\|\hat{\boldsymbol{g}}_k\|$ then
5          return $\boldsymbol{s}_k = E^{-T}\hat{\boldsymbol{s}}$
6      end
7      $\hat{\boldsymbol{v}} \leftarrow E^{-1}(\nabla^2 f(\boldsymbol{w}_k)(E^{-T}\hat{\boldsymbol{d}}))$,
8      $\alpha \leftarrow \|\hat{\boldsymbol{r}}\|^2/(\hat{\boldsymbol{d}}^T\hat{\boldsymbol{v}}), \hat{\boldsymbol{s}} \leftarrow \hat{\boldsymbol{s}} + \alpha\hat{\boldsymbol{d}}$
9      if $\|\hat{\boldsymbol{s}}\| \geq \hat{\Delta}_k$ then
10         $\hat{\boldsymbol{s}} \leftarrow \hat{\boldsymbol{s}} - \alpha\hat{\boldsymbol{d}}$
11         compute $\tau$ such that $\left\|\hat{\boldsymbol{s}} + \tau\hat{\boldsymbol{d}}\right\| = \hat{\Delta}_k$
12         return $\boldsymbol{s}_k = E^{-T}(\hat{\boldsymbol{s}} + \tau\hat{\boldsymbol{d}})$
13     end
14     $\hat{\boldsymbol{r}} \leftarrow \hat{\boldsymbol{r}} - \alpha\hat{\boldsymbol{v}}$
15     $\texttt{rTr}_{\texttt{new}} \leftarrow \hat{\boldsymbol{r}}^T\hat{\boldsymbol{r}}$
16     $\beta \leftarrow \texttt{rTr}_{\texttt{new}}/\texttt{rTr}, \hat{\boldsymbol{d}} \leftarrow \hat{\boldsymbol{r}} + \beta\hat{\boldsymbol{d}}$
17     $\texttt{rTr} \leftarrow \texttt{rTr}_{\texttt{new}}$
18 end
```

iterates exceed the trust region boundary (lines 9-11), we now need matrix-vector products $M\boldsymbol{s}$ and $M\boldsymbol{d}$. If each product is more expensive than a vector operation, a trick to save the computational cost is by using cheap vector operations to maintain these products. Specifically, after $\bar{\boldsymbol{s}}$ is updated we calculate

$$M\bar{\boldsymbol{s}} \leftarrow (M\bar{\boldsymbol{s}}) + \alpha(M\boldsymbol{d}), \tag{xiv}$$

and after $\bar{\boldsymbol{d}}$ is updated we calculate

$$M\boldsymbol{d} \leftarrow \boldsymbol{r} + \beta(M\boldsymbol{d}). \tag{xv}$$

Thus the cost of Algorithm 3 is still similar to that of Algorithm I.

IV. Implementation Details of Trust-region Newton PCG

We implement the proposed modified diagonal preconditioner (21). By using PCG rather than CG, many implementation details must be addressed.

IV.1. Matrix-vector Products in PCG

We begin with discussing if Algorithm I or 3 should be considered. We list the operations of the two algorithms by a line-by-line comparison in Table I. Note that we only include $\mathcal{O}(n)$ operations and exclude operations in the last CG step if the direction exceeds the

Algorithm I	Algorithm 3
$\boxed{E^{-T}\hat{\boldsymbol{d}}}$	
$\nabla^2 f(\boldsymbol{w}_k)(E^{-T}\hat{\boldsymbol{d}})$	$\nabla^2 f(\boldsymbol{w}_k)\boldsymbol{d}$
$\boxed{E^{-1}(\nabla^2 f(\boldsymbol{w}_k)(E^{-T}\hat{\boldsymbol{d}}))}$	
$\hat{\boldsymbol{r}}^T\hat{\boldsymbol{r}}$	$\boldsymbol{r}^T\boldsymbol{z}$
$\hat{\boldsymbol{d}}^T\hat{\boldsymbol{v}}$	$\boldsymbol{d}^T\boldsymbol{v}$
$\hat{\boldsymbol{s}} \leftarrow \hat{\boldsymbol{s}} + \alpha\hat{\boldsymbol{d}}$	$\bar{\boldsymbol{s}} \leftarrow \bar{\boldsymbol{s}} + \alpha\boldsymbol{d}$
	$\boxed{M\bar{\boldsymbol{s}}}$
$\hat{\boldsymbol{s}}^T\hat{\boldsymbol{s}}$	$\bar{\boldsymbol{s}}^T M\bar{\boldsymbol{s}}$
$\hat{\boldsymbol{r}} \leftarrow \hat{\boldsymbol{r}} - \alpha\hat{\boldsymbol{v}}$	$\boldsymbol{r} \leftarrow \boldsymbol{r} - \alpha\boldsymbol{v}$
	$\boxed{\boldsymbol{z} \leftarrow M^{-1}\boldsymbol{r}}$
$\hat{\boldsymbol{d}} \leftarrow \hat{\boldsymbol{r}} + \beta\hat{\boldsymbol{d}}$	$\boldsymbol{d} \leftarrow \boldsymbol{z} + \beta\boldsymbol{d}$

Table I: A line-by-line comparison of Algorithms I and 3

trust region. In Table I, we can see that the operations in the two algorithms are almost the same except for the four framed equations. Because now M is a diagonal matrix, the cost of each matrix-vector product is the same as one vector operation. Therefore, the total complexity of Algorithms I and 3 are similar.

However, a possible advantage of using Algorithm 3 is the numerical stability. In Algorithm I, we do factorization on $M = EE^T$ and two additional matrix-vector products are calculated in $E^{-1}(\nabla^2 f(\boldsymbol{w}_k)(E^{-T}\hat{\boldsymbol{d}}))$. Now M is diagonal so M^{-1} can be directly calculated. In Algorithm 3, M is used in two places but only one is for updating the solution (the other is for checking if $\|\bar{\boldsymbol{s}}\| > \Delta_k$). Thus Algorithm 3 may have better stability then Algorithm I. We then choose to implement Algorithm 3.

If Algorithm 3 is used, because the preconditioner M is a diagonal matrix, the trick discussed in Section III of maintaining $M\boldsymbol{d}$ and $M\boldsymbol{s}$ may not be useful. Applying the trick makes the operation

$$M\bar{\boldsymbol{s}}$$

become

$$\begin{aligned} M\boldsymbol{d} &\leftarrow \boldsymbol{r} + \beta(M\boldsymbol{d}) \\ M\bar{\boldsymbol{s}} &\leftarrow M\bar{\boldsymbol{s}} + \alpha(M\boldsymbol{d}), \end{aligned}$$

which incurs additional costs and hence we do not consider it.

IV.2. Additional Memory Usage

The memory usage is slightly increased because a diagonal preconditioner (stored as an array) and an additional vector are needed. To be specific, by comparing Algorithm 1 with Algorithm 3, we can see M and $\boldsymbol{z}$ are used, where both of them are stored in an array with $\mathcal{O}(n)$ size. Another difference is $\|\bar{\boldsymbol{s}} + \tau\boldsymbol{d}\|_M$ calculation in line 11, which involves an $\boldsymbol{s}^T M\boldsymbol{d}$ type of computation. One way is to calculate the matrix-vector multiplication $M\boldsymbol{d}$ first and then do an inner product with $\boldsymbol{s}$, while this requires an additional vector to store the result. Now because M is a diagonal preconditioner, $\boldsymbol{s}^T M \boldsymbol{d}$ can be computed directly by multiplying the corresponding components in the three vectors. We then consider this implementation to save memory. Therefore, the additional memory costs in our implementation for using PCG is $2\mathcal{O}(n)$.

IV.3. Trust-region Update Rule in Newton PCG

In LIBLINEAR, the trust region rule is

$$\Delta_{k+1} = \begin{cases} \min((\max(\alpha_k^*, \gamma_1))\left\|\boldsymbol{s}^k\right\|, \gamma_2\Delta_k), & \text{if } \rho < \eta_0, \\ \max(\gamma_1\Delta_k, \min(\alpha_k^*\left\|\boldsymbol{s}^k\right\|, \gamma_2\Delta_k)), & \text{if } \rho \in [\eta_0, \eta_1], \\ \max(\gamma_1\Delta_k, \min(\alpha_k^*\left\|\boldsymbol{s}^k\right\|, \gamma_3\Delta_k)), & \text{if } \rho \in (\eta_1, \eta_2), \\ \max(\Delta_k, \min(\alpha_k^*\left\|\boldsymbol{s}^k\right\|, \gamma_3\Delta_k)), & \text{if } \rho \geq \eta_2 \text{ and } \left\|\boldsymbol{s}^k\right\| < \Delta_k, \\ \gamma_3\Delta_k, & \text{if } \rho \geq \eta_2 \text{ and } \left\|\boldsymbol{s}^k\right\| = \Delta_k. \end{cases}$$

where

$$\alpha_k^* = \frac{-\nabla f(\boldsymbol{w}^k)^T\boldsymbol{s}^k}{2(f(\boldsymbol{w}^k + \boldsymbol{s}^k) - f(\boldsymbol{w}^k) - \nabla f(\boldsymbol{w}^k)^T\boldsymbol{s}^k)}$$

is the minimum of $\phi(\alpha)$, a quadratic interpolation of $f(\boldsymbol{w}^k + \alpha\boldsymbol{s}^k)$ such that

$$\phi(0) = f(\boldsymbol{w}^k), \quad \phi'(0) = \nabla f(\boldsymbol{w}^k)^T\boldsymbol{s}^k, \quad \phi(1) = f(\boldsymbol{w}^k + \boldsymbol{s}^k).$$

Now

$$\begin{aligned} & f(\boldsymbol{w}^k + \alpha\boldsymbol{s}^k) \\ =& f(\boldsymbol{w}^k + \alpha E^{-T}\hat{\boldsymbol{s}}) \\ =& f(\boldsymbol{w}^k + E^{-T}(\alpha\hat{\boldsymbol{s}})). \end{aligned}$$

We conjecture that $\alpha_k^*\left\|\hat{\boldsymbol{s}}^k\right\|$ can be used as an estimate of the trust region rule. Therefore, the update rule becomes

$$\Delta_{k+1} = \begin{cases} \min((\max(\alpha_k^*, \gamma_1))\left\|\hat{\boldsymbol{s}}^k\right\|, \gamma_2\Delta_k), & \text{if } \rho < \eta_0, \\ \max(\gamma_1\Delta_k, \min(\alpha_k^*\left\|\hat{\boldsymbol{s}}^k\right\|, \gamma_2\Delta_k)), & \text{if } \rho \in [\eta_0, \eta_1], \\ \max(\gamma_1\Delta_k, \min(\alpha_k^*\left\|\hat{\boldsymbol{s}}^k\right\|, \gamma_3\Delta_k)), & \text{if } \rho \in (\eta_1, \eta_2), \\ \max(\Delta_k, \min(\alpha_k^*\left\|\hat{\boldsymbol{s}}^k\right\|, \gamma_3\Delta_k)), & \text{if } \rho \geq \eta_2 \text{ and } \left\|\hat{\boldsymbol{s}}^k\right\| < \Delta_k, \\ \gamma_3\Delta_k, & \text{if } \rho \geq \eta_2 \text{ and } \left\|\hat{\boldsymbol{s}}^k\right\| = \Delta_k. \end{cases}$$

Note that this setting is different from Lin and Moré (1999), in which they solve a trust-region sub-problem under the constraint

$$\|\hat{\boldsymbol{s}}\| \leq \Delta_k,$$

but use $\alpha_k^* \|\boldsymbol{s}^k\|$ in the update rule. Our setting here seems to be more reasonable.

For the initial Δ_0, past works (Lin et al., 2008) suggest $\Delta_0 = \|\nabla f(\boldsymbol{w}_k)\|$. Now for PCG, because $\hat{\boldsymbol{s}} = E\boldsymbol{s}$, we change Δ_0 to

$$\Delta_0 = \|E\nabla f(\boldsymbol{w}_k)\| = \|\nabla f(\boldsymbol{w}_k)\|_M$$

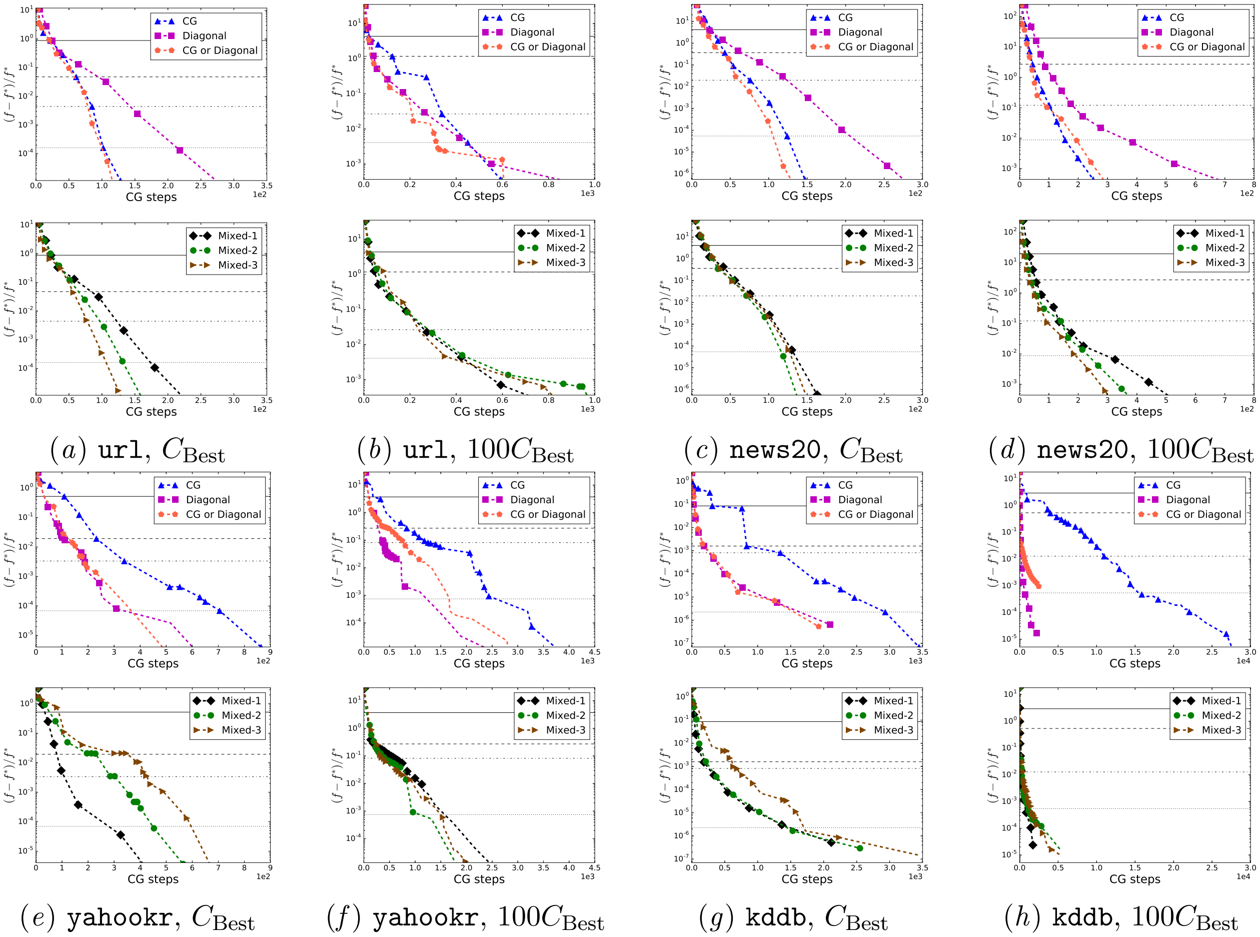
Experimental results on different choices of Δ_0 are in Section X.7.

V. More Details of Experimental Settings

Most timing experiments are conducted on a machine with Intel i7-6900K CPU and 128 GB memory. Data sets for the experiments are listed in Table II. For machine learning applications, it is not necessary to accurately solve the optimization problem because the test accuracy may be similar when $\boldsymbol{w}_k$ is close enough to $\boldsymbol{w}^*$. Therefore, in each figure we draw horizontal lines to indicate when the algorithm satisfies a stopping condition. We consider the following stopping condition used by LIBLINEAR:

$$\|\boldsymbol{\nabla} f(\boldsymbol{w}_k)\| \leq \epsilon \frac{\min(\#\text{pos}, \#\text{neg})}{l} \|\boldsymbol{\nabla} f(\boldsymbol{w}_0)\|, \tag{xvi}$$

where #pos, #neg are the numbers of positive- and negative-labeled instances respectively, and l is the total number of instances. Then in each figure several lines are provided to indicated that (xvi) is satisfied under $\epsilon = \{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$. Note that the default tolerance used in LIBLINEAR is $\epsilon = 10^{-2}$.



(a) url, C_{Best} (b) url, $100C_{\text{Best}}$ (c) news20, C_{Best} (d) news20, $100C_{\text{Best}}$

(e) yahookr, C_{Best} (f) yahookr, $100C_{\text{Best}}$ (g) kddb, C_{Best} (h) kddb, $100C_{\text{Best}}$

Figure i: Each sub-figure is separated to two parts because of too many curves. The upper part shows the convergence of the trust region Newton method without a preconditioner, with a diagonal preconditioner and by using techniques in Section 4.1. The lower one shows the convergence of using the preconditioner in (21) with different α values. Logistic loss is used. We show a relative difference to the optimal function value (log-scaled) versus the total CG steps. Horizontal lines show that LIBLINEAR's stopping condition with tolerances $10^{-1}, 10^{-2}(\text{default}), 10^{-3}$ and 10^{-4} is reached.

VI. The Sensitivity of the α Value in the Mixed Approach

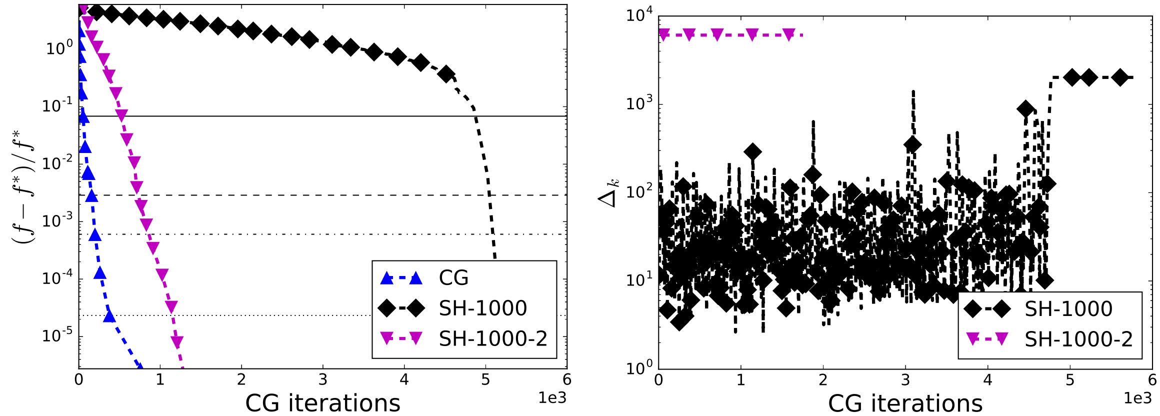
For the proposed method in Section 4 by combining a preconditioner and the identity matrix, we check the performance by considering the following settings.

- Mixed-1: the preconditioner in (21) with $\alpha = 10^{-1}$
- Mixed-2: the preconditioner in (21) with $\alpha = 10^{-2}$
- Mixed-3: the preconditioner in (21) with $\alpha = 10^{-3}$

Results in Figure i (lower part of each sub-figure) show that the proposed setting leads to a preconditioner more robust than the diagonal preconditioner.

For the selection of α, Figures i(e) and i(g) show that $\alpha = 10^{-3}$ may cause the resulting preconditioner to be close to the setting without preconditioning. Otherwise, the performance is not too sensitive to the change of α.

VII. Robustness of Sub-sampled Hessian Preconditioner



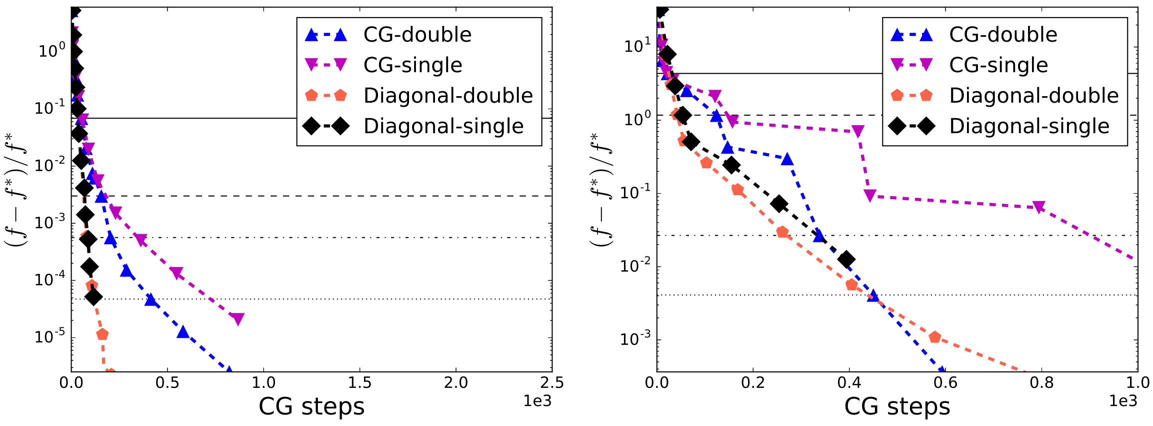
(a) w8a, $10C_{\text{Best}}$

(b) w8a, $10C_{\text{Best}}$

Figure ii: (a). Convergence of using different random seeds in SH-1000. (b). The changes of Δ_k in using different random seeds in SH-1000. In all figures x-axes are the cumulative number of CG iterations.

To investigate the robustness of the sub-sampled Hessian preconditioner, in Figure ii(a) by considering the data set w8a, we show the results of using two random seeds in selecting $\bar{l} = 1,000$ instances. The resulting preconditioners lead to very different convergence behavior. This data set has a small number of 300 features. The average number of non-zero feature values per instance is 11.65 ± 11.35. Further, the data set is unbalanced with only 3.06% instances as positive. Therefore, the variance in selecting the sub-sampled Hessian $\bar{H}_k$ might be high. We further present in Figure ii(b) the relation between the trust-region size Δ_k and the cumulative number of CG iterations. Under one random seed, probably because of bad approximations in the beginning, the truncated Newton method goes into a region in which it is more difficult to reduce the function value. Then we see that the size Δ_k of the trust region remains small for many iterations. This example fully demonstrates that because we solve a sequence of sub-problems in the truncated Newton framework, the change at one Newton iteration (e.g., using PCG rather than CG) can have a global impact. Our proposed methods in Section 4 takes this property into account. It avoids the situation where a poor preconditioner at one iteration causes subsequent iterations to go through a bad path of needing lengthy running time.

In summary, our experiments seem to indicate that the sub-sampled Hessian preconditioner is not robust enough for practical use.



(a) `w8a`, $100C_{\text{Best}}$ (b) `url`, $100C_{\text{Best}}$

Figure iii: A comparison between using single- or double-precision floating points. Logistic loss is used. We show a relative difference to the optimal function value (log-scaled) versus the cumulative number of CG iterations.

VIII. Improving Numerical Stability by Preconditioning

For all experimental results presented so far, we use double-precision floating-point numbers. We observe that using lower-precision (i.e., single precision) floating-point numbers may significantly hurt the convergence of the Newton method. In Figure iii, we compare the performance of single- and double-precision implementations. Results on more sets can be found in Section X.4. Clearly if no preconditioning, the implementation of using single precision has much slower convergence than that of using double precision. Apparently the larger numerical error by using lower precision leads to a poorer direction. Interestingly, for the same sets, if the diagonal preconditioner is applied, the two implementations have very similar convergence behavior. Therefore, besides the possible reduction of training time, a preconditioner may help to improve the numerical stability of Newton methods when a lower precision is considered.

Table II: Data statistics. C_{Best} is the regularization parameter selected by cross validation.

Data sets	#instances	#features	$\log_2(C_{\text{Best}})$	
			LR	L2
`news20`	19,996	1,355,191	9	3
`w8a`	49,749	300	8	2
`covtype`	581,012	54	-23	-26
`rcv1`	677,399	47,236	4	-1
`yahoojp`	176,203	832,026	3	-1
`yahookr`	460,554	3,052,939	6	1
`url`	2,396,130	3,231,962	-7	-10
`kdda`	8,407,752	20,216,831	-3	-5
`kddb`	19,264,097	29,890,095	-1	-4
`criteo`	45,840,617	1,000,000	-15	-12
`kdd12`	149,639,105	54,686,452	-4	-11

IX. Other Diagonal Preconditioners

It is proved in Roma (2005) that the diagonal preconditioner M' with

$$M'_{ii} = \|(H_k)_{i,:}\|_1$$

has the following property

$$\kappa_1(H_k(M')^{-1}) \leq \min_{D \in \text{ diagonal matrices}} \kappa_1(H_k D^{-1}),$$

where κ_1 is the 1-norm condition number of a matrix. Because we do not explicitly form the matrix H_k, the matrix M' cannot be easily obtained. Therefore, Roma (2005) proposes using

$$M_{ii} = \left| \sum_{j=1}^{n} (H_k)_{ij} \right| \approx \|(H_k)_{i,:}\|_1, \tag{xvii}$$

where a matrix-vector product

$$H_k \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}$$

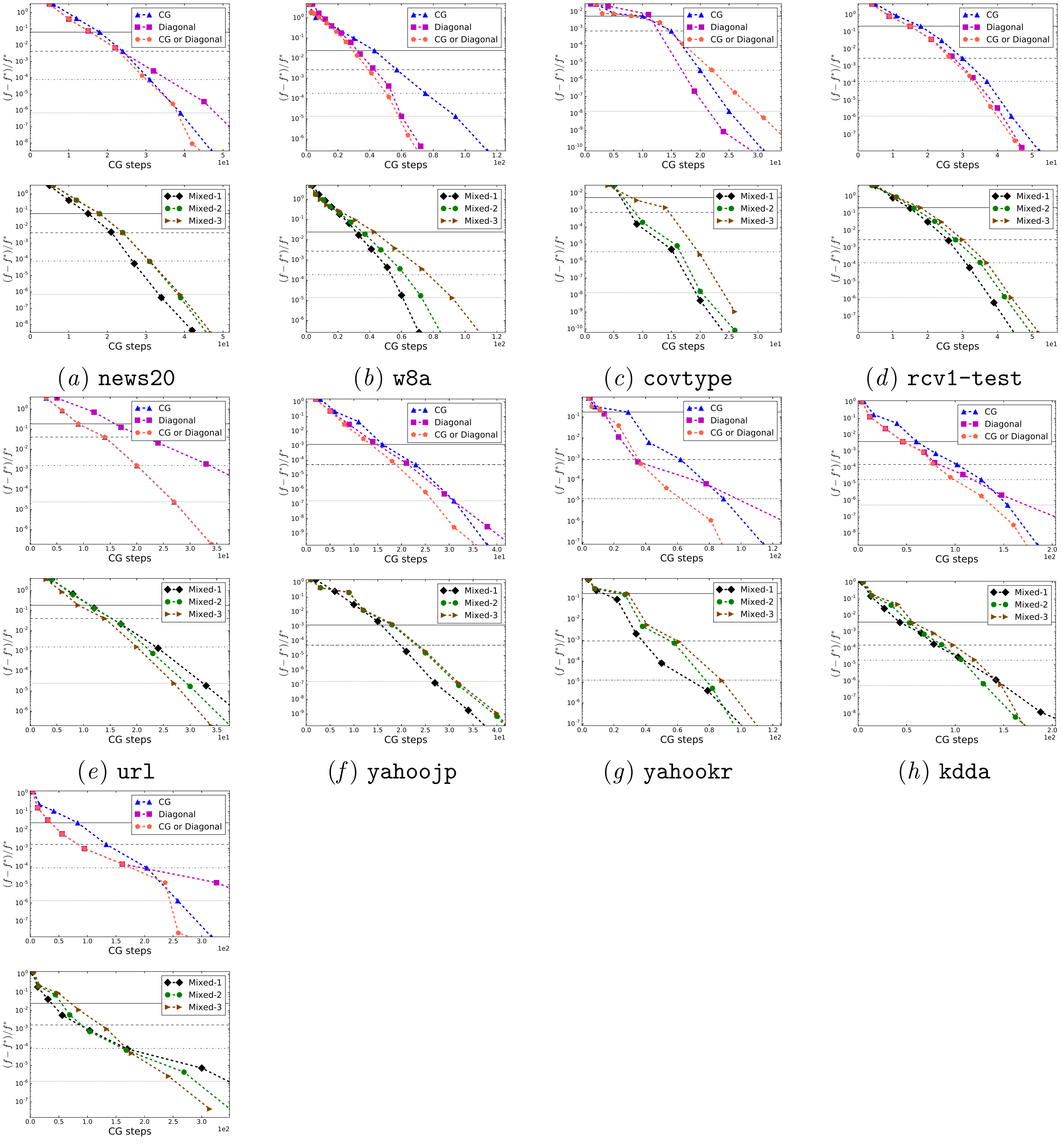
gives all the needed values.

However, a smaller κ_1 may not imply a smaller κ_2 used in Theorem V for the convergence of the CG procedure. Our preliminary results show that (xvii) does not lead to better convergence than the setting of using H_k's diagonal elements.

X. Additional Experimental Results

The following figures give complete results with more data sets listed in Table II of our experiments, where figures in Section X.1 are additional experimental results for Section 5.1, figures in Section X.2 are additional experimental results for Section 5.2, figures in Section X.3 are additional experimental results for Section 5.3, figures in Section X.4 are additional experimental results for Section VIII, figures in Section X.5 (logistic loss) and X.6 (L2 loss) are complete results under different C values, and figures in Section X.7 are experimental results of using different initial Δ values when using the diagonal preconditioner for PCG.

X.1. More Experimental Results in Section 5.1

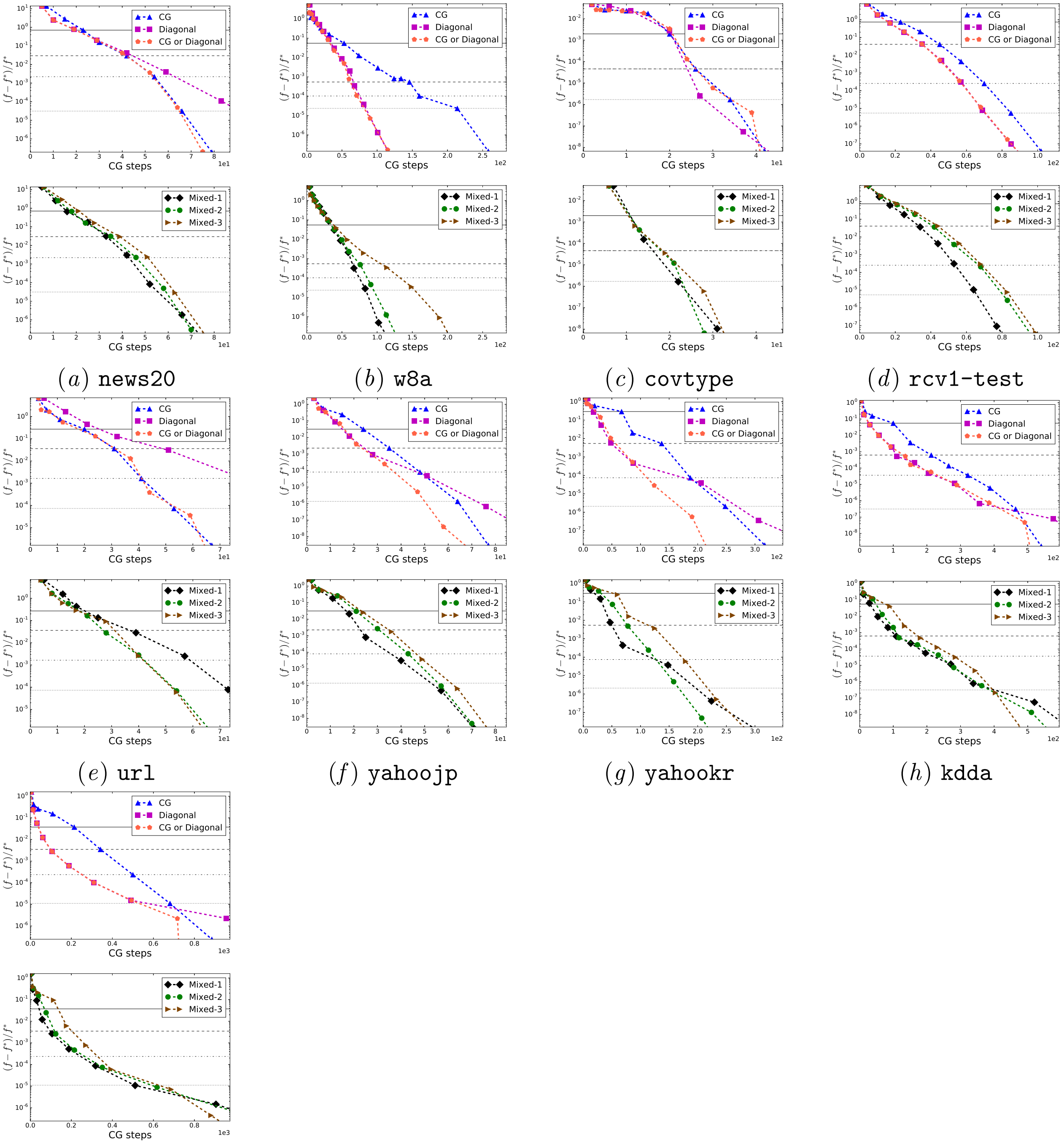


(a) news20 (b) w8a (c) covtype (d) rcv1-test

(e) url (f) yahoojp (g) yahookr (h) kdda

(i) kddb

Figure iv: For each sub-figure, the upper part shows the convergence of the trust region Newton method without a preconditioner, with a diagonal preconditioner and by using techniques in Section 4.1. The lower one shows the convergence of using the preconditioner in (21) with different α values. Logistic loss is used. $C = 0.01C_{\text{Best}}$. Other settings are the same as those in Figure 1.

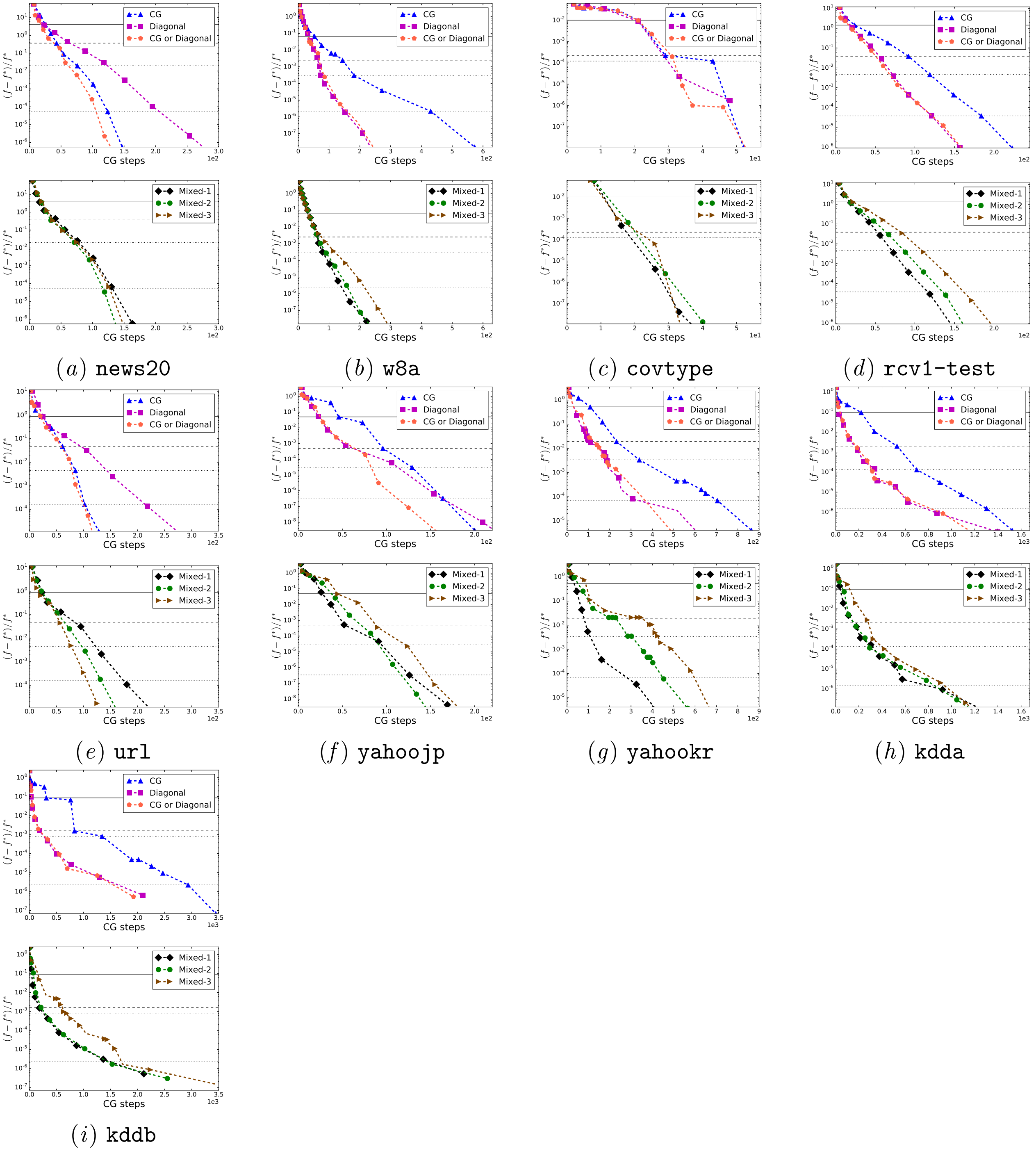


(a) `news20` (b) `w8a` (c) `covtype` (d) `rcv1-test`

(e) `url` (f) `yahoojp` (g) `yahookr` (h) `kdda`

(i) `kddb`

Figure v: For each sub-figure, the upper part shows the convergence of the trust region Newton method without a preconditioner, with a diagonal preconditioner and by using techniques in Section 4.1. The lower one shows the convergence of using the preconditioner in (21) with different α values. Logistic loss is used. $C = 0.1C_{\text{Best}}$. Other settings are the same as those in Figure 1.



(a) news20

(b) w8a

(c) covtype

(d) rcv1-test

(e) url

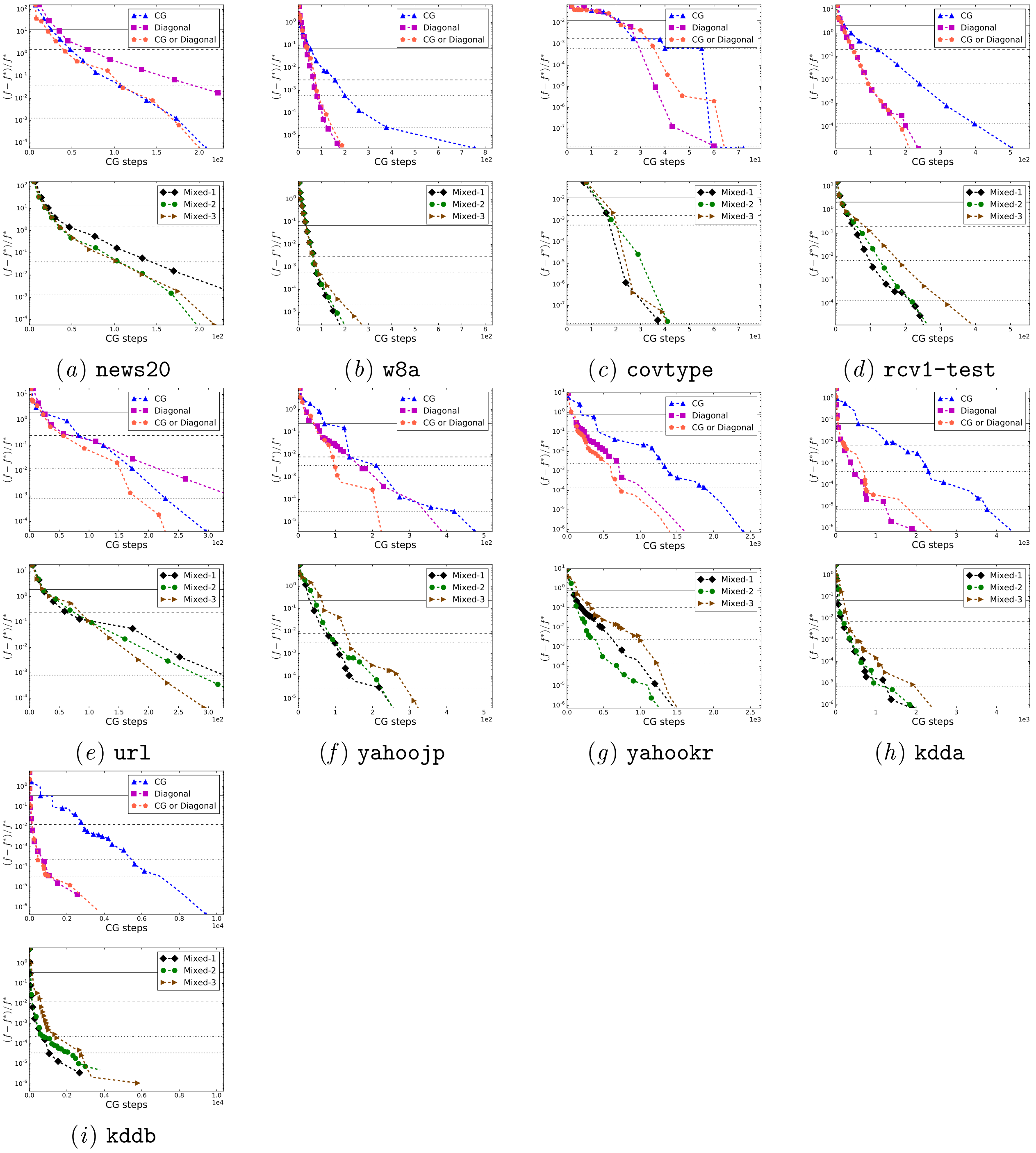
(f) yahoojp

(g) yahookr

(h) kdda

(i) kddb

Figure vi: For each sub-figure, the upper part shows the convergence of the trust region Newton method without a preconditioner, with a diagonal preconditioner and by using techniques in Section 4.1. The lower one shows the convergence of using the preconditioner in (21) with different α values. Logistic loss is used. $C = C_{\text{Best}}$. Other settings are the same as those in Figure 1.

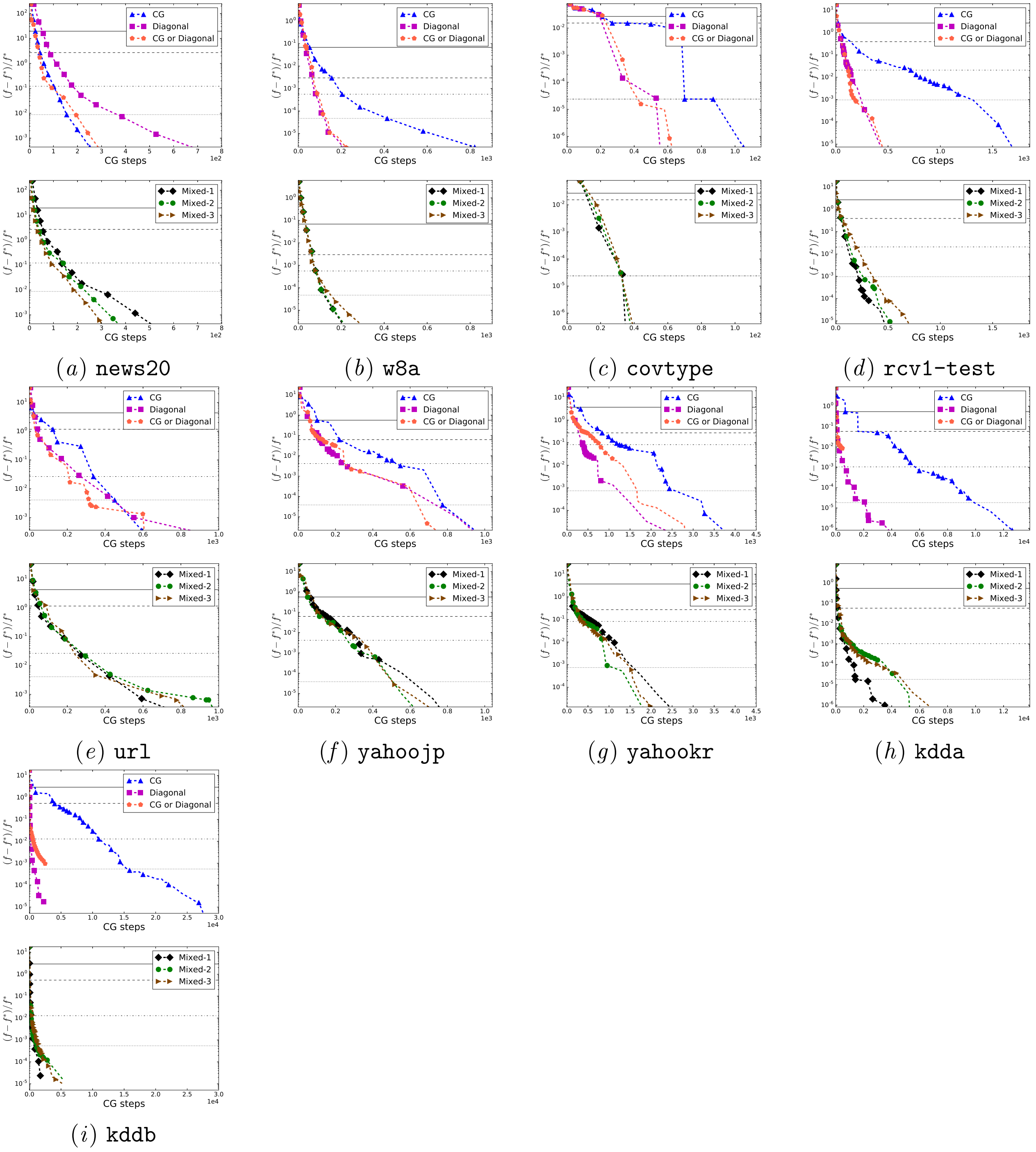


(*a*) news20 (*b*) w8a (*c*) covtype (*d*) rcv1-test

(*e*) url (*f*) yahoojp (*g*) yahookr (*h*) kdda

(*i*) kddb

Figure vii: For each sub-figure, the upper part shows the convergence of the trust region Newton method without a preconditioner, with a diagonal preconditioner and by using techniques in Section 4.1. The lower one shows the convergence of using the preconditioner in (21) with different α values. Logistic loss is used. $C = 10C_{\text{Best}}$. Other settings are the same as those in Figure 1.



(a) news20

(b) w8a

(c) covtype

(d) rcv1-test

(e) url

(f) yahoojp

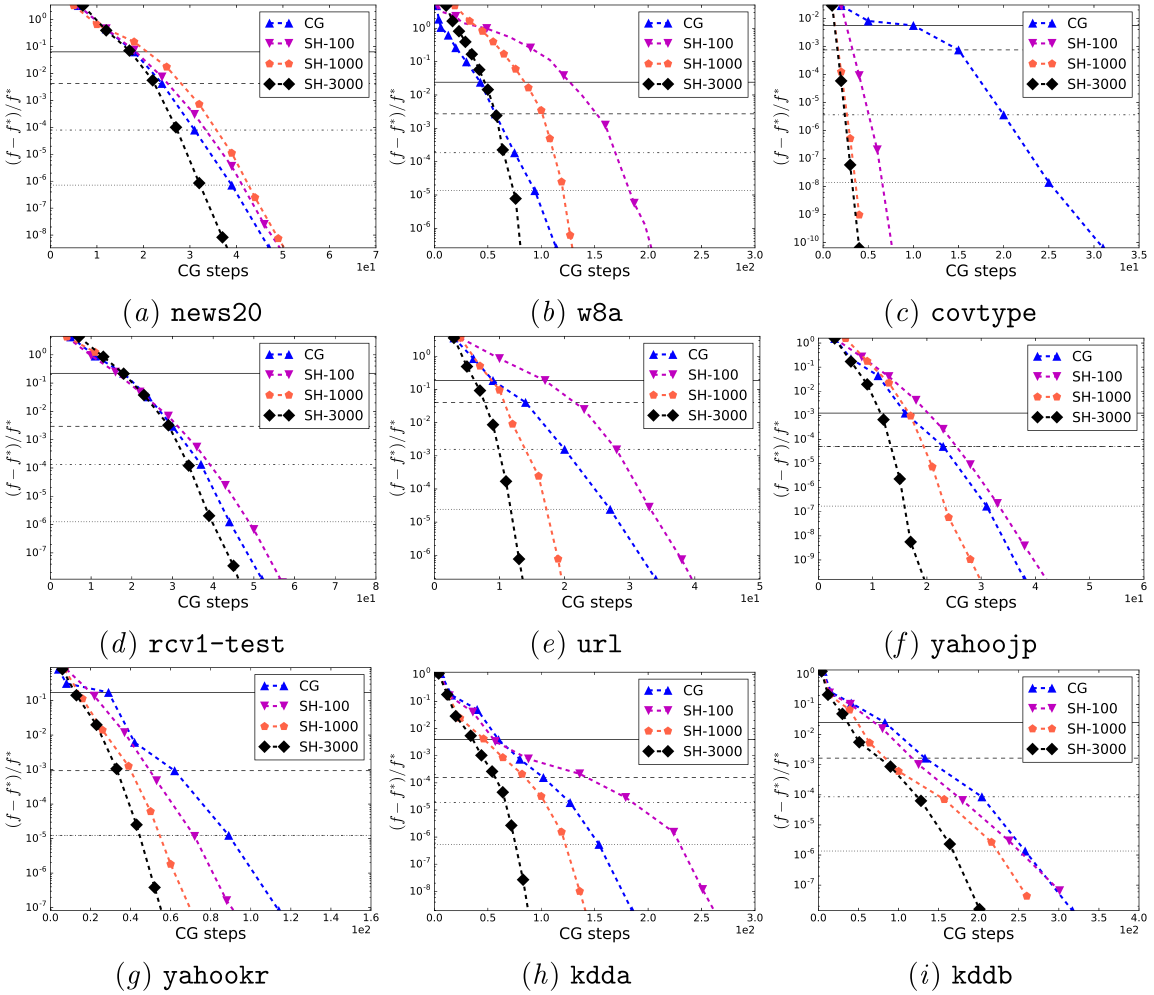
(g) yahookr

(h) kdda

(i) kddb

Figure viii: For each sub-figure, the upper part shows the convergence of the trust region Newton method without a preconditioner, with a diagonal preconditioner and by using techniques in Section 4.1. The lower one shows the convergence of using the preconditioner in (21) with different α values. Logistic loss is used. $C = 100C_{\text{Best}}$. Other settings are the same as those in Figure 1.

X.2. More Experimental Results in Section 5.2

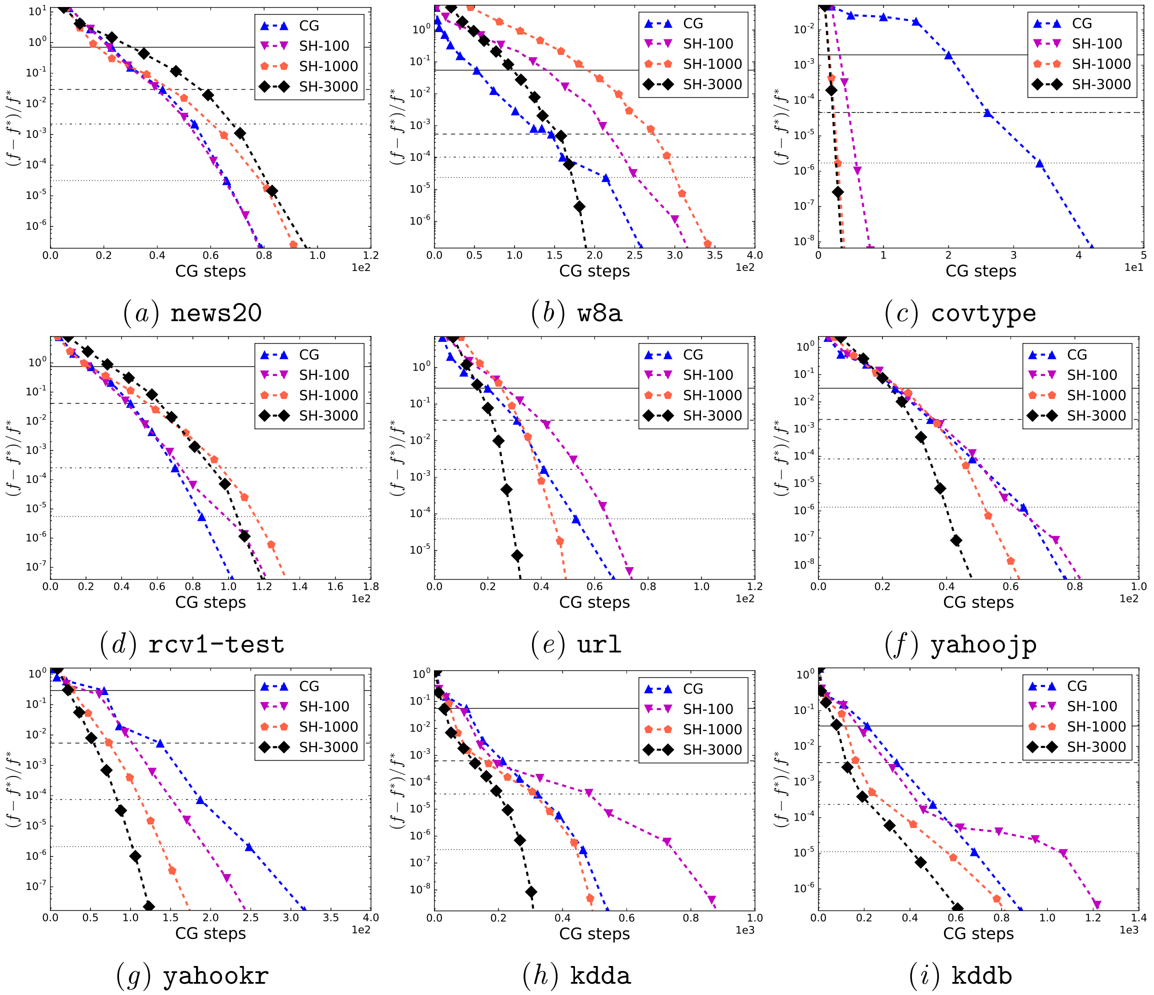


(a) news20 (b) w8a (c) covtype

(d) rcv1-test (e) url (f) yahoojp

(g) yahookr (h) kdda (i) kddb

Figure ix: Convergence of using different $\bar{l}$ values in the sub-sampled Hessian preconditioner. Other settings are the same as those in Figure 1. Loss=Logistic and $C = 0.01C_{\text{Best}}$.



(a) news20

(b) w8a

(c) covtype

(d) rcv1-test

(e) url

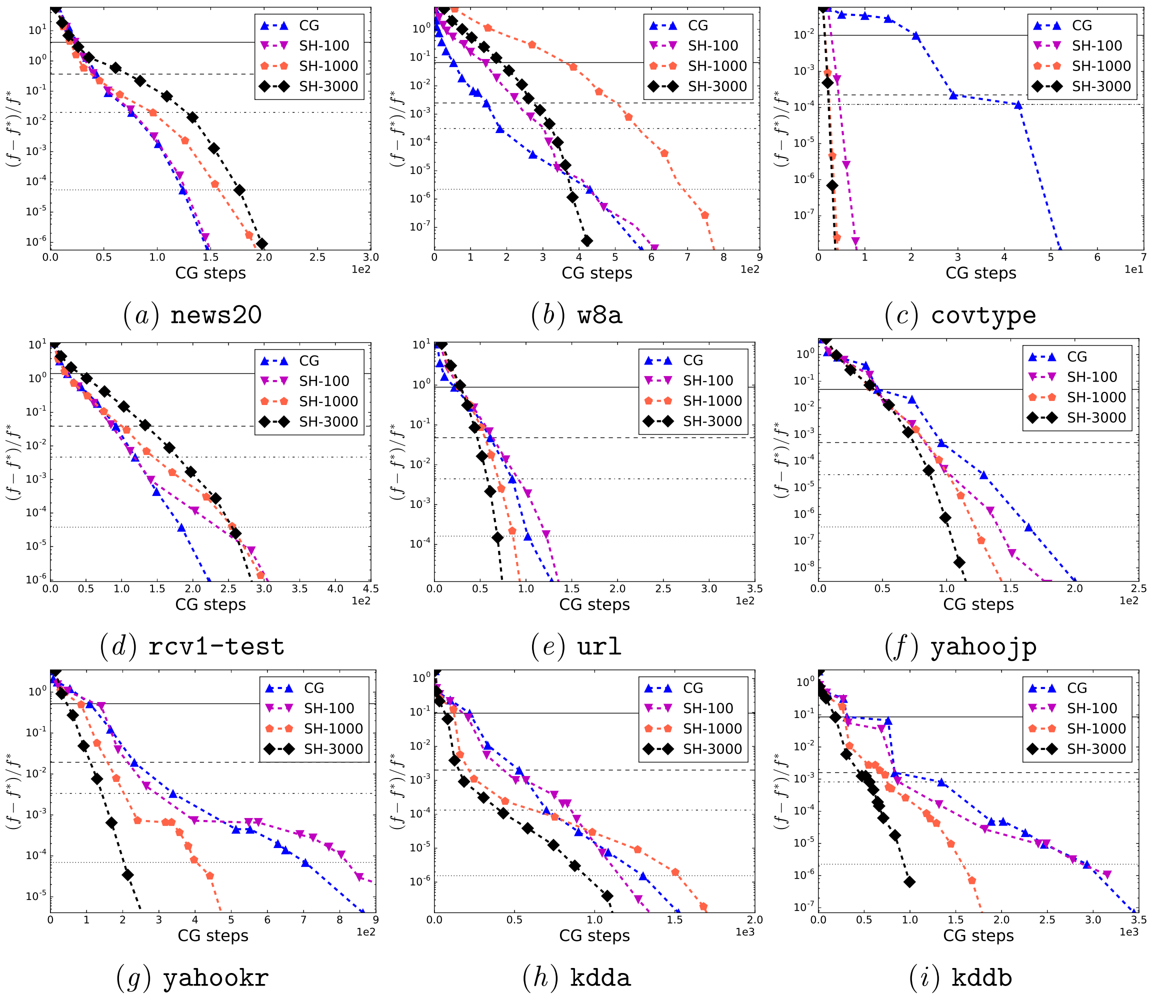
(f) yahoojp

(g) yahookr

(h) kdda

(i) kddb

Figure x: Convergence of using different $\bar{l}$ values in the sub-sampled Hessian preconditioner. Other settings are the same as those in Figure 1. Loss=Logistic and $C = 0.1C_{\text{Best}}$.



(a) news20

(b) w8a

(c) covtype

(d) rcv1-test

(e) url

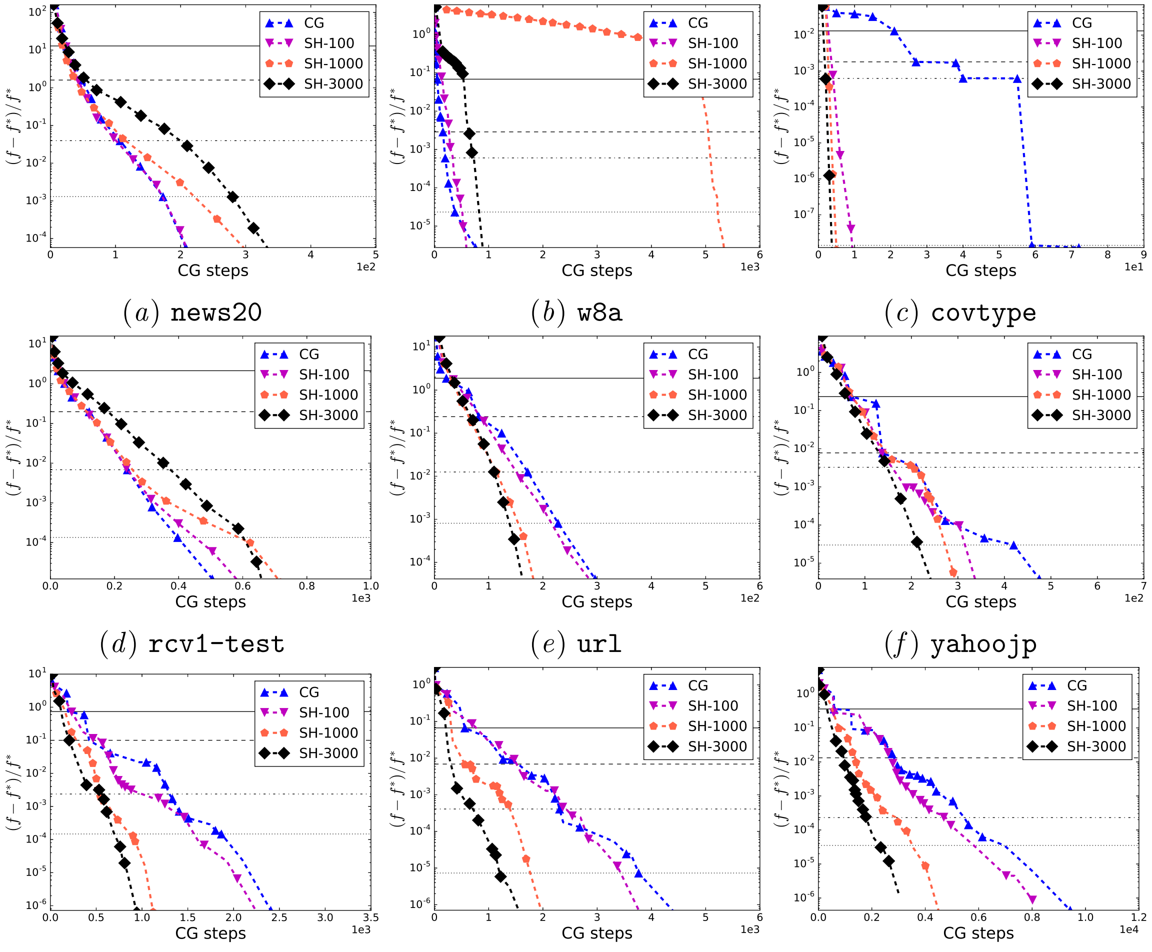
(f) yahoojp

(g) yahookr

(h) kdda

(i) kddb

Figure xi: Convergence of using different $\bar{l}$ values in the sub-sampled Hessian preconditioner. Other settings are the same as those in Figure 1. Loss=Logistic and $C = 1C_{\text{Best}}$.

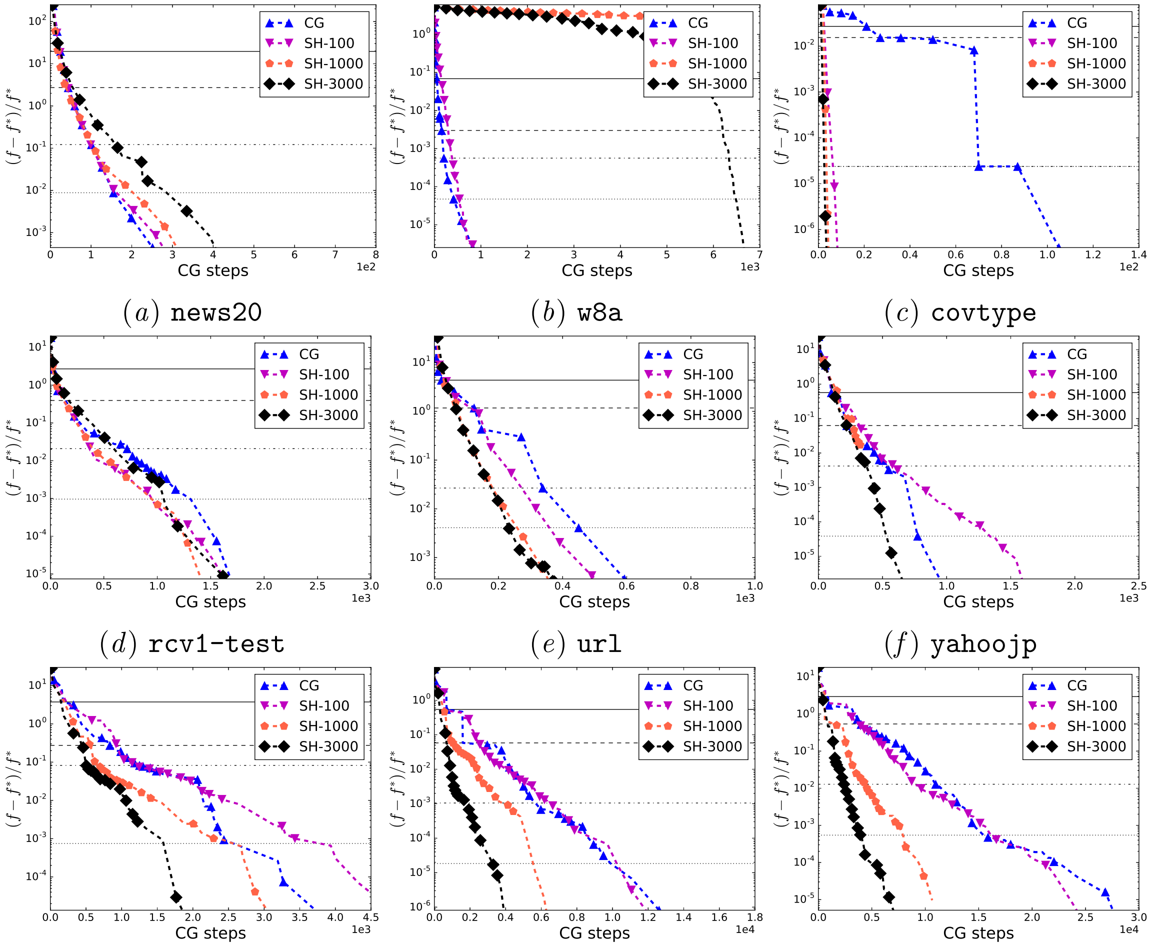


(a) news20 (b) w8a (c) covtype

(d) rcv1-test (e) url (f) yahoojp

(g) yahookr (h) kdda (i) kddb

Figure xii: Convergence of using different $\bar{l}$ values in the sub-sampled Hessian preconditioner. Other settings are the same as those in Figure 1. Loss=Logistic and $C = 10C_{\text{Best}}$.



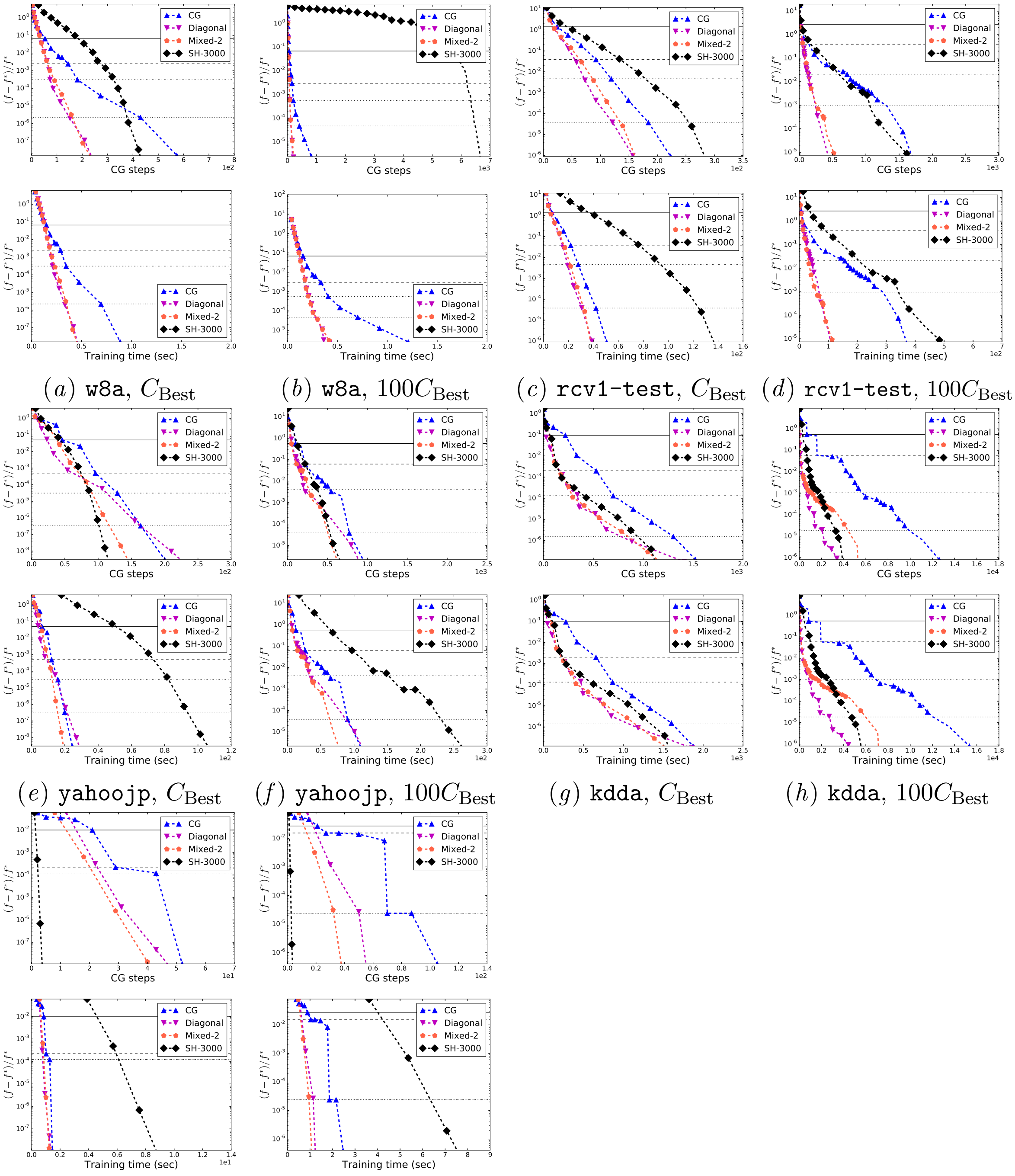
(a) news20 (b) w8a (c) covtype

(d) rcv1-test (e) url (f) yahoojp

(g) yahookr (h) kdda (i) kddb

Figure xiii: Convergence of using different $\bar{l}$ values in the sub-sampled Hessian preconditioner. Other settings are the same as those in Figure 1. Loss=Logistic and $C = 100C_{\text{Best}}$.

X.3. More Experimental Results in Section 5.3

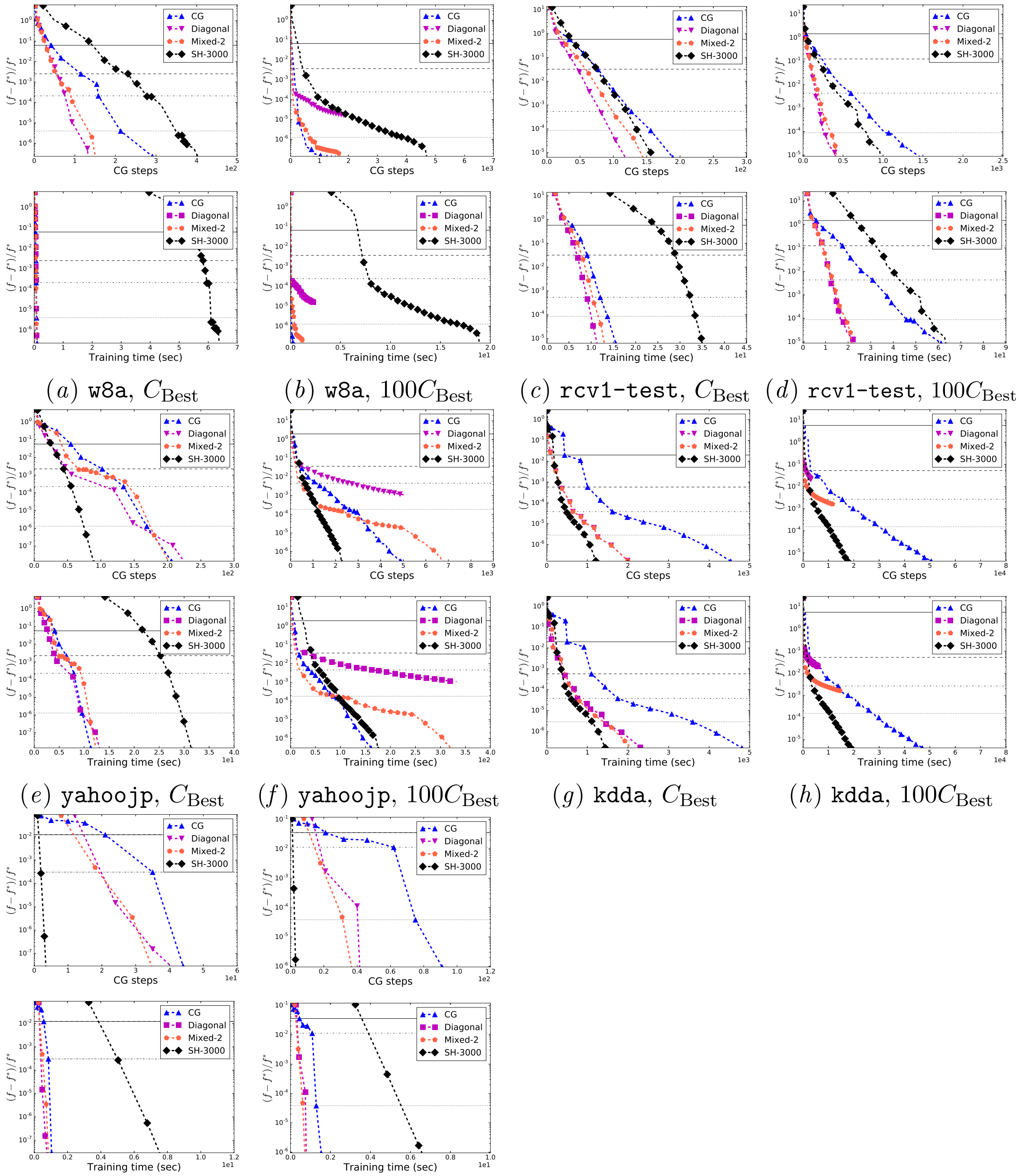


(a) `w8a`, C_{Best} (b) `w8a`, $100C_{\text{Best}}$ (c) `rcv1-test`, C_{Best} (d) `rcv1-test`, $100C_{\text{Best}}$

(e) `yahoojp`, C_{Best} (f) `yahoojp`, $100C_{\text{Best}}$ (g) `kdda`, C_{Best} (h) `kdda`, $100C_{\text{Best}}$

(i) `covtype`, C_{Best} (j) `covtype`, $100C_{\text{Best}}$

Figure xiv: Convergence of using different preconditioners. Logistic loss is considered. Other settings are the same as those in Figure 1.



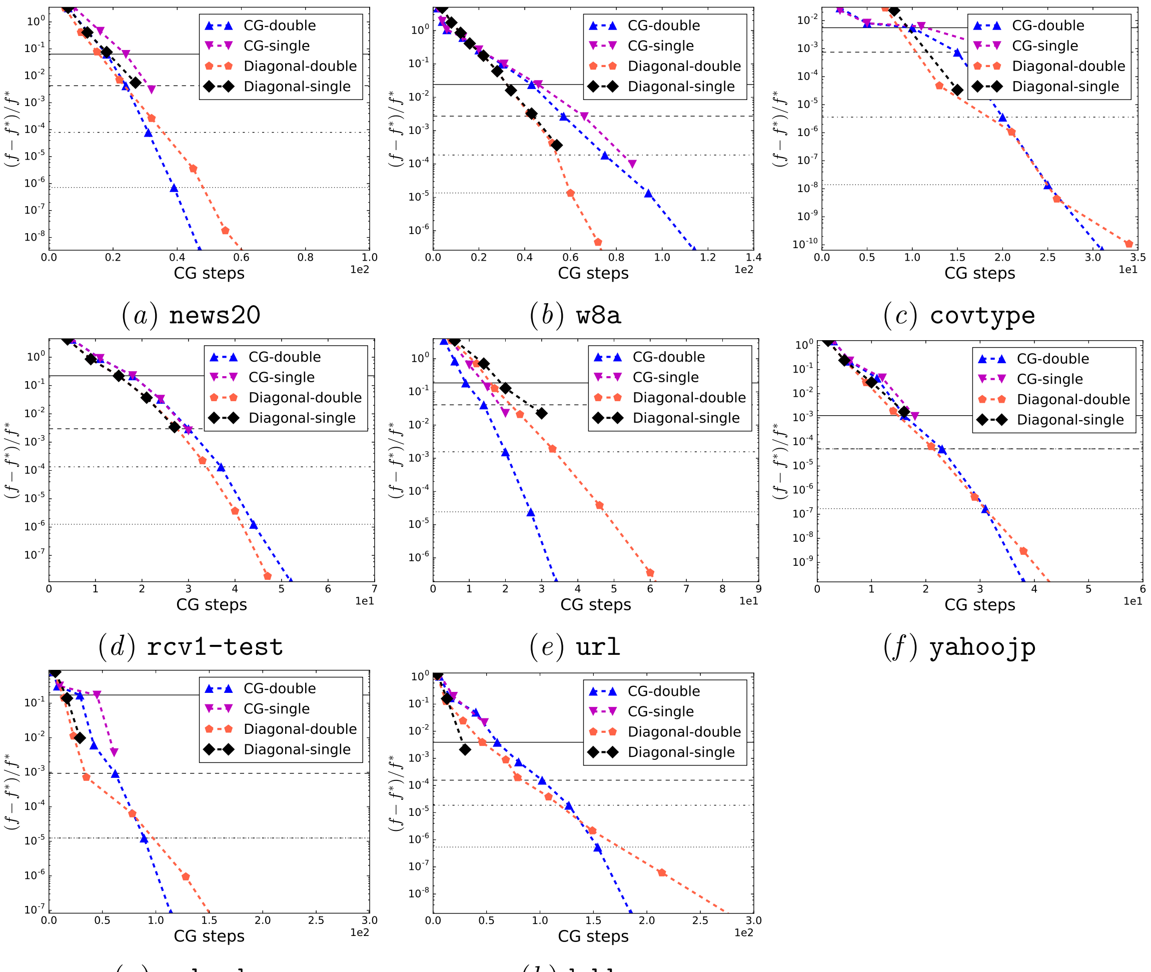
(a) `w8a`, C_{Best} (b) `w8a`, $100C_{\text{Best}}$ (c) `rcv1-test`, C_{Best} (d) `rcv1-test`, $100C_{\text{Best}}$

(e) `yahoojp`, C_{Best} (f) `yahoojp`, $100C_{\text{Best}}$ (g) `kdda`, C_{Best} (h) `kdda`, $100C_{\text{Best}}$

(i) `covtype`, C_{Best} (j) `covtype`, $100C_{\text{Best}}$

Figure xv: Convergence of using different preconditioners. L2 loss is considered. Other settings are the same as those in Figure 1.

X.4. Convergence of Using Single- or Double-precision Floating Points in a Trust Region Newton Method With/Without a Diagonal Preconditioner.



(a) news20

(b) w8a

(c) covtype

(d) rcv1-test

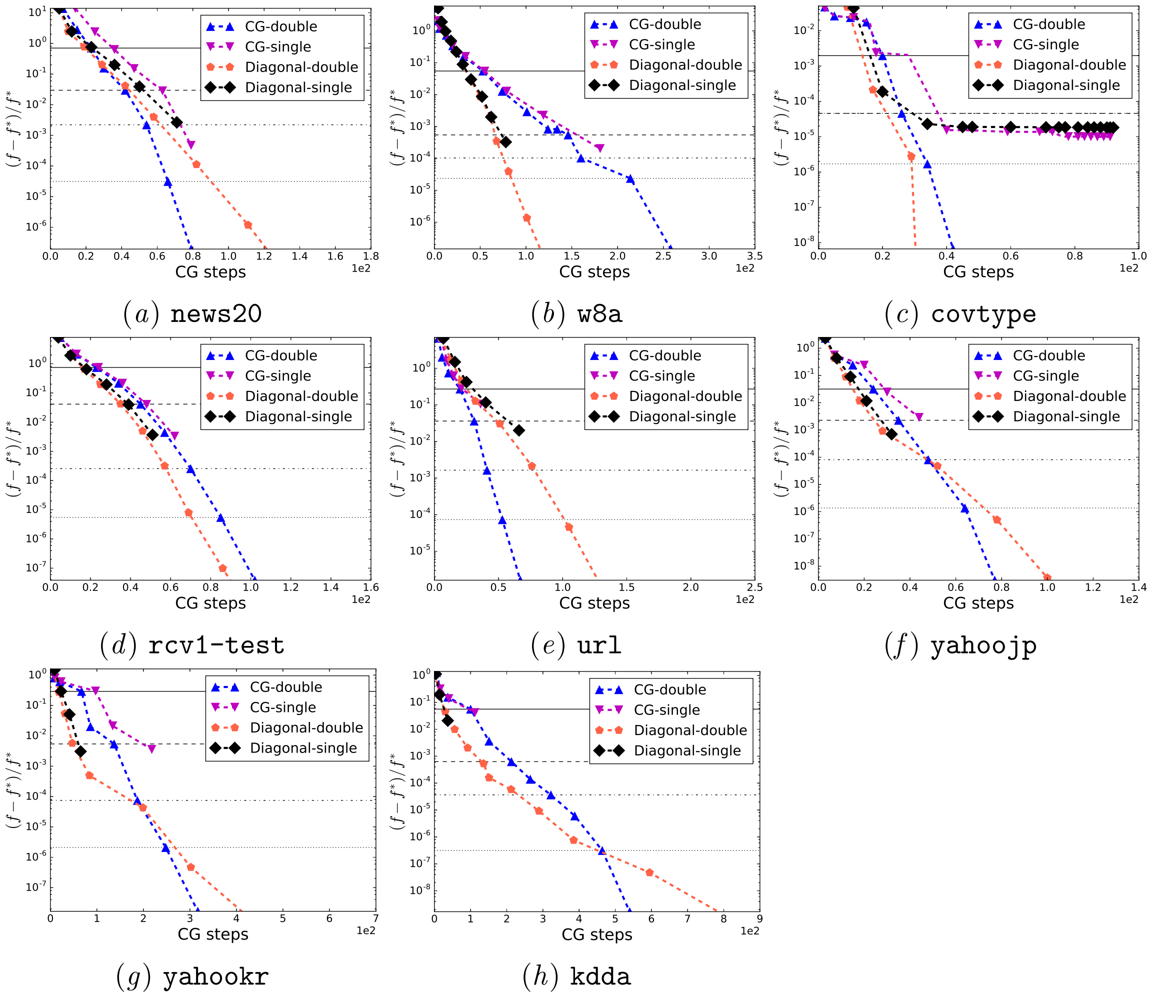
(e) url

(f) yahoojp

(g) yahookr

(h) kdda

Figure xvi: Convergence of using single or double precision floating points in a trust region Newton method without a preconditioner and with a diagonal preconditioner. We show a relative difference to the optimal function value (log scaled) versus the cumulative number of CG iterations. Note that for some data sets, the single precision version failed to continue at some point. Loss=Logistic and $C = 0.01C_{\text{Best}}$.



(a) `news20`

(b) `w8a`

(c) `covtype`

(d) `rcv1-test`

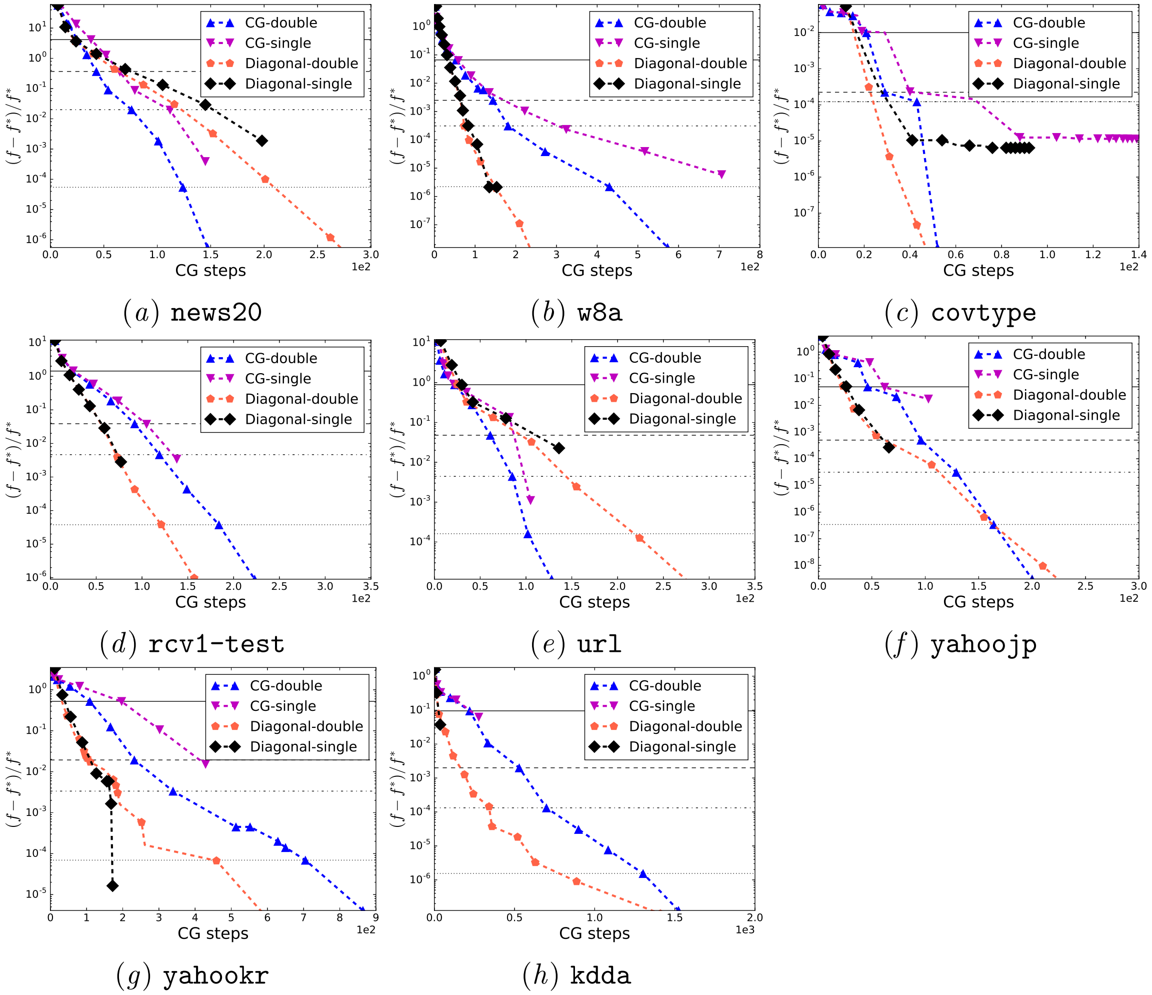
(e) `url`

(f) `yahoojp`

(g) `yahookr`

(h) `kdda`

Figure xvii: Convergence of using single or double precision floating points in a trust region Newton method without a preconditioner and with a diagonal preconditioner. We show a relative difference to the optimal function value (log scaled) versus the cumulative number of CG iterations. Note that for some data sets, the single precision version failed to continue at some point. Loss=Logistic and $C = 0.1C_{\text{Best}}$.

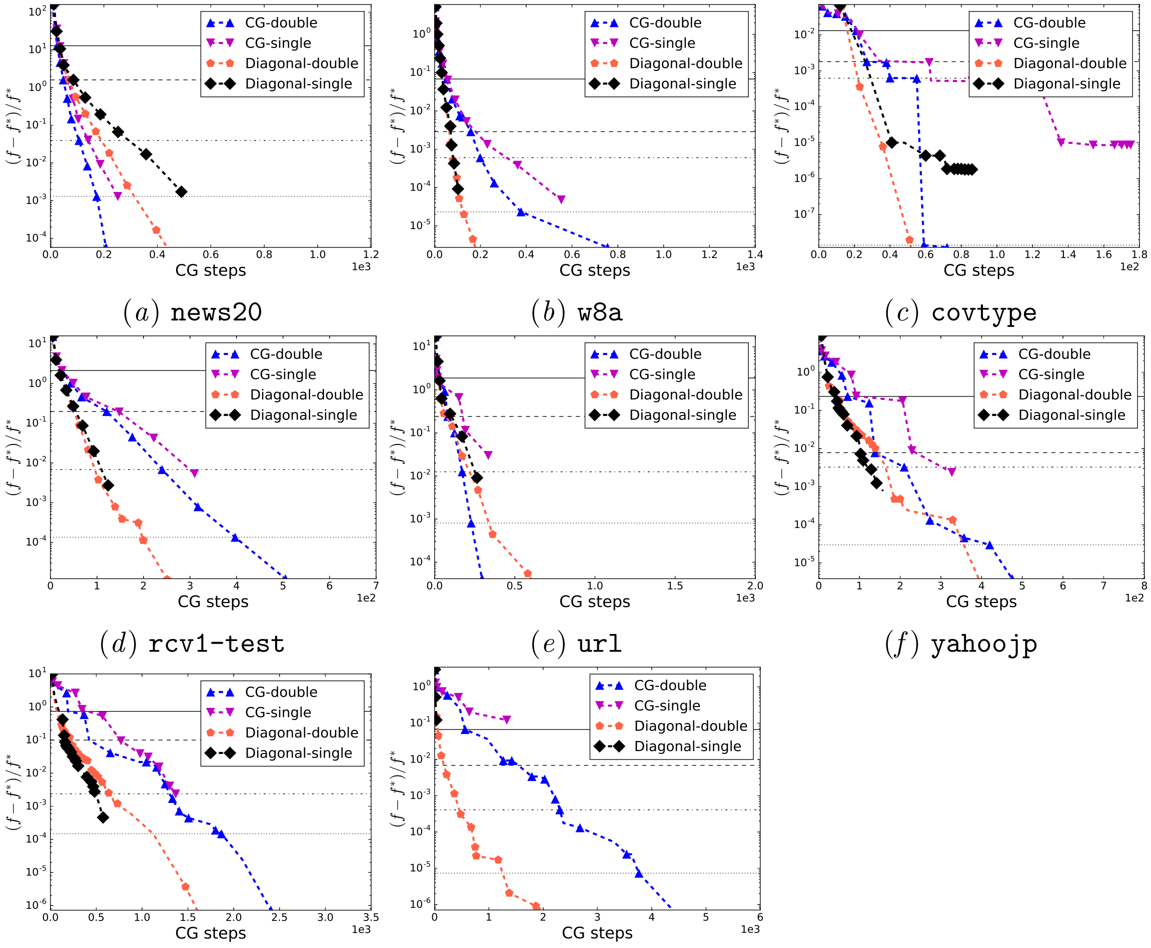


(*a*) `news20` (*b*) `w8a` (*c*) `covtype`

(*d*) `rcv1-test` (*e*) `url` (*f*) `yahoojp`

(*g*) `yahookr` (*h*) `kdda`

Figure xviii: Convergence of using single or double precision floating points in a trust region Newton method without a preconditioner and with a diagonal preconditioner. We show a relative difference to the optimal function value (log scaled) versus the cumulative number of CG iterations. Note that for some data sets, the single precision version failed to continue at some point. Loss=Logistic and $C = 1C_{\text{Best}}$.



(a) news20

(b) w8a

(c) covtype

(d) rcv1-test

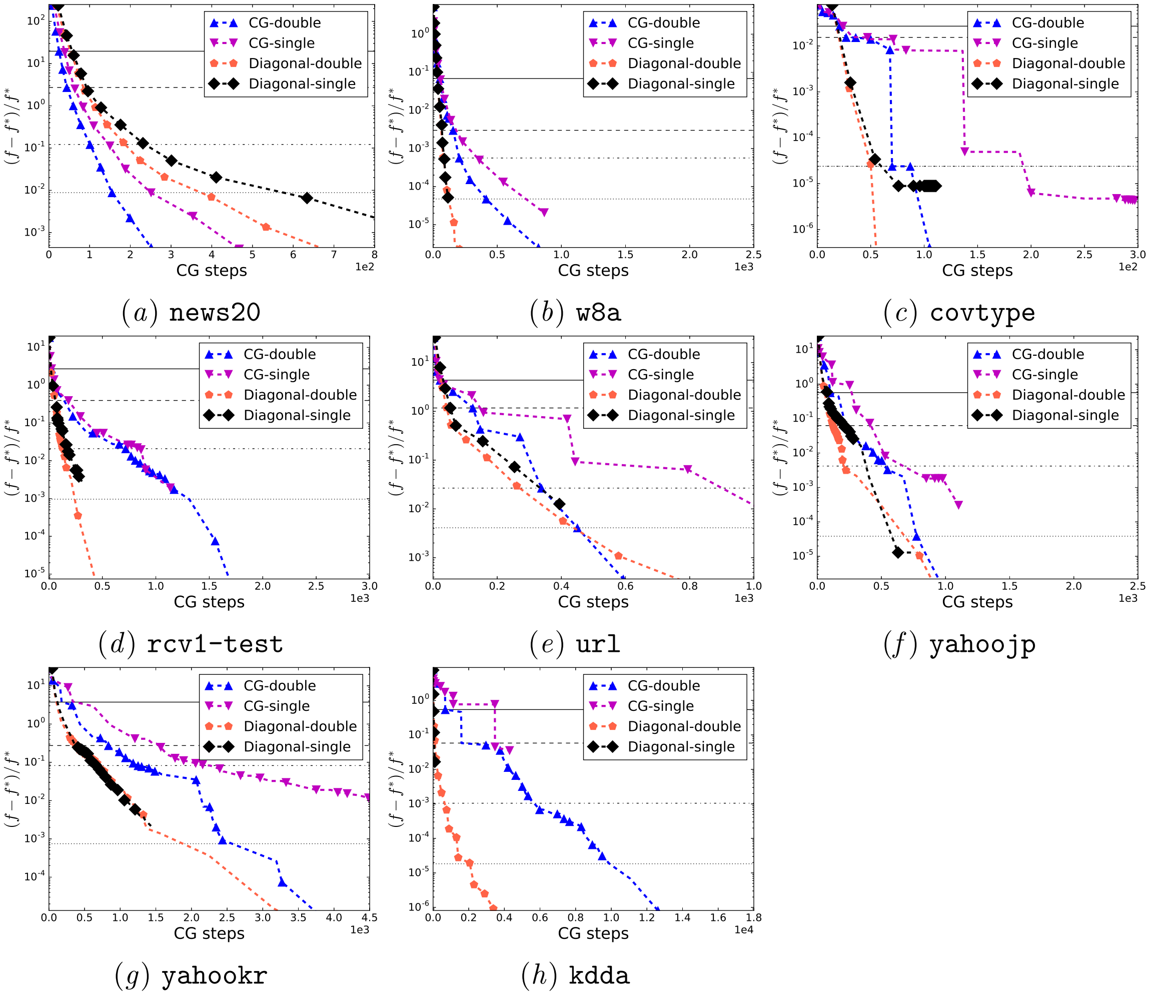
(e) url

(f) yahoojp

(g) yahookr

(h) kdda

Figure xix: Convergence of using single or double precision floating points in a trust region Newton method without a preconditioner and with a diagonal preconditioner. We show a relative difference to the optimal function value (log scaled) versus the cumulative number of CG iterations. Note that for some data sets, the single precision version failed to continue at some point. Loss=Logistic and $C = 10C_{\text{Best}}$.



(a) news20

(b) w8a

(c) covtype

(d) rcv1-test

(e) url

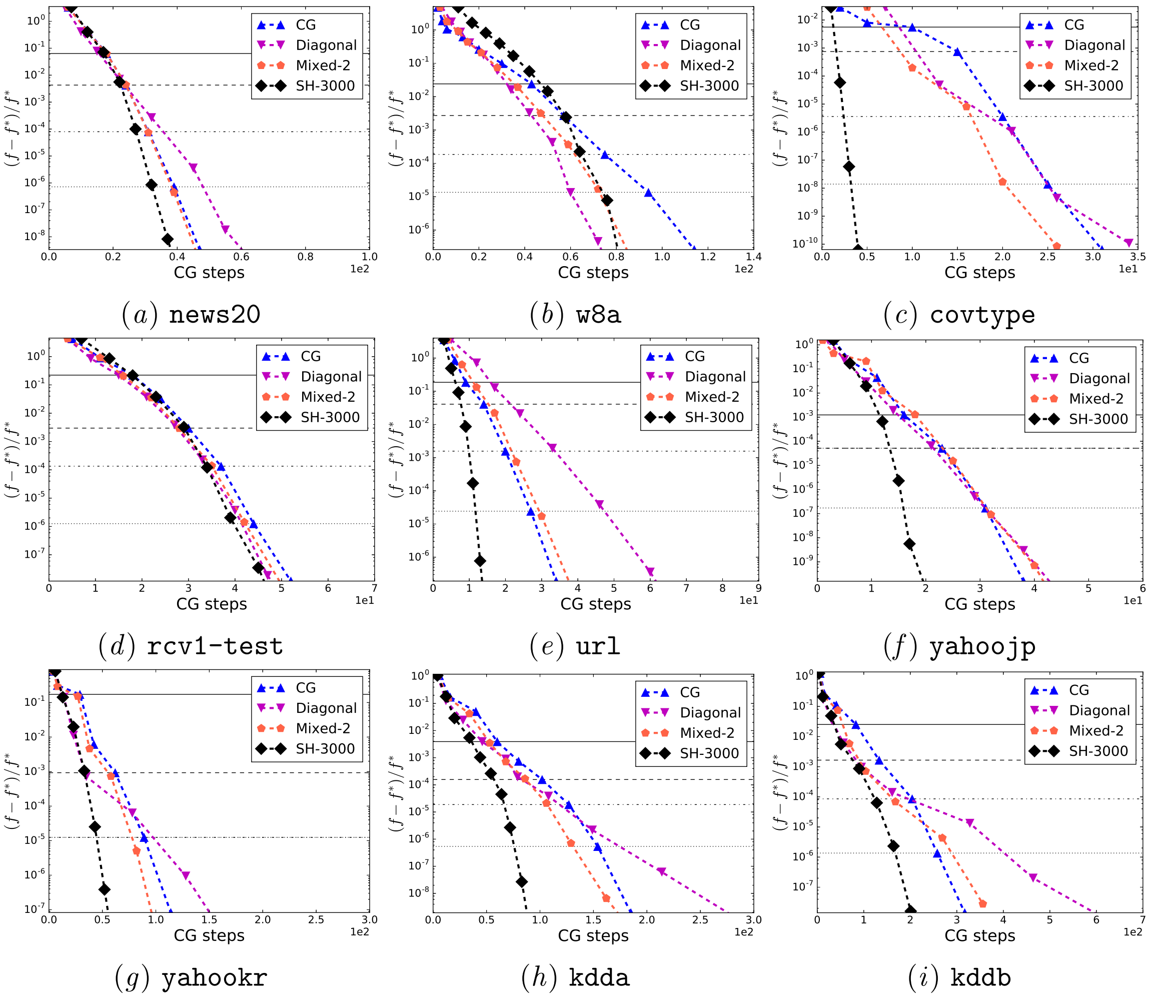
(f) yahoojp

(g) yahookr

(h) kdda

Figure xx: Convergence of using single or double precision floating points in a trust region Newton method without a preconditioner and with a diagonal preconditioner. We show a relative difference to the optimal function value (log scaled) versus the cumulative number of CG iterations. Note that for some data sets, the single precision version failed to continue at some point. Loss=Logistic and $C = 100C_{\text{Best}}$.

X.5. Detailed Results of Using Different C Values (Logistic loss)



(a) news20

(b) w8a

(c) covtype

(d) rcv1-test

(e) url

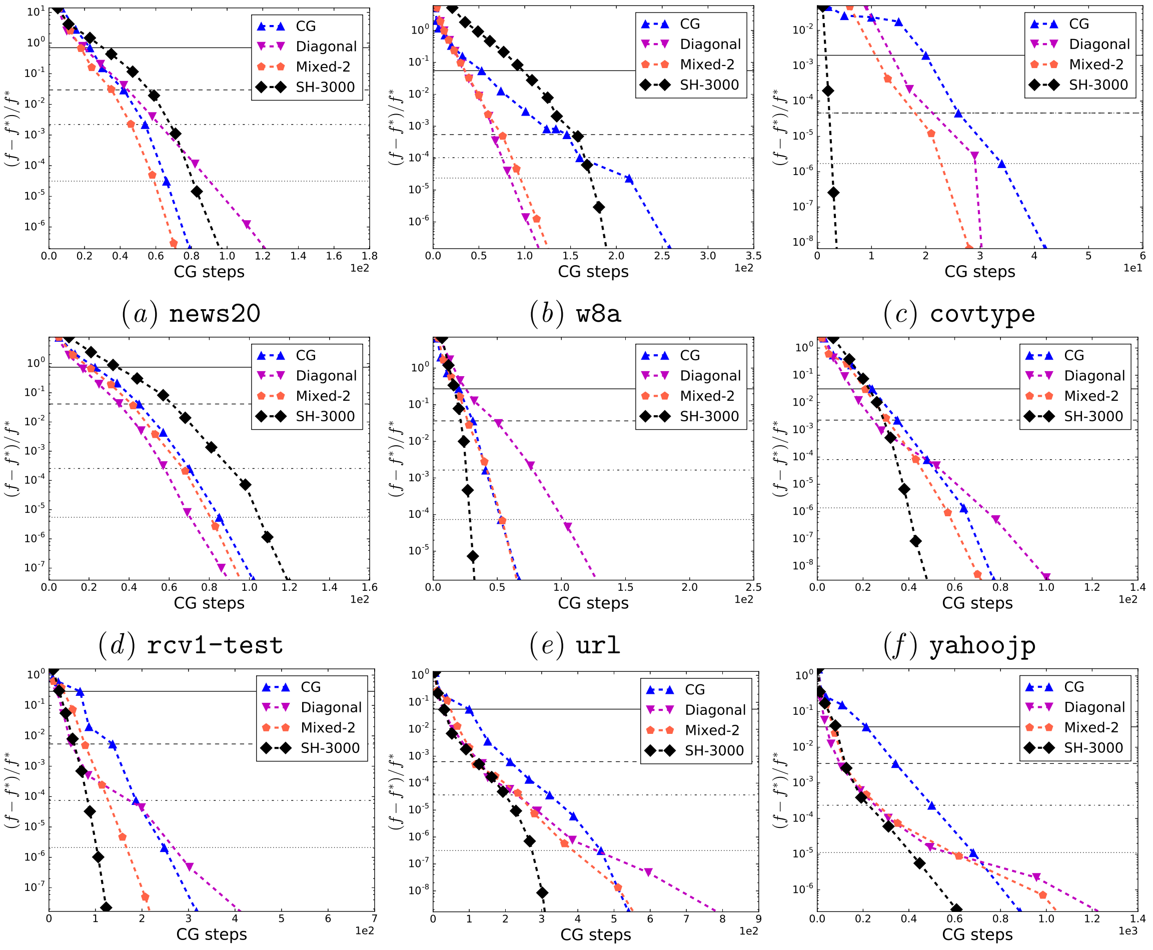
(f) yahoojp

(g) yahookr

(h) kdda

(i) kddb

Figure xxi: Convergence of using different preconditioners. The x-axis of the figures are the cumulative number of CG iterations. Other settings are the same as those in Figure 1. Loss=Logistic and $C = 0.01C_{\text{Best}}$.

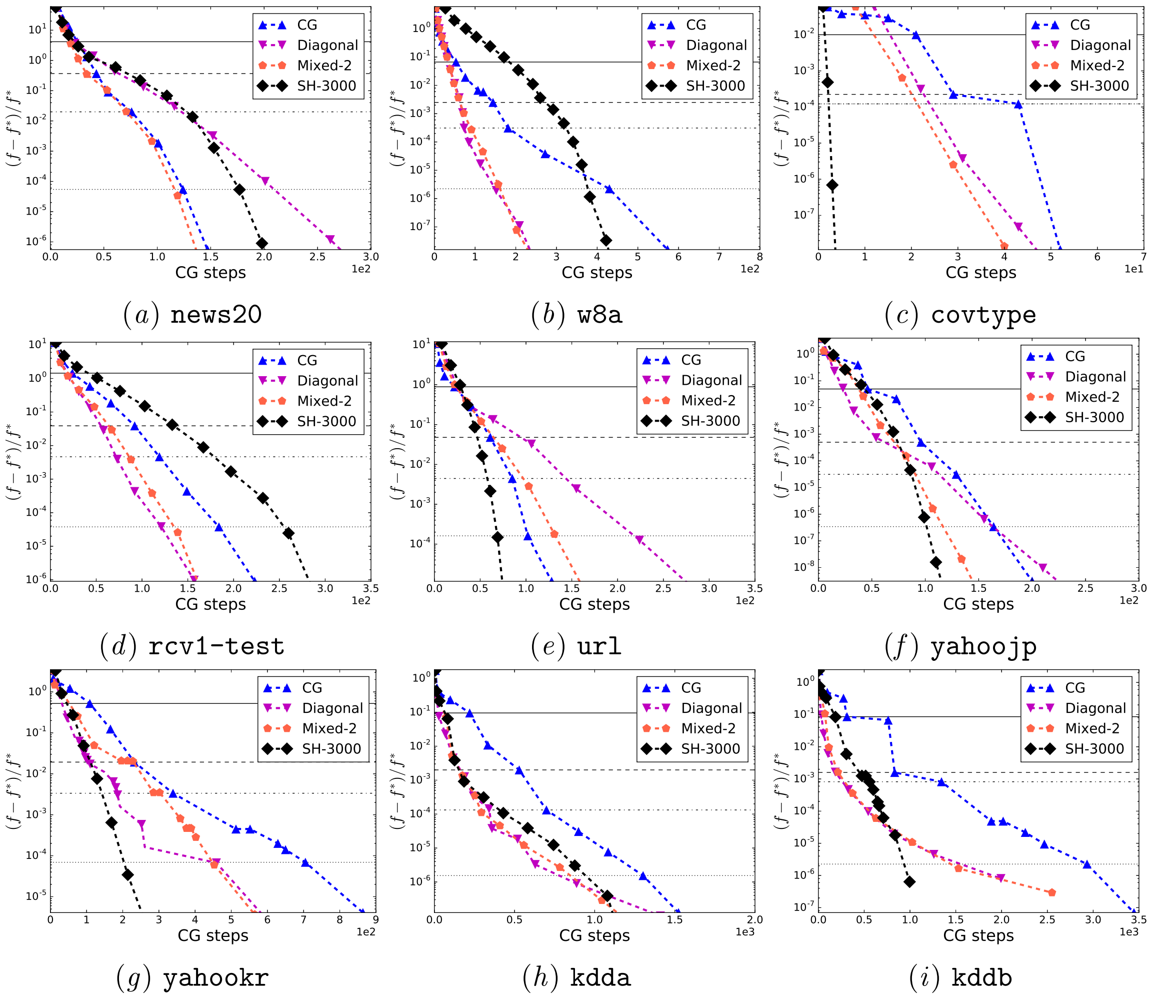


(a) `news20` (b) `w8a` (c) `covtype`

(d) `rcv1-test` (e) `url` (f) `yahoojp`

(g) `yahookr` (h) `kdda` (i) `kddb`

Figure xxii: Convergence of using different preconditioners. The x-axis of the figures are the cumulative number of CG iterations. Other settings are the same as those in Figure 1. Loss=Logistic and $C = 0.1C_{\text{Best}}$.

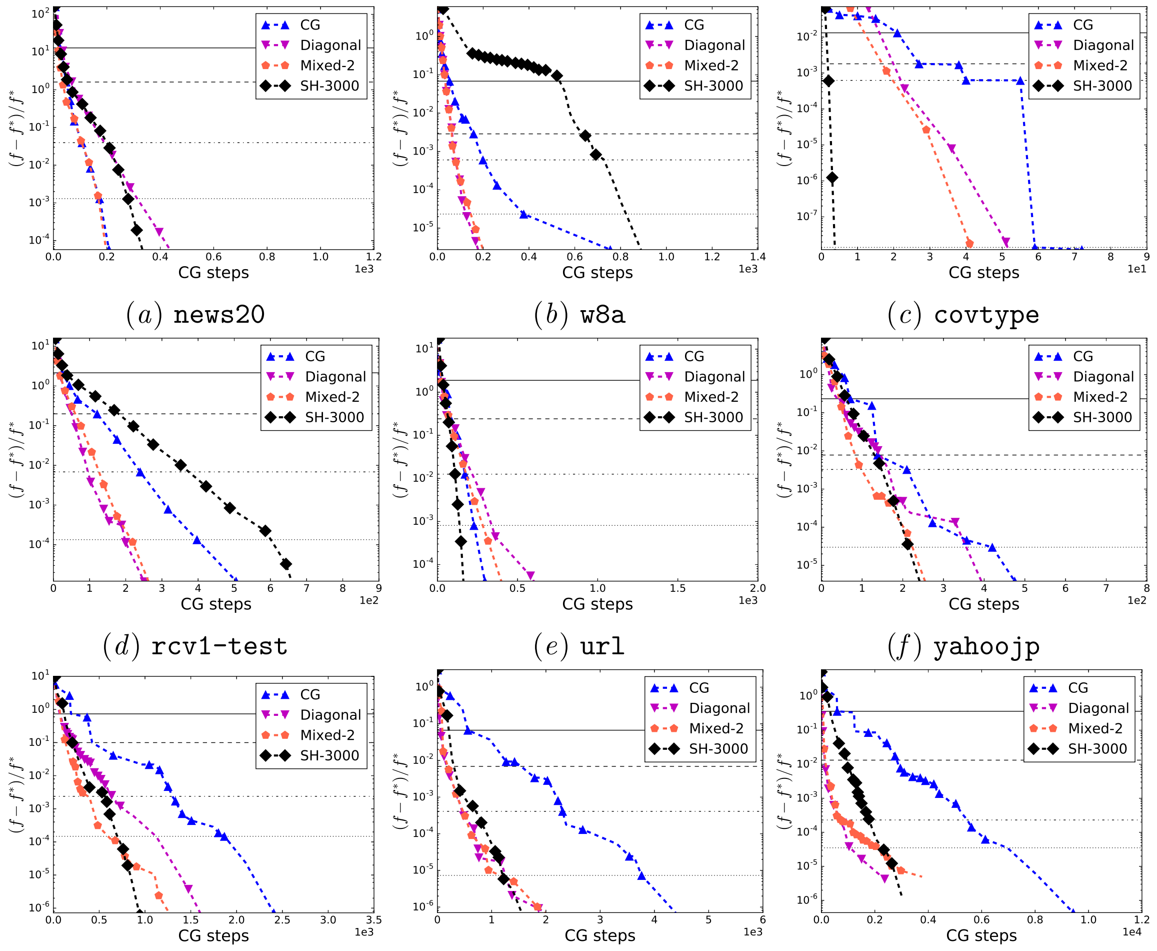


(a) news20 (b) w8a (c) covtype

(d) rcv1-test (e) url (f) yahoojp

(g) yahookr (h) kdda (i) kddb

Figure xxiii: Convergence of using different preconditioners. The x-axis of the figures are the cumulative number of CG iterations. Other settings are the same as those in Figure 1. Loss=Logistic and $C = 1C_{\text{Best}}$.

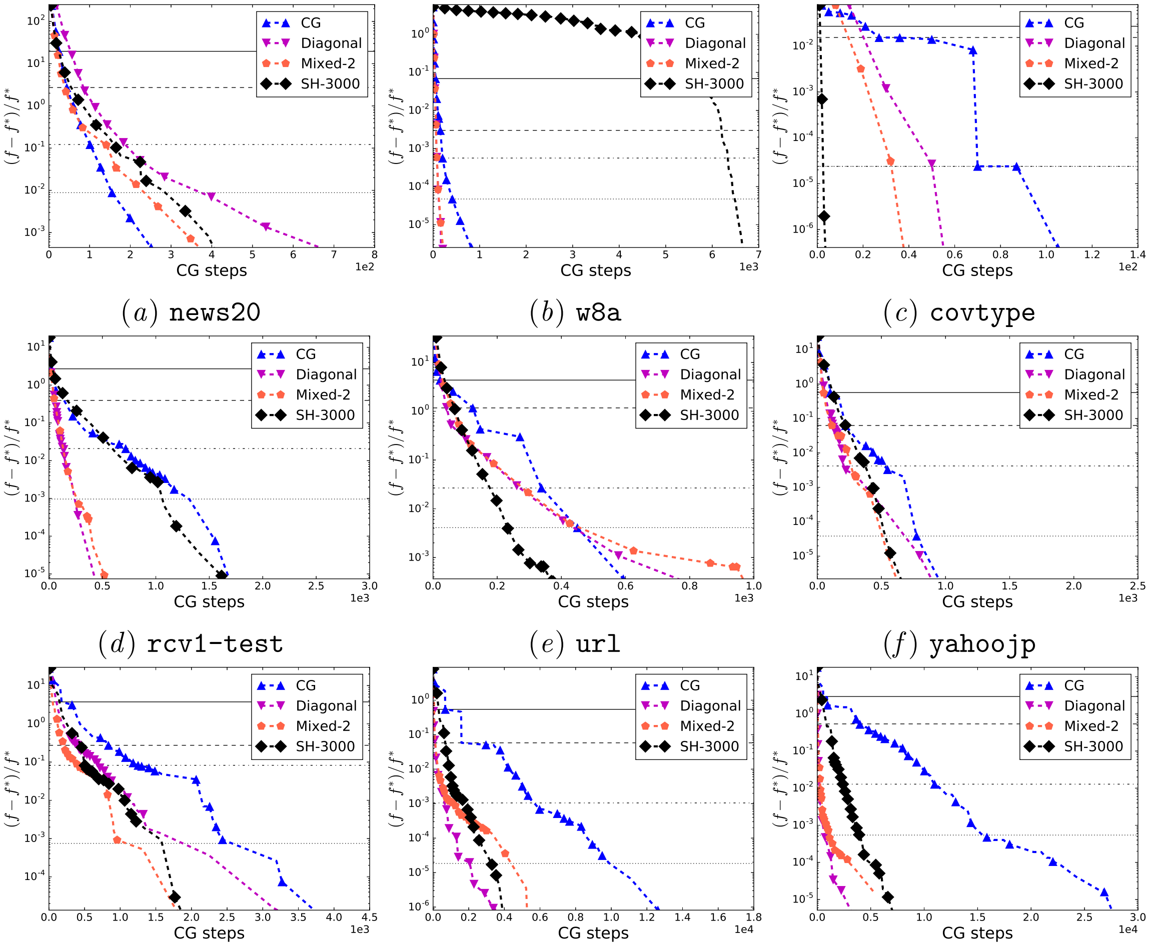


(a) news20 (b) w8a (c) covtype

(d) rcv1-test (e) url (f) yahoojp

(g) yahookr (h) kdda (i) kddb

Figure xxiv: Convergence of using different preconditioners. The x-axis of the figures are the cumulative number of CG iterations. Other settings are the same as those in Figure 1. Loss=Logistic and $C = 10C_{\text{Best}}$.



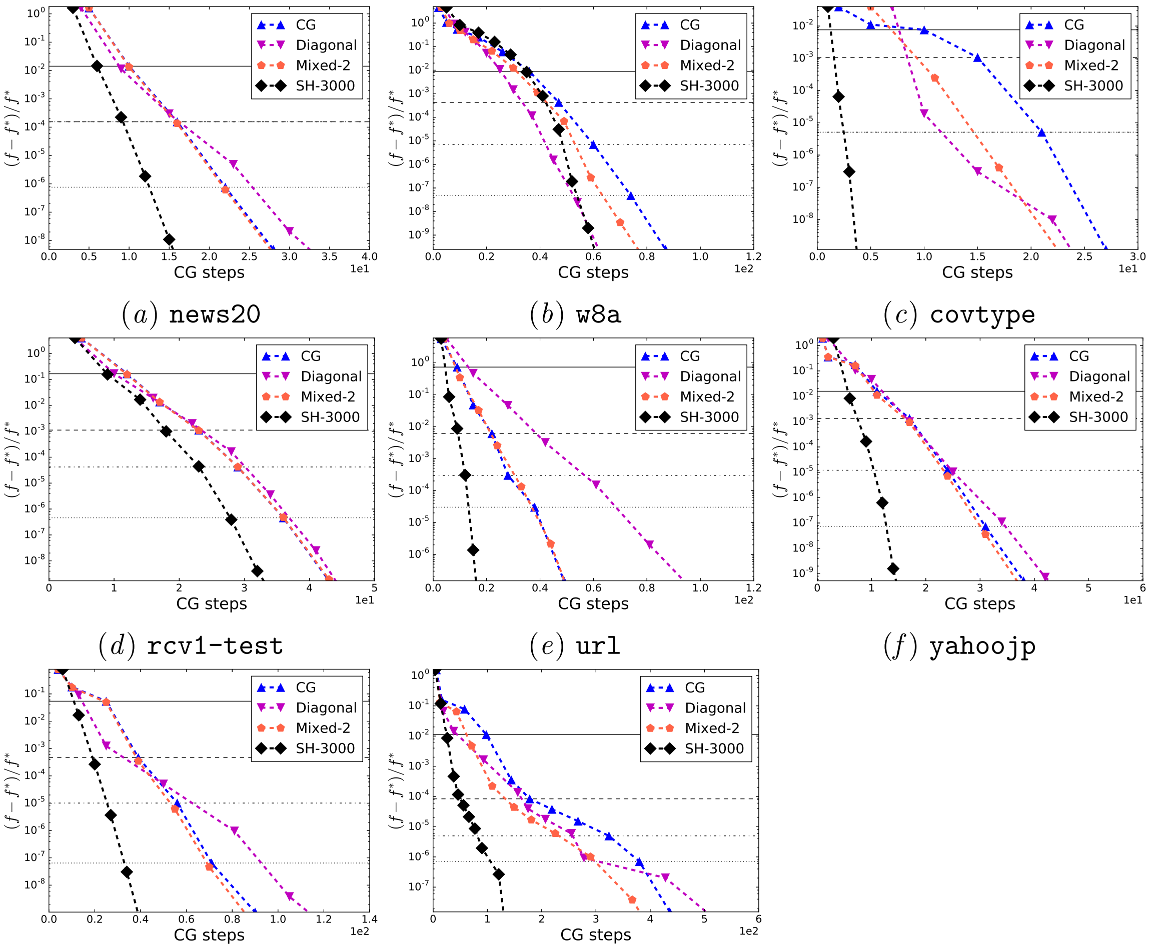
(*a*) `news20` (*b*) `w8a` (*c*) `covtype`

(*d*) `rcv1-test` (*e*) `url` (*f*) `yahoojp`

(*g*) `yahookr` (*h*) `kdda` (*i*) `kddb`

Figure xxv: Convergence of using different preconditioners. The x-axis of the figures are the cumulative number of CG iterations. Other settings are the same as those in Figure 1. Loss=Logistic and $C = 100C_{\text{Best}}$.

X.6. Detailed Results of Using Different C Values (L2 loss)



(a) `news20`

(b) `w8a`

(c) `covtype`

(d) `rcv1-test`

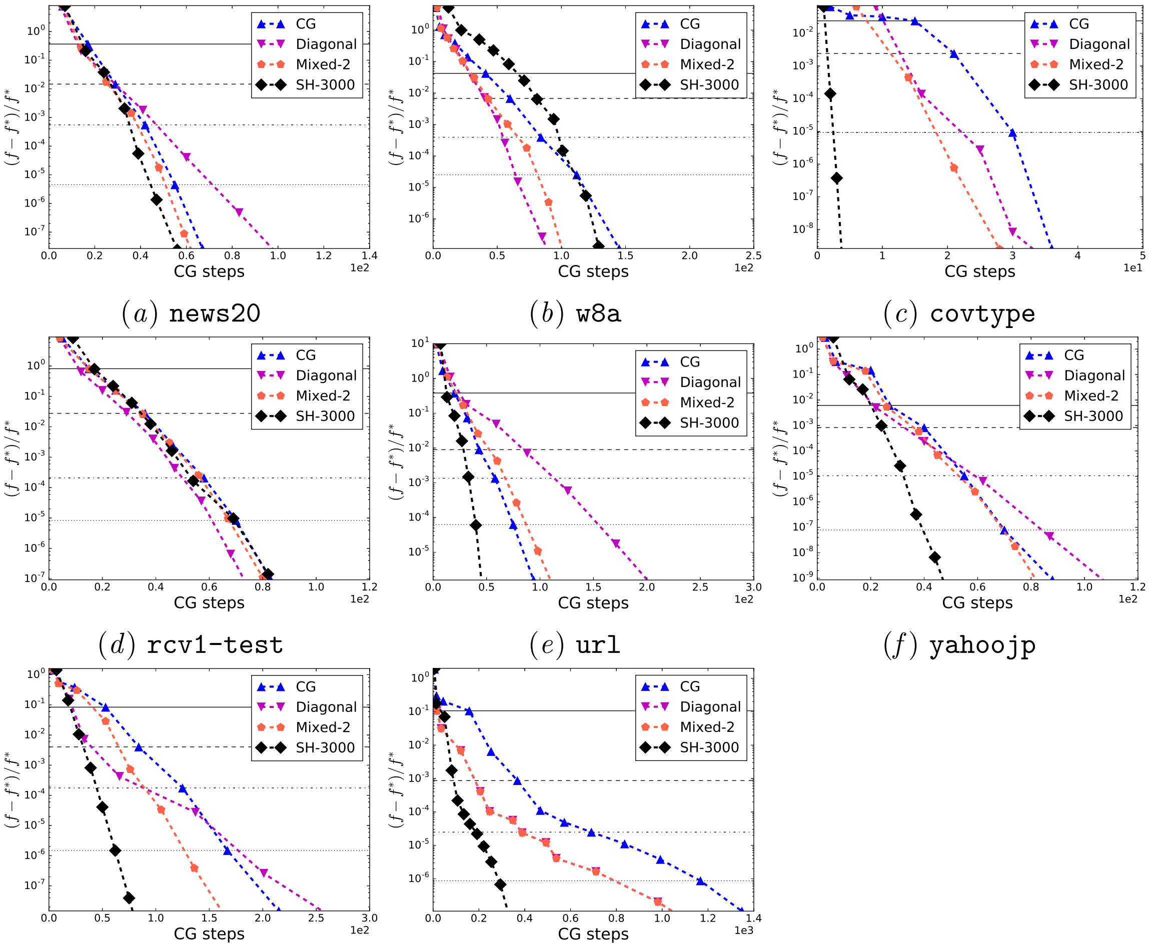
(e) `url`

(f) `yahoojp`

(g) `yahookr`

(h) `kdda`

Figure xxvi: Convergence of using different preconditioners. The x-axis of the figures are the cumulative number of CG iterations. Other settings are the same as those in Figure 1. Loss=L2 and $C = 0.01C_{\text{Best}}$.



(a) news20

(b) w8a

(c) covtype

(d) rcv1-test

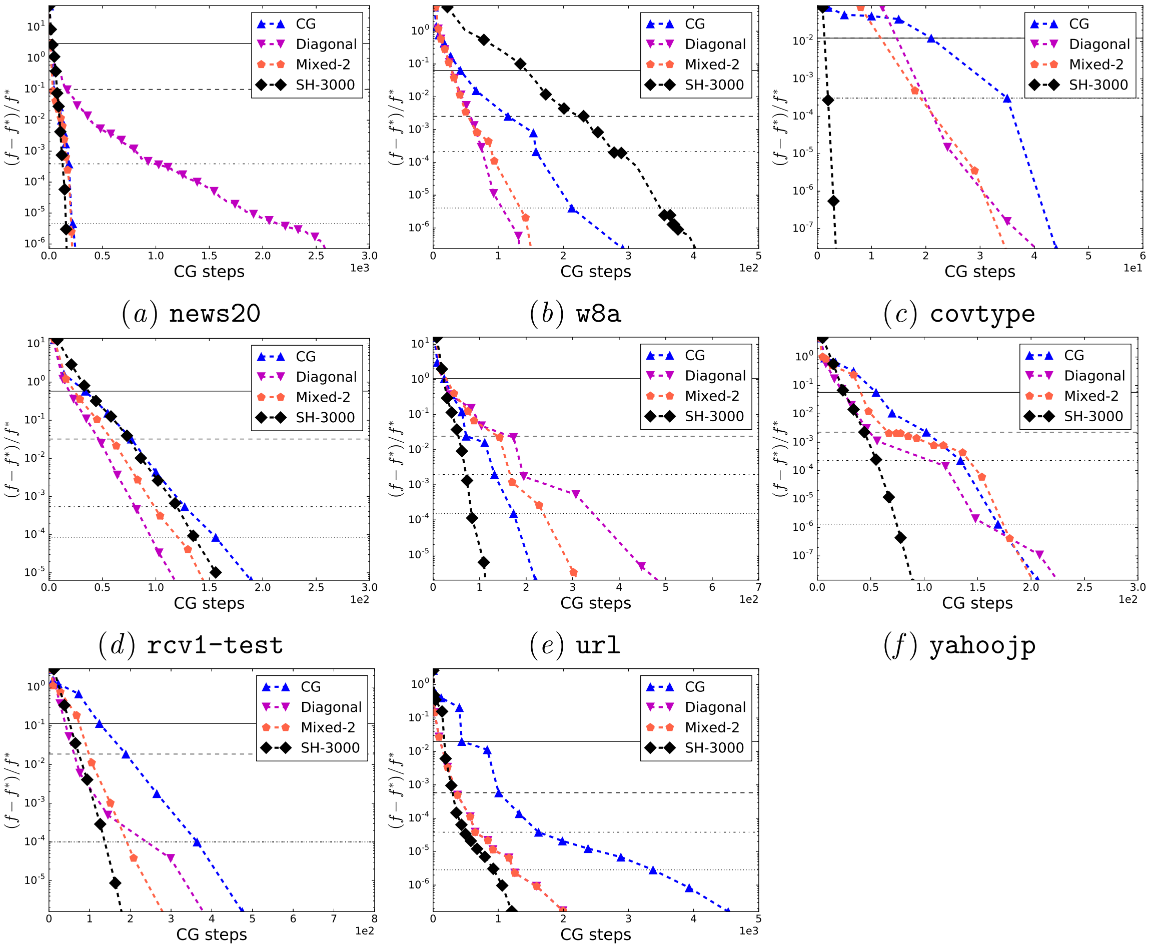
(e) url

(f) yahoojp

(g) yahookr

(h) kdda

Figure xxvii: Convergence of using different preconditioners. The x-axis of the figures are the cumulative number of CG iterations. Other settings are the same as those in Figure 1. Loss=L2 and $C = 0.1C_{\text{Best}}$.



(a) news20

(b) w8a

(c) covtype

(d) rcv1-test

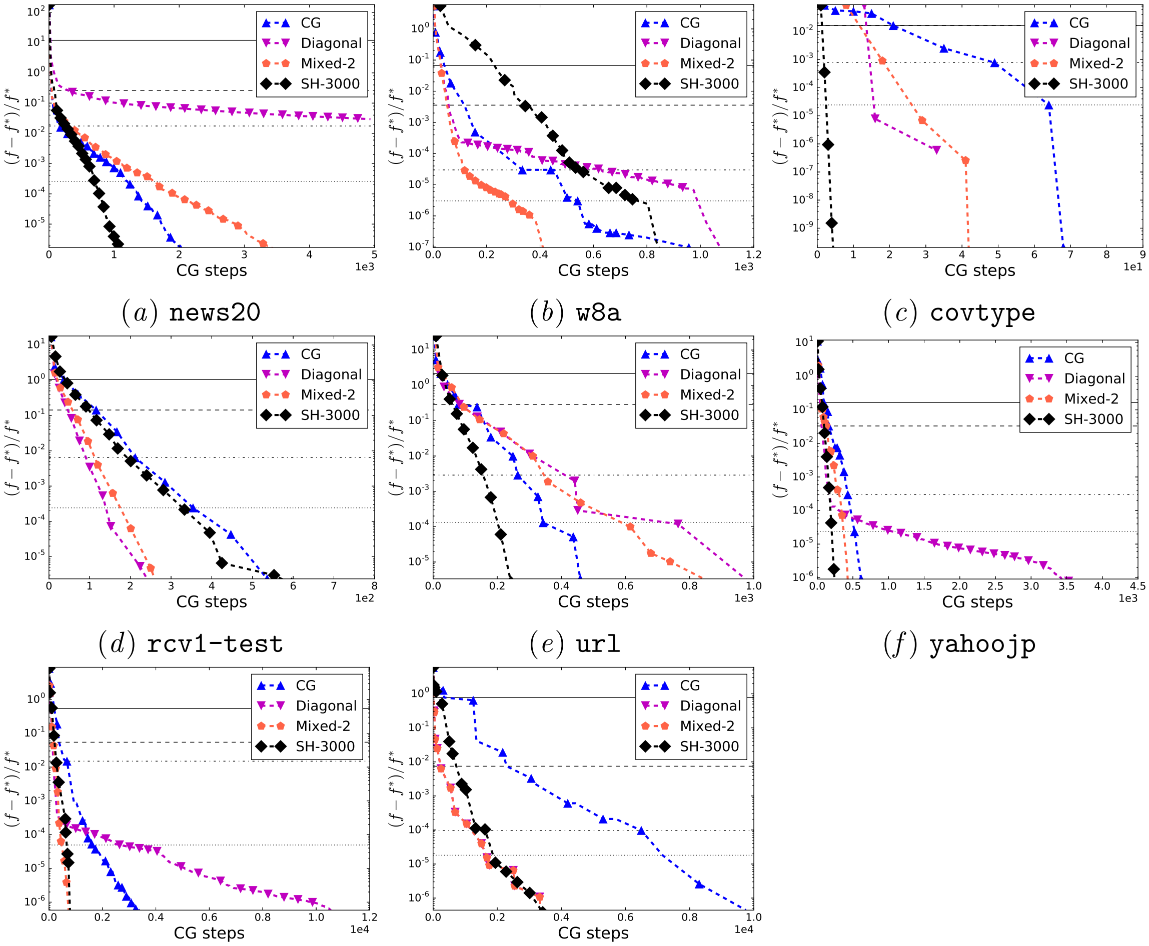
(e) url

(f) yahoojp

(g) yahookr

(h) kdda

Figure xxviii: Convergence of using different preconditioners. The x-axis of the figures are the cumulative number of CG iterations. Other settings are the same as those in Figure 1. Loss=L2 and $C = 1C_{\text{Best}}$.

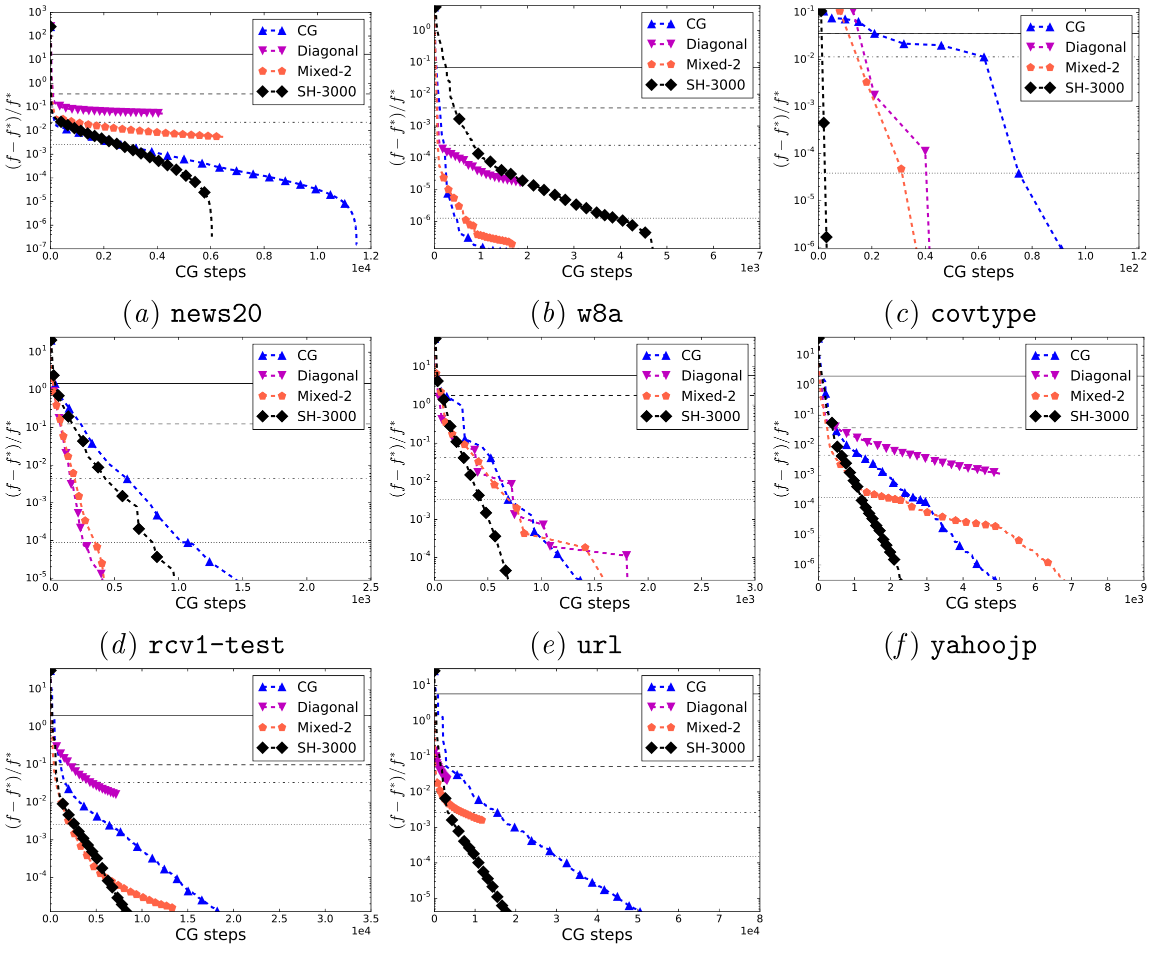


(a) news20 (b) w8a (c) covtype

(d) rcv1-test (e) url (f) yahoojp

(g) yahookr (h) kdda

Figure xxix: Convergence of using different preconditioners. The x-axis of the figures are the cumulative number of CG iterations. Other settings are the same as those in Figure 1. Loss=L2 and $C = 10C_{\text{Best}}$.



(a) news20

(b) w8a

(c) covtype

(d) rcv1-test

(e) url

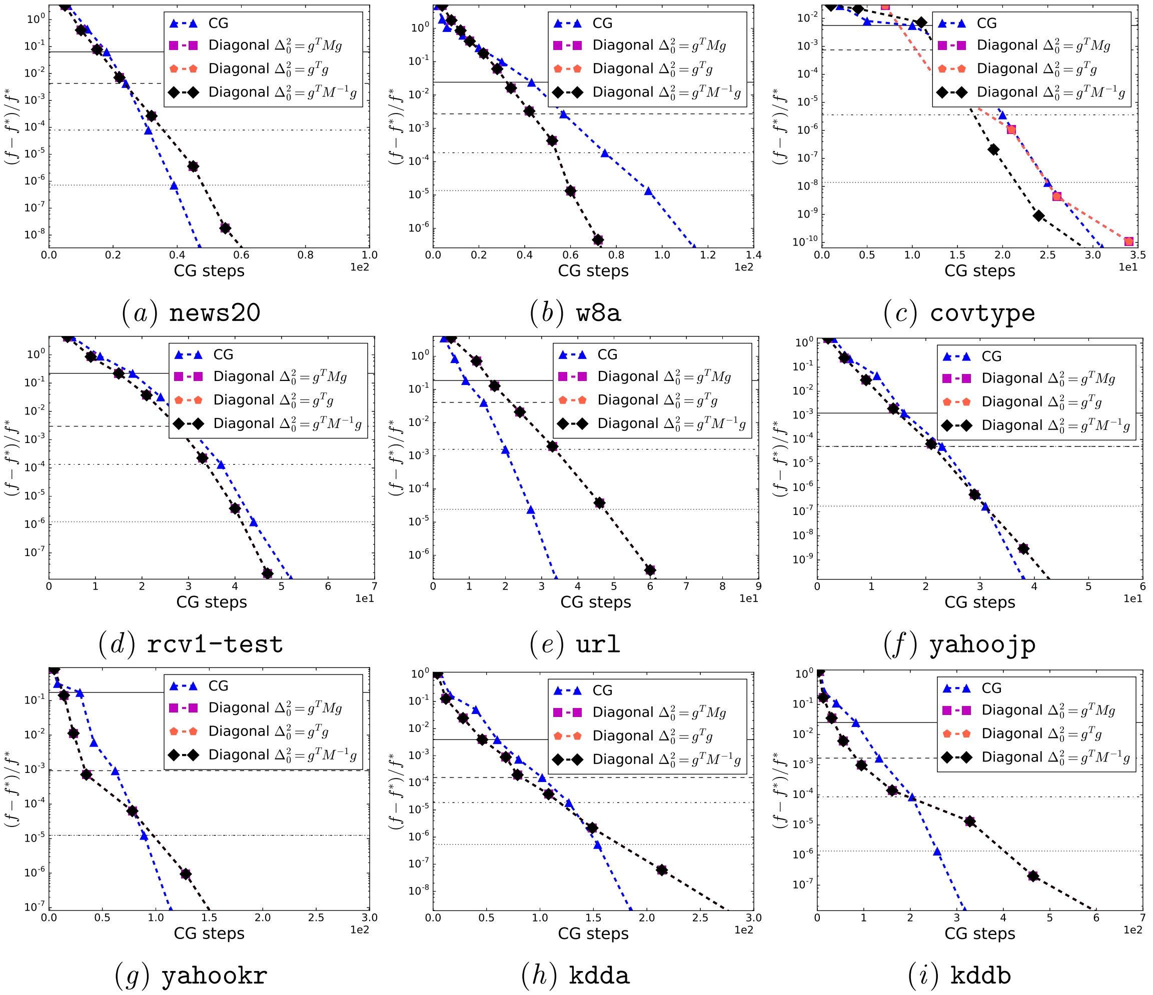
(f) yahoojp

(g) yahookr

(h) kdda

Figure xxx: Convergence of using different preconditioners. The x-axis of the figures are the cumulative number of CG iterations. Other settings are the same as those in Figure 1. Loss=L2 and $C = 100C_{\text{Best}}$.

X.7. Experimental Results of Using Different Initial Δ Values

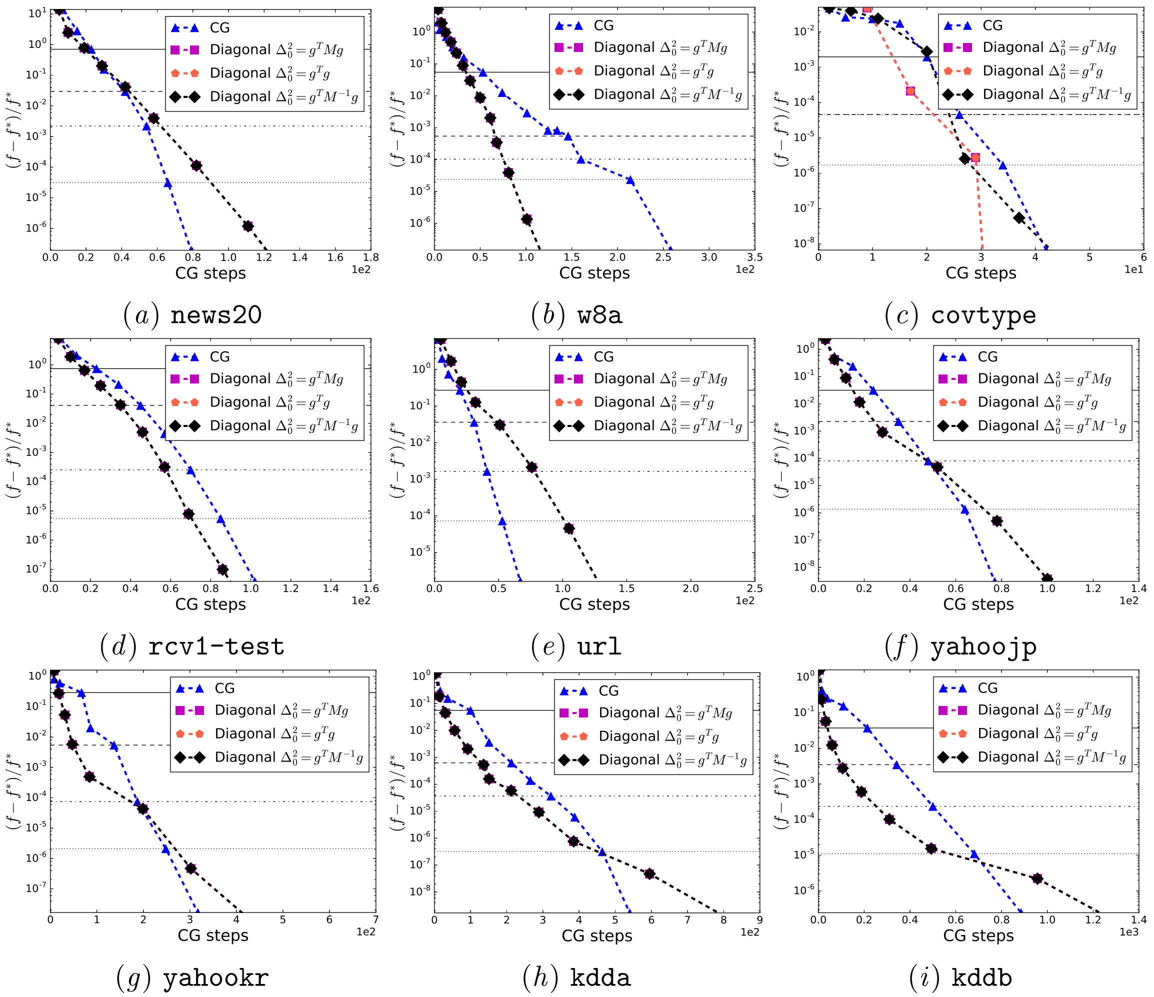


(a) news20 (b) w8a (c) covtype

(d) rcv1-test (e) url (f) yahoojp

(g) yahookr (h) kdda (i) kddb

Figure xxxi: Convergence of using different initial delta in PCG. Diagonal preconditioner is used. Other settings are the smae as those in Figure 1. Loss=Logistic and $C = 0.01C_{\text{Best}}$.



(a) news20

(b) w8a

(c) covtype

(d) rcv1-test

(e) url

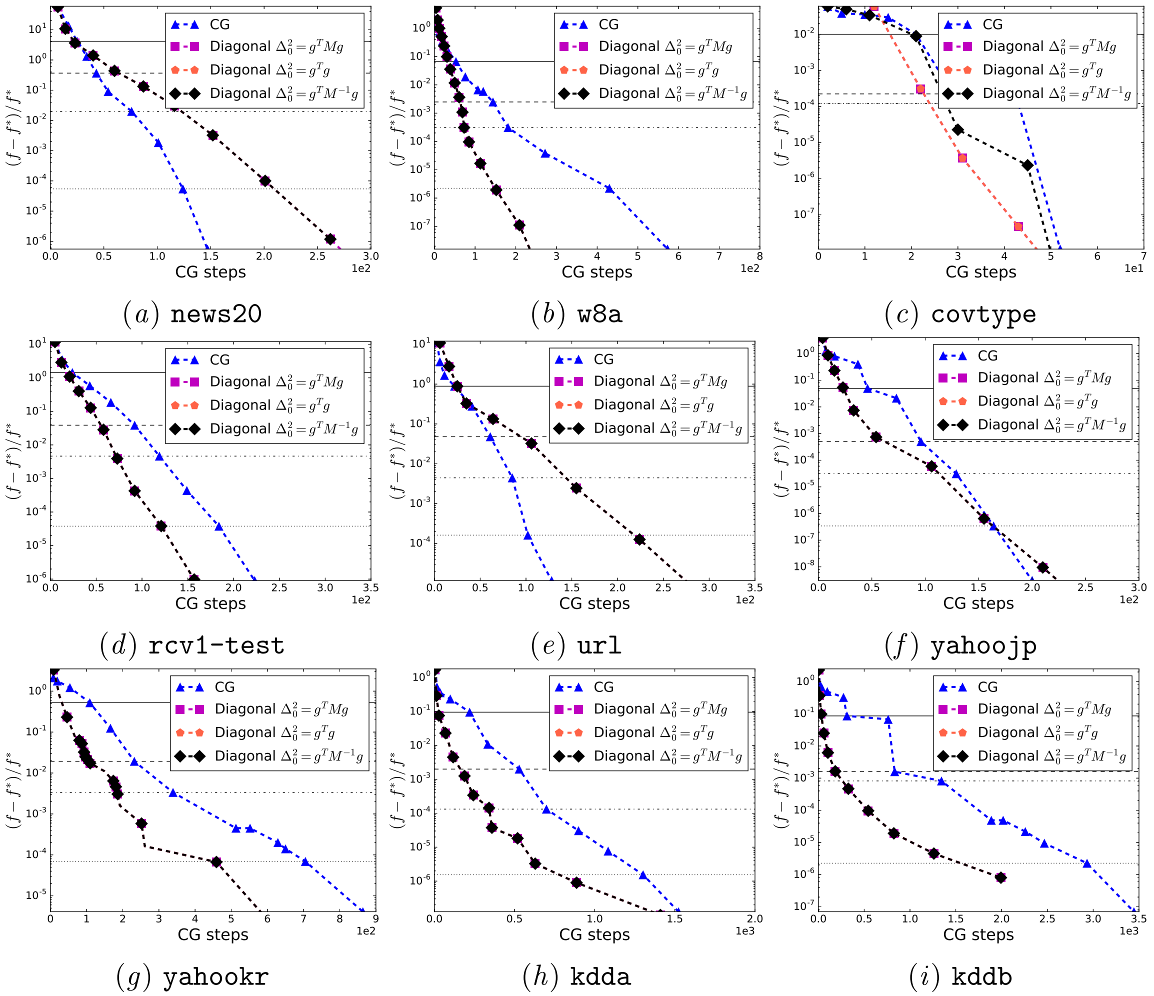
(f) yahoojp

(g) yahookr

(h) kdda

(i) kddb

Figure xxxii: Convergence of using different initial delta in PCG. Diagonal preconditioner is used. Other settings are the smae as those in Figure 1. Loss=Logistic and $C = 0.1C_{\text{Best}}$.

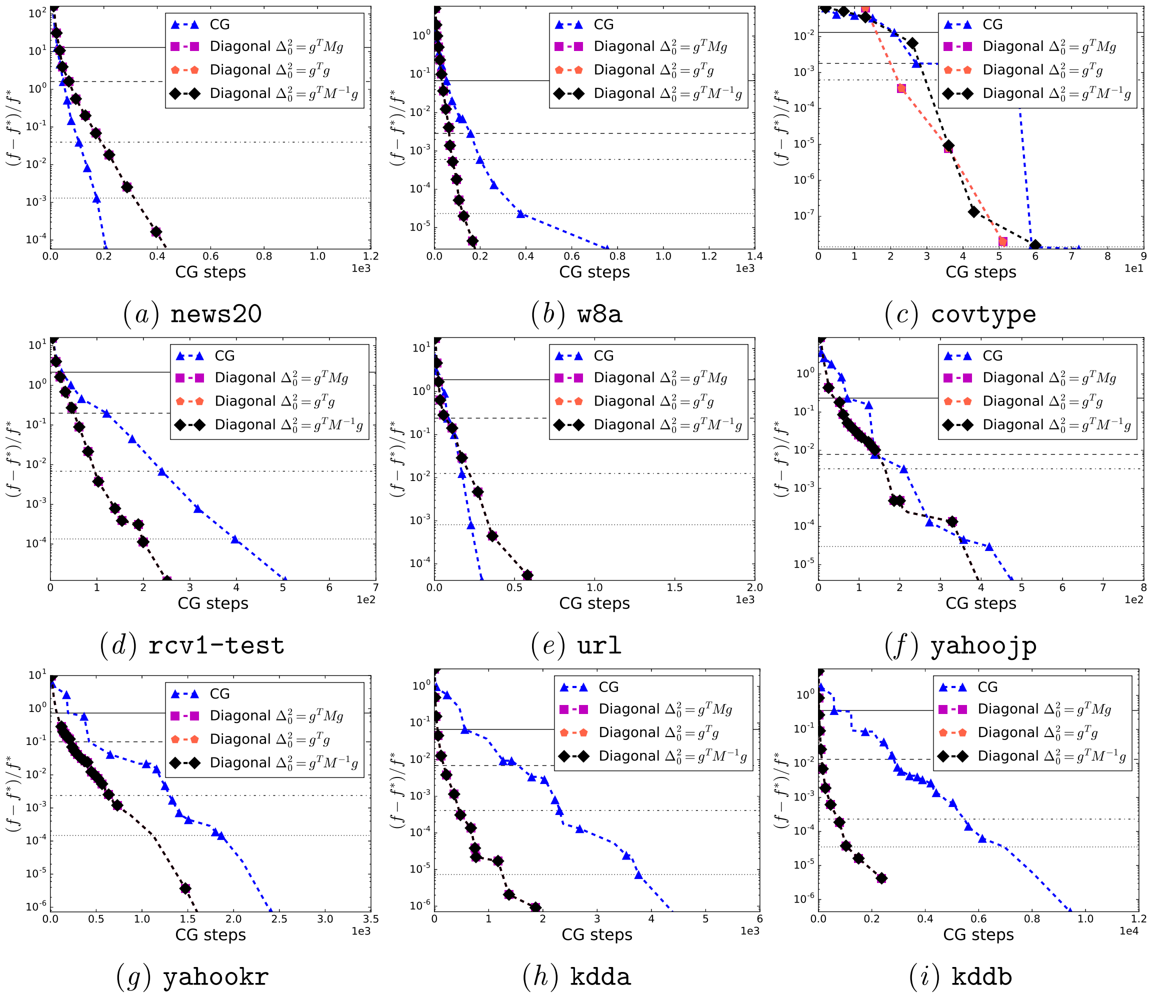


(*a*) news20 (*b*) w8a (*c*) covtype

(*d*) rcv1-test (*e*) url (*f*) yahoojp

(*g*) yahookr (*h*) kdda (*i*) kddb

Figure xxxiii: Convergence of using different initial delta in PCG. Diagonal preconditioner is used. Other settings are the smae as those in Figure 1. Loss=Logistic and $C = 1C_{\text{Best}}$.

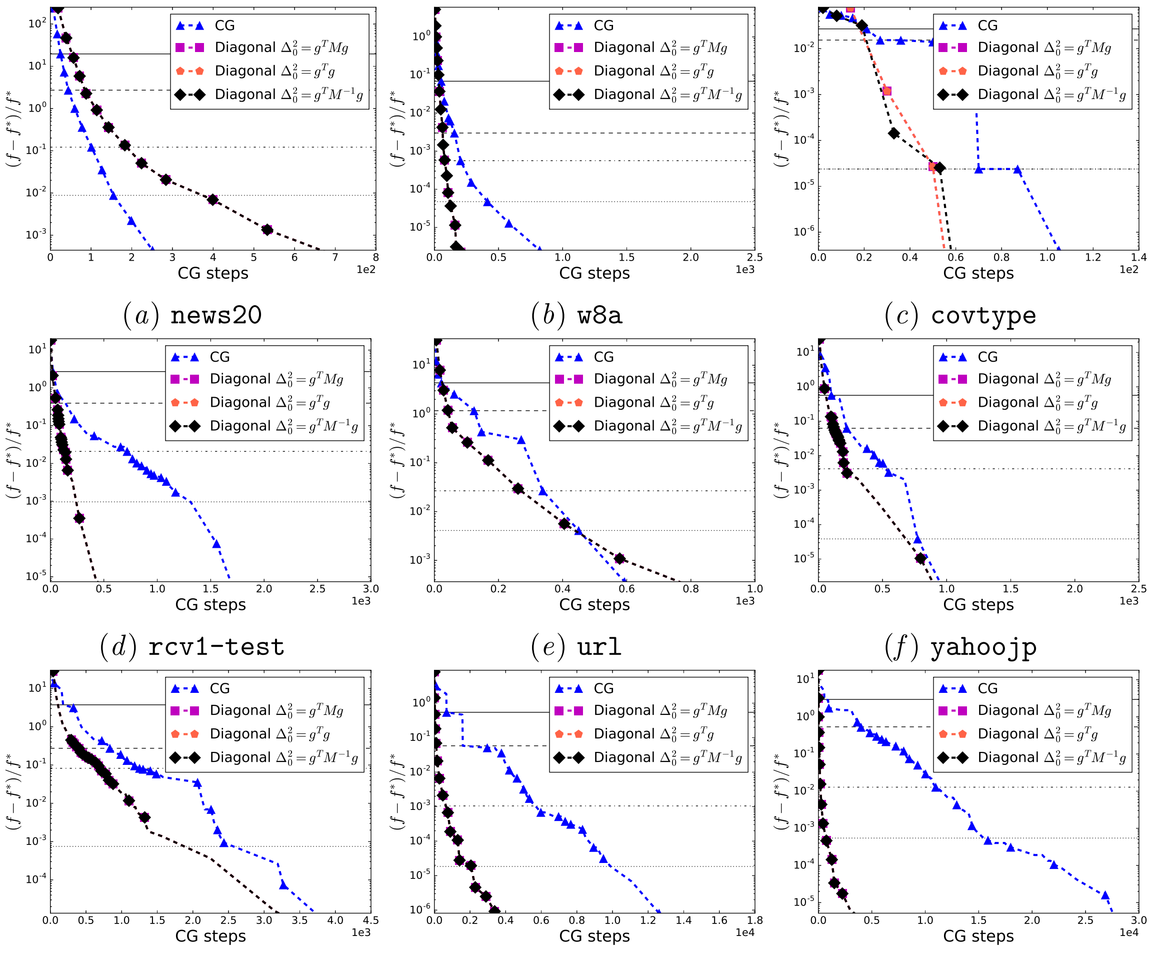


(a) `news20` (b) `w8a` (c) `covtype`

(d) `rcv1-test` (e) `url` (f) `yahoojp`

(g) `yahookr` (h) `kdda` (i) `kddb`

Figure xxxiv: Convergence of using different initial delta in PCG. Diagonal preconditioner is used. Other settings are the smae as those in Figure 1. Loss=Logistic and $C = 10C_{\text{Best}}$.



(a) news20 (b) w8a (c) covtype

(d) rcv1-test (e) url (f) yahoojp

(g) yahookr (h) kdda (i) kddb

Figure xxxv: Convergence of using different initial delta in PCG. Diagonal preconditioner is used. Other settings are the smae as those in Figure 1. Loss=Logistic and $C = 100C_{\text{Best}}$.

XI. Experiments on L2-regularized Support Vector Regression

Ho and Lin (2012) consider a trust region Newton method to solve the primal form of L2-regularized support vector regression (SVR). The method is further integrated into one of the SVR solvers in LIBLINEAR. In this section, we would like to investigate the effectiveness of our proposed preconditioning method on SVR problems. Here we consider six regression data sets and the statistics are presented in Table III. All data sets can be downloaded from LIBSVM Data Sets (2007).[1] Note that some sets used here have very few features. More larger SVR data sets may be needed for a thorough evaluation.

The following methods are considered in the experiments:

- `CG`: LIBLINEAR 2.11, a trust-region method without PCG.
- `Mixed-2`: the preconditioner in (21) with $\alpha = 10^{-2}$.
- `CG or Diag`: the technique in Section 4.1 to run CG and diagonal PCG in parallel.

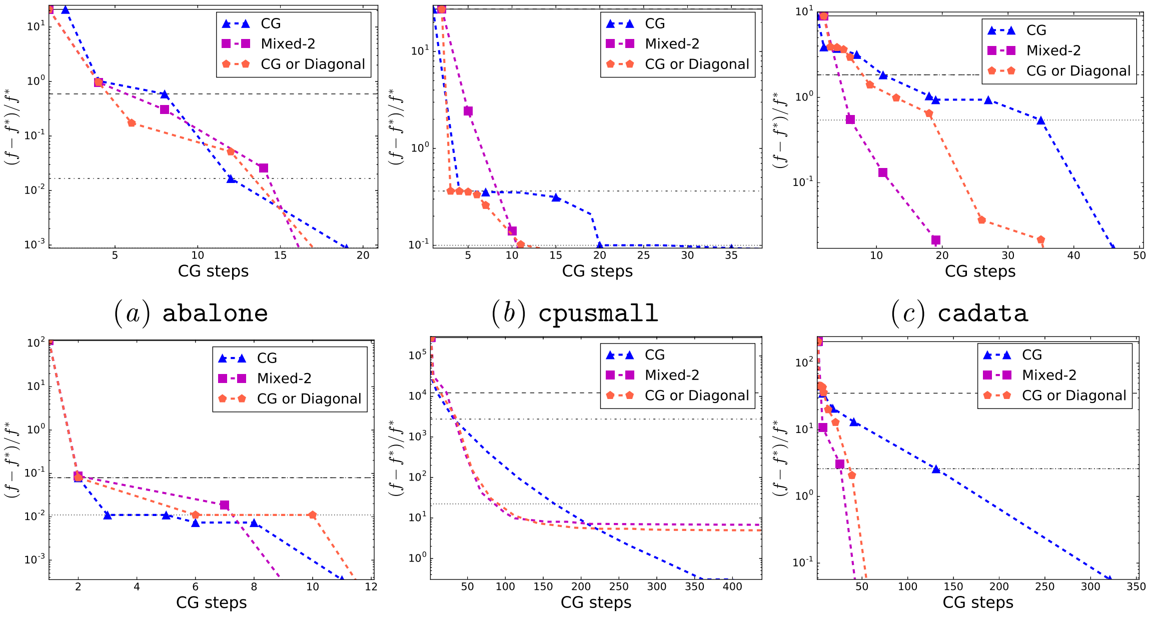
The setting of regularization parameter $C = \{1, 10, 100, 100\}$ and $\epsilon_p = 0.1$ (default setting in LIBLINEAR) are used in our experiments.

From Figures xxxvi to xxxix, we observe `Mixed-2` is similar or better than `CG` in most of cases if we consider LIBLINEAR's default stopping condition (the second horizontal line). However, we also observe that in the figures of `E2006-log1p-train`, `Mixed-2` and `CG or Diag` converge very slowly in the later stage. This situation is different from earlier results on classification, where `CG or Diag` is generally the fastest. A possible explanation is that they reach some bad points in the middle of the optimization process and gets worse optimization paths toward the end. Fortunately, the slow convergence only happens in a very late stage (below the fourth horizontal line). For applications like machine learning, in general a less accurate solution is enough so the above situation may be considered less important.

[1] https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/regression.html

Data sets	#instances	#features
`abalone`	4,177	8
`cpusmall`	8,192	12
`cadata`	20,640	8
`E2006-tfidf-train`	16,087	150,360
`E2006-log1p-train`	16,087	4,272,227
`YearPredictionMSD`	463,715	90

Table III: Statistics of Regression Data



(a) `abalone`

(b) `cpusmall`

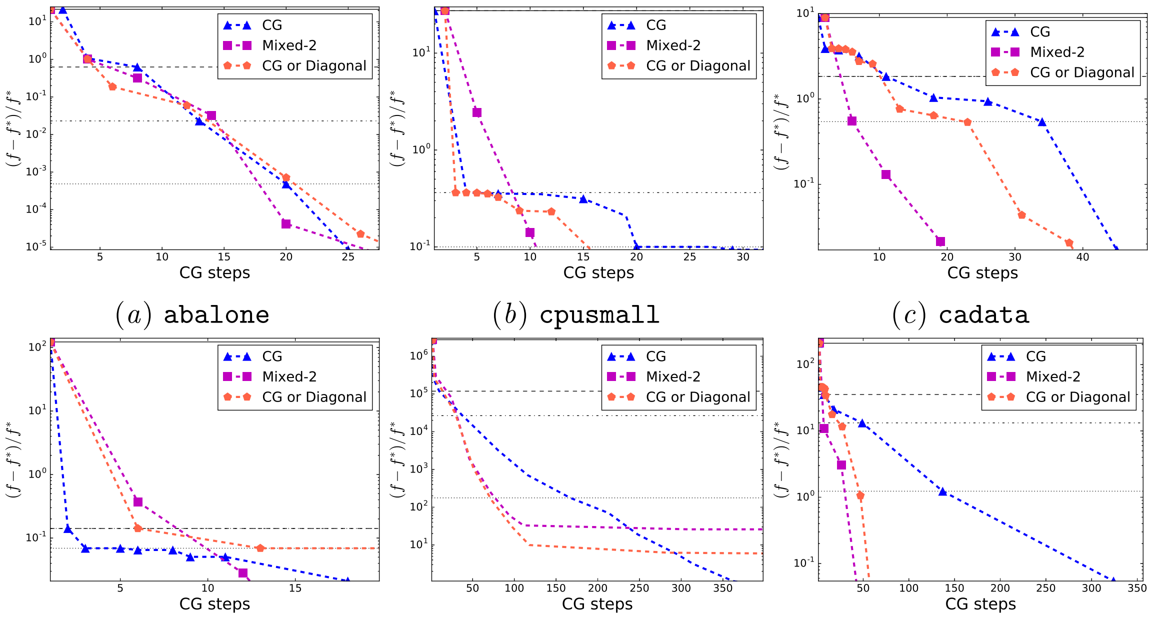
(c) `cadata`

(d) `E2006-tfidf-train`

(e) `E2006-log1p-train`

(f) `YearPredictionMSD`

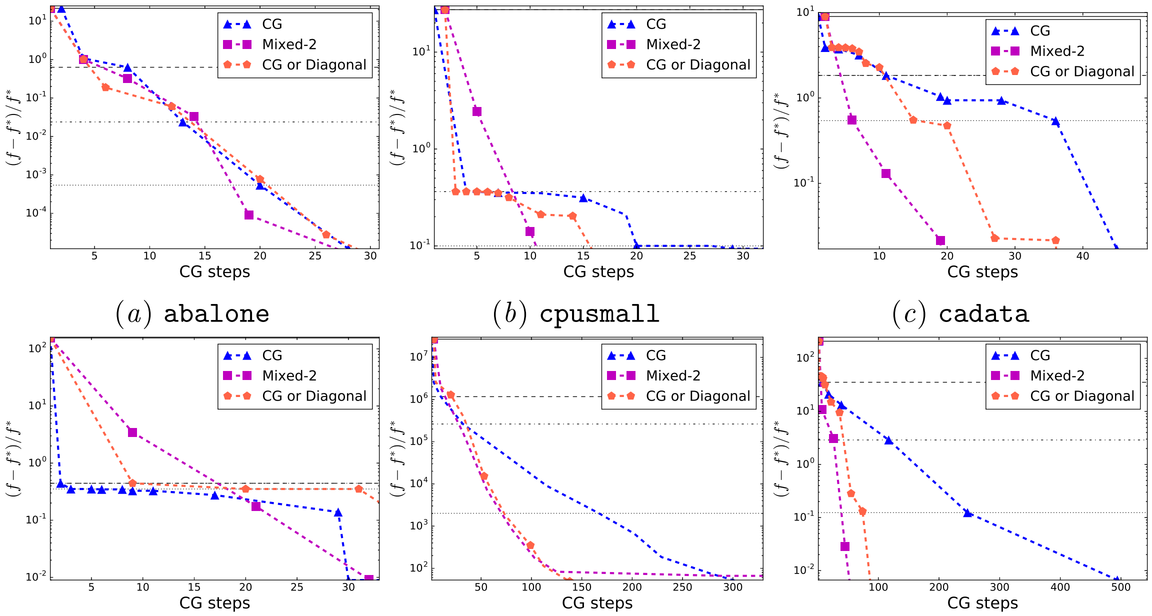
Figure xxxvi: Convergence of using different preconditioners on regression data sets. Other settings are the same as those in Figure 1. $C = 1$.



(a) `abalone` (b) `cpusmall` (c) `cadata`

(d) `E2006-tfidf-train` (e) `E2006-log1p-train` (f) `YearPredictionMSD`

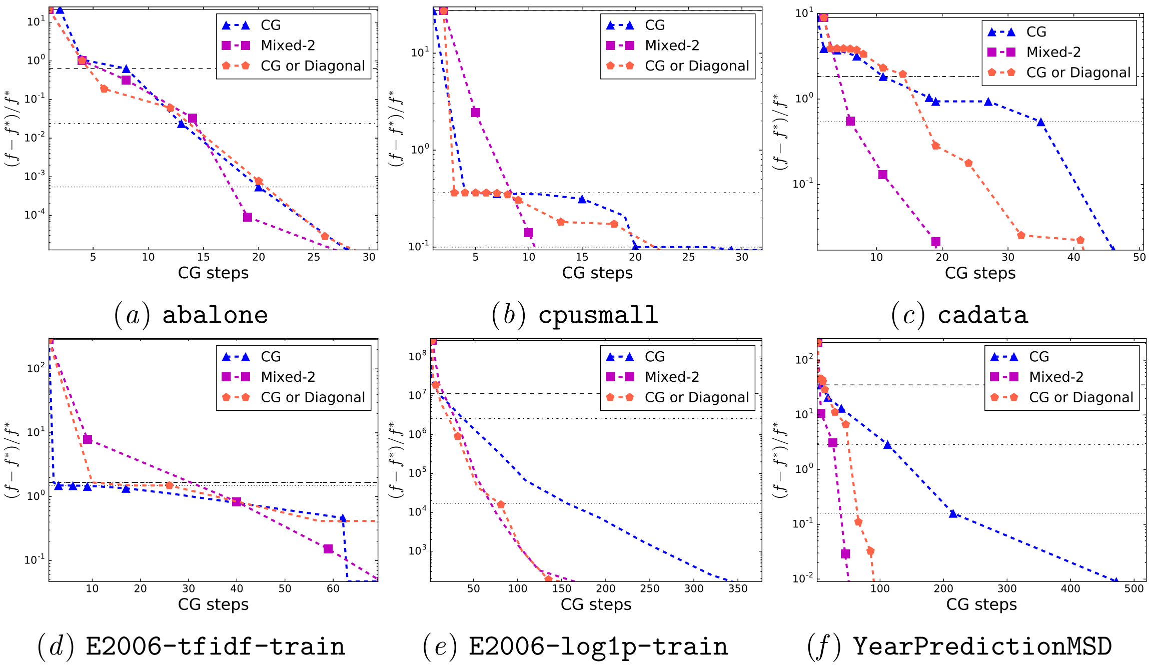
Figure xxxvii: Convergence of using different preconditioners on regression data sets. Other settings are the same as those in Figure 1. $C = 10$.



(a) `abalone` (b) `cpusmall` (c) `cadata`

(d) `E2006-tfidf-train` (e) `E2006-log1p-train` (f) `YearPredictionMSD`

Figure xxxviii: Convergence of using different preconditioners on regression data sets. Other settings are the same as those in Figure 1. $C = 100$.



(a) abalone

(b) cpusmall

(c) cadata

(d) E2006-tfidf-train

(e) E2006-log1p-train

(f) YearPredictionMSD

Figure xxxix: Convergence of using different preconditioners on regression data sets. Other settings are the same as those in Figure 1. $C = 1000$.

References

G. H. Golub and C. F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, third edition, 1996.

Chia-Hua Ho and Chih-Jen Lin. Large-scale linear support vector regression. *Journal of Machine Learning Research*, 13:3323–3348, 2012. URL http://www.csie.ntu.edu.tw/~cjlin/papers/linear-svr.pdf.

Roger A. Horn and Charles R. Johnson. *Matrix analysis*. Cambridge university press, 2012.

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

Chih-Jen Lin and Jorge J. Moré. Newton's method for large-scale bound constrained problems. *SIAM Journal on Optimization*, 9:1100–1127, 1999.

Chih-Jen Lin, Ruby C. Weng, and S. Sathiya Keerthi. Trust region Newton method for large-scale logistic regression. *Journal of Machine Learning Research*, 9:627–650, 2008. URL http://www.csie.ntu.edu.tw/~cjlin/papers/logistic.pdf.

Massimo Roma. Dynamic scaling based preconditioning for truncated Newton methods in large scale unconstrained optimization. *Optimization Methods and Software*, 20(6):693–713, 2005.