On the Weight Density of L_2-Regularized Linear Classification and Regression

He-Zhe Lin
National Taiwan University
B07902028@csie.ntu.edu.tw

Zhi-Bao Lu
National Taiwan University
R12922196@ntu.edu.tw

Sheng-Wei Chen
National Taiwan University
D09944003@ntu.edu.tw

Cheng-Hung Liu
National Taiwan University
D079944009@ntu.edu.tw

Chih-Jen Lin
National Taiwan University
cjlin@csie.ntu.edu.tw
Mohamed bin Zayed University
of Artificial Intelligence
chihjen.lin@mbzuai.ac.ae

Abstract

For traditional linear models with the widely used L_2-regularizer, it is often assumed that the resulting models are dense. As a result, little attention has been paid to when the optimal solution for an L_2-regularized problem can actually be sparse. In this work, we rigorously prove that for L_2-regularized support vector classification/regression, the theoretical optimum can indeed be sparse when the data have sparse feature values. Surprisingly, we observe that some optimization methods fail to preserve this sparsity and instead produce fully dense numerical solutions, leading to unnecessary storage overhead. We explain this phenomenon through detailed analysis. In particular, we novelly show that certain coordinate descent methods naturally yield sparser numerical solutions compared to other optimization algorithms. By applying suitable algorithms that preserve numerical sparsity, the storage can be reduced by up to 50%, which is highly advantageous for large-scale industrial applications.

Proceedings of the 29[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2026, Tangier, Morocco. PMLR: Volume 300.

1 INTRODUCTION

Even in the deep learning era, linear models remain valuable in industrial applications due to their superior efficiency and simplicity (Lin et al., 2023), especially when dealing with high-dimensional data. This work focuses on L_2-regularized linear classification and regression problems with ℓ training instances:

$$\boldsymbol{w}^* \in \arg\min_{\boldsymbol{w}} f(\boldsymbol{w}) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{i=1}^{\ell}\xi(y_i\boldsymbol{w}^T\boldsymbol{x}_i), \quad (1)$$

where $\boldsymbol{x}_i \in \mathbb{R}^n$ is the feature vector and $y_i \in \{+1, -1\}$ is the corresponding label, C is the regularization parameter and $\xi(\cdot)$ is a convex loss function such as the hinge (or squared hinge) loss (Boser et al., 1992; Cortes and Vapnik, 1995) or the logistic loss. In large-scale applications, L_2-regularization is the most widely adopted setting for empirical risk minimization, due to its simplicity and differentiability (e.g., Zhu and Zou, 2007; Yuan et al., 2010; Chiang et al., 2018; Dedieu et al., 2022). It is also the default setting in famous machine learning packages such as scikit-learn (Pedregosa et al., 2011) and LIBLINEAR (Fan et al., 2008).

Throughout the machine learning community, it is widely accepted that, due to the smoothness of the $\boldsymbol{w}^T\boldsymbol{w}$ term, the *theoretical* optimal solution to an L_2-regularized problem is typically dense, which means that every coordinate of the optimal $\boldsymbol{w}^*$ is nonzero. Nevertheless, Moore and DeNero (2011) surprisingly obtained *numerical* solutions with exact zeros even when solving L_2-regularized problems. A subsequent work (Marafino et al., 2015) also observed sparsity but remarked that "*the precise reasons for these behavior*

are unclear." These observations challenge the common belief that L_2-regularization always leads to dense solutions.

To clarify, let us first introduce the following concepts:

- Theoretical optimal solution: the exact minimizer $\boldsymbol{w}^*$ of problem (1) in $\mathbb{R}^n$.
- Numerical solution: the output produced by an optimization algorithm when solving (1). Since each number is represented in finite-precision (e.g., four or eight bytes per value), this solution can only approximate the theoretical optimum. Besides, we say a value is *numerically zero* if it is the floating point zero defined in the IEEE standard.

With these concepts, we formulate the above issues as two research questions.

(RQ 1) Is the theoretical optimal solution to problem (1) fully dense?

(RQ 2) If the answer to (RQ 1) is yes, why did Moore and DeNero (2011) and Marafino et al. (2015) obtain numerical solutions with numerical zeros? If the answer to (RQ 1) is no, can the numerical zeros reliably reflect the zero components in the theoretical optimum $\boldsymbol{w}^*$?

Unfortunately, the unavailability of the theoretical optimum $\boldsymbol{w}^*$ makes these questions more subtle. Our preliminary results in Section 2 show that while some algorithms return fully dense numerical weight vectors, others produce solutions with numerical zeros. Despite all of these algorithms converging to the same unique optimum $\boldsymbol{w}^*$, this discrepancy in the number of zeros shows that numerical solutions alone cannot determine whether $\boldsymbol{w}^*$ contains zero components or not.

In this work, we thoroughly investigate the sparsity for L_2-regularized linear classification and extend to regression.[1] Moore and DeNero (2011) conflated the sparsity of the theoretical optimum $\boldsymbol{w}^*$ with that of the numerical solutions returned by algorithms. In contrast, our study carefully separates the two aspects:

- Theoretical sparsity: We prove that when the input data is sparse, the theoretical solution $\boldsymbol{w}^*$ to L_2-regularized problems may contain exact zero components. This result, together with our experiments, shows that some optimization methods fail to preserve this inherent sparsity, instead producing fully dense numerical solutions.
- Numerical sparsity: We investigate the mechanism by which specific algorithms yield sparse numerical solutions. Notably, these methods possess the capability of recovering zeros during the optimization process and the power of maintaining floating-point zero upon reaching the optimal point.

Our analysis of weight density is particularly beneficial in applications involving many binary classifiers. For instance, extreme multi-label classification tasks with millions of labels typically require solving an equally large number of binary classification problems (Prabhu et al., 2018; Khandagale et al., 2020; Yu et al., 2022; Zhang et al., 2023). In such settings, if the optimal solutions for the binary problems are inherently sparse, applying a sparse-preserving method can avoid storing the unnecessary nonzero components that appear only in numerical approximations, thus saving a large amount of space (McMahan et al., 2013; Lin et al., 2024). Our experiments show that the model size can be reduced by an average of 30%, and by up to 50% on larger data sets.

Paper organization In Section 2, we start with a surprising observation on the number of zeros in the numerical solutions. In Section 3, we extend the comparison to more solvers. We investigate the sparsity of the optimal $\boldsymbol{w}^*$ in Section 4, while in Section 5, we comprehensively explain the reasons for the results of Section 3. We present the experimental results on extreme multi-label classification in Section 6 and conclusions are in Section 7. A table of notations is provided in Appendix A. Experimental codes are available at `https://www.csie.ntu.edu.tw/~cjlin/papers/dual-space/`. This work is an extension of the second author's master thesis (Lu, 2025).

2 A SURPRISING OBSERVATION: THE WEIGHT DENSITY VARIES AMONG OPTIMIZERS

Conceptually, if we use any optimization algorithm with convergence guarantee to solve (1), it is expected that the approximate solution is close to the unique optimum $\boldsymbol{w}^*$ and has similar weight densities (i.e., the percentage of nonzero elements). However, it is surprising, if not unbelievable, that in some experiments we see significant differences on the density.

Let us give an example by considering a binary data set news20 with $n = 1,355,191$ features. The set is sparse, meaning that only a few among the n feature values are nonzeros for each instance. We take a popular linear classification package LIBLINEAR (Fan et al., 2008) to solve L2-loss SVM, which is (1) with the squared hinge loss:

$$\xi^{\mathsf{L2}}(y\boldsymbol{w}^T\boldsymbol{x}) = (\max(0, 1 - y\boldsymbol{w}^T\boldsymbol{x}))^2. \tag{2}$$

[1] For clarity, we focus on L_2-regularized L2-loss in the main paper and defer other classification/regression losses to Appendices G, H, and I.

Table 1: The weight density and distance between numerical solutions obtained by two optimization methods on the news20 data set. We compute nnz(·) by counting the number floating-point zeros in $\overline{\boldsymbol{w}}$.

Stopping Condition	default	strict
$\text{nnz}(\overline{\boldsymbol{w}}_{\text{Method 1}})/n$	100.0%	100.0%
$\text{nnz}(\overline{\boldsymbol{w}}_{\text{Method 2}})/n$	74.5%	74.5%
$\|\overline{\boldsymbol{w}}_{\text{Method 1}} - \overline{\boldsymbol{w}}_{\text{Method 2}}\|$	2×10^{0}	5×10^{-7}

LIBLINEAR offers two methods to solve the problem (details given in Section 3).

- Method 1: A Newton method to solve (1).
- Method 2: A coordinate descent method to solve the dual problem of (1).

We do the experiments using a default and a strict stopping condition (details given in Appendix C) and leave other parameters (e.g., $C = 1$ as the regularization parameter) unmodified. After the optimization is finished, we compute $\text{nnz}(\overline{\boldsymbol{w}})/n$, the percentage of the nonzeros in the approximate $\overline{\boldsymbol{w}}$. From Table 1, we see that even when $\overline{\boldsymbol{w}}_{\text{Method 1}} \approx \overline{\boldsymbol{w}}_{\text{Method 2}}$ under a strict stopping condition, the huge density gap remains.

In practice, numerical solutions may contain tiny floating point numbers, which are often treated as mathematical zeros and attributed to numerical error. However, in this work we count zeros in a numerical solution strictly based on exact floating-point zeros. The reason is that small numerical values are often indistinguishable: as we show in Appendix E, it is nearly impossible to determine whether such a value arises from the numerical error or genuinely appears in the theoretical optimum.

3 INVESTIGATION OF THE WEIGHT DENSITY ON MORE OPTIMIZERS

Following the observation in Section 2, this section expands the study of the numerical solutions under L2-loss SVM to more solvers and data sets. We extend the study of L1-loss, logistic loss, and support vector regression in Appendices G, H, and I.

3.1 Preliminary: Primal and Dual Problems

Problem (1) with L2-loss (2) is referred to as the primal problem of SVM. Alternatively, one can solve the dual problem of L2-loss SVM, given by

$$\begin{aligned} \min_{\boldsymbol{\alpha} \in \mathbb{R}^{\ell}} \quad & f_{\mathsf{dual}}(\boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{\alpha}^T \overline{Q}\boldsymbol{\alpha} - \boldsymbol{e}^T\boldsymbol{\alpha} \\ \text{subject to} \quad & \alpha_i \geq 0, \forall i, \end{aligned} \tag{3}$$

where $\boldsymbol{e} \in \mathbb{R}^{\ell}$ is the column vector of all ones and $\overline{Q} \in \mathbb{R}^{\ell \times \ell}$ with

$$Q_{iu} = \begin{cases} y_i y_u \boldsymbol{x}_i^T \boldsymbol{x}_u & \text{if } i \neq u, \\ y_i y_u \boldsymbol{x}_i^T \boldsymbol{x}_u + 1/(2C) & \text{if } i = u. \end{cases}$$

Suppose $\boldsymbol{\alpha}^*$ is any optimal solution to the dual problem.[2] An important property that connects $\boldsymbol{\alpha}^*$ with the primal optimal $\boldsymbol{w}^*$ of (1) is the primal-dual relationship, given by

$$\boldsymbol{w}^* = \sum_{i=1}^{\ell} \alpha_i^* y_i \boldsymbol{x}_i. \tag{4}$$

3.2 Introduction of the Experimented Solvers

We categorize the solvers into two classes, depending on whether the solution is obtained directly by solving the primal problem (1) or indirectly via solving the dual problem (3).

- Newton (Galli and Lin, 2022, primal-based, "Method 1" in Section 2): This method iteratively updates $\boldsymbol{w}$ by
$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \gamma \boldsymbol{d}, \tag{5}$$
where γ is a carefully chosen step size and $\boldsymbol{d}$ is close to the Newton direction $\nabla^2 f(\boldsymbol{w})^{-1} \nabla f(\boldsymbol{w})$.
- LBFGS (Liu and Nocedal, 1989, primal-based): This method approximates the Newton direction by a so called limited BFGS update. Once the direction is obtained, the way to change $\boldsymbol{w}$ is the same as (5).
- SGD (primal-based): At each step, this method randomly selects a subset of instances B to calculate a gradient direction that mimics the full gradient. The update rule is $\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \tilde{\boldsymbol{g}}_B$, where $\tilde{\boldsymbol{g}}_B$ is the stochastic gradient of (1)
$$\tilde{\boldsymbol{g}}_B = \boldsymbol{w} + \frac{C\ell}{|B|} \sum\nolimits_{i \in B} \nabla_{\boldsymbol{w}} \xi(y_i \boldsymbol{w}^T \boldsymbol{x}_i). \tag{6}$$
In the experiments, we set $|B| = 1$. As explained in Section 5.2, we consider such a rarely used setting because a large batch size somewhat hinders SGD from getting a sparse weight vector.

[2] For L2-loss, the optimal $\boldsymbol{\alpha}^*$ is unique; however, for L1-loss discussed in Appendix G, the problem (3) may have multiple optimal $\boldsymbol{\alpha}^*$.

- CDprimal (Chang et al., 2008, primal-based): This is a coordinate descent method solving the primal problem (1). At each step, the method selects a feature j and updates the corresponding weight component w_j by considering the one-variable subproblem:

$$\min_z D_j(z) = f(\boldsymbol{w} + z\boldsymbol{e}_j), \tag{7}$$

where $\boldsymbol{e}_j = \left[0 \cdots 0\ 1\ 0 \cdots 0\right]^T$ is the j-th standard unit vector in $\mathbb{R}^n$. Usually, we need to exactly solve the subproblem (7) to ensure convergence to the optimal solution $\boldsymbol{w}^*$. However, for L2-loss, (7) lacks a closed-form solution. To avoid a costly inner iterative procedure, Chang et al. (2008) ensure convergence by requiring only a sufficient decrease condition in each coordinate update. This is achieved by first computing a Newton direction at $z = 0$, given by

$$\frac{-D_j'(0)}{D_j''(0)}, \tag{8}$$

where $D_j'(0)$ and $D_j''(0)$ are the first and generalized[3] second derivative at $z = 0$. Then w_j is updated as

$$w_j \leftarrow w_j + \eta \frac{-D_j'(0)}{D_j''(0)}, \tag{9}$$

where η is determined by a line search to ensure the sufficient decrease condition.

- CDdual (Hsieh et al., 2008, dual-based, "Method 2" in Section 2): This is a highly efficient and popular coordinate descent method solving the dual problem (3). At each step, CDdual only updates the i-th coordinate of $\boldsymbol{\alpha}$, denoted by α_i, by solving the one-variable subproblem

$$\min_d f_{\mathsf{dual}}(\boldsymbol{\alpha} + d\boldsymbol{e}_i) \ \text{ subject to } \ \alpha_i + d \geq 0, \tag{10}$$

where $\boldsymbol{e}_i$ is the i-th standard unit vector in $\mathbb{R}^\ell$. By minimizing the function in (10) under the constraint $\alpha_i \geq 0$, the new α_i is given by

$$\alpha_i^{\text{new}} \leftarrow \max\left(\alpha_i - \frac{\nabla_i f_{\mathsf{dual}}(\boldsymbol{\alpha})}{\overline{Q}_{ii}}, 0\right). \tag{11}$$

After we obtain α_i^{new} by (11), $\boldsymbol{w}$ is updated by

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + (\alpha_i^{\text{new}} - \alpha_i) y_i \boldsymbol{x}_i. \tag{12}$$

Note that both $\boldsymbol{\alpha}$ and $\boldsymbol{w}$ vectors converge to optimal $\boldsymbol{\alpha}^*$ and $\boldsymbol{w}^*$, respectively. We summarize the overall procedure of CDdual in Algorithm 1.

[3]Note that $D_j(z)$ is not twice differentiable, so we follow Chang et al. (2008) to define the generalized second derivative.

[4]We do not apply SGD and CDprimal to reach the strict stopping condition because of the slow convergence compared to other methods. Details are in Appendix C.

Algorithm 1 CDdual for L2-loss SVM

– Given $\boldsymbol{\alpha}$ and $\boldsymbol{w} = \sum_{i=1}^{\ell} \alpha_i y_i \boldsymbol{x}_i$.

– While $\boldsymbol{\alpha}$ is not optimal

For $i = 1, \ldots, \ell$

1. $\nabla_i f_{\mathsf{dual}}(\boldsymbol{\alpha}) = y_i \boldsymbol{w}^T \boldsymbol{x}_i - 1 + D_{ii}\alpha_i$
2. $\alpha_i^{\text{new}} \leftarrow \max(\alpha_i - \nabla_i f_{\mathsf{dual}}(\boldsymbol{\alpha})/\overline{Q}_{ii}, 0)$
   $\boldsymbol{w} \leftarrow \boldsymbol{w} + (\alpha_i^{\text{new}} - \alpha_i) y_i \boldsymbol{x}_i$
   $\alpha_i \leftarrow \alpha_i^{\text{new}}$

Table 2: The density of $\overline{\boldsymbol{w}}$ obtained by each solver for L2-loss SVM. To avoid the effect of a rough solution, we apply a strict stopping condition.[4]

Methods	Netwon	LBFGS	SGD[4]	CDprimal[4]	CDdual
news20	100.0%	100.0%	78.4%	72.7%	74.5%
rcv1	100.0%	100.0%	72.0%	67.4%	69.1%
real-sim	100.0%	100.0%	96.5%	93.9%	95.2%
kdd2010a	100.0%	100.0%	70.9%	58.1%	59.1%
kdd2010b	100.0%	100.0%	71.3%	57.8%	58.5%
kdd2012	100.0%	100.0%	80.0%	75.4%	80.6%

3.3 Comparison of the Weight Densities

Table 2 compares the densities of the numerical solution returned from solvers described in Section 3.2. Data statistics and the experimental settings are provided in Appendices B.1 and C, respectively. Our observations are summarized as below.

(1) The solutions produced by Newton and LBFGS are consistently fully dense.

(2) SGD produced weights with some sparsity.

(3) CDprimal and CDdual achieve the highest sparsity among five solvers. Though CDprimal exhibit slightly lower density in Table 2, we show in Section 5.4 that both methods actually achieve comparable levels of sparsity.

The density discrepancy among solvers shows the need of studying whether the optimal solution to problem (1) is inherently sparse. If the answer is yes, it follows that certain optimization methods introduce redundant nonzero components. Therefore, we analyze the theoretical sparsity of $\boldsymbol{w}^*$ in Section 4 and explore the underlying causes for these observations in Section 5.

4 THEORETICAL ANALYSIS ON THE SPARSITY OF OPTIMAL $\boldsymbol{w}^*$

Unluckily, studying the sparsity of $\boldsymbol{w}^*$ is challenging because it is impossible to get the theoretical optimum. To address this difficulty, we begin with the optimality

condition of any dual solution $\boldsymbol{\alpha}^*$:

$$y_i(\boldsymbol{w}^*)^T\boldsymbol{x}_i > 1 \implies \alpha_i^* = 0.^5 \tag{13}$$

Let the set

$$I_{\boldsymbol{w}^*} = \{i \mid y_i(\boldsymbol{w}^*)^T\boldsymbol{x}_i > 1\} \tag{14}$$

contain the indices of the instances that satisfy the condition (13), so we have $\alpha_i^* = 0$ for all $i \in I_{\boldsymbol{w}^*}$. If a feature j only has nonzero values in $I_{\boldsymbol{w}^*}$, i.e.,

$$(\boldsymbol{x}_i)_j = 0, \forall i \notin I_{\boldsymbol{w}^*}, \tag{15}$$

then from the primal-dual relationship (4),

$$w_j^* = \sum_{i=1}^{\ell} y_i\alpha_i^*(\boldsymbol{x}_i)_j = \sum_{i\in I_{\boldsymbol{w}^*}} y_i\alpha_i^*(\boldsymbol{x}_i)_j = 0, \tag{16}$$

where the second and the last equalities are from (15) and (13). If this occurs, $\boldsymbol{w}^*$ is not fully dense. This fact has been observed in Moore and DeNero (2011). Therefore, it seems that we should find the set $I_{\boldsymbol{w}^*}$ and check whether there are features only appearing in $I_{\boldsymbol{w}^*}$. However, the set $I_{\boldsymbol{w}^*}$ is unavailable because the optimal $\boldsymbol{w}^*$ is unknown. Subsequently, we develop a proxy set so that we do not rely on $I_{\boldsymbol{w}^*}$.

For any $\overline{\boldsymbol{w}} \in \mathbb{R}^n$, we define

$$I_{\overline{\boldsymbol{w}}} = \{i \mid y_i\overline{\boldsymbol{w}}^T\boldsymbol{x}_i \geq 1 + \Delta\}, \tag{17}$$

where $\Delta > 0$ is a small pre-specified number. If $\overline{\boldsymbol{w}}$ is close to $\boldsymbol{w}^*$ (e.g., $\overline{\boldsymbol{w}}$ is a good approximate solution from any optimization algorithm), then $I_{\overline{\boldsymbol{w}}}$ is likely a subset of $I_{\boldsymbol{w}^*}$. We then prove the following theorem to connect $I_{\overline{\boldsymbol{w}}}$ and any dual optimal solution $\boldsymbol{\alpha}^*$. The proof is in Appendix D.1.

Theorem 4.1. *Suppose $\boldsymbol{w}^*$ is optimal to problem (1) under L2-loss. For any $\overline{\boldsymbol{w}} \in \mathbb{R}^n$, if*

$$\|\overline{\boldsymbol{w}} - \boldsymbol{w}^*\|\|\boldsymbol{x}_i\| < \Delta,\ \forall i \in I_{\overline{\boldsymbol{w}}}, \tag{18}$$

then $I_{\overline{\boldsymbol{w}}} \subseteq I_{\boldsymbol{w}^*}$, i.e.,

$$y_i(\boldsymbol{w}^*)^T\boldsymbol{x}_i > 1, \forall i \in I_{\overline{\boldsymbol{w}}}. \tag{19}$$

Let $\boldsymbol{\alpha}^*$ be any dual optimal solution to problem (3). We further have

$$\alpha_i^* = 0, \forall i \in I_{\overline{\boldsymbol{w}}}. \tag{20}$$

The importance of Theorem 4.1 is to identify a suitable $I_{\overline{\boldsymbol{w}}}$ for checking if $\boldsymbol{w}^*$ contains zero components. By a similar derivation to (16), we have:

$$\begin{aligned} &\text{If} \quad (\boldsymbol{x}_i)_j = 0, \forall i \notin I_{\overline{\boldsymbol{w}}}, \\ &\text{then} \quad w_j^* = \sum_{i=1}^{\ell} y_i\alpha_i^*(\boldsymbol{x}_i)_j = \sum_{i\in I_{\overline{\boldsymbol{w}}}} y_i\alpha_i^*(\boldsymbol{x}_i)_j = 0. \end{aligned} \tag{21}$$

[5] See the equation (15) of Boser et al. (1992) for details.

In particular, if $\overline{\boldsymbol{w}}$ is close to $\boldsymbol{w}^*$ and Δ is small, the proxy set $I_{\overline{\boldsymbol{w}}}$ may contain most elements in $I_{\boldsymbol{w}^*}$, making (21) akin to (16). However, Theorem 4.1 does not give a practically feasible detector because, without $\boldsymbol{w}^*$, we cannot check $\|\overline{\boldsymbol{w}} - \boldsymbol{w}^*\|$. To this end, we derive the following theorem with the proof given in Appendix D.2.

Theorem 4.2. *Consider the dual problem (3) under L2-loss. For any dual feasible $\overline{\boldsymbol{\alpha}} \in \mathbb{R}^\ell$ and any primal $\overline{\boldsymbol{w}} \in \mathbb{R}^n$, if*

$$\sqrt{2\left(f(\overline{\boldsymbol{w}}) + f_{\mathsf{dual}}(\overline{\boldsymbol{\alpha}})\right)}\|\boldsymbol{x}_i\| < \Delta,\ \forall i \in I_{\overline{\boldsymbol{w}}}, \tag{22}$$

then $I_{\overline{\boldsymbol{w}}} \subseteq I_{\boldsymbol{w}^*}$ and

$$\alpha_i^* = 0, \forall i \in I_{\overline{\boldsymbol{w}}},$$

where $\boldsymbol{\alpha}^*$ is any dual optimal solution to (3).

Theorem 4.2 characterizes the condition (22) for $\overline{\boldsymbol{\alpha}}$ to identify the corresponding $I_{\overline{\boldsymbol{w}}}$ set. Now we show how to find such an $\overline{\boldsymbol{\alpha}}$ and apply Theorem 4.2. Consider any iterative algorithm that solves the dual problem (3). We generate a sequence of approximate solutions $\{\boldsymbol{\alpha}^k\}_{k=0}^{\infty}$ throughout iterations. If the algorithm possesses good theoretical properties, $\{\boldsymbol{\alpha}^k\}_{k=0}^{\infty}$ might globally converge to a dual optimal solution $\boldsymbol{\alpha}^*$. However, we consider a weaker requirement that $\{\boldsymbol{\alpha}^k\}_{k=0}^{\infty}$ has at least one convergent subsequence $\{\boldsymbol{\alpha}^k\}_{k\in\mathcal{K}}$ approaching an optimal $\boldsymbol{\alpha}^*$. We define $\overline{\boldsymbol{w}}^k$ as follows to have

$$\overline{\boldsymbol{w}}^k \equiv \sum_{i=1}^{\ell} \alpha_i^k y_i\boldsymbol{x}_i \to \sum_{i=1}^{\ell} \alpha_i^* y_i\boldsymbol{x}_i = \boldsymbol{w}^*$$

as $k \to \infty, k \in \mathcal{K}$. The primal-dual relationship shows

$$f(\overline{\boldsymbol{w}}^k) + f_{\mathsf{dual}}(\overline{\boldsymbol{\alpha}}^k) \to f(\boldsymbol{w}^*) + f_{\mathsf{dual}}(\boldsymbol{\alpha}^*) = 0$$

as $k \to \infty, k \in \mathcal{K}$. Thus, there exists an iteration t such that

$$\sqrt{2\left(f(\overline{\boldsymbol{w}}^t) + f_{\mathsf{dual}}(\overline{\boldsymbol{\alpha}}^t)\right)}\|\boldsymbol{x}_i\| < \Delta,\ \forall i \in I_{\overline{\boldsymbol{w}}^t}.$$

We can use $I_{\overline{\boldsymbol{w}}^t}$ to check if (21) holds.

After the theorems are prepared, it is the moment to witness the sparsity of the unavailable $\boldsymbol{w}^*$ on real data sets. We begin with choosing any small positive value as Δ. To find an $\overline{\boldsymbol{\alpha}}$ and define $\overline{\boldsymbol{w}}$, we consider CDdual introduced in Section 3, which guarantees global convergence and the existence of an $(\overline{\boldsymbol{\alpha}}, \overline{\boldsymbol{w}})$ satisfying (22). After using (17) to identify the corresponding $I_{\overline{\boldsymbol{w}}}$ set, we count how many feature j's only appear in the $I_{\overline{\boldsymbol{w}}}$ set, i.e., $(\boldsymbol{x}_i)_j = 0,\ \forall \boldsymbol{x}_i \notin I_{\overline{\boldsymbol{w}}}$. By (21), the counting result is a lower bound on number of zeros in $\boldsymbol{w}^*$. Based on the lower bound, in Table 3, we compute the corresponding upper bound of the weight density of the theoretical $\boldsymbol{w}^*$. The results indicate that

Table 3: A practical use of Theorem 4.2 to obtain an upper bound of the weight density for the theoretical optimal $\boldsymbol{w}^*$ under L2-loss SVM, computed by $1 - (\text{lower bound on \# zeros})/n$. We pick $\Delta = 0.003$ to identify an $I_{\overline{\boldsymbol{w}}}$ set close to $I_{\boldsymbol{w}^*}$.

Data sets	Upper bound on $\boldsymbol{w}^*$'s density
news20	73.0%
rcv1	67.5%
real-sim	93.9%
kdd2010a	58.2%
kdd2010b	57.9%
kdd2012	76.0%

(1) with upper bounds $< 100\%$, the theoretical $\boldsymbol{w}^*$ of the experimented problems are sparse, and

(2) some methods (e.g., Newton) fail to preserve the sparsity of $\boldsymbol{w}^*$ and instead return much denser $\overline{\boldsymbol{w}}$'s.

5 INVESTIGATION: WHY DIFFERENT DENSITY AMONG DIFFERENT SOLVERS?

After confirming the theoretical sparsity of $\boldsymbol{w}^*$, we now revisit the observations given in the end of Section 3 and explain why these algorithms do or do not retain this sparsity.

5.1 Newton and LBFGS: No Sparsity due to Full Gradient Updates

Each iteration in Newton and LBFGS involves using the gradient to obtain the update direction. For L2-loss, the derivative $(\xi^{\mathsf{L2}})'(s) = 0$ for $s \geq 1$, so the gradient for $f(\boldsymbol{w})$ in (1) is given by

$$\begin{aligned}\nabla f(\boldsymbol{w}) &= \boldsymbol{w} + C\sum_{i=1}^{\ell} \left(\xi^{\mathsf{L2}}\right)'(y_i \boldsymbol{w}^T \boldsymbol{x}_i) y_i \boldsymbol{x}_i \\ &= \boldsymbol{w} - 2C \sum_{i:y_i \boldsymbol{w}^T \boldsymbol{x}_i < 1} (1 - y_i \boldsymbol{w}^T \boldsymbol{x}_i) y_i \boldsymbol{x}_i. \quad (23)\end{aligned}$$

In the beginning of the optimization process, it is expected that $\boldsymbol{w}$ does not effectively separate the two classes of data. Hence, many $(y_i, \boldsymbol{x}_i)$ pairs satisfy $y_i \boldsymbol{w}^T \boldsymbol{x}_i < 1$ and contribute to the gradient calculation in (23), making $\nabla f(\boldsymbol{w})$ rather dense. As $\nabla f(\boldsymbol{w})$ is utilized for updating $\boldsymbol{w}$, the weight vector would also be dense. Take the common setting of initializing $\boldsymbol{w} = \boldsymbol{0}$, as an example. In the first iteration, $(\xi^{\mathsf{L2}})'(\boldsymbol{w}^T \boldsymbol{x}_i) = (\xi^{\mathsf{L2}})'(0) \neq 0,\ \forall i$. Since all data are used to compute the gradient, $\boldsymbol{w}$ becomes very dense after the first iteration. Once a w_j becomes nonzero, in subsequent iterations, the accumulation of numerical errors may cause the approximate solution to have $\overline{w}_j \neq 0$ even if $w_j^* = 0$. The discussion suggests we should avoid using too many instances at once for updating $\boldsymbol{w}$.

5.2 SGD: Sparsity Through Using a Subset of Data

Compared to Newton and LBFGS, SGD only uses a subset of instances at each update. For the experiments in Section 3, we consider the extreme batch size of one, say $B = \{i\}$. Therefore, from (6), the update rule for SGD is given by

$$\begin{aligned}\boldsymbol{w}^{\text{new}} &= \boldsymbol{w} - \eta \tilde{\boldsymbol{g}}_B \quad (24)\\ &= (1-\eta)\boldsymbol{w} + 2C\ell\eta \max(1 - y_i \boldsymbol{w}^T \boldsymbol{x}_i, 0) y_i \boldsymbol{x}_i.\end{aligned}$$

As mentioned earlier, in the first few updates $\boldsymbol{x}_i$ is very likely misclassified and satisfies $y_i \boldsymbol{w}^T \boldsymbol{x}_i < 1$. In such a situation, whenever a feature $\boldsymbol{x}_j$ has nonzero values, it is added to the corresponding component in $\boldsymbol{w}$. However, as the optimization keeps going, $\boldsymbol{w}$ may start making accurate predictions for some $\boldsymbol{x}_i$'s. For these instances, since $y \boldsymbol{w}^T \boldsymbol{x} \geq 1$ holds, they do not contribute to update of $\boldsymbol{w}$ throughout the whole optimization process. If some features only appear in these unadded instances, the returned $\overline{\boldsymbol{w}}$ can have some sparsity.

To obtain a sparse $\overline{\boldsymbol{w}}$, it is better to quickly get a sufficiently good $\boldsymbol{w}$ without seeing too many instances. This explains why we set $|B| = 1$ in Section 3. If $|B|$ is enlarged, there would be more instances visited in the early updates, causing many irreversible nonzeros in the weight vector. For example, on the set rcv1, after one epoch (i.e., seeing all instances once), we see that the weight density grows as B becomes larger.

$\lvert B\rvert$	1	2	4	8	16	32	64
Density (%)	59.0	67.2	74.4	81.2	87.8	93.7	97.7

Besides the choice of $|B|$, other settings of SGD, such as learning rate scheduling and momentum, also jointly affect the convergence speed and weight density. Therefore, although ideally SGD can return a sparse $\overline{\boldsymbol{w}}$, it is complicated to find a suitable setting which is both time- and space-efficient in practice. Furthermore, a careless tuning may even make SGD fail to converge.

5.3 CDprimal: Sparsity Through Recovering Zeros

To explain why CDprimal achieves superior numerical sparsity compared to the methods analyzed in Section 5.1 and Section 5.2, let us focus on any feature-j satisfying (15), which implies $w_j^* = 0$ in the optimal solution $\boldsymbol{w}^*$. Even if w_j becomes nonzero during optimization or initialization, CDprimal guarantees that the

final w_j returns to exactly zero based on the following properties:

- **Zero-Recovery**: The CDprimal update rule is capable of resetting a nonzero w_j to exactly zero.
- **Numerical Stability**: Once $\boldsymbol{w}$ is sufficiently close to $\boldsymbol{w}^*$ and $w_j = 0$, in later iterations w_j does not become nonzero again.

Below we illustrate the ideas and provide theoretical statements in Appendix D.3.

Zero-Recovery To demonstrate how CDprimal resets w_j of a feature-j satisfying (15) to zero, we first follow Chang et al. (2008) to write the derivatives $D'_j(0)$ and $D''_j(0)$ in the update rule (9):

$$D'_j(0) = w_j - 2C \sum_{i:y_i \boldsymbol{w}^T \boldsymbol{x}_i < 1} y_i (\boldsymbol{x}_i)_j (1 - y_i \boldsymbol{w}^T \boldsymbol{x}_i) \quad (25)$$

and

$$D''_j(0) = 1 + 2C \sum_{i:y_i \boldsymbol{w}^T \boldsymbol{x}_i < 1} ((\boldsymbol{x}_i)_j)^2. \quad (26)$$

Suppose that at the current iteration, the following two conditions are satisfied:

(i) $(\boldsymbol{x}_i)_j = 0$ for all i such that $y_i \boldsymbol{w}^T \boldsymbol{x}_i < 1$, and

(ii) the line search accepts a full step $\eta = 1$.

Under condition (i), the second term in (25) and (26) is zero, so with condition (ii) the update rule (9) yields

$$w_j^{\text{new}} = w_j - \eta \frac{D'_j(0)}{D''_j(0)} = w_j - 1 \cdot \frac{w_j}{1} = 0. \quad (27)$$

It is easy to see that condition (i) holds once $\boldsymbol{w}$ is close to the optimal $\boldsymbol{w}^*$. Specifically, the global convergence of CDprimal eventually makes

$$y_i (\boldsymbol{w}^*)^T \boldsymbol{x}_i > 1 \implies y_i \boldsymbol{w}^T \boldsymbol{x}_i > 1. \quad (28)$$

Thus, from (15), we have

$$y_i \boldsymbol{w}^T \boldsymbol{x}_i \leq 1 \implies i \notin I_{\boldsymbol{w}^*} \implies (\boldsymbol{x}_i)_j = 0. \quad (29)$$

In Appendix D.3, we formally prove that condition (ii) also holds eventually.

Numerical Stability in Maintaining $w_j = 0$ Suppose feature-j satisfies (15) and w_j is updated to be zero. If $\boldsymbol{w}$ is sufficiently close to $\boldsymbol{w}^*$ so that (29) holds, then we use the form of $D'_j(0)$ in (25) to have

$$\begin{aligned} D'_j(0) &= w_j - 2C \sum_{i:y_i \boldsymbol{w}^T \boldsymbol{x}_i < 1} y_i (\boldsymbol{x}_i)_j (1 - y_i \boldsymbol{w}^T \boldsymbol{x}_i) \\ &= 0 - 0 = 0, \end{aligned}$$

Table 4: The weight density of CDprimal for L2-loss under zero and random initialization of $\boldsymbol{w}$.

Initialization of $\boldsymbol{w}$	$\boldsymbol{w} = \mathbf{0}$	Random
news20	72.7%	72.7%
rcv1	67.4%	67.4%
real-sim	93.9%	93.9%
kdd2010a	58.1%	58.0%
kdd2010b	57.8%	57.8%
kdd2012	75.4%	75.4%

implying that $z = 0$ is optimal for the sub-problem $D_j(z)$ in (7). Therefore, $w_j = 0$ is not changed.

Our experiment in Table 4 shows that, even when $\boldsymbol{w}$ is initialized as a nonzero vector, CDprimal still recovers the zero components in the optimal $\boldsymbol{w}^*$. The result comes from the update of one w component at a time and the update rule in approximately minimizing the one-variable sub-problem (7). In contrast, Newton, LBFGS and SGD lack such properties. Earlier results indicate that once w_j becomes nonzero after an update, it is nearly impossible to return it to an exact zero.

5.4 Dual-based Methods: Sparsity by Representing $\overline{w}$ with Support Vectors

Interestingly, CDdual possesses similar properties to CDprimal despite solving the dual problem (3).

Zero-Recovery Due to the constraint $\alpha_i \geq 0$ in the dual problem, any algorithm that is able to operate on the boundary of the feasible region is likely to assign α_i to zero during the optimization process. Projected gradient methods represent a common class of algorithms that operate on the boundaries, and CDdual falls into this category because its update rule includes a projection of α_i onto the interval $[0, \infty)$. Therefore, even if $\alpha_i > 0$ at a given stage of the optimization, as long as

$$\alpha_i - \frac{\nabla_i f_{\mathsf{dual}}(\boldsymbol{\alpha})}{\overline{Q}_{ii}} < 0, \quad (30)$$

the projection $\max(\cdot, 0)$ yields $\alpha_i^{\text{new}} = 0$. However, it is worth noting that not all methods solving the dual problem can reach the boundary points. For instance, interior point methods operate strictly within the interior of the feasible region. Therefore, the resulting $\boldsymbol{\alpha}$ cannot contain exact zeros.

Numerical Stability in Maintaining $\alpha_i = 0$ Theorem 2 in Hsieh et al. (2008) guarantees that after a sufficient number of iterations, the value $\overline{\alpha}_i$ returned by CDdual will be exactly zero whenever the optimal

Table 5: The percentage of exact zeros in the returned $\overline{\boldsymbol{\alpha}}$ under zero initialization (i.e., $\boldsymbol{\alpha} = \mathbf{0}$) and random initialization, computed by $(1 - \text{nnz}(\overline{\boldsymbol{\alpha}})/\ell)$.

Initialization of $\boldsymbol{\alpha}$	Zero	Random
news20	42.7%	42.7%
rcv1	65.3%	65.3%
real-sim	79.1%	79.1%
kdd2010a	54.8%	54.8%
kdd2010b	56.8%	56.8%
kdd2012	46.4%	46.4%

Table 6: The weight density of CDdual for L2-loss with/without recalculating the final $\overline{\boldsymbol{w}}$ based on (31).

Recalculation?	No		Yes	
Initialization	$\boldsymbol{\alpha} = \mathbf{0}$	Random	$\boldsymbol{\alpha} = \mathbf{0}$	Random
news20	74.5%	88.5%	72.7%	72.7%
rcv1	69.1%	76.0%	67.4%	67.4%
real-sim	95.2%	99.7%	93.9%	93.9%
kdd2010a	59.1%	61.1%	58.1%	58.1%
kdd2010b	58.5%	60.1%	57.8%	57.8%
kdd2012	80.6%	86.0%	75.4%	75.4%

solution satisfies $\alpha_i^* = 0$ and $\nabla_i f_{\text{dual}}(\boldsymbol{\alpha}^*) > 0$.[6]

Table 5 illustrates the effect of moving along the boundaries: even under a randomly initialized $\boldsymbol{\alpha}$ where there is no zero component, in the end, CDdual is still able to produce a similar proportion of zero $\overline{\alpha}_i$'s as the zero initialization does. Therefore, both initialization methods can have sparse weights via the primal-dual relationship.

"Recalculation" with the primal-dual relationship Despite the similar number of zeros in the returned $\overline{\boldsymbol{\alpha}}$'s under zero/random initialization, there is a non-negligible difference between the densities of the returned $\overline{\boldsymbol{w}}$'s from LIBLINEAR, as shown in Table 6 (the left two columns). Here we explain why.

Conceptually, after obtaining an approximate $\overline{\boldsymbol{\alpha}}$, the returned $\overline{\boldsymbol{w}}$ is constructed by

$$\overline{\boldsymbol{w}} = \sum_{i=1}^{\ell} \overline{\alpha}_i y_i \boldsymbol{x}_i. \tag{31}$$

However, in practical implementations (e.g., LIBLINEAR), a vector $\boldsymbol{w}$ is maintained by (12) for calculating $\nabla_i f(\boldsymbol{\alpha})$ efficiently. After the optimization is finished, the maintained $\boldsymbol{w}$ is directly returned as the approximate solution $\overline{\boldsymbol{w}}$ so that we do not need to calculate $\overline{\boldsymbol{w}}$ by (31) in the end. It turns out that this convenient and cost-saving way to get $\overline{\boldsymbol{w}}$ developed in Hsieh et al. (2008) may lead to denser solutions than a naive construction via (31). Let us consider an α_i that is nonzero at some point in the iterative process but becomes zero in the end. According to (31), $\boldsymbol{x}_i$ is not needed in the construction of $\overline{\boldsymbol{w}}$. However, along the way of updating α_i from nonzero to exact zero, numerical errors may occur in the subtraction $(\alpha_i^{\text{new}} - \alpha_i)$ in (12), causing $\boldsymbol{x}_i$ still be used even though it should not be.

To further reduce the density, we propose recalculating $\overline{\boldsymbol{w}}$ by (31) after the optimization. This recalculation ensures that those $\boldsymbol{x}_i$'s with corresponding $\overline{\alpha}_i = 0$ are not added into $\overline{\boldsymbol{w}}$. As shown in Table 6, applying this technique consistently reduces the weight densities across all data sets, regardless of the initial value of $\boldsymbol{\alpha}$.

CDprimal vs. CDdual By comparing the results in Tables 3, 4, and 6, we observe that the densities returned from CDprimal and CDdual (with recalculation) are very close to the upper bound of the theoretical weight density shown in Table 3. Despite the similarity in sparsity, CDdual reaches the same level of convergence faster than CDprimal, as noted in Hsieh et al. (2008). Furthermore, CDdual can also be applied to optimize the non-differentiable L1-loss, while CDprimal cannot. As a result, CDdual is preferred for achieving theoretical sparsity in large-scale applications.

Table 7: Model size for extreme multi-label classification using different solvers. The numbers in parentheses indicate the size relative to that of Newton.

	Newton	CDdual
rcv1v2	0.058 GB	0.026 GB (45 %)
EUR-Lex	1.136 GB	0.827 GB (72 %)
AmazonCat-13K	4.477 GB	2.053 GB (45 %)
Wiki10-31K	8.279 GB	4.600 GB (55 %)
Wiki-500K	299.283 GB	164.952 GB (55 %)
Amazon-670K	37.184 GB	20.439 GB (55 %)
Amazon-3M	515.195 GB	224.994 GB (44 %)

6 EXPERIMENTS ON MULTI-LABEL CLASSIFICATION

We demonstrate the practical impact of our study on extreme multi-label classification, which requires solving a large number of binary linear classification problems. After training, the resulting weight vectors must be stored as part of the final model. Our analysis of weight density shows that using an inappropriate optimization algorithm can result in numerical solutions containing significantly more non-zero values than the theoretical

[6] Note that the optimality condition implies that if $\alpha_i^* = 0$, then $\nabla_i f_{\text{dual}}(\boldsymbol{\alpha}^*) \geq 0$. Thus, in general $\nabla_i f_{\text{dual}}(\boldsymbol{\alpha}^*) > 0$ holds together with $\alpha_i^* = 0$.

Table 8: Performance comparison using different solvers. We report the precision scores for Newton. For CDdual, we show the score differences compared to Newton.

	Precision@1		Precision@3		Precision@5	
	Newton	CDdual	Newton	CDdual	Newton	CDdual
rcv1v2	95.78	+0.01	80.02	+0.01	55.94	+0.01
EUR-Lex	82.36	−0.03	69.06	+0.00	57.73	+0.00
AmazonCat-13K	93.01	+0.01	79.24	+0.00	64.44	+0.00
Wiki10-31K	84.45	−0.02	74.34	+0.03	65.54	+0.01
Wiki-500K	67.92	+0.00	48.48	−0.01	37.71	−0.01
Amazon-670K	44.11	+0.00	38.93	−0.01	35.08	+0.00
Amazon-3M	46.88	−0.01	44.08	−0.01	41.93	−0.01

optimum. As shown in the following experiments, these unnecessary non-zero entries lead to substantial waste in model storage.

6.1 Experimental Settings

We consider several multi-label data sets, with statistics given in Appendix B.2. For training, we use the package LibMultiLabel,[7] which relies on LIBLINEAR to solve the underlying binary classification problems. We compare the two optimization methods used in Section 2: Newton and CDdual. These methods are selected because they exhibit markedly different sparsity behaviors – Newton yields fully dense solutions, while CDdual achieves high sparsity in our earlier analysis. To reflect practical usage, we apply both methods using the default stopping condition in LIBLINEAR. For CDdual, we additionally perform weight recalculation using (31) to improve sparsity. Further experimental details are provided in Appendix F.

6.2 Model Size Comparison

In Table 7, we report the size of the models trained by Newton and CDdual. We use Newton as a baseline and report the model size ratios for CDdual because Newton produces fully dense solutions. For all data sets, CDdual greatly reduces the model size under L2-loss, which aligns with the results for binary data sets. For the largest data set Amazon-3M, CDdual even saves more than half of the model size compared to Newton.

6.3 Comparison of Testing Set Performance

While reducing the model size is beneficial, sparsity should not come at the expense of predictive performance. In Table 8, we compare the "Precision at 1, 3, 5" metrics to evaluate the performance of extreme multi-label models. These precision scores measure the proportion of correctly predicted labels among the top-ranked predictions and are standard evaluation metrics in multi-label classification. We observe that the performance difference between CDdual and Newton is minimal, as both methods optimize the same L2-loss. Since they converge to the same optimal solution, their final models are expected to perform similarly. These results suggest that selecting an appropriate optimization method – such as CDdual – can significantly reduce model size without compromising predictive accuracy.

[7] https://www.csie.ntu.edu.tw/~cjlin/libmultilabel/

7 CONCLUSIONS

In this work, we analyze how L_2-regularized SVMs can have sparse optimal solutions when the data set is sparse. Although some previous works (Moore and DeNero, 2011; Marafino et al., 2015) also observed this phenomenon, our work provides formal theorems to explain the underlying reasons. Moreover, we investigate why different optimizers yield different numerical densities and point out that the high sparsity achieved by CDprimal and CDdual stems from their ability to recover zeros and maintain floating-point zero upon reaching the optimal point. In practical applications, some linear methods for solving multi-label classification problems rely on L_2-regularized SVMs. Our work suggests that by selecting an optimizer that preserves theoretical sparsity (e.g., CDdual), we may reduce the model size by nearly half.

Acknowledgements

This work was supported in part by National Science and Technology Council of Taiwan grants NSTC-113-2222-E-002-005MY3 and NSTC-114-2634-F-002-007.

References

Boser, B. E., Guyon, I., and Vapnik, V. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Com-*

putational Learning Theory, pages 144–152. ACM Press.

Chang, K.-W., Hsieh, C.-J., and Lin, C.-J. (2008). Coordinate descent method for large-scale L2-loss linear SVM. *Journal of Machine Learning Research*, 9:1369–1398.

Chiang, W.-L., Lee, M.-C., and Lin, C.-J. (2016). Parallel dual coordinate descent method for large-scale linear classification in multi-core environments. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*.

Chiang, W.-L., Li, Y.-S., Lee, C.-P., and Lin, C.-J. (2018). Limited-memory common-directions method for distributed l1-regularized linear classification. In *Proceedings of SIAM International Conference on Data Mining (SDM)*.

Cortes, C. and Vapnik, V. (1995). Support-vector network. *Machine Learning*, 20:273–297.

Dedieu, A., Mazumder, R., and Wang, H. (2022). Solving L1-regularized SVMs and related linear programs: revisiting the effectiveness of column and constraint generation. *Journal of Machine Learning Research*, 23(1).

Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J. (2008). LIBLINEAR: a library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

Galli, L. and Lin, C.-J. (2022). A study on truncated Newton methods for linear classification. *IEEE Transactions on Neural Networks and Learning Systems*, 33(7):2828–2841.

Ho, C.-H. and Lin, C.-J. (2012). Large-scale linear support vector regression. *Journal of Machine Learning Research*, 13:3323–3348.

Hsieh, C.-J., Chang, K.-W., Lin, C.-J., Keerthi, S. S., and Sundararajan, S. (2008). A dual coordinate descent method for large-scale linear SVM. In *Proceedings of the Twenty Fifth International Conference on Machine Learning (ICML)*.

Khandagale, S., Xiao, H., and Babbar, R. (2020). Bonsai: diverse and shallow trees for extreme multi-label classification. *Machine Learning*, 109:2099–2119.

Lin, H.-Z., Liu, C.-H., and Lin, C.-J. (2024). Exploring space efciency in a tree-based linear model for extreme multi-label classication. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 16245–16260.

Lin, Y.-C., Chen, S.-A., Liu, J.-J., and Lin, C.-J. (2023). Linear classifier: An often-forgotten baseline for text classification. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1876–1888. Short paper.

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

Lu, Z.-B. (2025). On the weight density of l2-regularized linear classification and regression. Master's thesis, Department of Computer Science and Information Engineering, National Taiwan University.

Marafino, B. J., Boscardin, W. J., and Dudley, R. A. (2015). Efficient and sparse feature selection for biomedical text classification via the elastic net: Application to ICU risk stratification from nursing notes. *Journal of Biomedical Informatics*, 54:114–120.

McMahan, H. B., Holt, G., Sculley, D., Young, M., Ebner, D., Grady, J., Nie, L., Phillips, T., Davydov, E., Golovin, D., Chikkerur, S., Liu, D., Wattenberg, M., Hrafnkelsson, A. M., Boulos, T., and Kubica, J. (2013). Ad click prediction: a view from the trenches. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*.

Moore, R. and DeNero, J. (2011). L1 and L2 regularization for multiclass hinge loss models. In *Symposium on Machine Learning in Speech and Language Processing*.

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

Prabhu, Y., Kag, A., Harsola, S., Agrawal, R., and Varma, M. (2018). Parabel: Partitioned label trees for extreme classification with application to dynamic search advertising. In *Proceedings of the 2018 World Wide Web Conference (WWW)*, pages 993–1002.

Rockafellar, R. T. (1970). *Convex Analysis*. Princeton University Press, Princeton, NJ.

Shalev-Shwartz, S., Singer, Y., and Srebro, N. (2007). Pegasos: primal estimated sub-gradient solver for SVM. In *Proceedings of the Twenty Fourth International Conference on Machine Learning (ICML)*.

Vapnik, V. (1995). *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, NY.

Wang, P.-W., Lee, C.-P., and Lin, C.-J. (2019). The common-directions method for regularized empirical risk minimization. *Journal of Machine Learning Research*, 20(58):1–49.

Yu, H.-F., Huang, F.-L., and Lin, C.-J. (2011). Dual coordinate descent methods for logistic regression and maximum entropy models. *Machine Learning*, 85(1-2):41–75.

Yu, H.-F., Zhong, K., Zhang, J., Chang, W.-C., and Dhillon, I. S. (2022). PECOS: Prediction for enormous and correlated output spaces. *Journal of Machine Learning Research*, 23(98):1–32.

Yuan, G.-X., Chang, K.-W., Hsieh, C.-J., and Lin, C.-J. (2010). A comparison of optimization methods and software for large-scale l1-regularized linear classification. *Journal of Machine Learning Research*, 11:3183–3234.

Zhang, J., Wang, Y.-S., Chang, W.-C., Li, W., Jiang, J.-Y., Hsieh, C.-J., and Yu, H.-F. (2023). Build faster with less: A journey to accelerate sparse model building for semantic matching in product search. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pages 4960–4966.

Zhu, J. and Zou, H. (2007). Variable selection for the linear support vector machine. In Chen, K. and Wang, L., editors, *Trends in Neural Computation*, pages 35–59. Springer Berlin Heidelberg.

CHECKLIST

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. Yes.
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. Not Applicable: In this work, no new algorithm is presented, but we properly cite the works for all algorithms mentioned so readers can check the details in the original works.
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. No. We will release the experimental code after the publication.

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. Yes.
   (b) Complete proofs of all theoretical results. Yes. See the appendix for details.
   (c) Clear explanations of any assumptions. Yes.

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). Yes. We clearly state the source and tools to run the experiments. We will include the experimental codes after publication.
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). Yes. See appendix for details.
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). Yes. See appendix for details.
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). Yes. See appendix for details.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. Yes.
   (b) The license information of the assets, if applicable. Yes.
   (c) New assets either in the supplemental material or as a URL, if applicable. Not Applicable.
   (d) Information about consent from data providers/curators. Not Applicable.
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. Not Applicable.

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. Not applicable.
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. Not Applicable.
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. Not Applicable.

Supplementary: On the Weight Density of L_2-Regularized Linear Classification and Regression

A NOTATION

A list of notations used in this paper is presented in Table 9.

Table 9: Main notation

Notation	Description
l, n, L	the number of training instances, features, and labels
$\boldsymbol{x}$	feature vector
y	label for classification or target value for regression
$\xi(\cdot)$	convex loss function
$f(\cdot), f_{\mathsf{dual}}(\cdot)$	primal and dual objective functions
$\boldsymbol{w}$	weight vector
$\boldsymbol{\alpha}$	dual-variable vector
$w_j\ \forall j \in [n]$	j-th component of the weight vector $\boldsymbol{w}$
$\alpha_i\ \forall i \in [l]$	i-th dual variable corresponding to instance i
$\boldsymbol{w}^*, \overline{\boldsymbol{w}}$	optimal and approximate solutions of the primal problem
$\boldsymbol{\alpha}^*, \overline{\boldsymbol{\alpha}}$	optimal and approximate solutions of the dual problem

Table 10: Data statistics for binary data sets, ordered by the number of instances. The column "used features" indicates that features with zero values in the entire training set are removed. The density is defined by the average number of non-zeros per instance divided by the number of used features. The imbalanced ratio is the ratio of majority class to minority class.

Data sets	ℓ #instances	n #features	#used features	imbalanced ratio	density
news20	19,996	1,355,191	1,355,191	1.0002	0.0336%
rcv1	20,242	47,236	44,504	1.0759	0.1664%
real-sim	72,309	20,958	20,958	2.2516	0.2448%
kdd2010a	8,407,752	20,216,830	19,306,083	5.8031	0.0002%
kdd2010b	19,264,097	29,890,095	28,875,157	6.1762	0.0001%
kdd2012	119,705,032	54,686,452	50,333,432	21.4798	0.00002%

Table 11: Data statistics for multilabel data sets, ordered by the number of labels. The density is defined by the average number of non-zeros per instance divided by the number of features.

Data sets	L #labels	ℓ #instances	n #features	#used features	density
rcv1v2	101	23,149	47,236	47,152	0.1610%
EUR-Lex	3,956	15,449	186,104	186,103	0.1459%
AmazonCat-13K	13,330	1,186,239	203,882	203,542	0.0419%
Wiki10-31K	30,938	14,146	104,374	104,374	0.6606%
Wiki-500K	501,070	1,779,881	2,381,304	2,379,703	0.0163%
Amazon-670K	670,091	490,449	135,909	135,876	0.0557%
Amazon-3M	2,812,281	1,717,899	337,067	336,774	0.0146%

B DATA STATISTICS

B.1 Binary Data Sets

Table 10 lists the statistics for six binary data sets. Below we provide the specific links for the data sets from "LIBSVM Data Sets."[8]

- news20: `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary/news20.binary.bz2`
- rcv1: `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary/rcv1_train.binary.bz2`
- real-sim: `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary/real-sim.bz2`
- kdd2010a: `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary/kdda.bz2`
- kdd2010b: `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary/kddb.bz2`
- kdd2012: `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary/kdd12.tr.xz`

We note that for rcv1 and all kdd sets, some features are zero across all training instances. We refer to these as "useless features" because, for all the methods we consider, the corresponding $\boldsymbol{w}$ components are not updated at all. Since we initialize $\boldsymbol{w}$ as a zero vector for all methods, these components remain zero for the useless features, regardless of which algorithm we use. In Table 10, we indicate the "#used features" to specify the number of features that are not "useless features." Additionally, we consider density based on the number of used features rather than the number of original features.

[8] `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`

B.2 Multi-label Data Sets

Table 11 lists the statistics for multi-label data sets. We download Wiki-500K and Amazon-3M from `https://archive.org/download/pecos-dataset/inference-models/`, while other data sets were obtained from "LIBSVM Data: Multi-label Classification."[9] Multiple versions of TF-IDF feature vectors may be available, so we specify the data sets used through the following list of links.

- rcv1v2:
  - training: `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multilabel/rcv1_topics_train.svm.bz2`
  - testing: `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multilabel/rcv1_topics_test.svm.bz2`
- EUR-Lex
  - training: `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multilabel/eurlex_tfidf_train.svm.bz2`
  - testing: `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multilabel/eurlex_tfidf_test.svm.bz2`
- AmazonCat-13K:
  - training: `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multilabel/AmazonCat-13K_tfidf_train_ver1.svm.bz2`
  - testing: `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multilabel/AmazonCat-13K_tfidf_test_ver1.svm.bz2`
- Wiki10-31K:
  - training: `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multilabel/wiki10_31k_tfidf_train.svm.bz2`
  - testing: `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multilabel/wiki10_31k_tfidf_test.svm.bz2`
- Wiki-500K (training and testing): `https://archive.org/download/pecos-dataset/xmc-base/wiki-500k.tar.gz`
- Amazon-670K:
  - training: `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multilabel/Amazon-670K_tfidf_train_ver2.svm.bz2`
  - testing: `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multilabel/Amazon-670K_tfidf_test_ver2.svm.bz2`
- Amazon-3M (training and testing): `https://archive.org/download/pecos-dataset/xmc-base/amazon-3m.tar.gz`

C EXPERIMENTAL SETTINGS FOR TABLE 2

We list the settings of different optimization methods on the following.

- Newton: We use LIBLINEAR version 2.47.0. In LIBLINEAR, the Newton method iteratively updates $\boldsymbol{w}$ until the gradient is small enough

$$\|\nabla f(\boldsymbol{w})\| \leq \epsilon \frac{\min\{\#\text{pos}, \#\text{neg}\}}{\ell} \|\nabla f(\boldsymbol{w}^{(0)})\|, \tag{32}$$

  where ϵ is the tolerance parameter, # pos (respectively, # neg) refers to the number of positive (respectively, negative) instances, and $\boldsymbol{w}^{(0)}$ is the initial point. The default ϵ in (32) is 10^{-2} and we set $\epsilon = 10^{-8}$ for the strict condition.

- LBFGS: We use the LBFGS implementation from Wang et al. (2019) with the same stopping condition as (32).

- SGD: In the beginning, we tried several learning rate schedules, including a constant rate and the scheduling used in Pegasos (Shalev-Shwartz et al., 2007). However, these configurations failed to satisfy the stopping condition (32) with the default $\epsilon = 10^{-2}$. After doing some experiments, with learning rate $\eta = 0.1$, the update rule

$$\eta \leftarrow 0.9\eta \text{ after each iteration (i.e., every } \ell \text{ instances)},$$

  and the momentum setting

$$\begin{cases} \boldsymbol{v} \leftarrow \tilde{\boldsymbol{g}}_B + 0.9\boldsymbol{v} \\ \tilde{\boldsymbol{g}}_B \leftarrow \boldsymbol{v} \end{cases},$$

  we can reach the default stopping condition for small data sets. For the larger data sets, the training were terminated at the 100-iteration limit.

- CDprimal: We use the code[10] from Chang et al. (2008). Since some data sets did not reach the strict stopping condition within a reasonable timeframe, we capped the training at 3,000 iterations (each iteration corresponding to one cycle of updating all $\boldsymbol{w}$'s components). While smaller data sets met the strict stopping condition earlier, larger data sets were terminated at the 3,000-iteration limit.

- CDdual: We use LIBLINEAR version 2.47.0. Inside the while-loop of Algorithm 1, a sequence of $\boldsymbol{\alpha}$'s is generated, denoted by $\{\boldsymbol{\alpha}^{k,1}, \ldots, \boldsymbol{\alpha}^{k,\ell}\}$. The iteration is terminated when

$$\left(\max_i \nabla_i^P f_{\mathsf{dual}}(\boldsymbol{\alpha}^{k,i})\right) - \left(\min_i \nabla_i^P f_{\mathsf{dual}}(\boldsymbol{\alpha}^{k,i})\right) < \epsilon, \tag{33}$$

  where $\nabla_i^P f_{\mathsf{dual}}(\boldsymbol{\alpha})$ is the i-th component of the projected gradient, defined as

$$\nabla_i^P f_{\mathsf{dual}}(\boldsymbol{\alpha}) = \begin{cases} \nabla_i f_{\mathsf{dual}}(\boldsymbol{\alpha}) & \text{if } 0 < \alpha_i \\ \min(0, \nabla_i f_{\mathsf{dual}}(\boldsymbol{\alpha})) & \text{if } \alpha_i = 0. \end{cases}$$

  The default ϵ in (33) is 10^{-1} and we set $\epsilon = 10^{-8}$ for the strict condition. Throughout this paper for solving L2-loss SVM, except specified, we use a strict stopping condition for CDdual.

[10] https://www.csie.ntu.edu.tw/~cjlin/liblinear/cdl2paper/

D PROOFS

D.1 Proof of Theorem 4.1

Proof. Since (19) directly implies (20) from (13), it suffices to prove (19). Assume for contradiction that (19) does not hold. That is, there exists some $i \in I_{\overline{\boldsymbol{w}}}$ such that $y_i(\boldsymbol{w}^*)^T\boldsymbol{x}_i \leq 1$. Therefore,

$$y_i\overline{\boldsymbol{w}}^T\boldsymbol{x}_i = y_i(\boldsymbol{w}^*)^T\boldsymbol{x}_i + y_i(\overline{\boldsymbol{w}} - \boldsymbol{w}^*)^T\boldsymbol{x}_i \leq 1 + \|\overline{\boldsymbol{w}} - \boldsymbol{w}^*\|\|\boldsymbol{x}_i\| < 1 + \Delta,$$

where the last inequality is from (18). However, this contradicts the definition of $I_{\overline{\boldsymbol{w}}}$. □

D.2 Proof of Theorem 4.2

Let us separate f into two parts $f = F_1 + F_2$, where

$$F_1(\boldsymbol{w}) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} \text{ and } F_2(\boldsymbol{w}) = C\sum\nolimits_{i=1}^{l}\xi(y_i\boldsymbol{w}^T\boldsymbol{x}_i).$$

Suppose that $\boldsymbol{w}^*$ is optimal to problem (1). The optimality condition of $\boldsymbol{w}^*$ implies that

$$\mathbf{0} \in \partial f(\boldsymbol{w}^*), \tag{34}$$

where $\partial f(\boldsymbol{w}^*)$ is the subdifferential of f at $\boldsymbol{w}^*$ (i.e., the set of all subgradients at $\boldsymbol{w}^*$).[11] Besides, by Rockafellar (1970, Theorem 23.8),[12] we have

$$\partial f(\boldsymbol{w}^*) = \partial F_1(\boldsymbol{w}^*) + \partial F_2(\boldsymbol{w}^*) = \nabla F_1(\boldsymbol{w}^*) + \partial F_2(\boldsymbol{w}^*),$$

where the differentiability of $F_1(\boldsymbol{w})$ leads to

$$\partial F_1(\boldsymbol{w}) = \{\nabla F_1(\boldsymbol{w})\}.$$

Therefore, (34) implies

$$-\nabla F_1(\boldsymbol{w}^*) \in \partial F_2(\boldsymbol{w}^*). \tag{35}$$

Because F_1 is a quadratic function with the identity matrix $\mathcal{I}$ as the Hessian, the Taylor expansion of F_1 is

$$F_1(\boldsymbol{w}) = F_1(\boldsymbol{w}^*) + \nabla F_1(\boldsymbol{w}^*)^T(\boldsymbol{w} - \boldsymbol{w}^*) + \frac{1}{2}(\boldsymbol{w} - \boldsymbol{w}^*)^T\mathcal{I}(\boldsymbol{w} - \boldsymbol{w}^*). \tag{36}$$

Furthermore, since F_2 is convex function, for any $\boldsymbol{g} \in \partial F_2(\boldsymbol{w}^*)$, we have

$$F_2(\boldsymbol{w}) \geq F_2(\boldsymbol{w}^*) + \boldsymbol{g}^T(\boldsymbol{w} - \boldsymbol{w}^*). \tag{37}$$

Combining (36) and (37), we have for each $\boldsymbol{g} \in \partial F_2(\boldsymbol{w}^*)$,

$$f(\boldsymbol{w}) \geq F_1(\boldsymbol{w}^*) + F_2(\boldsymbol{w}^*) + (\nabla F_1(\boldsymbol{w}^*) + \boldsymbol{g})^T(\boldsymbol{w} - \boldsymbol{w}^*) + \frac{1}{2}\|\boldsymbol{w} - \boldsymbol{w}^*\|^2. \tag{38}$$

From (35), we take $\boldsymbol{g} = -\nabla F_1(\boldsymbol{w}^*)$ in (38) and get

$$f(\boldsymbol{w}) \geq f(\boldsymbol{w}^*) + \frac{1}{2}\|\boldsymbol{w} - \boldsymbol{w}^*\|^2 \quad \forall \boldsymbol{w}. \tag{39}$$

Consider any dual feasible point $\overline{\boldsymbol{\alpha}}$ and any dual optimal solution $\boldsymbol{\alpha}^*$. Because $\overline{\boldsymbol{\alpha}}$ is feasible in the dual problem (3),

$$-f_{\mathsf{dual}}(\boldsymbol{\alpha}^*) \geq -f_{\mathsf{dual}}(\overline{\boldsymbol{\alpha}}). \tag{40}$$

Now consider any primal $\overline{\boldsymbol{w}}$. After combining (39), the property of duality $f(\boldsymbol{w}^*) = -f_{\mathsf{dual}}(\boldsymbol{\alpha}^*)$, and (40), we have

$$\begin{aligned} 2\left(f(\overline{\boldsymbol{w}}) + f_{\mathsf{dual}}(\overline{\boldsymbol{\alpha}})\right) \geq & 2\left(f(\boldsymbol{w}^*) + \frac{1}{2}\|\overline{\boldsymbol{w}} - \boldsymbol{w}^*\|^2 + f_{\mathsf{dual}}(\boldsymbol{\alpha}^*)\right) \\ = & 2\left(-f_{\mathsf{dual}}(\boldsymbol{\alpha}^*) + \frac{1}{2}\|\overline{\boldsymbol{w}} - \boldsymbol{w}^*\|^2 + f_{\mathsf{dual}}(\boldsymbol{\alpha}^*)\right) = \|\overline{\boldsymbol{w}} - \boldsymbol{w}^*\|^2. \end{aligned}$$

And the above inequality implies that

$$\sqrt{2\left(f(\overline{\boldsymbol{w}}) + f_{\mathsf{dual}}(\overline{\boldsymbol{\alpha}})\right)} \geq \|\overline{\boldsymbol{w}} - \boldsymbol{w}^*\|. \tag{41}$$

Finally, from Theorem 4.1, we have the results.

[11] See `https://stanford.edu/class/ee364b/lectures/subgradients_slides.pdf`.

[12] See also `https://www.math.cuhk.edu.hk/course_builder/1920/math4230/Note8.pdf`

D.3 Proof of the Zero-Recovery Property of CDprimal

In Section 5.3, we illustrate that if $w_j^* = 0$ and (15) holds, the update rule of CDprimal (9) guarantees the updated $w_j = 0$ provided that $\boldsymbol{w}$ is sufficiently close to the optimal $\boldsymbol{w}^*$. In this subsection, we give a rigorous theoretical statement and its formal proof.

Theorem D.1. *Assume $w_j^* = 0$ and (15) holds. Suppose* CDprimal *generates a sequence $\{\boldsymbol{w}^k\} \to \boldsymbol{w}^*$. Then there exists k_0 such that*

(i) (Zero-Recovery) If $w_j^{k_1} \neq 0$ for some $k_1 > k_0$, the update at this step is

$$w_j^{k_1+1} = w_j^{k_1} - w_j^{k_1} = 0.$$

(ii) (Numerical Stability of $w_j = 0$) Further, $w_j^k = 0$ for all $k > k_1$.

Proof. To begin, we recall the definition of the $I_{\boldsymbol{w}^*}$ set (14)

$$I_{\boldsymbol{w}^*} = \{i \mid y_i(\boldsymbol{w}^*)^T \boldsymbol{x}_i > 1\}$$

and define

$$\delta \equiv \min_{i \in I_{\boldsymbol{w}^*}} (y_i(\boldsymbol{w}^*)^T \boldsymbol{x}_i - 1) > 0.$$

Using the global convergence of CDprimal, there exists a k' such that for $k \geq k'$, the following three conditions hold:

$$I_{\boldsymbol{w}^*} \subseteq \{i \mid y_i(\boldsymbol{w}^k)^T \boldsymbol{x}_i > 1\}, \tag{42}$$

$$\left|y_i(\boldsymbol{w}^k)^T \boldsymbol{x}_i - y_i(\boldsymbol{w}^*)^T \boldsymbol{x}_i\right| < \frac{\delta}{2}, \ \forall i, \text{ and} \tag{43}$$

$$\left|w_j^k (\boldsymbol{x}_i)_j\right| < \frac{\delta}{2}, \ \forall i, \tag{44}$$

where (42) and (43) follow from $\boldsymbol{w}^k \to \boldsymbol{w}^*$, and (44) follows from $w_j^k \to w_j^* = 0$. We claim such a k' can be the k_0 mentioned in the statement. To see this, if $w_j^{k_1} \neq 0$ for some $k_1 > k'$, then the update at the k_1-th iteration is given by

$$w_j^{k_1+1} = w_j^{k_1} - \eta \frac{D_j'(0)}{D_j''(0)}. \tag{45}$$

Note that (42) implies $\{i \mid y_i(\boldsymbol{w}^{k_1})^T \boldsymbol{x}_1 \leq 1\} \subseteq \{1, \ldots, \ell\} \setminus I_{\boldsymbol{w}^*}$, so with (15), we have

$$(\boldsymbol{x}_i)_j = 0 \text{ for } i \in \{i' \mid y_{i'}(\boldsymbol{w}^{k_1})^T \boldsymbol{x}_{i'} \leq 1\}.^{13} \tag{46}$$

In (45), we have

$$D_j'(0) = w_j^{k_1} - 2C \sum_{i: y_i(\boldsymbol{w}^{k_1})^T \boldsymbol{x}_i < 1} y_i (\boldsymbol{x}_i)_j (1 - y_i(\boldsymbol{w}^{k_1})^T \boldsymbol{x}_i) = w_j^{k_1} \tag{47}$$

and

$$D_j''(0) = 1 + 2C \sum_{i: y_i(\boldsymbol{w}^{k_1})^T \boldsymbol{x}_i < 1} ((\boldsymbol{x}_i)_j)^2 = 1, \tag{48}$$

where the last equalities in (47) and (48) are from (46). Hence, (45) becomes

$$w_j^{k_1+1} = w_j^{k_1} - \eta w_j^{k_1}. \tag{49}$$

To decide the step size η in (49), CDprimal conduct a line search from $\eta = 1, \beta, \beta^2, \ldots$, where $\beta \in (0, 1)$, until the following sufficient decrease condition is satisfied

$$D_j(-\eta w_j^{k_1}) - D_j(0) \leq -\sigma \eta^2 (w_j^{k_1})^2, \tag{50}$$

[13] This corresponds to condition (i) in Section 5.3.

where σ is any constant in $(0, 1/2)$. Therefore, to show the resulting $w_j^{k_1+1} = 0$, it suffices to show that $\eta = 1$ meets the sufficient decrease condition (50). By the definition of $D_j(\cdot)$, we have

$$D_j(-w_j^{k_1}) = \frac{1}{2}\left(\boldsymbol{w}^{k_1} - w_j^{k_1}\boldsymbol{e}_j\right)^T\left(\boldsymbol{w}^{k_1} - w_j^{k_1}\boldsymbol{e}_j\right) + C\sum_{i=1}^{\ell}\left(\max\left(0, 1 - y_i\left(\boldsymbol{w}^{k_1} - w_j^{k_1}\boldsymbol{e}_j\right)^T\boldsymbol{x}_i\right)\right)^2 \tag{51}$$

and

$$D_j(0) = \frac{1}{2}\left(\boldsymbol{w}^{k_1}\right)^T\left(\boldsymbol{w}^{k_1}\right) + C\sum_{i=1}^{\ell}\left(\max\left(0, 1 - y_i\left(\boldsymbol{w}^{k_1}\right)^T\boldsymbol{x}_i\right)\right)^2. \tag{52}$$

For calculating (51) − (52), we have

$$\begin{aligned}
&D_j(-w_j^{k_1}) - D_j(0)\\
&= \frac{1}{2}\left(\boldsymbol{w}^{k_1} - w_j^{k_1}\boldsymbol{e}_j\right)^T\left(\boldsymbol{w}^{k_1} - w_j^{k_1}\boldsymbol{e}_j\right) - \frac{1}{2}\left(\boldsymbol{w}^{k_1}\right)^T\left(\boldsymbol{w}^{k_1}\right)\\
&\quad + C\sum_{i=1}^{\ell}\left[\left(\max\left(0, 1 - y_i\left(\boldsymbol{w}^{k_1} - w_j^{k_1}\boldsymbol{e}_j\right)^T\boldsymbol{x}_i\right)\right)^2 - \left(\max\left(0, 1 - y_i\left(\boldsymbol{w}^{k_1}\right)^T\boldsymbol{x}_i\right)\right)^2\right]\\
&= -\frac{(w_j^{k_1})^2}{2} + C\sum_{i=1}^{\ell}\left[\left(\max\left(0, 1 - y_i\left(\boldsymbol{w}^{k_1} - w_j^{k_1}\boldsymbol{e}_j\right)^T\boldsymbol{x}_i\right)\right)^2 - \left(\max\left(0, 1 - y_i\left(\boldsymbol{w}^{k_1}\right)^T\boldsymbol{x}_i\right)\right)^2\right]\\
&= -\frac{(w_j^{k_1})^2}{2} + C\sum_{i\in I_{\boldsymbol{w}^*}}\left[\left(\max\left(0, 1 - y_i\left(\boldsymbol{w}^{k_1} - w_j^{k_1}\boldsymbol{e}_j\right)^T\boldsymbol{x}_i\right)\right)^2 - \left(\max\left(0, 1 - y_i\left(\boldsymbol{w}^{k_1}\right)^T\boldsymbol{x}_i\right)\right)^2\right]\\
&\quad + C\sum_{i\notin I_{\boldsymbol{w}^*}}\left[\left(\max\left(0, 1 - y_i\left(\boldsymbol{w}^{k_1} - w_j^{k_1}\boldsymbol{e}_j\right)^T\boldsymbol{x}_i\right)\right)^2 - \left(\max\left(0, 1 - y_i\left(\boldsymbol{w}^{k_1}\right)^T\boldsymbol{x}_i\right)\right)^2\right].
\end{aligned} \tag{53}$$

In the following, we respectivly consider the case of $i \in I_{\boldsymbol{w}^*}$ and $i \notin I_{\boldsymbol{w}^*}$.

- If $i \in I_{\boldsymbol{w}^*}$, then according to the definition of $I_{\boldsymbol{w}^*}$, $1 - y_i(\boldsymbol{w}^*)^T\boldsymbol{x}_i < 0$. Further, by the definition of δ, (43) and (44), we have

$$\begin{aligned}
1 - y_i\left(\boldsymbol{w}^{k_1} - w_j^{k_1}\boldsymbol{e}_j\right)^T\boldsymbol{x}_i &= \left(1 - y_i\left(\boldsymbol{w}^*\right)^T\boldsymbol{x}_i\right) + \left(y_i\left(\boldsymbol{w}^*\right)^T\boldsymbol{x}_i - y_i\left(\boldsymbol{w}^{k_1}\right)^T\boldsymbol{x}_i\right) + y_i w_j^{k_1}(\boldsymbol{x}_i)_j\\
&< -\delta + \frac{\delta}{2} + \frac{\delta}{2} = 0.
\end{aligned}$$

  Therefore, the max function returns zero and we have

$$\sum_{i\in I_{\boldsymbol{w}^*}}\left[\left(\max\left(0, 1 - y_i\left(\boldsymbol{w}^{k_1} - w_j^{k_1}\boldsymbol{e}_j\right)^T\boldsymbol{x}_i\right)\right)^2 - \left(\max\left(0, 1 - y_i\left(\boldsymbol{w}^{k_1}\right)^T\boldsymbol{x}_i\right)\right)^2\right] = 0.$$

- If $i \notin I_{\boldsymbol{w}^*}$, by (15) we have $(\boldsymbol{x}_i)_j = 0$ and therefore

$$y_i\left(\boldsymbol{w}^{k_1} - w_j^{k_1}\boldsymbol{e}_j\right)^T\boldsymbol{x}_i = y_i(\boldsymbol{w}^{k_1})^T\boldsymbol{x}_i - y_i w_j^{k_1}(\boldsymbol{x}_i)_j = y_i\left(\boldsymbol{w}^{k_1}\right)^T\boldsymbol{x}_i.$$

  Hence, we have

$$\sum_{i\notin I_{\boldsymbol{w}^*}}\left[\left(\max\left(0, 1 - y_i\left(\boldsymbol{w}^{k_1} - w_j^{k_1}\boldsymbol{e}_j\right)^T\boldsymbol{x}_i\right)\right)^2 - \left(\max\left(0, 1 - y_i\left(\boldsymbol{w}^{k_1}\right)^T\boldsymbol{x}_i\right)\right)^2\right] = 0.$$

Combining the above items, (53) becomes

$$D_j(-w_j^{k_1}) - D_j(0) = -\frac{(w_j^{k_1})^2}{2} + 0 + 0 \le -\sigma(w_j^{k_1})^2.$$

Thus, the sufficient decrease condition holds at $\eta = 1$. This finishes the proof of the update in (i). The proof of (ii) has been provided in Section 5.3. □

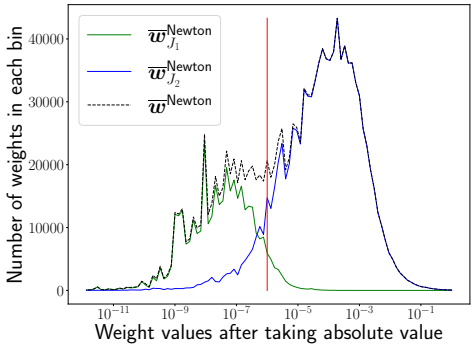

Figure 1: The distribution of the weight values in an approximate solution $\overline{\boldsymbol{w}}^{\mathsf{Newton}}$ by solving the primal problem. The green (respectively, blue) line is the distribution for $\overline{\boldsymbol{w}}^{\mathsf{Newton}}_{J_1}$ (respectively, $\overline{\boldsymbol{w}}^{\mathsf{Newton}}_{J_2}$). The red line corresponds to the weight value 10^{-6}.

E ON THE INFEASIBILITY OF TREATING SMALL NUMERICAL VALUES AS ZEROS

Throughout this work, we count zeros in numerical solutions strictly based on the floating-point zero defined in the IEEE standard. A natural question is whether one could instead treat very small numerical values as zero. For example, one might remove components in $\overline{\boldsymbol{w}}$ whose absolute values are below a threshold ϵ.

Based on the analysis in Section 5.4, we expect that the weights returned by CDdual may provide hints about the zero components in the theoretical optimum $\boldsymbol{w}^*$, since the numerical sparsity observed in CDdual closely reflects the theoretical sparsity of $\boldsymbol{w}^*$. Ideally, one might attempt to prune small values from the fully dense numerical solution $\overline{\boldsymbol{w}}^{\mathsf{Newton}}$ so that only entries corresponding to nonzero positions in $\overline{\boldsymbol{w}}^{\mathsf{CDdual}}$ are retained. To examine this possibility, we partition the feature indices $\{1, \ldots, n\}$ into two disjoint sets:

- $J_1 = \{j \mid \overline{w}_j^{\mathsf{CDdual}} = 0\}$, corresponding to the components whose values in $\overline{\boldsymbol{w}}^{\mathsf{Newton}}$ should ideally be removed.
- $J_2 = \{j \mid \overline{w}_j^{\mathsf{CDdual}} \neq 0\}$.

Figure 1 shows the histograms of the weight values for the two subsets $\overline{\boldsymbol{w}}^{\mathsf{Newton}}_{J_1}$ and $\overline{\boldsymbol{w}}^{\mathsf{Newton}}_{J_2}$. We observe that most values in $\overline{\boldsymbol{w}}^{\mathsf{Newton}}_{J_1}$ are smaller than those in $\overline{\boldsymbol{w}}^{\mathsf{Newton}}_{J_2}$, but the two distributions still overlap substantially. Hence, it is impossible to perfectly remove weights in $\overline{\boldsymbol{w}}^{\mathsf{Newton}}_{J_1}$ without also discarding some weights in $\overline{\boldsymbol{w}}^{\mathsf{Newton}}_{J_2}$. Moreover, selecting an appropriate threshold ϵ is infeasible without knowing J_1 and J_2.

In summary, treating small numerical values as zero cannot faithfully reproduce the sparsity pattern of the theoretical optimal $\boldsymbol{w}^*$. In contrast, CDdual inherently preserves this sparsity without any post-hoc processing.

Algorithm 2 A framework of parallel CDdual in LIBLINEAR-multicore (Chiang et al., 2016)

– Given $\boldsymbol{\alpha}$ and $\boldsymbol{w} = \sum_{i=1}^{\ell} \alpha_i y_i \boldsymbol{x}_i$.

– While $\boldsymbol{\alpha}$ is not optimal

1. Select a batch $\overline{B}$
2. Calculate $\nabla_{\overline{B}} f_D(\boldsymbol{\alpha})$ in parallel
3. Select $B \subset \overline{B}$ with $|B| \ll |\overline{B}|$
4. Update $\alpha_i, i \in B$ sequentially as in Algorithm 1

Table 12: The density of weights for Newton and CDdual with single and multicore for L2-loss SVM, with default setting in LIBLINEAR-multicore. The multicore setting applies 16 cores to train each binary classifier.

	Newton		CDdual	
	Single	Multi	Single	Multi
news20	100.0%	100.0%	74.3%	74.3%
rcv1	100.0%	100.0%	69.0%	69.0%
real-sim	100.0%	100.0%	95.3%	95.3%
kdd2010a	100.0%	100.0%	59.1%	59.1%
kdd2010b	100.0%	100.0%	58.5%	58.5%
kdd2012	100.0%	100.0%	79.7%	79.7%

F EXPERIMENTAL SETTINGS FOR LIBMULTILABEL

In this section we provide details of our LibMultiLabel setting not mentioned in the main context.

The tree-based method in LibMultiLabel originates from the method introduces in Khandagale et al. (2020), which uses hierarchical K-means to construct the label tree. Every path from a parent node to a child node corresponds to a binary classification problem. Because the K-means algorithm involves randomness, causing different sets of binary classification problems, we repeat the model training five times using various random seeds on rcv1v2, EUR-Lex, AmazonCat-13K, Wiki10-31K and Amazon-670K. We report their average in Table 7 and Table 8. For the two large data sets Wiki-500K and Amazon-3M, we only conduct the experiments for one seed due to resource limitation. We apply 5-fold cross validation to select the regularization parameter C for all binary problems in the tree model.

Besides, LibMultiLabel uses the package LIBLINEAR-multicore to speed up the optimization for CDdual. Its general framework is shown in Algorithm 2. One may wonder if the weight density obtained from Algorithm 2 is different from the weight density using Algorithm 1. We now explain that the multi-core implementation keeps the weight density the same. The implementation of LIBLINEAR-multicore starts by first splitting all instance $\{1, \ldots, \ell\}$ into several batches. Within each batch, it calculates $\nabla_{\overline{B}} f_D(\boldsymbol{\alpha})$ for a batch $\overline{B}$ in parallel, and then selects a subset $B \subset \overline{B}$. Once the update batch B is determined, the update for α_i is carried out sequentially for $i \in B$. Since the parallelization is only for the calculation of gradients, the returned weights remain the same, regardless a single/multi core implementation.

In Table 12 we show the weight density using both single and multi-core configurations for L2-loss SVM, with the default setting in LIBLINEAR-multicore. For the multi-core setup, we assign half of the available CPU cores to each binary problem. The results show that the parallelization of CDdual does not affect the weight density.

G EXTENSION TO L1-LOSS SUPPORT VECTOR CLASSIFICATION

Due to space limitation, we only focus on the L2-loss in the main paper. In this section, we extend our results to L1-loss (hinge loss)

$$\xi^{\mathsf{L1}}(y\boldsymbol{w}^T\boldsymbol{x}) = \max(0, 1 - y\boldsymbol{w}^T\boldsymbol{x}). \tag{54}$$

The optimization problem is given by

$$\min_{\boldsymbol{w}} f(\boldsymbol{w}) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{i=1}^{\ell}\xi^{\mathsf{L1}}(y_i\boldsymbol{w}^T\boldsymbol{x}_i), \tag{55}$$

and its dual problem is given by

$$\min_{\boldsymbol{\alpha}} f_{\mathsf{dual}}(\boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{\alpha}^T Q\boldsymbol{\alpha} - \boldsymbol{e}^T\boldsymbol{\alpha} \quad \text{subject to } 0 \le \alpha_i \le C, \forall i, \tag{56}$$

where $Q_{iu} = y_i y_u \boldsymbol{x}_i^T \boldsymbol{x}_u$.

G.1 Theoretical Sparsity of $\boldsymbol{w}^*$

Since L1-loss shares a similar mathematical structure with L2-loss, Theorem 4.1 and Theorem 4.2 also apply to the L1-loss case. The proofs follow directly from the derivations in Appendix D.

G.2 The Weight Density of the Numerical Solutions Obtained from Different Solvers

To investigate whether optimization methods can return solutions with numerical zeros when optimizing L_2-regularized L1-loss SVM, we consider the two algorithms:

- Pegasos (Shalev-Shwartz et al., 2007): Since L1-loss is not differentiable, the three primal-based methods considered in Section 3 cannot be applied. Thus, we consider Pegasos, an algorithm similar to SGD. The main difference is that the stochastic gradient $\tilde{\boldsymbol{g}}_B$ in (6) becomes a stochastic subgradient. The update rule of Pegasos is given by

$$\begin{aligned}\boldsymbol{w}^{\text{new}} &= \boldsymbol{w} - \eta\tilde{\boldsymbol{g}}_B \\ &= (1-\eta)\boldsymbol{w} + C\ell\eta\mathbb{1}[y_i\boldsymbol{w}^T\boldsymbol{x}_i < 1]y_i\boldsymbol{x}_i,\end{aligned} \tag{57}$$

  where the indicator function is

$$\mathbb{1}[y_i\boldsymbol{w}^T\boldsymbol{x}_i < 1] = \begin{cases} 1 & \text{if } y_i\boldsymbol{w}^T\boldsymbol{x}_i < 1, \\ 0 & \text{otherwise.}\end{cases}$$

  In Shalev-Shwartz et al. (2007), they solve the following problem

$$\min_{\boldsymbol{w}} \ \frac{\lambda}{2}\boldsymbol{w}^T\boldsymbol{w} + \frac{1}{\ell}\sum_{i=1}^{\ell}\xi(y_i\boldsymbol{w}^T\boldsymbol{x}_i). \tag{58}$$

  After taking

$$\lambda = \frac{1}{C\ell},$$

  problem (58) is equivalent to our primal problem (1) by scaling a positive constant. Thus, the optimal solutions of these two problems are the same. We set the batch size equal to one for solving the problem by (24). Note that Pegasos sets the initial learning rate as

$$\eta_0 = \frac{1}{\lambda} = C\ell,$$

  and utilizing the scheduling

$$\eta_k = \frac{1}{k}\eta_0$$

  in the k-th update (epoch). We report the results after running 10,000 epochs.

Table 13: The weight density using different solvers for L1/L2-SVM losses. The settings are given in Appendix C. For CDdual, we re-calculate $\overline{\boldsymbol{w}}$ based on (31) to avoid numerical errors from (12).

Methods	L1-loss		L2-loss	
	Pegasos	CDdual	SGD	CDdual
news20	68.7%	62.9%	78.4%	72.7%
rcv1	64.3%	58.7%	72.0%	67.4%
real-sim	94.2%	90.1%	96.5%	93.9%
kdd2010a	67.6%	55.0%	70.9%	58.1%
kdd2010b	68.0%	53.4%	71.3%	57.8%
kdd2012	69.3%	50.0%	80.0%	75.4%

Table 14: The percentage of support vectors (#SV)/l obtained by running CDdual on both losses.

Methods	L1-loss	L2-loss
news20	43.7%	57.3%
rcv1	24.5%	34.7%
real-sim	15.2%	20.9%
kdd2010a	40.2%	45.2%
kdd2010b	37.6%	43.2%
kdd2012	24.9%	53.9%

- CDdual (Hsieh et al., 2008): For dual-based solvers, CDdual can also be used to solve L1-loss SVM through the same derivation in Section 3. For solving L1-loss SVM using CDdual, we only run the algorithm to reach the default stopping condition, though we observe that the results for L2-loss under default/strict conditions are similar.

In Table 13, we list the weight densities of the numerical solutions using Pegasos and CDdual for L1-loss. We also copy the results of L2-loss from Table 2. Both Pegasos and CDdual achieve some sparsity in the weights under L1-loss.

To understand why CDdual leads to greater sparsity under L1-loss, according to the primal-dual relationship (4), we hypothesize that if fewer instances are used to construct $\boldsymbol{w}$, the final $\overline{\boldsymbol{w}}$ is likely to be sparser. To support this conjecture, we compute the number of instances with $\overline{\alpha}_i > 0$ (i.e., support vectors) for CDdual with L1/L2-loss under the same setting in Table 2. The results are in Table 14. We observe that the support vectors using L1-loss are generally fewer than using L2-loss, which aligns with our hypothesis.

G.3 Model Size Reduction for L1-loss in Extreme Multi-label Classification

In Table 15, we report the model sizes obtained by solving each binary classifier with CDdual under the L1-loss. For comparison, the corresponding results for the L2-loss (from Table 7) are also included. Across all data sets, training with L1-loss using CDdual leads to smaller models than with L2-loss. This observation aligns with Table 13: since the numerical solutions obtained from L1-loss are sparser than those from L2-loss, fewer nonzero weights need to be stored.

However, in terms of prediction performance, we observe a noticeable gap between solving L1-loss and L2-loss. Because the underlying optimization objectives differ, their respective optimal solutions can vary significantly. Therefore, a fair comparison requires careful tuning of hyperparameters such as the regularization parameter C for both settings. The tuned results are summarized in Table 16. On large-scale sets such as Amazon-670K and Amazon-3M, the performance of L1-loss is worse than that of L2-loss. Investigating the reasons behind this performance drop remains an interesting direction for future work.

Table 15: Model size for extreme multi-label classification using different solvers. The numbers in parentheses indicate the size relative to that of Newton.

	Newton(L2)	CDdual(L2)	CDdual(L2)/Newton	CDdual(L1)	CDdual(L1)/Newton
rcv1v2	0.058 GB	0.026 GB	45 %	0.022 GB	37 %
EUR-Lex	1.136 GB	0.827 GB	72 %	0.743 GB	65 %
AmazonCat-13K	4.477 GB	2.053 GB	45 %	1.706 GB	38 %
Wiki10-31K	8.279 GB	4.600 GB	55 %	3.609 GB	43 %
Wiki-500K	299.283 GB	164.952 GB	55 %	139.952 GB	47 %
Amazon-670K	37.184 GB	20.439 GB	55 %	18.314 GB	49 %
Amazon-3M	515.195 GB	224.994 GB	44 %	193.054 GB	37 %

Table 16: Performance comparison using different solvers. We report the precision scores for Newton. For CDdual, we show the score differences compared to Newton.

	Precision@1			Precision@3			Precision@5		
	Newton	CDdual(L2)	CDdual(L1)	Newton	CDdual(L2)	CDdual(L1)	Newton	CDdual(L2)	CDdual(L1)
rcv1v2	95.78	+0.01	+0.12	80.02	+0.01	+0.15	55.94	+0.01	+0.05
EUR-Lex	82.36	−0.03	−0.97	69.06	+0.00	−0.66	57.73	+0.00	−0.95
AmazonCat-13K	93.01	+0.01	−1.78	79.24	+0.00	−0.32	64.44	+0.00	−0.39
Wiki10-31K	84.45	−0.02	−1.10	74.34	+0.03	+0.01	65.54	+0.01	−0.07
Wiki-500K	67.92	+0.00	−1.12	48.48	−0.01	−0.89	37.71	−0.01	−0.85
Amazon-670K	44.11	+0.00	−1.01	38.93	−0.01	−1.24	35.08	+0.00	−1.58
Amazon-3M	46.88	−0.01	−1.21	44.08	−0.01	−1.26	41.93	−0.01	−1.23

H EXTENSION TO LOGISTIC LOSS

The classification method logistic regression solves (1) with the following logistic loss:

$$\xi^{\mathsf{LR}}(y\boldsymbol{w}^T\boldsymbol{x}) = \log(1+\exp(-y\boldsymbol{w}^T\boldsymbol{x})).$$

Different from the situation of L1/L2-loss, we show that the optimal $\boldsymbol{w}^*$ is in general fully dense. In Yu et al. (2011), they showed that the dual problem takes the following form

$$\begin{aligned}\min_{\boldsymbol{\alpha}} \quad & \frac{1}{2}\boldsymbol{\alpha}^T Q^{\mathsf{LR}}\boldsymbol{\alpha} + \sum\nolimits_{i=1}^{\ell} \alpha_i \log\alpha_i + (C-\alpha_i)\log(C-\alpha_i),\\ \text{subject to} \quad & 0 \le \alpha_i \le C, \forall i,\end{aligned}$$

where $Q^{\mathsf{LR}} \in \mathbb{R}^{\ell\times\ell}$ with

$$Q^{\mathsf{LR}}_{iu} = y_i y_u \boldsymbol{x}_i^T \boldsymbol{x}_u.$$

Moreover, they proved that the optimal $\alpha_i^* > 0$ for all i. Then, from the primal-dual relationship (4), every feature is used for constructing $\boldsymbol{w}^*$. Thus, $\boldsymbol{w}^*$ is likely to be fully dense. In this situation, an optimization method with convergence guarantee returns a dense $\overline{\boldsymbol{w}}$ close to $\boldsymbol{w}^*$.

We check the weight density using Newton and CDdual for logistic loss in Table 17 with the default setting in LIBLINEAR. Both methods return dense weights. Therefore, we confirm that even the coordinate descent method to solve the dual problem of logistic regression returns fully dense $\overline{\boldsymbol{w}}$.

Table 17: The density of $\overline{\boldsymbol{w}}$ for logistic regression using Newton and CDdual solvers.

	Newton	CDdual
news20	100%	100%
rcv1	100%	100%
real-sim	100%	100%
kdd2010a	100%	100%
kdd2010b	100%	100%
kdd2012	100%	100%

I EXTENSION TO SUPPORT VECTOR REGRESSION

Due to space limitations, the main paper primarily focuses on the weight density in L_2-regularized linear classification, specifically for support vector classification. In this section, we extend our discussion to include linear support vector regression (SVR).

I.1 Problem Formulation[14]

Given a set of ℓ training instance-target pairs $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{\ell}$, where $\boldsymbol{x}_i \in \mathbb{R}^n$ is the feature vector and $y_i \in \mathbb{R}$ is the target value for $\boldsymbol{x}_i$. Linear SVR finds a model $\boldsymbol{w}$ such that $\boldsymbol{w}^T\boldsymbol{x}_i$ is close to the target value y_i. It solves the following L_2-regularized problem:

$$\min_{\boldsymbol{w}} f(\boldsymbol{w}) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum\nolimits_{i=1}^{l} \xi_\epsilon(\boldsymbol{w}; \boldsymbol{x}_i, y_i), \tag{59}$$

where ξ_ϵ is the ϵ-insensitive loss

$$\xi_\epsilon(\boldsymbol{w}; \boldsymbol{x}_i, y_i) = \max(|\boldsymbol{w}^T\boldsymbol{x}_i - y_i| - \epsilon, 0) \quad \text{(L1-loss)}, \tag{60}$$

$$\xi_\epsilon(\boldsymbol{w}; \boldsymbol{x}_i, y_i) = \max(|\boldsymbol{w}^T\boldsymbol{x}_i - y_i| - \epsilon, 0)^2 \quad \text{(L2-loss)}. \tag{61}$$

The parameter ϵ is given so that the loss is zero if $|\boldsymbol{w}^T\boldsymbol{x}_i - y_i| \le \epsilon$.

According to Vapnik (1995), the dual problem of SVR is given by:

$$\min_{\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-} f_{\mathsf{dual}}(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) \quad \text{subject to} \quad 0 \le \alpha_i^+, \alpha_i^- \le U, \quad \forall i = 1, \ldots, \ell, \tag{62}$$

where the dual objective is:

$$\begin{aligned} f_{\mathsf{dual}}(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-) = {} & \frac{1}{2}(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-)^T Q(\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-) \\ & + \sum\nolimits_{i=1}^{\ell} \left(\epsilon(\alpha_i^+ + \alpha_i^-) - y_i(\alpha_i^+ - \alpha_i^-) + \frac{\lambda}{2}\left((\alpha_i^+)^2 + (\alpha_i^-)^2\right) \right). \end{aligned} \tag{63}$$

In equation (63), $Q \in \mathbb{R}^{l \times l}$ is a matrix with $Q_{ij} = \boldsymbol{x}_i^T\boldsymbol{x}_j$ and (λ, U) depends on the loss type:

$$(\lambda, U) = \begin{cases} (0, C) & \text{if L1-loss SVR,} \\ \left(\frac{1}{2C}, \infty\right) & \text{if L2-loss SVR.} \end{cases}$$

To simplify the notation, we can combine $\boldsymbol{\alpha}^+$ and $\boldsymbol{\alpha}^-$ such that

$$\boldsymbol{\alpha} = \begin{bmatrix} \boldsymbol{\alpha}^+ \\ \boldsymbol{\alpha}^- \end{bmatrix} \quad \text{and} \quad f_{\mathsf{dual}}(\boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{\alpha}^T \begin{bmatrix} \overline{Q} & -Q \\ -Q & \overline{Q} \end{bmatrix} \boldsymbol{\alpha} + \begin{bmatrix} \epsilon\boldsymbol{e} - \boldsymbol{y} \\ \epsilon\boldsymbol{e} + \boldsymbol{y} \end{bmatrix}^T \boldsymbol{\alpha},$$

[14] We follow the description for support vector regression in Ho and Lin (2012).

Table 18: A practical use of Theorem I.2 to obtain an upper bound of the weight density for the theoretical optimal $\boldsymbol{w}^*$ under L2-loss SVR, computed by $1-$(lower bound on # zeros)$/n$. We pick $\Delta = 0.005$ for all datasets to identify an $I_{\overline{\boldsymbol{w}}}$ set close to $I_{\boldsymbol{w}^*}$.

Data sets	Upper bound on $\boldsymbol{w}^*$'s density
E2006-tfidf	63.7 %
E2006-log1p	65.1 %
kdd2012	80.3 %

where $\overline{Q} = Q + \lambda\mathcal{I}$, $\mathcal{I}$ is the identity matrix, and $\boldsymbol{e}$ is the vector of ones. Therefore, the dual problem (62) can be re-written as

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \ & f_{\mathsf{dual}}(\boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{\alpha}^T \begin{bmatrix} \overline{Q} & -Q \\ -Q & \overline{Q} \end{bmatrix} \boldsymbol{\alpha} + \begin{bmatrix} \epsilon\boldsymbol{e} - \boldsymbol{y} \\ \epsilon\boldsymbol{e} + \boldsymbol{y} \end{bmatrix}^T \boldsymbol{\alpha} \\ \text{subject to} \ & 0 \le \alpha_i^+, \alpha_i^- \le U, \quad \forall i = 1, \ldots, \ell. \end{aligned} \tag{64}$$

The primal-dual relationship between the primal optimal solution $\boldsymbol{w}^*$ and the dual optimal solution $\boldsymbol{\alpha}^*$ is given by:

$$\boldsymbol{w}^* = \sum\nolimits_{i=1}^{\ell} ((\alpha_i^+)^* - (\alpha_i^-)^*)\boldsymbol{x}_i. \tag{65}$$

I.2 Theoretical Analysis on the Sparsity of Optimal $\boldsymbol{w}^*$ in Support Vector Regression

Following what we have developed under L1/L2-loss support vector classification in Section 4 of the main paper, we now analyze whether the optimal solution $\boldsymbol{w}^*$ in SVR can also exhibit sparsity.

From the KKT optimality condition of the dual problem, for any optimal dual solution $\boldsymbol{\alpha}^*$, we have

$$|(\boldsymbol{w}^*)^T\boldsymbol{x}_i - y_i| < \epsilon \implies (\alpha_i^+)^* = 0 \text{ and } (\alpha_i^-)^* = 0. \tag{66}$$

Therefore, we turn to consider the set

$$I_{\boldsymbol{w}^*} = \{i \mid \left|(\boldsymbol{w}^*)^T\boldsymbol{x}_i - y_i\right| < \epsilon\}, \tag{67}$$

so that $(\alpha_i^+)^* = (\alpha_i^-)^* = 0$ for all $i \in I_{\boldsymbol{w}^*}$. By a similar argument in the main paper, if a feature j only has values in $I_{\boldsymbol{w}^*}$, i.e.,

$$(\boldsymbol{x}_i)_j = 0, \forall i \notin I_{\boldsymbol{w}^*}, \tag{68}$$

then from the primal-dual relationship (65),

$$w_j^* = \sum_{i=1}^{\ell}((\alpha_i^+)^* - (\alpha_i^-)^*)(\boldsymbol{x}_i)_j = \sum_{i\in I_{\boldsymbol{w}^*}} ((\alpha_i^+)^* - (\alpha_i^-)^*)(\boldsymbol{x}_i)_j = 0. \tag{69}$$

If this occurs, $\boldsymbol{w}^*$ is not fully dense. However, the set $I_{\boldsymbol{w}^*}$ is also unavailable, as what we faced with in the main paper. We then follow the same strategy to construct a proxy index set that avoids relying on $I_{\boldsymbol{w}^*}$. For any $\overline{\boldsymbol{w}} \in \mathbb{R}^n$, we define

$$I_{\overline{\boldsymbol{w}}} = \{i \mid \left|\overline{\boldsymbol{w}}^T\boldsymbol{x}_i - y_i\right| < \epsilon - \Delta\}, \tag{70}$$

where $0 < \Delta < \epsilon$ is a small pre-specified number. We then state and prove the following theorem to connect $I_{\overline{\boldsymbol{w}}}$ and any dual optimal solution $\boldsymbol{\alpha}^*$ to problem (64).

Theorem I.1. *Suppose $\boldsymbol{w}^*$ is optimal to problem* (59) *under* L1-/L2-*loss. For any $\overline{\boldsymbol{w}} \in \mathbb{R}^n$, if*

$$\|\overline{\boldsymbol{w}} - \boldsymbol{w}^*\|\|\boldsymbol{x}_i\| < \Delta, \ \forall i \in I_{\overline{\boldsymbol{w}}}, \tag{71}$$

then $I_{\overline{\boldsymbol{w}}} \subseteq I_{\boldsymbol{w}^*}$, i.e.,

$$\left|(\boldsymbol{w}^*)^T\boldsymbol{x}_i - y_i\right| < \epsilon, \forall i \in I_{\overline{\boldsymbol{w}}}. \tag{72}$$

Further, by (66)*, we have*

$$(\alpha_i^+)^* = 0 \text{ and } (\alpha_i^-)^* = 0, \forall i \in I_{\overline{\boldsymbol{w}}}, \tag{73}$$

where $\boldsymbol{\alpha}^*$ is any dual solution to problem (64)*.*

Table 19: Data statistics for regression data sets, ordered by the number of instances. The column "used features" indicates that features with zero values in the entire training set are removed. The density is defined by the average number of non-zeros per instance divided by the number of used features.

Data sets	ℓ #instances	n #features	#used features	density
E2006-tfidf	16,087	150,360	150,348	0.8256 %
E2006-log1p	16,087	4,272,227	4,265,669	0.1407 %
kdd2012	119,705,032	54,686,452	50,333,432	0.00002%

Table 20: Weight densities of $\overline{\boldsymbol{w}}$ for L2-loss SVR. For both methods, we set the tolerance parameter in LIBLINEAR to be 10^{-8}. For CDdual, we recalculate $\overline{\boldsymbol{w}}$ based on (65) to ensure numerical precision.

Methods	Netwon	CDdual
E2006-tfidf	100.0%	63.2 %
E2006-log1p	100.0%	64.4 %
kdd2012	100.0%	75.4 %

Proof. Since (72) directly implies (73) from (66), it suffices to prove (72). We have for all $i \in I_{\overline{\boldsymbol{w}}}$,

$$\begin{aligned}\left|(\boldsymbol{w}^*)^T\boldsymbol{x}_i - y_i\right| &\le \left|(\overline{\boldsymbol{w}})^T\boldsymbol{x}_i - y_i\right| + \left|(\boldsymbol{w}^*)^T\boldsymbol{x}_i - (\overline{\boldsymbol{w}})^T\boldsymbol{x}_i\right| \\ &= \left|(\overline{\boldsymbol{w}})^T\boldsymbol{x}_i - y_i\right| + \left|(\boldsymbol{w}^* - \overline{\boldsymbol{w}})^T\boldsymbol{x}_i\right| \\ &< \epsilon - \Delta + \Delta = \epsilon.\end{aligned}$$

Therefore, $i \in I_{\boldsymbol{w}^*}$. □

Then, we develop a theorem similar to Theorem 4.2 in the main paper for practically finding an $\overline{\boldsymbol{\alpha}}$ satisfying (71).

Theorem I.2. *Consider the dual problem* (64) *under* L1-/L2-*loss. For any dual feasible* $\overline{\boldsymbol{\alpha}} \in \mathbb{R}^{2\ell}$*, we define* $\overline{\boldsymbol{w}} = \sum_{i=1}^{\ell}\left((\alpha_i^+)^* - (\alpha_i^-)^*\right)\boldsymbol{x}_i$*. If*

$$\sqrt{2\left(f(\overline{\boldsymbol{w}}) + f_{\mathsf{dual}}(\overline{\boldsymbol{\alpha}})\right)}\|\boldsymbol{x}_i\| < \Delta,\ \forall i \in I_{\overline{\boldsymbol{w}}}, \tag{74}$$

then $I_{\overline{\boldsymbol{w}}} \subseteq I_{\boldsymbol{w}^*}$ *and*

$$(\alpha_i^+)^* = 0 \text{ and } (\alpha_i^-)^* = 0, \forall i \in I_{\overline{\boldsymbol{w}}},$$

where $\boldsymbol{\alpha}^*$ *is any dual optimal solution to* (64)*.*

Proof. By checking Appendix D.2, we find that the theorem relies on proving (41) and then applying Theorem D.1. We see that (41) holds for any loss function convex in $y\boldsymbol{w}^T\boldsymbol{x}$. Here, the ϵ-insensitive losses (60) and (61) for support vector regression are convex, so the proof in Appendix G.2 can be applied. □

With the theorems, we also compute the upper bound of the weight density for the unknown optimal $\boldsymbol{w}^*$ and show the results in Table 18. We see that for L2-regularized support vector regression, the optimal solutions also show some sparsity.

I.3 Comparison of the Numerical Weight Density under Primal- and Dual-based Solvers

We use Newton to solve the primal problem (59) and CDdual to solve the dual. Both methods are developed by Ho and Lin (2012) and implemented in LIBLINEAR (Fan et al., 2008). We consider two regression data sets, E2006-tfidf and E2006-log1p, and a binary classification data set, kdd2012. We treat the classification data set as a regression data set because we cannot find many large sparse regression sets. The data statistics are given in Table 19, and all sets can be downloaded from "LIBSVM Data Sets."[15] We apply the same experimental setup for support vector classification in the main paper to SVR. Since the losses in (60) and (61) involve one more

[15] https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/

parameter ϵ, we select $\epsilon = 0.3$ for all SVR experiments. Empirically, we observe that ϵ significantly affects the number of zeros in the returned $\overline{\boldsymbol{\alpha}}$ and the density of $\overline{\boldsymbol{w}}$.

The results shown in Table 20 are quite similar to the classification case. Newton always returns fully dense solutions, while CDdual gives sparse solutions.