OPT2022: 14th Annual Workshop on Optimization for Machine Learning

Boosting as Frank-Wolfe

Ryotaro Mitsuboshi RYOTARO.MITSUBOSHI@INF.KYUSHU-U.AC.JP
Kyushu University/RIKEN AIP

Kohei Hatano HATANO@INF.KYUSHU-U.AC.JP
Kyushu University/RIKEN AIP

Eiji Takimoto EIJI@INF.KYUSHU-U.AC.JP
Kyushu University

Abstract

Some boosting algorithms, such as LPBoost, ERLPBoost, and C-ERLPBoost, aim to solve the soft margin optimization problem with the ℓ_1-norm regularization. LPBoost rapidly converges to an ϵ-approximate solution in practice, but it is known to take $\Omega(m)$ iterations in the worst case, where m is the sample size. On the other hand, ERLPBoost and C-ERLPBoost are guaranteed to converge to an ϵ-approximate solution in $O(\frac{1}{\epsilon^2} \ln \frac{m}{\nu})$ iterations. However, the computation per iteration is very high compared to LPBoost.

To address this issue, we propose a generic boosting scheme that combines the Frank-Wolfe algorithm and any secondary algorithm and switches one to the other iteratively. We show that the scheme retains the same convergence guarantee as ERLPBoost and C-ERLPBoost. One can incorporate any secondary algorithm to improve in practice. This scheme comes from a unified view of boosting algorithms for soft margin optimization. More specifically, we show that LPBoost, ERLPBoost, and C-ERLPBoost are instances of the Frank-Wolfe algorithm. In experiments on real datasets, one of the instances of our scheme exploits the better updates of the secondary algorithm and performs comparably with LPBoost.

1. Introduction

The ℓ_1-norm regularized soft margin optimization problem, defined later, is a formulation of finding sparse large-margin classifiers based on the linear program (LP). This problem aims to optimize the ℓ_1-margin by combining multiple hypotheses from some hypothesis class $\mathcal{H}$. The resulting classifier tends to be sparse, so ℓ_1-margin optimization is helpful for feature selection tasks. Off-the-shelf LP solvers can solve the problem, but they are still not efficient enough for a huge class $\mathcal{H}$.

Boosting is a framework for solving the ℓ_1-norm regularized margin optimization even though $\mathcal{H}$ is infinitely large. Various boosting algorithms have been invented. LPBoost [2] is a practical algorithm and often works efficiently in practice. Although LPBoost terminates rapidly, It is shown that it takes $\Omega(m)$ iterations in the worst case, where m is the number of training examples [8]. Shalev-Shwartz and Singer [7] invented an algorithm called Corrective ERLPBoost (we call this algorithm C-ERLPBoost for shorthand) in the paper on ERLPBoost [9]. C-ERLPBooost and ERLPBoost find ϵ-approximate solutions in $O(\ln(m/\nu)/\epsilon^2)$ iterations, where $\nu \in [1, m]$ is the soft margin parameter. The difference is the time complexity per iteration; ERLPBoost solves a convex program (CP) for each

Table 1: Comparison of the boosting algorithms. C-ERLPBoost solves the problem per iteration by sorting based algorithm, while our work and LPBoost solves linear programming (LP). ERLPBoost solves convex programming (CP) per iteration.

	LPBoost	C-ERLPBoost	ERLPBoost	One of our work
Iter. bound	$\Omega(m)$	$O\left(\frac{1}{\epsilon^2}\ln\frac{m}{\nu}\right)$	$O\left(\frac{1}{\epsilon^2}\ln\frac{m}{\nu}\right)$	$O\left(\frac{1}{\epsilon^2}\ln\frac{m}{\nu}\right)$
Problem per iter.	LP	Sorting	CP	LP

iteration, while C-ERLPBooost solves a sorting-like problem. Although ERLPBoost takes much time per iteration, it takes fewer iterations than C-ERLPBoost in practical applications. For this reason, ERLPBoost is faster than C-ERLPBoost. Our primary motivation is to investigate boosting algorithms with provable iteration bounds which perform as fast as LPBoost.

This paper has two contributions. Our first contribution is to give a unified view of boosting for soft margin optimization. We show that LPBoost, ERLPBoost, and C-ERLPBoost are instances of the Frank-Wolfe algorithm.

Our second contribution is to propose a generic scheme for boosting based on the unified view. Our scheme combines a standard Frank-Wolfe algorithm and *any* algorithm and switch one to the other at each iteration in a non-trivial way. We show that this scheme guarantees the same convergence rate, $O(\ln(m/\nu)/\epsilon^2)$, as ERLPBoost and C-ERLPBoost. One can incorporate any update rule to this scheme without losing the convergence guarantee, so that it take advantage of better updates of the second algorithm in practice. In particular, we propose to choose LPBoost as the second algorithm to combine and we call the resulting algorithm Modified LPBoost (MLPBoost).

In experiments on real datasets, MLPBoost works comparably with LPBoost and it is MLPBoost is the fastest among theoretically guaranteed algorithms as expected.

2. Basic definitions

This paper considers binary classification boosting. We use the same notations as in [7]. Let $S := ((\boldsymbol{x}_i, y_i))_{i=1}^m \in (\mathcal{X} \times \{\pm 1\})^m$ be a sequence of m-examples, where $\mathcal{X}$ is some set. Let $\mathcal{H} \subset [-1, +1]^{\mathcal{X}}$ be a set of hypotheses. For simplicity, we assume $|\mathcal{H}| = |\{h_1, h_2, \ldots, h_n\}| = n$. It is convenient to regard each $h_j \in \mathcal{H}$ as a canonical basis vector $\boldsymbol{e}_j \in \mathbb{R}^n$. Let $A = (y_i h_j(\boldsymbol{x}_i)) \in [-1, +1]^{m \times n}$ be a matrix of size $m \times n$. We denote m-dimensional capped probability simplex as $\mathcal{P}_\nu^m := \{\boldsymbol{d} \in [0, 1/\nu]^m \mid \|\boldsymbol{d}\|_1 = 1\}$, where $\nu \in [1, m]$. We write $\mathcal{P}^m = \mathcal{P}_1^m$ for shorthand. For a set $\mathcal{C} \subset \mathbb{R}^n$, we denote the convex hull of $\mathcal{C}$ as $\mathrm{CH}(\mathcal{C}) := \{\sum_k w_k \boldsymbol{s}_k \mid \sum_k w_k = 1, w_k \geq 0, \{\boldsymbol{s}_k\}_k \subset \mathcal{C}\}$. We say that a function $f : \mathbb{R}^m \to \mathbb{R}$ is η-smooth over a convex set $\mathcal{C} \subset \mathbb{R}^m$ w.r.t. a norm $\|\cdot\|$ if $f(\boldsymbol{y}) \leq f(\boldsymbol{x}) + (\boldsymbol{y} - \boldsymbol{x})^\top \nabla f(\boldsymbol{x}) + \frac{\eta}{2}\|\boldsymbol{y} - \boldsymbol{x}\|^2$ for all $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{C}$. The Fenchel conjugate $f^\star$ of a function $f : \mathbb{R}^m \to [-\infty, +\infty]$ is defined as $f^\star(\boldsymbol{\theta}) = \sup_{\boldsymbol{d} \in \mathbb{R}^m} \left(\boldsymbol{d}^\top \boldsymbol{\theta} - f(\boldsymbol{d})\right)$. It is well known that if f is a $1/\eta$-strongly convex function w.r.t. the norm $\|\cdot\|$ for some $\eta > 0$, $f^\star$ is an η-smooth function w.r.t. the dual norm

$\| \cdot \|_\star$. Further, if f is a strongly convex function, the gradient vector of $f^\star$ is written as $\nabla f^\star(\boldsymbol{\theta}) = \arg\sup_{\boldsymbol{d} \in \mathbb{R}^m} \left(\boldsymbol{d}^\top \boldsymbol{\theta} - f(\boldsymbol{d}) \right)$. You can find the proof of these properties here [1, 7].

We define the soft margin optimization problem as the dual problem of the edge minimization problem. The edge minimization problem is defined as $\gamma := \min_{\boldsymbol{d}} \max_{j \in [n]} (\boldsymbol{d}^\top A)_j + f(\boldsymbol{d})$, where $f : \mathbb{R}^m \to \{0, +\infty\}$ is the barrier function such that $f(\boldsymbol{d}) = 0 \iff \boldsymbol{d} \in \mathcal{P}^m_\nu$. The quantity $(\boldsymbol{d}^\top A)_j = \sum_{i=1}^m d_i y_i h_j(\boldsymbol{x}_i)$ is often called the *edge* of the hypothesis h_j w.r.t. the distribution $\boldsymbol{d}$. [7] proved that the ℓ_1-norm regularized soft margin optimization problem is derived as the Fenchel dual problem of edge minimization: $\rho := \max_{\boldsymbol{w} \in \mathcal{P}^n} -f^\star(-A\boldsymbol{w}) = \max_{\boldsymbol{w} \in \mathcal{P}^n} \min_{\boldsymbol{d} \in \mathcal{P}^m_\nu} \boldsymbol{d}^\top A \boldsymbol{w}$. Since the strong duality holds, $\gamma = \rho$. The goal of the soft margin optimization is to find an optimal combined hypothesis $H_T = \sum_{j=1}^n w^\star_j h_j$, where $\boldsymbol{w}^\star \in \mathcal{P}^n$ is an optimal solution of the soft margin optimization. Although the edge minimization and soft margin optimization problems are formulated as a linear program, solving the problem for a huge class $\mathcal{H}$ is hard. Boosting is a standard approach to dealing with the problem.

Boosting is formulated by a protocol between two algorithms; the booster and the weak learner. For each iteration $t = 0, 1, 2, \ldots, T$, the booster chooses a distribution $\boldsymbol{d}^t \in \mathcal{P}^m_\nu$ over the training examples S. Then, the weak learner returns a hypothesis $h_{j_{t+1}} \in \mathcal{H}$ to the booster that satisfies $({\boldsymbol{d}_t}^\top A)_{j_{t+1}} \geq g$ for some unknown guarantee $g > 0$. The boosting algorithm aims to produce a convex combination $H_T = \sum_{t=1}^T w_{T,t} h_{j_t}$ of the hypotheses $\{h_{j_1}, h_{j_2}, \ldots, h_{j_T}\} \subset \mathcal{H}$ that satisfies

$$-f^\star(-A\boldsymbol{w}_T) = \min_{\boldsymbol{d} \in \mathcal{P}^m_\nu} \boldsymbol{d}^\top A \boldsymbol{w}_T = \min_{\boldsymbol{d} \in \mathcal{P}^m_\nu} \sum_{i=1}^m d_i y_i H_T(\boldsymbol{x}_i) \geq g - \epsilon \tag{1}$$

for any predefined $\epsilon > 0$. Suppose that the weak learner always returns a max-edge hypothesis. In that case, the goal is to find an ϵ-approximate solution of the soft margin optimization.

3. Main results

We first show the unified view of the boosting algorithms via Fenchel duality. From this view, LPBoost, ERLPBoost, and C-ERLPBoost can be seen as instances of the Frank-Wolfe algorithm with different step sizes and objectives. Using this knowledge, we derive a new boosting scheme. Note that all proofs are found in appendix.

3.1. A unified view of boosting for soft margin optimization

This section assumes that the weak learner always returns a hypothesis $h \in \mathcal{H}$ that maximizes the edge w.r.t. the given distribution. We start by revisiting C-ERLPBoost. C-ERLPBoost aims to solve the convex program $\min_{\boldsymbol{d}} \max_{j \in [n]} (\boldsymbol{d}^\top A)_j + \tilde{f}^\star(\boldsymbol{d})$,, where $\tilde{f} = f + \frac{1}{\eta}\Delta$. Since $\frac{1}{\eta}\Delta$ is a $\frac{1}{\eta}$-strongly convex function w.r.t. ℓ_1-norm, so does $\tilde{f}$. One can easily verify that the dual problem is $\max_{\boldsymbol{w} \in \mathcal{P}^n} -\tilde{f}^\star(-A\boldsymbol{w})$. Furthermore, $\tilde{f}^\star$ is an η-smooth function w.r.t. ℓ_∞-norm. Thus, the soft margin optimization problem becomes a minimization problem of a smooth function.

C-ERLPBoost updates the distribution $\boldsymbol{d}_t \in \mathcal{P}^m_\nu$ as $\boldsymbol{d}_t = \nabla \tilde{f}^\star(\boldsymbol{\theta}_t) = \arg\min_{\boldsymbol{d}} \boldsymbol{d}^\top \boldsymbol{\theta}_t + \tilde{f}(\boldsymbol{d})$, where $\theta_t = -A\boldsymbol{w}_t$. Then, obtain a hypothesis $h_{j_{t+1}} \in \mathcal{H}$ that maximizes the edge; $j_{t+1} \in$

Algorithm 1 A theoretically guaranteed boosting scheme

```
Input: A matrix $A = (y_i h_j(\boldsymbol{x}_i))_{i,j} \in [-1,1]^{m \times n}$, a Frank-Wolfe rule $\mathcal{F}$, a secondary
    algorithm $\mathcal{B}$, and parameters $\nu > 0$ and $\epsilon > 0$.
1: Set $\boldsymbol{w}_0 = \boldsymbol{0}$.
2: for $t = 0, 1, 2, \ldots, T$ do
3:     Compute the distribution $\boldsymbol{d}_t = \nabla \tilde{f}^\star(-A\boldsymbol{w}_t)$.
4:     Obtain a hypothesis $h_{j_{t+1}} \in \mathcal{H}$ and set $\mathcal{E}_{t+1} = \{\boldsymbol{e}_{j_\tau}\}_{\tau=1}^{t+1}$.
5:     if $\epsilon_t := \min_{0 \le \tau \le t} (\boldsymbol{d}_\tau^\top A)_{j_{\tau+1}} + \tilde{f}^\star(-A\boldsymbol{w}_t) \le \epsilon/2$ then Set $T = t$, break.
6:     Compute the Frank-Wolfe weight $\boldsymbol{w}_{t+1}^{(1)} = \mathcal{F}(A, \boldsymbol{w}_t, \boldsymbol{e}_{j_{t+1}}, \mathcal{E}_t, \boldsymbol{d}_t)$.
7:     Compute the secondary weight $\boldsymbol{w}_{t+1}^{(2)} = \mathcal{B}(A, \mathcal{E}_{t+1})$.
8:     Update the weight $\boldsymbol{w}_{t+1} \leftarrow \arg\min_{\boldsymbol{w}_{t+1}^{(k)} : k \in \{1,2\}} \tilde{f}^\star(-A\boldsymbol{w}_{t+1}^{(k)})$.
9: end for
Output: Combined classifier $H_T = \sum_{t=1}^{T} w_{T,t} h_t$.
```

$\arg\max_{j \in [n]} (\boldsymbol{d}_t^\top A)_j$. We can rewrite this step as

$$\arg \max_{\boldsymbol{e}_j : j \in [n]} \boldsymbol{d}_t^\top A \boldsymbol{e}_j = \arg \min_{\boldsymbol{e}_j : j \in [n]} (-A\boldsymbol{e}_j)^\top \nabla \tilde{f}^\star(\boldsymbol{\theta}_t) = \arg \min_{\boldsymbol{\theta} \in -A\mathcal{P}^n} \boldsymbol{\theta}^\top \nabla \tilde{f}^\star(\boldsymbol{\theta}_t)$$

Thus, finding a hypothesis that maximizes edge corresponds to solving linear programming in the Frank-Wolfe algorithm. Further, C-ERLPBoost updates the weights as $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \lambda_t(\boldsymbol{e}_{j_{t+1}} - \boldsymbol{w}_t)$, where λ_t is the short-step, as in algorithm 3. From these observations, we can say that the C-ERLPBoost is an instance of the Frank-Wolfe algorithm. Since $\tilde{f}^\star$ is η-smooth, we can say that C-ERLPBoost converges in $O(\eta/\epsilon) = O(\ln(m/\nu)/\epsilon^2)$ iterations for a max-edge weak learner.

Similarly, one can easily verify that ERLPBoost and C-ERLPBoost are also instances of the Frank-Wolfe algorithm. See the proof in appendix.

Theorem 1 *LPBoost, ERLPBoost, and C-ERLPBoost are instances of the Frank-Wolfe algorithm.*

3.2. A generic scheme for margin-maximizing boosting

Algorithm 2 LPBoost rule $\mathcal{B}(A, \mathcal{E}_{t+1})$

```
Input: A matrix $A = (y_i h_j(\boldsymbol{x}_i))_{i,j} \in [-1,+1]^{m \times n}$ and a set of basis vectors $\mathcal{E}_{t+1} \subset \mathcal{P}^n$.
Output: $\boldsymbol{w} \leftarrow \arg\max_{\boldsymbol{w} \in \mathrm{CH}(\mathcal{E}_{t+1})} \min_{\boldsymbol{d} \in \mathcal{P}_\nu^m} \boldsymbol{d}^\top A \boldsymbol{w}$.
```

Algorithm 3 Short step rule $\mathcal{F}(A, \boldsymbol{w}_t, \boldsymbol{e}_{j_{t+1}}, \mathcal{E}_t, \boldsymbol{d}_t)$

```
1: $\boldsymbol{w}_{t+1}^{(1)} = \boldsymbol{w}_t + \lambda_t(\boldsymbol{e}_{j_{t+1}} - \boldsymbol{w}_t)$, where $\lambda_t = \min\left\{1, \max\left\{0, \frac{\boldsymbol{d}_t^\top A(\boldsymbol{e}_{j+1} - \boldsymbol{w}_t)}{\eta \|A(\boldsymbol{e}_{j+1} - \boldsymbol{w}_t)\|_\infty^2}\right\}\right\}$.
```

We propose a Frank-Wolfe-like boosting scheme, shown in algorithm 1. Intuitively, the Frank-Wolfe rule $\mathcal{F}$ is a safety net for the convergence guarantee. Further, the convergence analysis only depends on the Frank-Wolfe rule $\boldsymbol{w}_{t+1}^{(1)}$ so one can incorporate any update rule to $\mathcal{B}$. For example, one can use $\boldsymbol{w}_{t+1}^{\mathrm{E}}$ as $\mathcal{B}(A, \mathcal{E}_{t+1})$. Algorithm 1 becomes ERLPBoost in this case since $\boldsymbol{w}_{t+1} = \boldsymbol{w}_{t+1}^{(2)}$ holds for all t.

Recall that our primary objective is to find a weight vector $\boldsymbol{w}$ that optimizes the soft margin, formulated as LP. The most practical algorithm, LPBoost, solves the optimization problem over past hypotheses, so using the solution as $\mathcal{B}$ is a natural choice. Algorithm 2 summarizes this update. As described in [7], one can compute the distribution $\boldsymbol{d}_t = \nabla \tilde{f}^\star(-A\boldsymbol{w}_t)$ by a sorting-based algorithm, which takes $O(m \ln m)$ iterations[1]. Thus, the time complexity of our scheme depends on the secondary algorithm $\mathcal{B}$. The following theorem guarantees the convergence rate for algorithm 1. We give the proof in the appendix.

Theorem 2 (A convergence rate for algorithm 1) *Assume that the weak learner returns a hypothesis $h_{j_{t+1}} \in \mathcal{H}$ that satisfies $(\boldsymbol{d}_t^\top A)_{j_{t+1}} \geq g$ for some unknown guarantee g. Let $\mathcal{F}$ be a Frank-Wolfe rule with classic step $\lambda_t = \frac{2}{t+2}$, or short-step as in algorithm 3. Then, for any secondary algorithm $\mathcal{B}$, algorithm 1 outputs $H_T = \sum_{t=1}^T w_{j_t} h_{j_t}$ satisfying (1) in $O\left(\frac{1}{\epsilon^2} \ln \frac{m}{\nu}\right)$ iterations.*

4. Experiments

We compared the boosting algorithms on Gunnar Rätsch's benchmark dataset [2]. We call our scheme with secondary algorithm 2 as MLPBoost. MLPB. (SS) uses the short-step Frank-Wolfe algorithm 3. As table 2 shows, MLPB. (SS) terminates much faster than C-ERLPB. This result indicates that the secondary update significantly improves the objective.

Table 2: Comparison of the computation time (seconds). Some algorithms do not terminate in a few hours so we abort them within some appropriate time.

	m	LPB.	ERLPB.	Frank-Wolfe	MLPB. (SS)
Diabetes	768	47.53	1478.77	$> 10^4$	201.46
German	1000	77.56	1391.91	$> 10^4$	181.43
Heart	270	10.03	193.58	$> 10^3$	44.11
Image	2086	8.25	107.52	$> 10^3$	32.01
R.norm	7400	22.09	1148.16	$> 10^4$	26.76
Splice	2991	19.35	490.92	$> 10^4$	122.08
Twonorm	7400	105.40	$> 10^4$	$> 10^4$	478.22
Waveform	5000	437.29	9018.54	$> 10^4$	2243.07

The worst case for LPB. Although LPB. outperforms the running time in Table 2, it takes $m/2$ iterations for the worst case [8]. Even in this case, the other algorithms terminate in 2 iterations.

1. They also suggest a linear time algorithm, see [4].
2. Datasets are obtained from `http://theoval.cmp.uea.ac.uk/~gcc/matlab/default.html#benchmarks`.

References

[1] Jonathan M. Borwein and Adrian S. Lewis. *Convex Analysis*, pages 65–96. Springer New York, 2006.

[2] A Demiriz, K P Bennett, and J Shawe-Taylor. Linear Programming Boosting via Column Generation. *Machine Learning*, 46(1-3):225–254, 2002.

[3] Marguerite Frank and Philip Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3(1-2):95–110, 1956.

[4] Mark Herbster and Manfred K. Warmuth. Tracking the best linear predictor. *J. Mach. Learn. Res.*, 1:281–309, sep 2001.

[5] Martin Jaggi. Revisiting frank-wolfe: Projection-free sparse convex optimization. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013*, volume 28 of *JMLR Workshop and Conference Proceedings*, pages 427–435. JMLR.org, 2013.

[6] Simon Lacoste-Julien and Martin Jaggi. On the global linear convergence of frank-wolfe optimization variants. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 496–504, 2015.

[7] Shai Shalev-Shwartz and Yoram Singer. On the equivalence of weak learnability and linear separability: new relaxations and efficient boosting algorithms. *Mach. Learn.*, 80 (2-3), 2010.

[8] M Warmuth, K Glocer, and G Rätsch. Boosting Algorithms for Maximizing the Soft Margin. In *Advances in Neural Information Processing Systems 20 (NIPS 2007)*, pages 1585–1592, 2007.

[9] Manfred K. Warmuth, Karen A. Glocer, and S. V. N. Vishwanathan. Entropy regularized lpboost. In *Algorithmic Learning Theory, 19th International Conference, ALT 2008, Budapest, Hungary, October 13-16, 2008. Proceedings*, volume 5254 of *Lecture Notes in Computer Science*, pages 256–271. Springer, 2008.

Appendix A. Fenchel duality review

First, we introduce some additional notations. For a function $f : \mathbb{R}^m \to (-\infty, +\infty]$, let $\operatorname{dom} f = \{\boldsymbol{d} | f(\boldsymbol{d}) < +\infty\}$. For a convex set $C \subset \mathbb{R}^n$, define the set of interior points $\operatorname{int}(C)$ as

$$\operatorname{int}(C) = \{\boldsymbol{w} \in C \mid \forall \boldsymbol{d} \in \mathbb{R}^n, \exists t > 0, \forall \tau \in [0, t], \boldsymbol{w} + \tau \boldsymbol{d} \in C\}.$$

For sets A and B over $\mathbb{R}^n$, define $A - B = \{\boldsymbol{a} - \boldsymbol{b} \mid \boldsymbol{a} \in A, \boldsymbol{b} \in B\}$.

Theorem 3 ([1]) *Let $f : \mathbb{R}^m \to (-\infty, +\infty]$ and $g : \mathbb{R}^n \to (-\infty, +\infty]$ be convex functions and a linear map $M : \mathbb{R}^m \to \mathbb{R}^n$. Define the Fenchel problems*

$$\gamma = \inf_{\boldsymbol{d}} f(\boldsymbol{d}) + g(M^\top \boldsymbol{d}), \tag{2}$$

$$\rho = \sup_{\boldsymbol{w}} -f^\star(-M\boldsymbol{w}) - g^\star(\boldsymbol{w}). \tag{3}$$

Then, $\gamma \geq \rho$ holds. Further, $\gamma = \rho$ holds if

$$\mathbf{0} \in \operatorname{int}\left(\operatorname{dom} g - M^\top \operatorname{dom} f\right). \tag{4}$$

Further, points $\boldsymbol{d}^\star \in \mathcal{P}_\nu^m$ and $\boldsymbol{w}^\star \in \mathcal{P}^n$ are optimal solutions for problems (2) and (3), respectively, if and only if $-M\boldsymbol{w}^\star \in \partial f(\boldsymbol{d}^\star)$ and $\boldsymbol{w}^\star \in \partial g(M^\top \boldsymbol{d}^\star)$.

The following lemma is useful to prove our result.

Lemma 4 *Let $f, \tilde{f} : \mathbb{R}^m \to (-\infty, +\infty]$ be functions such that $\exists c > 0, \forall \boldsymbol{\theta}, f(\boldsymbol{\theta}) \leq \tilde{f}(\boldsymbol{\theta}) \leq f(\boldsymbol{\theta}) + c$. Then, $f^\star(\boldsymbol{\mu}) - c \leq \tilde{f}^\star(\boldsymbol{\mu}) \leq f^\star(\boldsymbol{\mu})$ holds for all $\boldsymbol{\mu}$.*

Appendix B. Technical Lemmas

The following lemma shows the maximum value of the relative entropy from the uniform distribution over the capped probability simplex $\mathcal{P}_\nu^m$.

Lemma 5 $\max_{\boldsymbol{d} \in \mathcal{P}_\nu^m} \Delta(\boldsymbol{d}) \leq \ln \frac{m}{\nu}$.

Proof Since the relative entropy from the uniform distribution achieves its maximal value at the extreme points of $\mathcal{P}_\nu^m$, a maximizer has the form

$$\boldsymbol{d} = (\underbrace{1/\nu, 1/\nu, \ldots, 1/\nu}_{k \text{ elements}}, s, 0, 0, \ldots, 0), \qquad s = 1 - \frac{k}{\nu} \leq \frac{1}{\nu}$$

for some $k \in [m]$. Plugging the maximizer into $\Delta(\boldsymbol{d})$, we can write the objective function as $\Delta(\boldsymbol{d}) = (k/\nu)\ln(m/\nu) + s\ln(sm)$. If $s = 0$, $\Delta(\boldsymbol{d}) = \ln(m/\nu)$ holds since $k = \nu$. If $s > 0$, $k/\nu < 1$ so that

$$\Delta(\boldsymbol{d}) \leq \frac{k}{\nu} \ln \frac{m}{\nu} + \left(1 - \frac{k}{\nu}\right) \ln \frac{m}{\nu} = \ln \frac{m}{\nu}.$$

■

The next lemma shows the dual problem of the edge minimization.

Lemma 6 *Let $f : \mathbb{R}^m \to \{0, +\infty\}$, $g : \mathbb{R}^n \to \mathbb{R}$ be functions defined as*

$$f(\boldsymbol{d}) = \begin{cases} 0 & \boldsymbol{d} \in \mathcal{P}_\nu^m \\ +\infty & \boldsymbol{d} \notin \mathcal{P}_\nu^m \end{cases}, \qquad g(\boldsymbol{\theta}) = \max_{j \in [n]} \theta_j.$$

Then, the dual problem of edge minimization

$$\min_{\boldsymbol{d}} f(\boldsymbol{d}) + g(A^\top \boldsymbol{d}) \tag{5}$$

is the soft margin maximization

$$\max_{\boldsymbol{w} \in \mathcal{P}^n} -f^\star(-A\boldsymbol{w}). \tag{6}$$

Further, the strong duality holds.

Proof We can use Theorem 3 to derive the dual problem. Since

$$\boldsymbol{g}^\star(\boldsymbol{w}) = \begin{cases} 0 & \boldsymbol{w} \in \mathcal{P}^n \\ +\infty & \boldsymbol{w} \notin \mathcal{P}^n \end{cases},$$

you can verify the dual form becomes as in (6). To prove the strong duality, it is enough to prove eq. (4). By definition, $\operatorname{dom} g = \mathbb{R}^n$ and $\operatorname{dom} f = \mathcal{P}^m_\nu$ and hence

$$\operatorname{dom} g - A^\top \operatorname{dom} f = \left\{ \boldsymbol{w} - A^\top \boldsymbol{d} \mid \boldsymbol{w} \in \mathbb{R}^n, \boldsymbol{d} \in \mathcal{P}^m_\nu \right\}.$$

Obviously, $\boldsymbol{0} \in \operatorname{int}(\operatorname{dom} g - A^\top \operatorname{dom} f)$ and thus the strong duality holds. ■

Since

$$f^\star(-A\boldsymbol{w}) = \sup_{\boldsymbol{d}} \left[-\boldsymbol{d}^\top A\boldsymbol{w} - f(\boldsymbol{d}) \right] = \max_{\boldsymbol{d} \in \mathcal{P}^m_\nu} -\boldsymbol{d}^\top A\boldsymbol{w} = \min_{\boldsymbol{d} \in \mathcal{P}^m_\nu} \boldsymbol{d}^\top A\boldsymbol{w},$$

we can write the dual problem (6) explicitly:

$$\max_{\boldsymbol{w} \in \mathcal{P}^n} -f^\star(-A\boldsymbol{w}) = -\min_{\boldsymbol{w} \in \mathcal{P}^n} \max_{\boldsymbol{d} \in \mathcal{P}^m_\nu} \boldsymbol{d}^\top A\boldsymbol{w}.$$

By the same derivation, we get the dual problem for the regularized edge minimization problem.

Corollary 7 *Let f, g be the functions defined in Lemma 6 and let $\Delta(\boldsymbol{d}) = \sum_{i=1}^m d_i \ln d_i + \ln(m)$ be the relative entropy function from the uniform distribution. Define $\tilde{f} = f + (1/\eta)\Delta$ for some $\eta > 0$. Then, the dual problem of*

$$\min_{\boldsymbol{d}} \tilde{f}(\boldsymbol{d}) + g(A^\top \boldsymbol{d}) \qquad \textit{is} \qquad \max_{\boldsymbol{w} \in \mathcal{P}^n} -\tilde{f}^\star(-A\boldsymbol{w}).$$

Appendix C. Proofs

This section is dedicated for the proof of the main result. We first show the fact that LPBoost and ERLPBoost are also instances of FW. Before getting into the proof, we first review the Frank-Wolfe algorithm.

C.1. The Frank-Wolfe algorithms

The original Frank-Wolfe (FW) algorithm is a first-order iterative algorithm invented by [3]. The FW algorithm solves the problems of the form: $\min_{\boldsymbol{x}\in\mathcal{C}} f(\boldsymbol{x})$, where $\mathcal{C} \subset \mathbb{R}^m$ is a closed convex set and $f : \mathcal{C} \to \mathbb{R}$ is an η-smooth and convex function.

In each step t, FW seeks an extreme point $\boldsymbol{s}_{t+1} \in \arg\min_{\boldsymbol{s}\in\mathcal{C}} \boldsymbol{s}^\top \nabla f(\boldsymbol{x}_t)$. Then, it updates the iterate as $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \lambda_t(\boldsymbol{s}_{t+1} - \boldsymbol{x}_t)$ for some $\lambda_t \in [0, 1]$. Although the classical result [3, 5] suggests $\lambda_t = 2/(t + 2)$, there are many choices of λ_t. For example, one can choose λ_t as $\lambda_t := \text{clip}_{[0,1]} \frac{(\boldsymbol{x}_t - \boldsymbol{s}_{t+1})^\top \nabla f(\boldsymbol{x}_t)}{\eta \|\boldsymbol{s}_{t+1} - \boldsymbol{x}_t\|^2}$, where $\text{clip}_{[0,1]}\, x = \max\{0, \min\{1, x\}\}$. This is the optimal solution that minimizes the RHS of the strongly-smooth bound and is often called a short-step strategy. The FW algorithms converge to an ϵ-approximate solution in $O(\eta/\epsilon)$ iterations if the objective function is η-smooth w.r.t. some norm over $\mathcal{C}$ [3, 5]. The best advantage of the FW algorithm is the projection-free property; there is no projection onto $\mathcal{C}$, so the running time per step is faster than the projected gradient methods.

C.2. Proof of Theorem 1

LPBoost and ERLPBoost update the distribution as

$$\begin{aligned} (\text{LPBoost}) \quad & \boldsymbol{d}_t^{\mathrm{L}} \in \arg\min_{\boldsymbol{d}} \max_{j\in[n]} (\boldsymbol{d}^\top A)_j + f(\boldsymbol{d}), \\ (\text{ERLPBoost}) \quad & \boldsymbol{d}_t^{\mathrm{E}} = \arg\min_{\boldsymbol{d}} \max_{j\in[n]} (\boldsymbol{d}^\top A)_j + \tilde{f}(\boldsymbol{d}). \end{aligned}$$

By definition of the Fenchel problems, these distribution correspond to the (sub)gradient vectors $\boldsymbol{d}_t^{\mathrm{L}} \in \partial f^\star(\boldsymbol{\theta}_t^{\mathrm{L}})$ and $\boldsymbol{d}_t^{\mathrm{E}} = \nabla \tilde{f}^\star(\boldsymbol{\theta}_t^{\mathrm{E}})$, where $\boldsymbol{\theta}_t^{\mathrm{L}} = -A\boldsymbol{w}_t^L$ and $\boldsymbol{\theta}_t^{\mathrm{E}} = -A\boldsymbol{w}_t^E$ are

$$\begin{aligned} (\text{LPBoost}) \quad & \boldsymbol{w}_t^{\mathrm{L}} \leftarrow \arg\max_{\boldsymbol{w}\in\mathrm{CH}(\mathcal{E}_t)} -f^\star(-A\boldsymbol{w}), \\ (\text{ERLPBoost}) \quad & \boldsymbol{w}_t^{\mathrm{E}} \leftarrow \arg\max_{\boldsymbol{w}\in\mathrm{CH}(\mathcal{E}_t)} -\tilde{f}^\star(-A\boldsymbol{w}). \end{aligned}$$

These properties are the immediate consequence of Fenchel duality theorem. Thus, we can say that LPBoost and ERLPBoost are instances of the fully-corrective Frank-Wolfe algorithm.

C.3. Proof of our scheme

We prove the convergence guarantee for algorithm 1. Before getting into the main result, we first justify the stopping criterion.

Lemma 8 *Let $\epsilon_t := \min_{0\le\tau\le t}(\boldsymbol{d}_\tau^\top A)_{j_{\tau+1}} + \tilde{f}^\star(-A\boldsymbol{w}_t)$ be the optimality gap defined in algorithm 2 and let $\eta = 2\ln(m/\nu)/\epsilon$. Then, $\epsilon_t \le \epsilon/2$ implies $-f^\star(-A\boldsymbol{w}_t) \ge g - \epsilon$.*

Proof By the weak-learnability assumption, $\epsilon_t \ge g + \tilde{f}^\star(-A\boldsymbol{w}_t)$. The statement follows from Lemma 4. ■

Note that this criterion is better than the one of C-ERLPBoost.

The following theorem shows the convergence rate for our scheme.

Theorem 9 (Recap.) *Assume that the weak learner returns a hypothesis $h_{j_{t+1}} \in \mathcal{H}$ that satisfies $(\boldsymbol{d}_t^\top A)_{j_{t+1}} \geq g$ for some unknown guarantee g. Let $\mathcal{F}$ be a FW update with classic step $\lambda_t = \frac{2}{t+2}$, or short-step as in algorithm 3. Then, for any secondary algorithm $\mathcal{B}$, algorithm 1 outputs $H_T = \sum_{t=1}^{T} w_{j_t} h_{j_t}$ satisfying (1) in $O\left(\frac{1}{\epsilon^2} \ln \frac{m}{\nu}\right)$ iterations.*

Proof First of all, we prove the bound for the classic step size. We start by showing the recursion

$$\epsilon_{t+1} \leq (1 - \lambda_t)\epsilon_t + 2\eta\lambda_t^2. \tag{7}$$

By using the definition of $\boldsymbol{w}_{t+1}$ and the η-smoothness of $\tilde{f}^\star$,

$$\begin{aligned}
\epsilon_t - \epsilon_{t+1} &\geq \tilde{f}^\star(-A\boldsymbol{w}_t) - \tilde{f}^\star(-A\boldsymbol{w}_t^{(1)}) \\
&= \tilde{f}^\star(-A\boldsymbol{w}_t) - \tilde{f}^\star(-A\boldsymbol{w}_t + \lambda_t A(\boldsymbol{w}_t - \boldsymbol{e}_{j_{t+1}})) \\
&\geq \lambda_t (A(\boldsymbol{e}_{j_{t+1}} - \boldsymbol{w}_t))^\top \nabla \tilde{f}^\star(-A\boldsymbol{w}_t) - 2\eta\lambda_t^2, \qquad (8)
\end{aligned}$$

where eq. (8) holds since $A \in [-1, +1]^{m \times n}$ and $\boldsymbol{e}_{j_{t+1}}, \boldsymbol{w}_t \in \mathcal{P}^n$. By the non-negativity of the entropy function and the definition of $\boldsymbol{d}_t$, we get

$$\begin{aligned}
\lambda_t (A(\boldsymbol{e}_{j_{t+1}} - \boldsymbol{w}_t))^\top \nabla \tilde{f}^\star(-A\boldsymbol{w}_t) &= \lambda_t \boldsymbol{d}_t^\top A(\boldsymbol{e}_{j_{t+1}} - \boldsymbol{w}_t) \\
&\geq \lambda_t \left[\min_{0 \leq \tau \leq t} (\boldsymbol{d}_\tau^\top A)_{j_{\tau+1}} - \boldsymbol{d}_t^\top A\boldsymbol{w}_t - \frac{1}{\eta}\Delta(\boldsymbol{d}_t)\right] \\
&= \lambda_t \left[\min_{0 \leq \tau \leq t} (\boldsymbol{d}_\tau^\top A)_{j_{\tau+1}} + \tilde{f}^\star(-A\boldsymbol{w}_t)\right] = \lambda_t \epsilon_t. \qquad (9)
\end{aligned}$$

Combining eq. (8) and (9), we obtain (7).

Now, we prove the following inequality by induction on t.

$$\epsilon_t \leq \frac{8\eta}{t+2}, \quad \forall t = 1, 2, \ldots \tag{10}$$

For the base case $t = 1$, the inequality (10) holds; $\epsilon_1 \leq (1 - \lambda_0)\epsilon_0 + 2\eta\lambda_0^2 = 2\eta \leq \frac{8\eta}{0+2}$. Assume that (10) holds for $t \geq 1$. By the inductive assumption,

$$\epsilon_{t+1} \leq (1 - \lambda_t)\epsilon_t + 2\eta\lambda_t^2 \leq \frac{t}{t+2}\frac{8\eta}{t+2} + 2\eta\left(\frac{2}{t+2}\right)^2 = 8\eta\frac{t}{t+2}\frac{t+1}{t+2} \leq \frac{8\eta}{t+3}.$$

Therefore, (10) holds for all $t \geq 1$.

By definition of η, $\epsilon_T \leq \frac{\epsilon}{2}$ holds after $T \geq \frac{32}{\epsilon^2} \ln \frac{m}{\nu} - 2$ iterations. Lemma 8 yields the convergence rate.

For the short-step case, we get a similar recursion:

$$\begin{aligned}
\epsilon_t - \epsilon_{t+1} &\geq \tilde{f}^\star(-A\boldsymbol{w}_t) - \tilde{f}^\star(-A\boldsymbol{w}_t^{(1)}) \\
&\geq \lambda_t (A(\boldsymbol{e}_{j_{t+1}} - \boldsymbol{w}_t))^\top \nabla \tilde{f}^\star(-A\boldsymbol{w}_t) - \frac{\eta}{2}\lambda_t^2 \|A(\boldsymbol{w}_t - \boldsymbol{e}_{j_{t+1}})\|_\infty^2 \qquad (11) \\
&\geq \lambda (A(\boldsymbol{e}_{j_{t+1}} - \boldsymbol{w}_t))^\top \nabla \tilde{f}^\star(-A\boldsymbol{w}_t) - 2\eta\lambda^2, \qquad \forall \lambda \in [0, 1].
\end{aligned}$$

Optimizing λ in RHS and applying the inequality (9), we get $\epsilon_t - \epsilon_{t+1} \geq \epsilon_t^2 / 8\eta$. With this inequality, one can easily verify that the same iteration bound (10) holds for this case. ■

Algorithm 4 Pairwise rule $\mathcal{F}(A, \boldsymbol{w}_t, \boldsymbol{e}_{j_{t+1}}, \mathcal{E}_t, \boldsymbol{d}_t)$

```
1: Let $\boldsymbol{w}_t = \sum_{\boldsymbol{e} \in E_t} \alpha_{t,\boldsymbol{e}} \boldsymbol{e}$ be the current representation of $\boldsymbol{w}_t$ w.r.t. the basis vectors $E_t \subset \mathcal{E}_t$
   with positive coefficients $\{\alpha_{t,\boldsymbol{e}}\}_{\boldsymbol{e} \in E_t}$.
2: Compute an away basis $\boldsymbol{e}^{\text{Away}} \in \arg\min_{\boldsymbol{e} \in E_t} \boldsymbol{d}_t^\top A \boldsymbol{e}$ and set $\lambda_{t,\max} = \alpha_{t,\boldsymbol{e}^{\text{Away}}}$.
3: Compute the step size $\lambda_t \leftarrow \arg\min_{\lambda \in [0, \lambda_{t,\max}]} \tilde{f}^\star(-A(\boldsymbol{w}_t + \lambda(\boldsymbol{e}_{t+1} - \boldsymbol{e}^{\text{Away}})))$.
Output: $\boldsymbol{w}_{t+1}^{(1)} = \boldsymbol{w}_t + \lambda_t(\boldsymbol{e}_{j_{t+1}} - \boldsymbol{e}^{\text{Away}})$.
```

C.4. Pairwise rule

The FW update rule $\boldsymbol{w}_{t+1}^{(1)}$ of algorithm 3 comes from the FW algorithm with short-step sizes. One can apply other update rules as $\mathcal{F}$. The pairwise Frank-Wolfe (PFW) is the one of a state-of-the-art Frank-Wolfe algorithm [6]. The basic idea of PFW is to move the weight from the most worthless hypothesis to the newly attained one. We can derive a similar bound for this case. See appendix. As described in the main section, one can apply the other variants of the FW algorithm. Algorithm 4 summarizes this variant.

Appendix D. Additional experiments

This section includes experiments, not in the main paper. MLPB. (PFW) in the figures and tables corresponds to MLPBoost with the PFW algorithm 4.

Settings. We set the capping parameters $\nu \in \{pm \mid p = 0.1, 0.2, \ldots, 0.5\}$ and the tolerance parameter $\epsilon = 0.01$, where m is the number of training instances. We use the weak learner that returns the best decision tree of depth 2. We measure the CPU time and the System time using `/usr/bin/time -v` command. We measure the running time with capping parameters over the capping parameters for each dataset and took their average. Table 4 shows the results. As this table indicates, MLPB. (SS) is faster than ERLPB. We first show the test error comparison. We performed 5-fold cross validation to find the best capping parameter $\nu \in [1, m]$. After finding the best one, we trained the model on overall dataset and measure the test data. Table 3 shows the result. Our algorithm somehow achieves the smallest test error for most dataset.

Next we show the time comparison for all dataset. Table 4 summarizes the result.

Figure 1 shows the convergence curve. As expected, our algorithm converges faster than ERLPBoost and is competitive with LPBoost.

Further, we compare the test error decrease. Figure 2 shows the test error curves. Our algorithm somehow achieves low test errors in most datasets.

Now, we compare the Frank-Wolfe algorithms, SS and PFW. SS indicates the *short-steps* strategy, and PFW indicates the *pairwise* strategy. Figure 3 shows the number of iterations that adopt $\boldsymbol{w}_{t+1}^{(2)}$ as the next iterate.

Finally, we verify the effectiveness of the secondary rule, $\boldsymbol{w}_{t+1}^{(2)}$. For comparison, we measured the soft margin objective and time for MLPB. (SS), MLPB. (PFW), FW, and PFW. FW is the classic FW algorithm with short-step strategy and PFW is the Pairwise FW algorithm. Note that FW is equivalent to C-ERLPBoost. Figure 4 shows the curve. As this figure shows, the secondary rule, $\boldsymbol{w}_{t+1}^{(2)}$ improves the objective value significantly.

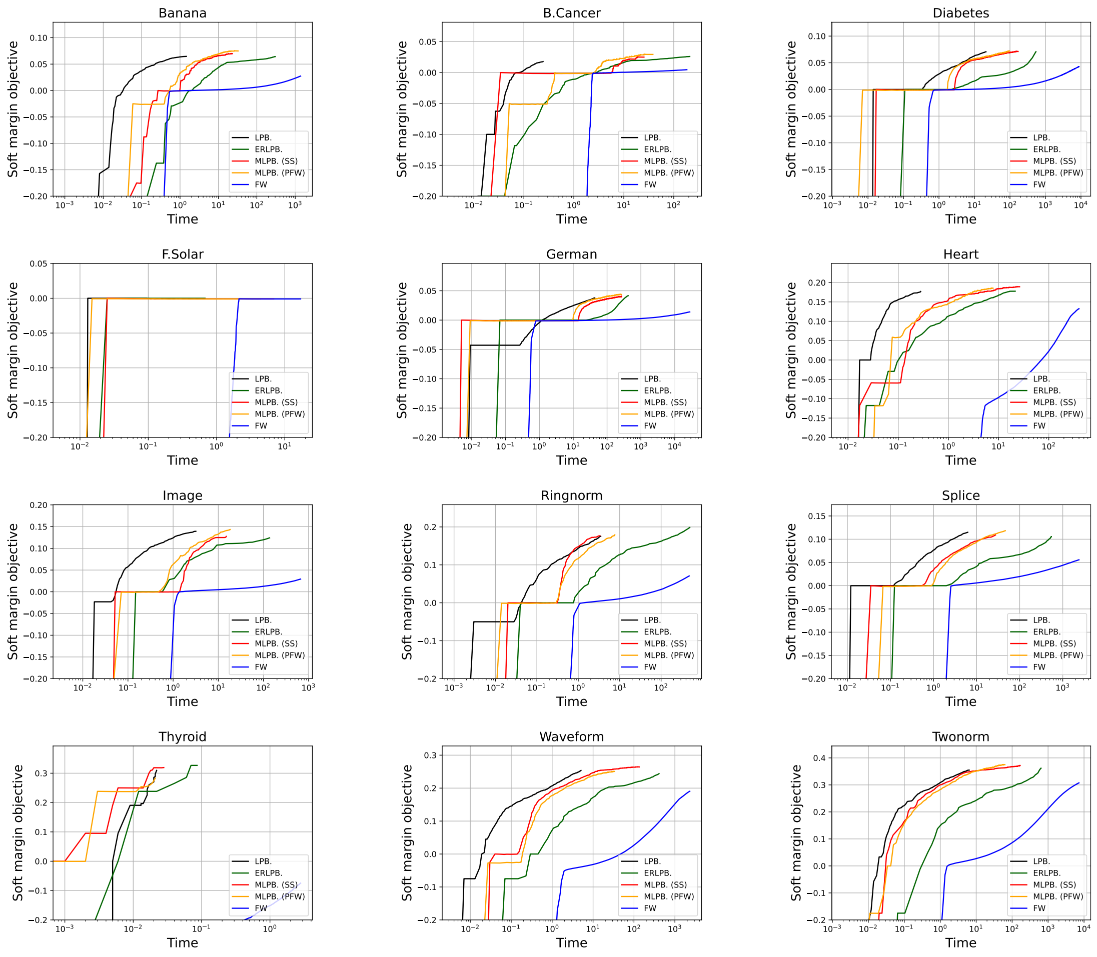

Figure 1: Time vs. soft margin objective with parameters $\nu = 0.1m$ and $\epsilon = 0.01$. Note that the time axis is log-scale. For many datasets, MLPBoost tends to achieve a large margin rapidly.

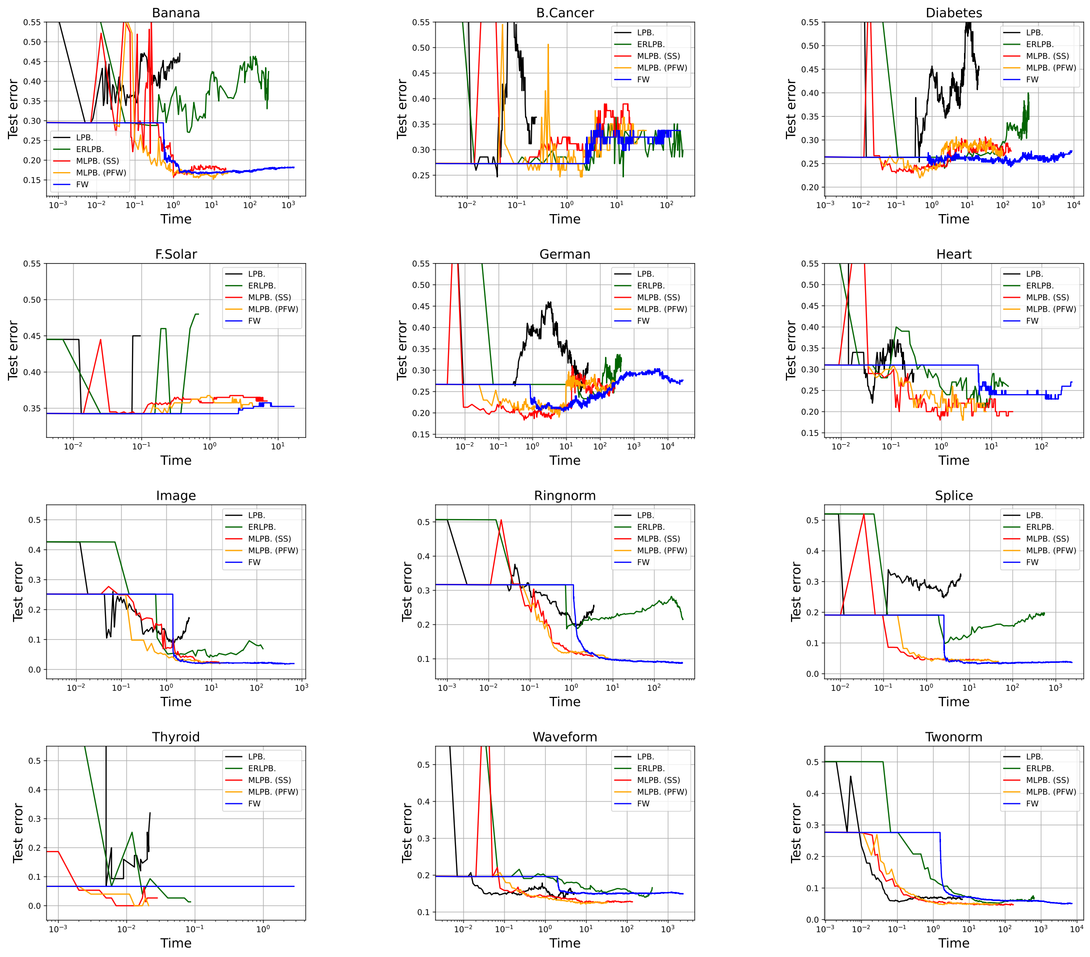

Figure 2: Time vs. test errors for the 0th fold of each dataset with parameters $\nu = 0.1m$ and $\epsilon = 0.01$. Note that the time axis is log-scale. For many datasets, MLPBoost tends to decrease the test error.

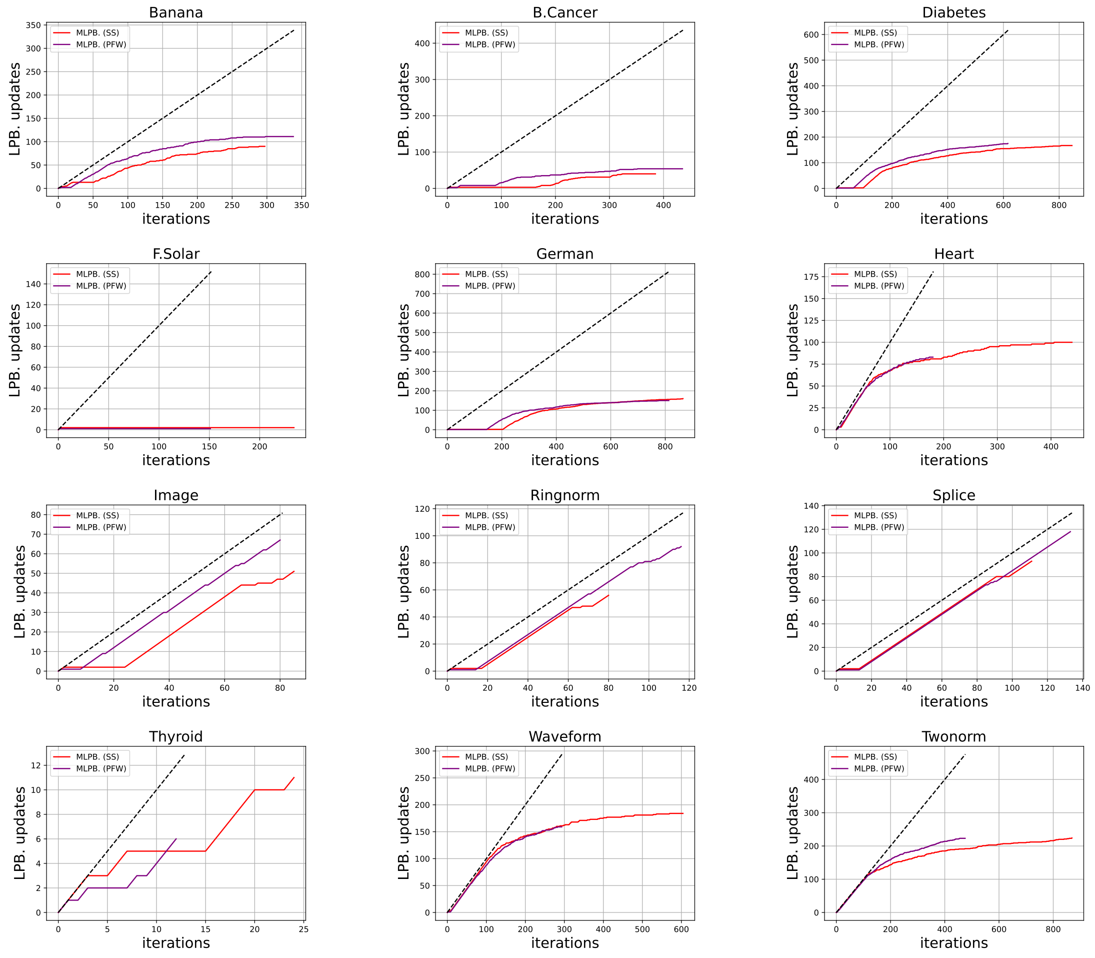

Figure 3: The number of $\mathcal{B}$ updates for each benchmark dataset with parameters $\nu = 0.1m$ and $\epsilon = 0.01$. The dotted line indicates the linear function for comparison. Since the shape of the titanic dataset is the same as the F. Solar dataset, we omit it.

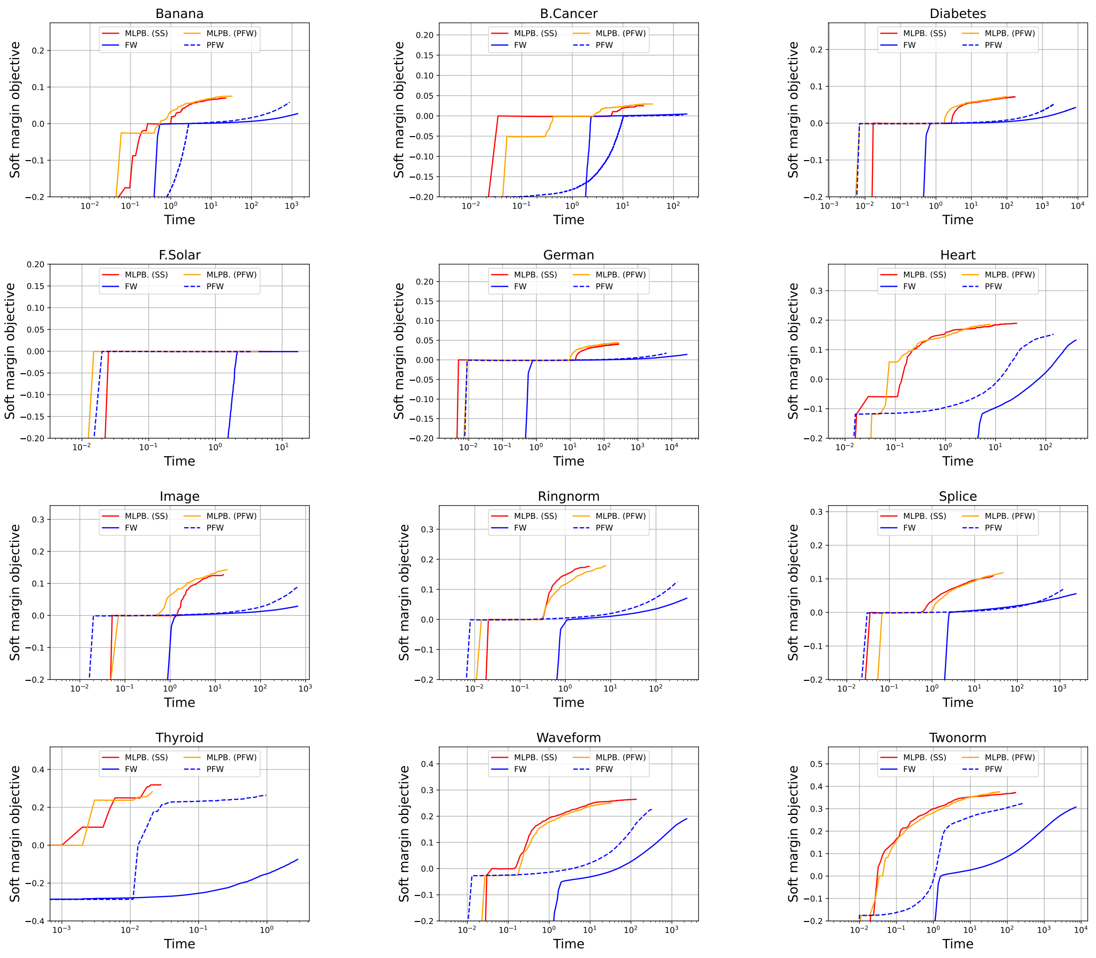

Figure 4: Comparison of MLPBoost algorithms. As this figure shows, $\boldsymbol{w}_t^{(2)}$ update yields huge progress.

Table 3: Test errors for 5-fold cross validation for the best parameters for each algorithms.

	LPB.	ERLPB.	MLPB. (SS)
Banana	0.28	0.37	0.10
B.Cancer	0.40	0.49	0.28
Diabetes	0.26	0.26	0.24
F.Solar	0.38	0.52	0.69
German	0.28	0.35	0.27
Heart	0.24	0.29	0.17
Image	0.10	0.20	0.02
Ringnorm	0.18	0.18	0.03
Splice	0.11	0.10	0.05
Thyroid	0.09	0.05	0.05
Titanic	0.60	0.60	0.60
Twonorm	0.03	0.04	0.03

Table 4: Comparison of the computation time (seconds). Each cell is the average computation time over the capping parameters over N. Some algorithm does not terminate in a few hours so we abort them within some appropriate time.

	m	LPB.	ERLPB.	FW	PFW	MLPB. (SS)	MLPB. (PFW)
Banana	5300	168.26	3434.75	$> 10^4$	$> 10^4$	1418.41	1398.68
B.Cancer	263	3.61	73.45	180.16	270.50	23.43	19.81
Diabetes	768	47.53	1478.77	$> 10^4$	3471.77	201.46	270.51
F.Solar	144	2.30	2.46	13.34	80.73	31.64	46.45
German	1000	77.56	1391.91	$> 10^4$	5692.32	181.43	201.88
Heart	270	10.03	193.58	$> 10^3$	183.09	44.11	24.26
Image	2086	8.25	107.52	$> 10^3$	502.83	32.01	10.51
R.norm	7400	22.09	1148.16	$> 10^4$	3350.87	26.76	36.73
Splice	2991	19.35	490.92	$> 10^4$	943.98	122.08	37.88
Thyroid	215	0.70	0.66	367.51	0.35	2.71	0.61
Titanic	24	0.25	0.13	0.58	0.10	1.96	0.12
Twonorm	7400	105.40	$> 10^4$	$> 10^4$	989.54	478.22	397.91
Waveform	5000	437.29	9018.54	$> 10^4$	$> 10^4$	2243.07	1619.56