BLITZ: A Principled Meta-Algorithm for Scaling Sparse Optimization

Tyler B. Johnson
Carlos Guestrin
University of Washington, Seattle, WA 98195, USA

Abstract
By reducing optimization to a sequence of small subproblems, working set methods achieve fast convergence times for many challenging problems. Despite excellent performance, theoretical understanding of working sets is limited, and implementations often resort to heuristics to determine subproblem size, makeup, and stopping criteria. We propose BLITZ, a fast working set algorithm accompanied by useful guarantees. Making no assumptions on data, our theory relates subproblem size to progress toward convergence. This result motivates methods for optimizing algorithmic parameters and discarding irrelevant variables as iterations progress. Applied to $\ell_1$-regularized learning, BLITZ convincingly outperforms existing solvers in sequential, limited-memory, and distributed settings. BLITZ is not specific to $\ell_1$-regularized learning, making the algorithm relevant to many applications involving sparsity or constraints.

1. Introduction
With user-specific features for recommendation, n-gram phrases in text, or high-order transformations for feature engineering, many learning problems involve large numbers of features. In these cases, $\ell_1$ regularization is a popular tool, as it biases learning toward sparse solutions. Sparsity offers many advantages, including reduced resources needed at test time, more interpretable models, and statistical efficiency, as the feature space may increase exponentially with sample size (Ng, 2004; Wainwright, 2009).

Unfortunately, convergence times for $\ell_1$-regularized loss minimization tend to grow linearly with the number of features. For faster solutions, recent works have considered parallel algorithms (Boyd et al., 2011; Bradley et al., 2011; Fercoq & Richtárik, 2013). Despite parallel speedups, these algorithms in their basic form share a significant inefficiency: equal priority is assigned to all features. Due to sparsity, most features are instead irrelevant to the solution!

We propose BLITZ, a general optimization algorithm that prioritizes resources on important parts of the problem. For $\ell_1$-regularized learning, BLITZ solves a sequence of sub-problems restricted to small subsets of features using an existing solver, converging quickly to the original problem’s solution. Known as a working set method, this concept is not new. GLMNET (Friedman et al., 2010) and LIBLINEAR (Yuan et al., 2012), two libraries for $\ell_1$-regularized learning, prioritize computation with working set heuristics. More broadly, working sets have been applied successfully to a diverse set of optimization problems involving sparsity or constraints; see Fan et al. (2005), Tsochantaridis et al. (2005), and Kim & Park (2008) as examples.

Given the practical success of working set methods, theoretical understanding of these algorithms is surprisingly limited. How to choose a subproblem, how large it should be, and when it should terminate are questions inadequately answered by existing theory. We present novel analysis to offer such perspective. Without assumptions on data, our theory explains how to choose working sets to guarantee a desired amount of progress toward convergence. This motivates methods for eliminating irrelevant variables and optimizing algorithmic parameters, making BLITZ’s choices of subproblem size, variables, and stopping criteria more principled and robust than previous approaches allow.

In practice, our theoretical insights lead to very fast convergence times for $\ell_1$-regularized learning. In the sequential setting, BLITZ outperforms solvers such as GLMNET and LIBLINEAR, making BLITZ one of the fastest algorithms for high dimensional lasso and sparse logistic regression on a single machine. We then show additional gains for BLITZ in limited-memory and distributed regimes. By considering data in subsets, BLITZ prioritizes not only computation but also memory and bandwidth usage, directly targeting I/O and communication bottlenecks for problems at scale.

Importantly, BLITZ directly extends to objectives other
than $\ell_1$-regularized loss minimization. Given the performance of BLITZ for this well-studied application, an intriguing open question is whether similar performance is achievable for additional objectives.

In summary of our contributions, we propose BLITZ, a working set algorithm that:

- Selects theoretically justified subproblems to maximize guaranteed progress toward convergence.
- Applies theoretical analysis to automatically tune algorithmic parameters and discard irrelevant constraints as the algorithm runs.
- Achieves very fast convergence times when applied to $\ell_1$-regularized learning in a variety of settings.
- Provides a novel proof path for analyzing working set methods for sparse or constrained optimization.

2. The BLITZ Algorithm

In this section, we introduce BLITZ, including convergence analysis and numerical experiments examining our bounds.

2.1. Problem Formulation

We consider the convex problem

$$\begin{align*}
\text{minimize} & \quad f(x) \\
\text{s.t.} & \quad h_j(x) \leq 0 \quad j = 1, \ldots, m,
\end{align*}$$

where $x \in \mathbb{R}^n$, and $h_j$ is convex for all $j$. We assume $f$ is $\gamma$-strongly convex, and we denote (P1)’s solution by $x^\ast$. We define the feasible region

$$D = \{x : h_j(x) \leq 0 \text{ for all } j = 1, \ldots, m\}. \quad (1)$$

We focus on instances of (P1) with large $m$. While not obvious, many unconstrained problems involving sparsity are instances of (P1), as sparsity often appears as constraints in a problem’s dual (see Section 3 or Bach et al. (2012)).

Define the set of active constraints at $x^\ast$:

$$C^\ast = \{h_j : h_j(x^\ast) = 0\}. \quad (2)$$

In addition to (P1), $x^\ast$ solves the modified problem

$$\begin{align*}
\text{minimize} & \quad f(x) \\
\text{s.t.} & \quad h_j(x) \leq 0 \quad \text{for all } h_j \in C^\ast.
\end{align*}$$

2.2. BLITZ Algorithm Overview

BLITZ is defined in Algorithm 2. $x$ is initialized as the unconstrained minimizer of $f$ (unique due to strong convexity), while $y$ is a point in $D$. We update $y$ via $y \leftarrow \alpha x + (1 - \alpha) y$, where $\alpha$ is the largest coefficient in $D$.
[0, 1] such that y remains in D. Constraints are prioritized
according to the Euclidean distance
\[ \text{dist}(h_j, y) = \inf_{z : h_j(z) = 0} \| z - y \|_2, \]
where constraints with boundaries closest to y receive
highest priority. Often (3) can be computed in closed form
(and often lower bounded for more complex h_j), and we
include examples for doing so in Appendix F. A constraint
h_j is included in the working set C if (i.) \text{dist}(h_j, y) \leq \tau, or (ii.) h_j(x) = 0, meaning h_j is active
at x. \tau controls the size of each subproblem. Upon
determining C, x is set to the minimizer of f subject only
to constraints in C. BLITZ reaches optimality when x no
longer violates any constraints.

Before considering analysis, we can observe two intuitive
advantages Algorithm 2 has over Algorithm 1:

- Scale invariance: Consider h_j(x) = \sum_i x_i. In this
case, h_j and h_k = 100h_j are effectively the same
constraint. However, in Algorithm 1, h_k may be included
in C when h_j is not. BLITZ is invariant to this scaling.
- Feasibility regularization: BLITZ chooses constraints
C that are close to a feasible point y or tight at x. This
ensures both f(y) decreases and f(x) increases during
an iteration. Algorithm 1 chooses constraints that
are active or most violated by x, which only ensures
f(x) increases. By using y to choose C, BLITZ com-
penstates for constraints that are greatly violated by x.

2.3. Convergence Analysis

We now analyze the convergence of BLITZ. For now, we
assume each iteration’s subproblem is solved exactly. All
proofs are provided in supplementary material.

For all iterations t, since y_t \in D and x_t minimizes f
subject to a subset of constraints, we have
\[ f(x_t) \leq f(x^*) \leq f(y_t). \]

Thus, we may define an optimality gap
\[ \Delta_t = f(y_t) - f(x_t) \geq f(y_t) - f(x^*). \]

A strength of BLITZ is that both f(y_t) and f(x_t) converge
monotonically to f(x^*). At each iteration, substantial
improvement must be made in f(y_t), f(x_t), or both. This is
the intuition of our first theorem:

**Theorem 2.1 (Convergence Progress at Iteration t).** Let \Delta_t
and \Delta_{t+1} be the optimality gaps after iterations t and t+1
of Algorithm 2. Then for all t \geq 1 if the algorithm does not
converge at iteration t + 1, we have
\[ \Delta_{t+1} \leq \Delta_t - \left( \frac{2}{\tau + 1} \right)^{1/3} \Delta_t^2 . \]

If \tau is held constant for all t, Algorithm 2 converges in
a fixed number of subproblems. In practice, \tau should
decrease over time to ensure |C| remains small. The following
corollary suggests a scaling of \tau for fast convergence:

**Corollary 2.2 (Linear Convergence).** For t \geq 1, define
\[ \Delta'_t = f(y_t) - f(x_{t-1}), \]
and suppose we run Algorithm 2 choosing \tau_t as
\[ \tau_t = \sqrt{\frac{2}{5}} (1 - r)^3 \Delta'_t \]
for some r \in [0, 1). Then for t \geq 1, we have
\[ f(y_t) - f(x^*) \leq r^{t-1} \Delta_0 . \]

Another consequence of Theorem 2.1 is a method for iden-
tifying constraints guaranteed to be inactive at x^*. This
is similar to prescreening, a useful preprocessing step that
eliminates irrelevant constraints for particular instances
of (P1) (El Ghaoui et al., 2012; Liu et al., 2014). Finding \tau_t
such that \Delta_t \leq 0 in (6), we arrive at the following corol-
ary:

**Corollary 2.3 (Constraint Elimination).** For t \geq 1, define
\Delta_t as in (7). If
\[ \text{dist}(h_j, y_t) > \sqrt{\frac{2}{5}} \Delta'_t, \]
then h_j(x^*) < 0, and h_j may be eliminated from (P1).

Compared to prescreening, Corollary 2.3 is more general
and can be applied at any iteration of BLITZ; however,
fewer constraints may be discarded initially. Elaboration
on this topic is included in Appendix C.

2.4. Experiments with Bounds

To examine our bounds numerically, we instantiate (P1) as
\[
\begin{align*}
\text{minimize} & \quad \| x - b \|_2^2 \\
\text{subject to} & \quad | A_j^T x | \leq \lambda, \quad j = 1, \ldots, m. \quad (P3)
\end{align*}
\]

(Later we will see (P3) is dual to the lasso.) We let
m = 10,000 and n = 100. Elements of b and A_j are drawn
i.i.d. from \mathcal{N}(0, 1). We set \lambda = \frac{1}{r} \max_j | A_j^T b |, resulting
in approximately 30 active constraints at x^*.

In Figure 2, we compare results solving (P3) with BLITZ
to our worst-case bounds. Figure 2(a) plots convergence
vs. iteration choosing \tau with (8) and r = 0.95. Figure 2(b)
plots optimality gaps after 2 iterations using a range of r
values. Each plot aggregates 15 problem instances. The
solid green line is our analytical bound. Axes are scaled so
that the bound displays as a line.

From Figure 2, we see that while convergence is faster
in practice than our bounds guarantee, theory and practice
agree well on the scaling of \tau and \Delta.
BLITZ

Table 1. Summary of Quantities for $\ell_1$-Regularized Learning. Table includes loss $\phi$, convex conjugate $\phi^*$, primal-dual mapping $p$, and smoothness constant $L$ for lasso and logistic regression.

<table>
<thead>
<tr>
<th>LOSS</th>
<th>$\phi_i(a_i^T w)$</th>
<th>$\phi^*_i(x_i)$</th>
<th>$[p(Aw^*, b)]_i$</th>
<th>$L$</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQUARES</td>
<td>$\frac{1}{2} (a_i^T w - b_i)^2$</td>
<td>$-\frac{2}{b_i^2} (b_i + x_i)^2 - \frac{1}{b_i^2} b_i^2$</td>
<td>$a_i^T w^* - b_i$</td>
<td>1</td>
</tr>
<tr>
<td>LOGISTIC</td>
<td>$\log (1 + \exp(-b_i a_i^T w))$</td>
<td>$(\frac{2}{b_i^2} + \log(1 + \frac{2}{b_i^2}))$</td>
<td>$-2 b_i \exp(-b_i a_i^T w^*)$</td>
<td>$\frac{1}{2}$</td>
</tr>
</tbody>
</table>

For arbitrary loss $\phi_i$, we require a single assumption:

**Assumption 3.1 (Smooth Loss).** The derivative $\phi_i'$ exists and is Lipschitz continuous with constant $L$:

$$|\phi_i'(x) - \phi_i'(y)| \leq L|x - y| \quad \text{for all } x, y \in \mathbb{R}.$$  

3.2. $\ell_1$ Duality

To solve (P4) with BLITZ, we transform (P4) into its dual:

$$\begin{align*}
\min_{x \in \mathbb{R}^n} & \quad \sum_{i=1}^{n} \phi^*_i(x_i) \\
\text{s.t.} & \quad |A_i^T x| \leq \lambda \quad j = 1, \ldots, m.
\end{align*}$$

(P5)

Here $\phi_i^*$ is the convex conjugate of $\phi_i$. $f(x) = \sum_i \phi^*(x_i)$ is strongly convex due to the following proposition:

**Proposition 3.2 (Strong Convexity of $\ell_1$ Dual).** Given Assumption 3.1, $f(x)$ is strongly convex with parameter $\frac{1}{L}$.

Strong duality holds for this problem ($f(x^*) = g(w^*)$), and there exists a mapping $p$ between optimal variables:

$$x^* = p(Aw^*, b) .$$

Table 1 summarizes relevant quantities for (P4) and (P5). Derivations are included in Appendix E.

3.3. Partial Subproblem Convergence

(P5) can be solved naturally with BLITZ. Minimizing (P5) subject to a subset of constraints corresponds to maximizing (P4) over a subset of variables, prioritizing resources on important features. However, Algorithm 2 requires exact subproblem solutions, which is impractical. To accommodate partial solutions in our analysis, we require the subproblem solver returns a primal-dual pair $(x_i, w_i)$, where

$$x_i = \xi_i \cdot p(Aw_i, b) ,$$

and $\xi_i$ is the largest scalar in $(0, 1]$ such that $|A_i^T x_i| \leq \lambda$ for all constraints in $C_i$. Here we must redefine $\Delta_i$ as

$$\Delta_i = f(y_i) - g(w_i) ,$$

as well as sparse logistic regression (Ng, 2004), for which $B = [-1, 1]$ and

$$g(w) = -\sum_{i=1}^{n} \log (1 + \exp(-b_i a_i^T w)) - \lambda \|w\|_1 .$$

Figure 2. Theory vs. Practice. (a) For $r = 0.95$, 15 trials of observed optimality gap and bound (Corollary 2.2) vs. iteration. (b) After 2 iterations, optimality gap and bound (Corollary 2.2) vs. decrease ratio $r$. Convergence is faster than theory guarantees, but theory and experiments agree on the scaling of $r$ and $\Delta$ (plotted appropriately, trends are approximately linear).

3. Application: $\ell_1$-Regularized Learning

We now apply BLITZ to $\ell_1$-regularized optimization. This class of problems is widely used for supervised learning, compressed sensing, and algorithms for more complex problems in which $\ell_1$ penalties appear in subproblems.

3.1. $\ell_1$-Regularized Loss Minimization

We consider problems for which a feature vector $a_i \in \mathbb{R}^m$ is used to predict a label $b_i \in \mathbb{B}$. Our prediction function is parameterized by a vector $w^* \in \mathbb{R}^m$, which is computed by maximizing an $\ell_1$-regularized likelihood function over a set of $n$ training examples $\{(a_1, b_1), \ldots, (a_n, b_n)\}$:

$$\text{maximize } g(w) = -\sum_{i=1}^{n} \phi_i(a_i^T w) - \lambda \|w\|_1 .$$

(A)

Above $\phi_i : \mathbb{R} \to \mathbb{R}_{\geq 0}$ is a convex loss function parameterized by $b_i$, $\lambda > 0$ is a tuning parameter. For large enough $\lambda$, many values of $w^*$ are exactly zero. We let $A \in \mathbb{R}^{n \times m}$ denote the design matrix, its $i$th row $a_i$ and $j$th column $A_j$, while $b \in \mathbb{B}^n$ denotes a labels vector with $i$th element $b_i$.

We focus on two popular forms of (A): the lasso (Tibshirani, 1996), for which $\mathbb{B} = \mathbb{R}$ and

$$g(w) = -\frac{1}{2} \|Aw - b\|_2^2 - \lambda \|w\|_1 ,$$

as well as sparse logistic regression (Ng, 2004), for which $B = [-1, 1]$ and

$$g(w) = -\sum_{i=1}^{n} \log (1 + \exp(-b_i a_i^T w)) - \lambda \|w\|_1 .$$

For arbitrary loss $\phi_i$, we require a single assumption:

**Assumption 3.1 (Smooth Loss).** The derivative $\phi_i'$ exists and is Lipschitz continuous with constant $L$:

$$|\phi_i'(x) - \phi_i'(y)| \leq L|x - y| \quad \text{for all } x, y \in \mathbb{R}.$$  

3.2. $\ell_1$ Duality

To solve (P4) with BLITZ, we transform (P4) into its dual:

$$\begin{align*}
\min_{x \in \mathbb{R}^n} & \quad \sum_{i=1}^{n} \phi^*_i(x_i) \\
\text{s.t.} & \quad |A_i^T x| \leq \lambda \quad j = 1, \ldots, m.
\end{align*}$$

(P5)

Here $\phi_i^*$ is the convex conjugate of $\phi_i$. $f(x) = \sum_i \phi^*(x_i)$ is strongly convex due to the following proposition:

**Proposition 3.2 (Strong Convexity of $\ell_1$ Dual).** Given Assumption 3.1, $f(x)$ is strongly convex with parameter $\frac{1}{L}$.

Strong duality holds for this problem ($f(x^*) = g(w^*)$), and there exists a mapping $p$ between optimal variables:

$$x^* = p(Aw^*, b) .$$

Table 1 summarizes relevant quantities for (P4) and (P5). Derivations are included in Appendix E.

3.3. Partial Subproblem Convergence

(P5) can be solved naturally with BLITZ. Minimizing (P5) subject to a subset of constraints corresponds to maximizing (P4) over a subset of variables, prioritizing resources on important features. However, Algorithm 2 requires exact subproblem solutions, which is impractical. To accommodate partial solutions in our analysis, we require the subproblem solver returns a primal-dual pair $(x_i, w_i)$, where

$$x_i = \xi_i \cdot p(Aw_i, b) ,$$

and $\xi_i$ is the largest scalar in $(0, 1]$ such that $|A_i^T x_i| \leq \lambda$ for all constraints in $C_i$. Here we must redefine $\Delta_i$ as

$$\Delta_i = f(y_i) - g(w_i) ,$$

as well as sparse logistic regression (Ng, 2004), for which $B = [-1, 1]$ and

$$g(w) = -\sum_{i=1}^{n} \log (1 + \exp(-b_i a_i^T w)) - \lambda \|w\|_1 .$$

Figure 2. Theory vs. Practice. (a) For $r = 0.95$, 15 trials of observed optimality gap and bound (Corollary 2.2) vs. iteration. (b) After 2 iterations, optimality gap and bound (Corollary 2.2) vs. decrease ratio $r$. Convergence is faster than theory guarantees, but theory and experiments agree on the scaling of $r$ and $\Delta$ (plotted appropriately, trends are approximately linear).
so that \( \Delta_t \) upper bounds \( f(y_t) - f(x^*) \) and \( g(w^*) - g(w_t) \) for all \( t \). We avoid spending excessive time on subproblem \( t \) by monitoring its duality gap, terminating when

\[
f(x_t) - g(w_t) \leq \epsilon_t \left(f(y_t) - g(w_t)\right)
\]

for a tolerance \( \epsilon_t \in [0, 1) \). This enables our next theorem:

**Theorem 3.3** (Progress for \( \ell_t \) with Approximate Solver). For (P5), define \( \Delta_t \) as in (16), and assume \( x_t \) and \( w_t \) satisfy (17). If \( \alpha_{t+1} = 1 \), assume \( g(w_{t+1}) \geq g(w_t) \). If \( \alpha_{t+1} < 1 \), let \( h_j \) be the (possibly non-unique) constraint such that \( h_j(x_t) > 0 \) and \( h_j(y_{t+1}) = 0 \) and assume \( g(w_{t+1}) \geq \max \delta g(w_t + \delta e_j) \). Then for \( t \geq 1 \), we have

\[
\Delta_{t+1} \leq \max \left\{ \Delta_t - \left( \frac{1}{2\epsilon_t} (1 - \epsilon_t)^2 \epsilon_t^2 \Delta_t^2 \right)^{1/3}, \epsilon_t \Delta_t \right\}.
\]

(18)

Note that when \( \epsilon_t = 0 \), we recover Theorem 2.1. The technical condition \( g(w_{t+1}) \geq \max \delta g(w_t + \delta e_j) \) can easily be satisfied with one coordinate descent update of \( w_j \).

### 3.4. Optimizing Algorithmic Parameters

The performance of working set algorithms is sensitive to subproblem size and stopping criteria. We apply Theorem 3.3 to optimize \( \tau \) and \( \epsilon \) at runtime. This procedure is not meant to be exact, rather to provide BLITZ with a basic mechanism for adjusting these parameters. We model the duration of iteration \( t \) as \( T_\alpha + T_{\text{solved}}(\tau, \epsilon) \), where

\[
T_\alpha = C_\alpha, \quad T_{\text{solved}}(\tau, \epsilon) = C_{\text{solved}} \frac{\text{NNZ}(\tau, t)}{\epsilon}.
\]

(19)

Above \( T_\alpha \) is the time to compute \( \alpha \). \( T_{\text{solved}}(\tau, \epsilon) \) estimates the time to solve the subproblem, increasing proportional to the number of nonzero elements in columns \( A_j \) for which \( h_j \in C_t \) and inversely proportional to \( \epsilon \). \( C_\alpha \) and \( C_{\text{solved}} \) are constants, which are computed using runtime data by solving for \( C_\alpha \) and \( C_{\text{solved}} \) in (19) after each iteration and taking median values over this history. Applying Theorem 3.3, we model convergence progress as

\[
\hat{\Delta}_{t+1}(\tau, \epsilon) = \max \left\{ \Delta'_t - C_P ((1 - \epsilon) \tau \Delta'_t)^2, \epsilon \Delta'_t \right\}.
\]

(20)

Above, \( \Delta'_t = f(y_t) - g(w_{t-1}) \), which is used as an approximation to \( \Delta_t \) since \( \Delta_t \) cannot be computed before choosing \( \tau_t \). The constant \( C_P \) accounts for bound looseness (see Figure 2), estimated using an analogous procedure to that for \( C_\alpha \) and \( C_{\text{solved}} \). Finally, we choose \( \tau_t \) and \( \epsilon_t \) by solving

\[
\tau_t, \epsilon_t = \arg\min_{\tau, \epsilon} \frac{\hat{\Delta}_{t+1}(\tau, \epsilon)}{\exp(-C_{\text{TC}}[T_\alpha + T_{\text{solved}}(\tau, \epsilon)])}
\]

approximately with grid search. The time constant \( C_{\text{TC}} \) accounts for empirical evidence that BLITZ’s overall convergence rate should be closer to linear than sublinear (see Figure 3).

![Figure 3. Optimizing Parameters](image)

Above) Squared loss. (below) Logistic loss. For synthetic problem, BLITZ is run multiple times for 15 seconds using different \( \epsilon \) and \( r \) which are fixed as BLITZ runs. Plotted is resulting optimality gap. Green curve fixes best-case \( r \) and varies \( \epsilon \). Purple curve fixes best-case \( \epsilon \) and varies \( r \). Blue line is result of automatically tuning via (21). In these cases, parameter adaption is better than any fixed \( (r, \epsilon) \) pair.

Figure 4). We set \( C_{\text{TC}} \) to the ratio of elapsed time to \( \log (\Delta_0/\Delta'_t) \). Since \( C_\alpha, C_{\text{solved}}, C_P, \) and \( C_{\text{TC}} \) cannot be computed before the first iteration, we initialize BLITZ with a relatively small, easy subproblem (100 features in sequential setting and \( \epsilon_1 = 0.5 \)).

We experiment with this approach using two synthetic datasets, each containing \( 5 \times 10^3 \) examples, \( 1 \times 10^5 \) features and elements drawn i.i.d. from \( \mathcal{N}(0, 1) \). We solve lasso on the first dataset using labels drawn from \( \mathcal{N}(0, 1) \), and we solve logistic regression on the second dataset assigning labels \( \pm 1 \) with equal probability. We solve for 15 seconds using regularization \( \lambda = 0.05 \lambda_{\text{max}}^{-1} \) and a variety of fixed \( r \) (from (8)) and \( \epsilon \) values, comparing to the proposed auto-adjustment method. As Figure 3 illustrates, performance varies for choice of \( r \) and \( \epsilon \), but our tuning method makes BLITZ robust to this effect and improves upon any single choice of parameters by an order of magnitude in this case.

### 3.5. Sequential Comparisons

We now demonstrate the performance of BLITZ in practice. Our comparisons begin with the case that the dataset \( (A, b) \) fits in memory of a single machine. For this setting, we implement BLITZ in C++ using a coordinate descent-based proximal Newton method to solve each subproblem.

In this setting, we compare BLITZ to seven alternatives:

- **PROXNEWT**: Our subproblem solver for BLITZ (no

\[1\lambda_{\text{max}} \] is the smallest \( \lambda \) for which \( w^* = 0 \).
prioritization of features).
- GLMNET 1.9-8 (Friedman et al., 2010): Popular R package for lasso and sparse logistic regression; implemented in Fortran; uses working set heuristics.
- LIBLINEAR 1.94 (Yuan et al., 2012): Widely-used C++ solver for sparse logistic regression (lasso not implemented); uses working set heuristics.
- L1_LS (Kim et al., 2007): Interior point method for lasso implemented in MATLAB®.
- L1_LOGREG 0.8.2 (Koh et al., 2007): Interior point method for sparse logistic regression written in C.
- APPROX (Fercoq & Richtárik, 2013): Parallel, accelerated coordinate descent for lasso; pre-computed step sizes ensure convergence; C++ implementation.
- CD: C++ implementation of coordinate descent for sparse logistic regression.

With the exception of L1_LS, each solver is compiled with version 4.8.2 of the applicable GNU C/C++/Fortran compiler and "-O3" optimization flag. Our hardware is a 64-bit Intel i7-2630QM processors, 8 GB memory, and 6 MB cache. Solvers that utilize parallelism (APPROX, L1_LS, and L1_LR) use up to 8 threads.

We found the performance of GLMNET depends significantly on its termination threshold—even during early iterations. We run GLMNET using only its default stopping condition.

We include results for two problem instances listed in Table 2. Datasets are publicly available from LIBSVM. To emphasize the high dimensional setting, we expand RCV1, including features formed by taking the element-wise product of each pair of original features, disregarding new features that contain five or fewer nonzeros. Since L1_LS and APPROX do not support an unregularized intercept term, we include this variable for logistic regression but not lasso. We standardize columns to have zero mean and unit variance.

We quantify the performance of each solver using three metrics. The first metric measures convergence progress (||w*||₀ significantly smaller than min(n,m) while still resulting in a difficult problem).

Table 2. Problem Instances for Sequential Comparisons. We choose λ = 0.05λ_max to select a desirable number of features.

| DATASET   | LOSS   | n   | m   | NNZ | ||w*||₀ |
|-----------|--------|-----|-----|-----|--------|
| FINANCE   | SQUARED| 1.6×10⁴ | 1.6×10⁶ | 9.2×10⁷ | 1419   |
| RCV1      | LOGISTIC| 2.0×10⁴ | 2.4×10⁶ | 6.2×10⁷ | 537    |

URL: http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/.

---

Footnotes:
3 We found the performance of GLMNET depends significantly on its termination threshold—even during early iterations. We run GLMNET using only its default stopping condition.
4 URL: http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/.
relative suboptimality:
\[ \left| g(w^*) - g(w_t) \right| / |g(w^*)|. \tag{22} \]

w* is approximated as the solution returned by BLITZ after solving to machine precision. We also plot precision and recall for nonzero weight variables wj. Define \( S^* = \{ j : w^*_j \neq 0 \} \) and \( S_t \) as the analogous support set for w_t. (For solvers that do not set values \( w_j \) to exactly 0, we take \( w_j \) to be nonzero i.f.f. \( |w_j| \geq 10^{-3} \).) We measure

\[
\text{Precision} = \frac{|S_t \cap S^*|}{|S_t|}, \quad \text{and} \quad \text{Recall} = \frac{|S_t \cap S^*|}{|S^*|}. \tag{23}
\]

Precision and recall are suitable metrics for \( \ell_1 \)-regularized learning, since \( \ell_1 \) regularization is most prominently used for feature selection, while generalization performance can be suppressed by coefficients overly biased toward zero.

Results of our comparison are included in Figure 4. Comparing BLITZ to its subproblem solver, PROXNEWT, as well as other methods without working sets, we see prioritizing computation provides extreme gains. With 8 threads, APPROX requires at least 6 minutes to solve a lasso problem that our sequential implementation of BLITZ completes in fewer than 30 seconds. Compared to other working set algorithms (GLMNET and LIBLINEAR), we see BLITZ still can be faster. While GLMNET and LIBLINEAR are highly optimized implementations, we see precision and recall results are superior for BLITZ, suggesting computation is better-focused on relevant features.

3.6. Limited Memory Comparison

Often datasets are too large to fit in the memory of a single machine. To solve (P4), one option is to load data multiple times from disk. While disk I/O becomes a bottleneck, BLITZ can be used to prioritize memory usage.

Applying BLITZ is straightforward in this setting if the set \( \{ A_j : w^*_j \neq 0 \} \) fits comfortably in memory. At each iteration, \( \tau \) is chosen such that the resulting subproblem includes as many features as memory limitations allow. Computing this \( \tau \) requires a single pass over the data. Each subproblem is then solved with (in-memory) BLITZ.

We compare this approach to three alternatives:

- **ADARDA** (Duchi et al., 2011): Stochastic gradient descent method with adaptive step-sizes. RDA is well-suited for \( \ell_1 \)-regularized learning (Xiao, 2010).
- **STRONG** (Tibshirani et al., 2012): Like BLITZ but features are prioritized according to the “Strong Rule.” Regularization is initialized to \( \lambda_{\text{MAX}} \) and decreased at each iteration until reaching the target \( \lambda \). **STRONG** uses (in-memory) BLITZ to solve subproblems.
- **CD**: A memory-limited coordinate descent implementation. \( A_j \) is loaded, (P4) is maximized with respect to \( w_j \), then memory for \( A_j \) is deallocated.

We implement each method in C++. To enable sequential loads, training data is stored on disk in compressed row format for ADARDA and compressed column format for all other methods. Data is stored in binary format and compressed with gzip. Our hardware is a 64-bit machine with 2.60 GHz Intel i5-4278U processors and a SATA HDD that achieves read rates of 100 MB/s.

We compare algorithms using the Webspam dataset from LIBSVM and logistic loss. This dataset contains \( 3.5 \times 10^5 \) examples, \( 6.8 \times 10^5 \) features, and \( 1.8 \times 10^9 \) nonzero entries. We set \( \lambda = 0.01\lambda_{\text{MAX}} \), resulting in 762 selected features. We normalize features to have unit variance. Under default compression, the dataset occupies approximately 12 GB. To emphasize the limited memory setting, we allow each algorithm use of just 1 GB memory.

Results of this experiment are included in Figure 5. BLITZ and **STRONG** greatly outperform alternative solvers that do not use more of the available memory. Clearly for some large problems, one need not settle for approximate solutions when the solution is sparse.
3.7. Distributed Comparison

For the largest problems, it is necessary to distribute data among many machines. Often distributed solvers for (P4) partition data by training examples and communicate gradient vectors of length m, the number of features, at each iteration. With m exceeding one billion in some industrial applications (Chen et al., 2014), communication becomes a bottleneck to optimization. In this setting, BLITZ can be used to drastically decrease the communication needed.

As a concrete example, consider a bulk synchronous proximal gradient descent implementation with data partitioned by examples. During an iteration, each node computes the gradient contribution of its local partition, and an $O(m)$ reduce operation aggregates these contributions to determine the global gradient. By solving subproblems with only $|C|$ features, BLITZ reduces the time complexity of this reduce operation to $O(|C|)$ per subproblem iteration. A “KKT filter” heuristic with similar motivation was recently proposed by Li et al. (2014). Communication of gradient values that are small in magnitude is prolonged until later iterations, which greatly improves convergence times.

We compare BLITZ with the KKT filter approach and a proximal gradient method with no prioritization of communication. The underlying solver for each method is an identical proximal gradient descent implementation which uses backtracking as detailed by Beck & Teboulle (2009) to ensure convergence. We implement this method in C++ using Rabit\(^5\), a reliable all-reduce communication library. The KKT filtering step is directly translated from the implementation of Li et al. (2014).

We compare methods using sparse logistic regression and the Criteo click-through rate dataset\(^6\). This dataset has $4.6 \times 10^7$ examples, $3.3 \times 10^7$ features, and $1.5 \times 10^9$ nonzero entries. We normalize features to have unit variance. Using $\lambda = 0.01\lambda_{\text{MAX}}$, the solution contains 5717 nonzero elements. We use 64 workers on 16 servers connected with 1 Gb/s networking. We approximate the optimal solution by running BLITZ for 200 minutes.

Results of this experiment are provided in Figure 6. By prioritizing communication, BLITZ and the KTT filtering method converge an order of magnitude faster than the naïve proximal gradient algorithm.

4. Discussion

$\ell_1$-regularized learning owes its popularity to the practical and statistical benefits of sparsity. In this work, we propose BLITZ, a method for exploiting sparsity during optimization. Unlike previous working set heuristics, BLITZ enables theoretically justified methods for choosing the contents, size, and stopping criteria of subproblems.

In several settings, BLITZ converges extremely quickly for $\ell_1$-regularized learning. Given such performance, it is important to consider additional problems for which BLITZ can work well. As a beginning, the analogy between constraint elimination (Corollary 2.3) and screening methods suggest BLITZ may work well for other applications for which screening has found traction (for example Wang et al. (2014)). It would also be interesting to consider more challenging objectives, including graphical lasso and problems with trace or total variation norms.

Another remaining challenge is to apply BLITZ to problems for which the constraint space is intractably large and cannot be enumerated. This includes structured prediction (Tschantaridis et al., 2005) and submodular minimization (Fujishige & Isotani, 2011). We view BLITZ as a very promising starting point for future work on these problems and large-scale machine learning in general.

---

5URL: https://github.com/tqchen/rabit.
Acknowledgments

The authors would like to thank the reviewers for their thoughtful suggestions as well as Joseph Bradley for his feedback on early versions of BLITZ. This work is supported in part by PECASE N00014-13-1-0023, NSF IIS-1258741, and the TerraSwarm Research Center 00008169.

References


We organize the appendices as follows:

- **Appendix A**: Proof of Theorem 2.1
- **Appendix B**: Proof of Corollary 2.2
- **Appendix C**: Alternative Proof and Generalized Version of Corollary 2.3
- **Appendix D**: Proof of Theorem 3.3
- **Appendix E**: Derivation of Dual Problem for $\ell_1$-Regularized Loss Minimization
- **Appendix F**: Examples of Computing $\text{dist}(h_j, y)$
- **Appendix G**: Remarks on Computing $\alpha$

### A. Proof of Theorem 2.1

**Theorem 2.1** (Convergence Progress at Iteration $t$). *Let $\Delta_t$ and $\Delta_{t+1}$ be the optimality gaps after iterations $t$ and $t+1$ of Algorithm 2. Then for all $t \geq 1$ if the algorithm does not converge at iteration $t+1$, we have*

$$\Delta_{t+1} \leq \Delta_t - \left(\frac{5}{2} \tau_t^2 \Delta_t^2\right)^{1/3}.$$  \hfill (6)

**Proof.** Note: throughout this proof, we use $\alpha$ to refer to $\alpha_{t+1}$ in order to simplify notation.

When $\alpha = 1$, we have

$$\Delta_{t+1} = f(y_{t+1}) - f(x_{t+1}) \leq f(x_t) - f(x_{t+1}).$$  \hfill (24)

This is because $C_{t+1}$ includes all constraints active at $x_t$, ensuring $f(x_{t+1}) \geq f(x_t)$. Thus, when $\alpha = 1$, the algorithm converges at iteration $t+1$, and the theorem holds.

To consider the case $\alpha < 1$, we begin by writing

$$\Delta_{t+1} = f(y_{t+1}) - f(x_{t+1}) = [f(y_{t+1}) - f(x_t)] + [f(x_t) - f(x_{t+1})].$$  \hfill (28)

Our approach is to bound these terms as functions of $\Delta_t$, $\tau_t$, and $\alpha$. We will eliminate $\alpha$ from this result by bounding over all $\alpha \in [0, 1]$.

**Bounding First Term in (28):** Because $f$ is strongly convex with parameter $\gamma$, we can write

$$f(y_{t+1}) = f(\alpha x_t + (1 - \alpha)y_t) \leq \alpha f(x_t) + (1 - \alpha)f(y_t) - \alpha(1 - \alpha)\frac{2}{\gamma} ||x_t - y_t||_2^2.$$  \hfill (30)

This implies

$$f(y_{t+1}) - f(x_t) \leq (1 - \alpha) [f(y_t) - f(x_t)] - \alpha(1 - \alpha)\frac{2}{\gamma} ||x_t - y_t||_2^2$$  \hfill (31)

$$= (1 - \alpha)\Delta_t - \alpha(1 - \alpha)\frac{2}{\gamma} ||x_t - y_t||_2^2.$$  \hfill (32)

Furthermore, since $y_{t+1} = \alpha x_t + (1 - \alpha)y_t$, we have

$$\alpha(1 - \alpha)\frac{2}{\gamma} ||x_t - y_t||_2^2 = \alpha(1 - \alpha)\frac{2}{\gamma} \left[\frac{||y_{t+1} - y_t||_2^2}{\alpha^2}\right].$$  \hfill (33)

$$\geq \frac{(1 - \alpha)}{\alpha} \frac{2}{\gamma} \tau_t^2.$$  \hfill (34)
Above, the inequality is true because \( \alpha < 1 \) implies \( \|y_{t+1} - y_t\|_2 \geq \tau_t \); there exists an \( h_j \notin C_t \) such that \( h_j(y_{t+1}) = 0 \), but since \( h_j \notin C_t \), no point on the boundary of \( h_j - y_{t+1} \) included—may be within a radius \( \tau_t \) of \( y_t \).

Combining (32) and (34), we have

\[
\Delta_t + \gamma \frac{(1 - \alpha)}{\alpha} \frac{\gamma^2}{2} \tau_t^2.
\]

Bounding the Second Term in (28): To bound the second term for the case that \( \alpha < 1 \), let \( h_j \) be the (possibly non-unique) constraint such that \( h_j(y_{t+1}) = 0 \) and \( h_j(x_t) > 0 \), and recall the definition

\[
\text{dist}(h_j, x_t) = \inf_{z : h_j(z) = 0} \|z - x_t\|_2.
\]

Because \( h(y_{t+1}) = 0 \), the set \( \{z : h_j(z) = 0\} \) is non-empty, and we can define \( z_t \) as a value of \( z \) that minimizes \( \|z - x_t\|_2 \) over this set. We have

\[
\text{dist}(h_j, x_t) = \|z_t - x_t\|_2 = \left\| z_t - \frac{1}{\alpha} (y_{t+1} - (1 - \alpha)y_t) \right\|_2
\]

\[
= \frac{1 - \alpha}{\alpha} \left\| \frac{-\alpha}{1 - \alpha} z_t + \frac{1}{1 - \alpha} y_{t+1} - y_t \right\|_2
\]

\[
\geq \frac{1 - \alpha}{\alpha} \tau_t.
\]

The last step is due to the convexity of \( h_j \) and the fact that \( h_j \notin C_t \) (otherwise we could not have \( h_j(x_t) > 0 \) since \( x_t \) is feasible for all constraints in \( C_t \)). Applying convexity of \( h_j \), we have

\[
h_j \left( \frac{-\alpha}{1 - \alpha} z_t + \frac{1}{1 - \alpha} y_{t+1} \right) \geq 0,
\]

since \( h_j(z_t) = h_j(y_{t+1}) = 0 \), and \( \frac{-\alpha}{1 - \alpha} + \frac{1}{1 - \alpha} = 1 \) with the first term being negative. The fact that this affine combination of \( z_t \) and \( y_{t+1} \) violates (or is tight at) \( h_j \) while \( y_t \) is feasible for \( h_j \) implies (40) since \( y_t \) is at least a distance \( \tau_t \) from the boundary \( \{z : h_j(z) = 0\} \) (since \( h_j \notin C_t \)).

We can use our bound on \( \text{dist}(h_j, x_t) \) to bound \( f(x_t) - f(x_{t+1}) \).

\[
\Delta_t \leq \Delta_t + \frac{(1 - \alpha)}{\alpha} \frac{\gamma^2}{2} \tau_t^2.
\]

Above the first inequality results from strong convexity. The second inequality requires an optimality conditions argument. In particular, \( x_t \) minimizes \( f \) subject to constraints \( \{h_j : h_j(x_t) = 0\} \), while \( x_{t+1} \) minimizes \( f \) subject to a superset of these constraints. This means \( x_{t+1} \) is feasible for the first problem and \( (x_{t+1} - x_t)^T \nabla f(x_t) \geq 0 \). Finally, the third inequality results from the fact that \( x_{t+1} \) cannot violate \( h_j \).

Completing the Proof: Adding (35) and (45), we have

\[
\Delta_t + \frac{(1 - \alpha)}{\alpha} \frac{\gamma^2}{2} \tau_t^2 = \left( 1 - \alpha \right) \Delta_t - \frac{1 - \alpha}{\alpha^2} \frac{\gamma^2}{2} \tau_t^2.
\]

It is worth noting that this partial result formalizes the main intuition for BLITZ. When \( \alpha \) is close to 1, \( y_t \) becomes close to \( x_{t-1} \) and the resulting suboptimality gap becomes small (via the left part of (48)). At the same time, if \( \alpha \) is close to 0,
there must exist an $h_j$ that is substantially violated by $x_{t-1}$. As a result, $f(x_t)$ improves significantly from $f(x_{t-1})$ and the resulting suboptimality gap again becomes small (this time via the right side of (48)).

We complete our proof by bounding (48) over all $\alpha \in [0, 1]$. A relatively simple bound is the following:

$$\Delta_{t+1} \leq \Delta_t - \left(\frac{2}{3} \gamma \tau_t^2 \Delta_t^2\right)^{1/3}. \quad (49)$$

This can be obtained by solving for $\alpha$ in

$$\alpha \Omega = \frac{1}{\alpha} \frac{2}{3} \gamma \tau_t^2$$

and then writing \(\Delta_{t+1} \leq (1 - \alpha')\Delta_t\) where $\alpha'$ is the solution from above.

B. Proof of Corollary 2.2

Corollary 2.2 (Linear Convergence). For $t \geq 1$, define

$$\Delta'_t = f(y_t) - f(x_{t-1}), \quad (7)$$

and suppose we run Algorithm 2 choosing $\tau_t$ as

$$\tau_t = \sqrt{\frac{2}{\gamma}(1 - r)^3 \Delta'_t} \quad (8)$$

for some $r \in [0, 1)$. Then for $t \geq 1$, we have

$$f(y_t) - f(x^*) \leq r^t - 1 \Delta_0. \quad (9)$$

Proof. The proof is a direct application of Theorem 2.1. However, since $\Delta_t$ is not known when selecting $\tau_t$, we instead use $\Delta'_t$ to upper-bound $\Delta_t$. (To see that $\Delta'_t$ upper-bounds $\Delta_t$, note that since all constraints that are tight at $x_{t-1}$ are included in the working set at iteration $t$, we have $f(x_t) \geq f(x_{t-1})$. Plugging into the definitions of $\Delta_t$ and $\Delta'_t$, we have $\Delta'_t \geq \Delta_t$.)

Applying Theorem 2.1 while choosing $\tau_t$ as in (8), we have

$$\Delta_t \leq \Delta_{t-1} - \left(\frac{1}{2} \gamma \tau_{t-1}^2 \Delta_{t-1}^2\right)^{1/3} \quad (51)$$

$$= \Delta_{t-1} - ((1 - r)^3 \Delta'_{t-1} \Delta_{t-1}^2)^{1/3} \quad (52)$$

$$\leq \Delta_{t-1} - (1 - r)\Delta_{t-1} \quad (53)$$

$$= r\Delta_{t-1}. \quad (54)$$

This completes our proof.

C. Alternative Proof and Generalized Version of Corollary 2.3

Corollary 2.3 immediately follows from Theorem 2.1. In this appendix, we present a simpler alternative proof. Furthermore, this proof leads to a more general constraint elimination rule. In particular, while Corollary 2.3 is assumed to be used with the BLITZ algorithm (and subproblems are assumed to be solved exactly), the more general rule can be applied with any feasible point $y$ and suboptimality gap $\Delta$.

Recall Corollary 2.3:

Corollary 2.3 (Constraint Elimination). For $t \geq 1$, define $\Delta'_t$ as in (7). If

$$\text{dist}(h_j, y_t) > \sqrt{\frac{2}{\gamma} \Delta'_t}, \quad (10)$$

then $h_j(x^*) < 0$, and $h_j$ may be eliminated from (P1).

Here we prove the following:
Theorem C.1 (FLEX Constraint Elimination). For (P1), let \( y \) be any feasible point and let \( \Delta \) be a suboptimality gap such that \( f(y) - f(x^*) \leq \Delta \). If
\[
\text{dist}(h_j, y) > \sqrt{\frac{2}{\gamma}} \Delta ,
\]
then \( h_j(x^*) < 0 \), and \( h_j \) may be eliminated from (P1).

Proof. By optimality conditions of (P1), we know
\[
(\nabla f(x^*), y - x^*) \geq 0 .
\]
By strong convexity of \( f \), we have
\[
f(y) \geq f(x^*) + (\nabla f(x^*), y - x^*) + \frac{\gamma}{2} \| y - x^* \|_2^2 \\
\geq f(x^*) + \frac{\gamma}{2} \| y - x^* \|_2^2 .
\]
Assume \( \text{dist}(h_j, y) > \sqrt{\frac{2}{\gamma}} \Delta \). This implies
\[
\| y - x^* \|_2^2 \leq \frac{\gamma}{2} [f(y) - f(x^*)] \\
\leq \frac{\gamma}{2} \Delta \\
< \text{dist}(h_j, y)^2 .
\]
We have shown \( \text{dist}(h_j, y) > \| y - x^* \|_2 \). By definition of \( \text{dist}(h_j, y) \), we must have \( h_j(x^*) < 0 \). Therefore, \( h_j \) is not active at the solution.

We note that in our experience, such screening/constraint elimination rules are rather conservative in general. This means that for many problems, few constraints are eliminated unless the problem is somehow easy to begin with (in our case, if the feasible point \( y \) is already close to the solution \( x^* \); in the \( \ell_1 \)-regularized learning case, screening rules perform best when the regularization \( \lambda \) is large).

As a result, for hard problems, we find it much more efficient to be aggressive eliminating constraints and then periodically reconsider constraints later. When reconsidering constraint \( h_j \) in BILTZ, we compute \( \text{dist}(h_j, y) \) to determine whether \( h_j \) should be added to \( C \). With \( \text{dist}(h_j, y) \) already computed, applying Theorem C.1 requires negligible additional computation.

D. Proof of Theorem 3.3

Theorem 3.3 (Progress for \( \ell_1 \) with Approximate Solver). For (P5), define \( \Delta_t \) as in (16), and assume \( x_t \) and \( w_t \) satisfy (17). If \( \alpha_{t+1} = 1 \), assume \( g(w_{t+1}) \geq g(w_t) \). If \( \alpha_{t+1} < 1 \), let \( h_j \) be the (possibly non-unique) constraint such that \( h_j(x_t) > 0 \) and \( h_j(y_{t+1}) = 0 \) and assume \( g(w_{t+1}) \geq \max_{\delta} g(w_t + \delta e_j) \). Then for \( t \geq 1 \), we have
\[
\Delta_{t+1} \leq \max \left\{ \Delta_t - \left( \frac{1}{2\epsilon^2}(1 - \epsilon_t)^2\tau_t^2\Delta_t^2 \right)^{1/3}, \epsilon_t \Delta_t \right\} .
\]

This proof is similar to the proof of Theorem 2.1. The main addition is the incorporation of partial subproblem solutions. The relation between \( x_t \) and \( w_t \) as defined in (15) is important, and for this reason, our proof applies only to the \( \ell_1 \)-regularized loss minimization problem and not the general setting of Theorem 2.1.

Like in the proof of Theorem 2.1, we use \( \alpha \) to refer to \( \alpha_{t+1} \). Note that when \( \alpha = 1 \), we have
\[
\Delta_{t+1} = f(y_{t+1}) - g(w_{t+1}) \\
= f(x_t) - g(w_{t+1}) \leq f(x_t) - g(w_t) \leq \epsilon_t (f(x_t) - f(y_t)) = \epsilon_t \Delta_t .
\]
Thus, when $\alpha = 1$, the theorem holds. For the remainder of the proof, we consider the case $\alpha < 1$. We write

$$
\Delta_{t+1} = f(y_{t+1}) - g(w_{t+1})
= f((1 - \alpha)y_t + \alpha x_t) - g(w_{t+1})
\leq (1 - \alpha)f(y_t) + \alpha f(x_t) - \frac{1}{2L}(1 - \alpha) \|x_t - y_t\|^2_2
\leq (1 - \alpha)[f(y_t) - g(w_t)] + \alpha[f(x_t) - g(w_t)] + [g(w_t) - g(w_{t+1})] - \frac{1}{2L}(1 - \alpha) \|x_t - y_t\|^2_2
\leq (1 - \alpha)\Delta_t + \alpha \epsilon_t \Delta_t + [g(w_t) - g(w_{t+1})] - \frac{1}{2L}(1 - \alpha) \|x_t - y_t\|^2_2
= (1 - \alpha)(1 - \epsilon_t)\Delta_t - \frac{1}{2L}(1 - \alpha) \|x_t - y_t\|^2_2 + [g(w_t) - g(w_{t+1})].
$$

The remaining steps of the proof bound the second and third terms of (72) as functions of $\alpha$ and $\tau_t$. We then achieve the final result by bounding over all $\alpha \in [0, 1]$.

For the second term of (72), we have

$$
\frac{1}{2L}(1 - \alpha) \|x_t - y_t\|^2_2 = \frac{1}{2L}\alpha(1 - \alpha) \left[ \frac{\|y_{t+1} - y_t\|^2_2}{\alpha^2} \right] \geq \frac{1 - \alpha}{\alpha} \frac{1}{2L}\tau_t^2.
$$

The inequality above results from the definition of $\alpha$, the condition $\alpha < 1$, and the definition of $\tau_t$ ($\|y_{t+1} - y_t\|^2_2$ must be at least $\tau_t$, otherwise $\alpha$ must be 1).

Now let us consider the third term of (72). Recall that $x_t = \xi_t \cdot p(Aw_t, b)$, where $p$ maps dual variables $w_t$ to the primal variables $x_t$ and $\xi_t \in [0, 1]$ scales this result toward 0 so that $x_t$ satisfies all constraints in $C_t$. Since $\alpha < 1$, there must be an $h_j$ such that $h_j(x_t) > 0$, $h_j(y_{t+1}) = 0$, and $h_j(x_{t+1}) \leq 0$. For this $h_j$, we have

$$
h_j(y_{t+1}) = 0
\Rightarrow |A_j^T y_{t+1}| - \lambda = 0
\Rightarrow |A_j^T (\alpha x_t + (1 - \alpha)y_t)| - \lambda = 0
\Rightarrow \alpha |A_j^T x_t| + (1 - \alpha) |A_j^T y_t| - \lambda \geq 0
\Rightarrow |A_j^T x_t| - \lambda \geq \frac{(1 - \alpha)}{\alpha} (\lambda - |A_j^T y_t|)
\Rightarrow \frac{|A_j^T x_t| - \lambda}{|A_j|_2} \geq \frac{(1 - \alpha)}{\alpha} \frac{\lambda - |A_j^T y_t|}{|A_j|_2}
\Rightarrow \frac{|A_j^T x_t| - \lambda}{|A_j|_2} \geq \frac{(1 - \alpha)}{\alpha} \tau_t.
$$

Above we have used the fact that $\text{dist}(h_j, y_t) = \frac{\lambda - |A_j^T y_t|}{|A_j|_2} \geq \tau_t$. Otherwise, $h_j$ would have been included in $C_t$, making $h_j(x_t) \leq 0$. Since $\xi_t \in [0, 1]$ we have

$$
\frac{|A_j^T p(Aw_t, b)| - \lambda}{|A_j|_2} \geq \frac{(1 - \alpha)}{\alpha} \tau_t.
$$

However, $A_j^T p(Aw_t, b)$ is also the derivative of the loss $\sum_i \phi_i(a_i^T w)$ with respect to $w_j$. Using standard coordinate
We complete our proof by bounding over all $\alpha$. Above, the second inequality comes from our assumption that $\phi_i$ is smooth. Combining (86) and (74) with (72), we have

\[
\Delta_{t+1} \leq (1 - \alpha(1 - \epsilon_t)) \Delta_t - \frac{(1 - \alpha)(1 - \epsilon_t)}{2\lambda} - \frac{(1 - \alpha)^2}{2L} \frac{\tau_t^2}{t}
\]

\[
= (1 - \alpha(1 - \epsilon_t)) \Delta_t - \frac{1 - \alpha}{\alpha^2} \frac{\tau_t^2}{t}
\]

\[
= \epsilon_t \Delta_t + (1 - \alpha) \left( (1 - \epsilon_t) \Delta_t - \frac{1}{\alpha^2} \frac{\tau_t^2}{2L} \right)
\]

We complete our proof by bounding over all $\alpha \in [0, 1]$. A relatively simple bound is the following:

\[
\Delta_{t+1} \leq \epsilon_t \Delta_t + \max_{\alpha \in [0, 1]} \min \left\{ (1 - \alpha)(1 - \epsilon_t) \Delta_t, \left( (1 - \epsilon_t) \Delta_t - \frac{1}{\alpha^2} \frac{\tau_t^2}{2L} \right) \right\}
\]

\[
= \max \left\{ \epsilon_t \Delta_t, \Delta_t - \left( \frac{1}{2L} (1 - \epsilon_t)^2 \frac{\tau_t^2}{2} \right)^{1/3} \right\}
\]

**E. Derivation of Dual Problem for $\ell_1$-Regularized Loss Minimization**

In this appendix, we derive the dual of the $\ell_1$-regularized learning problem from Section 3.

\[
\min_w \sum_{i=1}^n \phi_i(a_i^T w) + \lambda \|w\|_1 = \min_w \sum_{i=1}^n \phi_i^*(a_i^T w) + \lambda \|w\|_1
\]

\[
= \min_w \sum_{i=1}^n \max_{x_i} \left[ (a_i^T w)x_i - \phi_i^*(x_i) \right] + \lambda \|w\|_1
\]

\[
= \min_w \max_x - \sum_{i=1}^n \phi_i^*(x_i) + \langle Aw, x \rangle + \lambda \|w\|_1
\]

\[
= \max_x \min_w - \sum_{i=1}^n \phi_i^*(x_i) + \langle Aw, x \rangle + \lambda \|w\|_1
\]

\[
= \max_x - \sum_{i=1}^n \phi_i^*(x_i) + \min_w \langle Aw, x \rangle + \lambda \|w\|_1
\]

\[
= \max_x \sum_{i=1}^n - \phi_i^*(x_i)
\]

Note that $\phi_i^*$ refers to the conjugate function of $\phi_i$:

\[
\phi_i^*(x_i) = \max_v \langle v, x_i \rangle - f(v)
\]

We now derive this function for squared and logistic loss.
E.1. Conjugate Function for Squared Loss

\[
\max_v \langle v, x_i \rangle - \frac{1}{2} (v - b_i)^2 = -\frac{1}{2} b_i^2 + \max_v (x_i + b_i) v - \frac{1}{2} v^2
\]

\[
= -\frac{1}{2} b_i^2 + \frac{1}{2} (b_i + x_i)^2
\]

by setting \( v = x_i + b_i \).

E.2. Conjugate Function for Logistic Loss

We are looking to solve

\[
\max_v \langle v, x_i \rangle - \log(1 + \exp(-b_i v)).
\]

Differentiating, we have

\[
x_i + \frac{b_i \exp(-b_i v)}{1 + \exp(-b_i v)} = 0
\]

\[
\Rightarrow x_i = -\frac{b_i \exp(-b_i v)}{1 + \exp(-b_i v)}
\]

\[
\Rightarrow v = -\frac{1}{b_i} \log \left( \frac{-x_i}{x_i + b_i} \right).
\]

We can substitute this into (101) to obtain

\[
\phi^*(x_i) = -\frac{x_i}{b_i} \log \left( \frac{-x_i}{x_i + b_i} \right) - \log \left( 1 - \frac{x_i}{x_i + b_i} \right)
\]

\[
= -\frac{x_i}{b_i} \log \left( \frac{x_i}{b_i} \right) + \frac{x_i}{b_i} \log \left( 1 + \frac{x_i}{b_i} \right) - \log \left( 1 - \frac{x_i}{x_i + b_i} \right)
\]

\[
= -\frac{x_i}{b_i} \log \left( -\frac{x_i}{b_i} \right) + \left( 1 + \frac{x_i}{b_i} \right) \log \left( 1 + \frac{x_i}{b_i} \right).
\]

F. Examples of Computing \( \text{dist}(h_j, y) \)

In this appendix, we briefly include examples for evaluating \( \text{dist}(h_j, y) \).

F.1. Linear Constraints

The most common scenario is that \( h_j \) is linear. For some vector \( a \) and scalar \( b \), let

\[
h_j(x) = a^T x + b.
\]

In this case,

\[
\text{dist}(h_j, y) = \inf_{z : h_j(z) = 0} \| z - y \|_2
\]

\[
= \| (y + \mu a) - y \|_2
\]

\[
= |\mu| \| a \|_2,
\]

where the scalar \( \mu \) is such that

\[
h_j(y + \mu a) = \langle a, y \rangle + \mu \| a \|_2^2 + b = 0.
\]

This leaves us with

\[
\text{dist}(h_j, y) = \frac{\| \langle a, y \rangle + b \|}{\| a \|_2}.
\]
F.2. Constraints for $\ell_1$-Regularized Loss Minimization

When $h_j(x) = |A_j^T x| - \lambda$, the constraint $h_j$ can be viewed as the combination of two linear constraints:

$$h_j^+(x) = A_j^T x - \lambda, \quad \text{and} \quad h_j^-(x) = -A_j^T x - \lambda.$$  \hfill (114)

In the BLITZ algorithm, the fact $y$ is feasible implies $|A_j^T y| \leq \lambda$ and we have

$$\text{dist}(h_j, y) = \lambda - |A_j^T y|.$$  \hfill (116)

F.3. Spherical Constraints

dist$(h_j, y)$ is also easy to compute when $\{x : h_j(x) = 0\}$ is a sphere. Specifically, let

$$h_j(x) = a \|x - b\|^2_2 - c.$$  \hfill (117)

Assume $a > 0$ and also assume that $c \geq 0$ since $h_j(x) \leq 0$ could never be satisfied otherwise. The minimizer of $\|z - y\|_2$ subject to $h_j(z) = 0$ is given by

$$z^* = b + \mu(y - b),$$  \hfill (118)

where $\mu \geq 1$ is chosen such that $h_j(z^*) = 0$. More specifically, we have

$$a\mu^2 \|y - b\|^2_2 - c = 0 \quad \Rightarrow \quad \mu = \sqrt{\frac{1}{a \|y - b\|^2_2}}.$$  \hfill (120)

This implies

$$\|z^* - y\|_2 = (\mu - 1)\|b - y\|_2$$  \hfill (122)

$$= (\mu - 1) \|y - b\|_2$$  \hfill (123)

$$= \sqrt{\frac{c}{a}} - \|y - b\|_2.$$  \hfill (124)

F.4. Smooth Constraints

For arbitrary $h_j$, evaluating dist$(h_j, y)$ is potentially difficult. Despite $h_j$ being convex, minimizing $\|z - y\|_2$ subject to $h_j(z) = 0$ is not a convex problem in general due to the domain $\{z : h_j(z) = 0\}$.

The guarantees of BLITZ still hold, however, if we use a lower bound of dist$(h_j, y)$ when determining the working set. If the gradient of $h_j$ exists and is Lipschitz continuous with constant $L$, then obtaining a lower bound is straightforward. We can define

$$h_j'(x) = h_j(y) + (x - y)^T \nabla h_j(y) + \frac{L}{2} \|x - y\|^2_2.$$  \hfill (125)

$h_j'(x)$ upper-bounds $h_j(x)$ for all $x$. As a result, the set $\{x : h_j'(x) \leq 0\}$ is a subset of $\{x : h_j(x) \leq 0\}$, and we have

$$\text{dist}(h_j', y) \leq \text{dist}(h_j, y).$$  \hfill (126)

Evaluating dist$(h_j', y)$ is straightforward since $\{x : h_j'(x) = 0\}$ is a sphere.
G. Remarks on Computing $\alpha$

Here we briefly discuss how to compute $\alpha$. Recall that

$$\alpha = \max \{ \alpha' \in [0,1] : \alpha' x + (1 - \alpha') y \in D \} .$$  \hspace{1cm} (127)

That is, $\alpha$ is chosen such that $y = \alpha x + (1 - \alpha) y$ is the closest feasible point to $x$ on the line segment $[y, x]$. One way to find $\alpha$ is to define an $\alpha_j$ for each constraint $h_j$ as

$$\alpha_j = \max \{ \alpha' \in [0,1] : h_j(\alpha' x + (1 - \alpha') y) \leq 0 \} .$$  \hspace{1cm} (128)

Then we simply set

$$\alpha = \min_j \alpha_j .$$  \hspace{1cm} (129)

If $h_j(x) \leq 0$, then clearly $\alpha_j = 1$. Otherwise, for general $h_j$, evaluating (128) can be accomplished in logarithmic time using the bisection algorithm. For the common case that $h_j$ is linear, $\alpha_j$ can be computed in closed form:

$$h_j(\alpha_j x + (1 - \alpha_j) y) = 0$$  \hspace{1cm} (130)

$$\Rightarrow \alpha_j h_j(x) + (1 - \alpha_j) h_j(y) = 0$$  \hspace{1cm} (131)

$$\Rightarrow \alpha_j = \frac{-h_j(y)}{h_j(x) - h_j(y)} .$$  \hspace{1cm} (132)

Note that in BLITZ, $h_j(y) \leq 0$, and since any constraint for which $h_j(y) = 0$ is included in $\mathcal{C}$, it is always the case that $\alpha_j > 0$. 