Preprint – IJCAI 2025: This is the accepted version made available for conference attendees.
Do not cite. The final version will appear in the IJCAI 2025 proceedings.

Smoothed Online Convex Optimization with Delayed Feedback

Sifan Yang[1,2], **Wenhao Yang**[1,2], **Wei Jiang**[1], **Yuanyu Wan**[3] and **Lijun Zhang**[1,2*]

[1]National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China
[2]School of Artificial Intelligence, Nanjing University, Nanjing 210023, China
[3]School of Software Technology, Zhejiang University, Ningbo 315100, China
{yangsf, yangwh, jiangw, zhanglj}@lamda.nju.edu.cn, wanyy@zju.edu.cn

Abstract

Smoothed online convex optimization (SOCO), in which the online player incurs both a hitting cost and a switching cost for changing its decisions, has garnered significant attention in recent years. While existing studies typically assume that the gradient information is revealed immediately, such an assumption may not hold in some real-world applications. To overcome this limitation, we investigate SOCO with delayed feedback, and develop two online algorithms that can minimize the dynamic regret with switching cost. Firstly, we extend Mild-OGD, an existing algorithm that adopts the meta-expert framework for online convex optimization with delayed feedback, to account for switching cost. Specifically, we analyze the switching cost in the expert-algorithm of Mild-OGD, and then modify its meta-algorithm to incorporate this cost when assigning the weight to each expert. We demonstrate that our proposed method, Smelt-DOGD can achieve an $O(\sqrt{dT(P_T+1)})$ dynamic regret bound with switching cost, where d is the maximum delay and P_T is the path-length. Secondly, we develop an efficient variant to reduce the number of projections per round from $O(\log T)$ to 1, yet maintaining the same theoretical guarantee. The key idea is to construct a new surrogate loss defined over a simpler domain for expert-algorithms so that these experts do not need to perform the complex projection operations in each round. Finally, we conduct experiments to validate the effectiveness and efficiency of our algorithms.

1 Introduction

This paper investigates smoothed online convex optimization (SOCO) [Goel and Wierman, 2019], a popular variant of online convex optimization (OCO) [Shalev-Shwartz, 2012]. SOCO is motivated by real-world scenarios where the changes in states introduces additional costs, such as thermal management [Zanini *et al.*, 2010], dynamic right-sizing for data centers [Lin *et al.*, 2011], video streaming [Joseph and de Veciana, 2012], spatiotemporal sequence prediction [Kim *et al.*, 2015], speech animation [Kim *et al.*, 2015], etc. Specifically, SOCO is typically formulated as a game between a player and an adversary. In each round $t \in [T]$, the player begins by selecting a decision $\mathbf{x}_t$ from a convex feasible set $\mathcal{X} \subseteq \mathbb{R}^n$, where n is the dimensionality. When the online player makes a decision $\mathbf{x}_t$, the adversary simultaneously chooses a convex loss function $f_t(\cdot)\colon \mathcal{X} \mapsto \mathbb{R}$. The player then incurs both a hitting cost $f_t(\mathbf{x}_t)$ and a switching cost $m(\mathbf{x}_t, \mathbf{x}_{t-1})$ for changing its decisions between rounds. For general convex functions, a natural choice of switching cost is the distance between the successive decisions, i.e., $m(\mathbf{x}_t, \mathbf{x}_{t-1}) = \|\mathbf{x}_t - \mathbf{x}_{t-1}\|$ [Zhang *et al.*, 2022].

Following the previous work [Chen *et al.*, 2018; Zhang *et al.*, 2021], we adopt the dynamic regret with switching cost to measure the performance of the online player, which is formally defined as:

$$\begin{aligned}&\text{D-R-SC}(\mathbf{u}_1, ..., \mathbf{u}_T)\\ &= \sum_{t=1}^{T} \left(f_t(\mathbf{x}_t) + \|\mathbf{x}_t - \mathbf{x}_{t-1}\|\right) - \sum_{t=1}^{T} f_t(\mathbf{u}_t),\end{aligned} \tag{1}$$

where $\mathbf{u}_1, ..., \mathbf{u}_T \in \mathcal{X}$ is a sequence of any feasible comparators. Over the past years, a few algorithms [Li *et al.*, 2018; Chen *et al.*, 2018; Goel *et al.*, 2019; Zhang *et al.*, 2021; Zhang *et al.*, 2022] have been proposed to minimize the dynamic regret with switching cost for SOCO. The optimal dynamic regret bound for SOCO is $O(\sqrt{T(P_T+1)})$ [Zhang *et al.*, 2021], where

$$P_T(\mathbf{u}_1, ..., \mathbf{u}_T) = \sum_{t=1}^{T} \|\mathbf{u}_t - \mathbf{u}_{t-1}\| \tag{2}$$

denotes the path length of an arbitrary comparator sequence $\mathbf{u}_1, ..., \mathbf{u}_T \in \mathcal{X}$.

However, existing work typically assumes that information about the hitting cost, such as the gradient $\nabla f_t(\mathbf{x}_t)$, can be obtained immediately after the player makes a decision $\mathbf{x}_t$, which is often hard to satisfy in some practical scenarios. For example, in online advertising, the decision involves selecting ad delivery strategies for users. Advertisers adjust their strategies based on the interactions of users, like advertisement clicks, but there is often a delay in receiving the feedback. [McMahan *et al.*, 2013; Joulani *et al.*, 2013;

*Lijun Zhang is the corresponding author.

Preprint – IJCAI 2025: This is the accepted version made available for conference attendees.
Do not cite. The final version will appear in the IJCAI 2025 proceedings.

He *et al.*, 2014; Wan *et al.*, 2022a; Wan *et al.*, 2022c; Wan *et al.*, 2023; Yang *et al.*, 2025]. Motivated by this issue, we focus on SOCO with arbitrary delays, in which the gradient of the loss function $\nabla f_t(\mathbf{x}_t)$ arrives at the end of the round $t + d_t - 1$, where $d_t \geq 1$ represents an arbitrary delay at round t.

Delayed feedback has been investigated in the standard OCO setting over the past decades [Weinberger and Ordentlich, 2002; Joulani *et al.*, 2013; Quanrud and Khashabi, 2015; Korotin *et al.*, 2020; Wang *et al.*, 2021; Wan *et al.*, 2022b; Wan *et al.*, 2022c]. Recently, Wan *et al.* [2024] propose a delayed variant of OGD [Zinkevich, 2003], termed delayed online gradient descent (DOGD), and establish an $O(\sqrt{dT}(P_T+1))$ dynamic regret bound, where d is the maximum delay and P_T is the path-length defined in (2). To further enhance the dynamic regret bound, they develop a two-layer structured algorithm, named as Mild-OGD, which runs several DOGD instances as the expert-algorithms at the same time and uses delayed Hedge [Korotin *et al.*, 2020] as the meta-algorithm to track the best one. Mild-OGD achieves an $O(\sqrt{dT(P_T+1)})$ dynamic regret bound. However, it lacks consideration for the switching cost, which restricts its applicability in SOCO. Additionally, owing to maintaining $O(\log T)$ expert-algorithms simultaneously, Mild-OGD requires multiple projections onto the feasible domain per round, which could be computationally expensive when the feasible domain is complex.

In this paper, we develop a smoothed variant of Mild-OGD, named as smoothed multiple delayed online gradient descent (Smelt-DOGD), to deal with SOCO under delayed feedback. Similar to Mild-OGD, Smelt-DOGD adopts the meta-expert framework but explicitly accounts for the switching cost. Specifically, we prove that the expert-algorithm, DOGD, can obtain an $O(\sqrt{dT}(P_T+1))$ dynamic regret bound with switching cost. Then we modify the meta-algorithm by additionally considering the switching cost when deriving the weight for each expert. Smelt-DOGD achieves an $O(\sqrt{dT(P_T+1)})$ dynamic regret bound, which is on the same order as the dynamic regret bound without switching cost [Wan *et al.*, 2024]. Moreover, if the delay does not alter the arrival order of gradients, the dynamic regret bound with switching cost can be further improved to $O(\sqrt{S(P_T+1)})$, where $S = \sum_{t=1}^{T} d_t \leq dT$ represents the sum of delays.

Furthermore, to develop an efficient method, we employ a black-box technique [Cutkosky and Orabona, 2018; Cutkosky, 2020; Zhao *et al.*, 2022], which reduces the projection complexity from $O(\log T)$ to 1. Particularly, we simplify the optimization task over the complicated domain $\mathcal{X}$ to an optimization problem within the smallest Euclidean ball $\mathcal{Y}$ encompassing $\mathcal{X}$ so that the projection onto $\mathcal{Y}$ is much cheaper. We construct a surrogate loss over the domain $\mathcal{Y}$, which is minimized by the expert-algorithm. In this way, the expert-algorithm only performs a simple rescaling operation instead of the complex projection and the meta-algorithm projects the weighted decision onto the domain $\mathcal{X}$ just once per round. The theoretical analysis indicates that the efficient variant achieves a regret bound of the same order as Smelt-DOGD. Finally, we conduct numerical experiments on online classification and online regression problems, and the results demonstrate the superiority and efficiency of our algorithms.

2 Related Work

In this section, we provide a brief review of related research in dynamic regret, SOCO and OCO with arbitrary delays.

2.1 Dynamic Regret

Dynamic regret is introduced by Zinkevich [2003] to deal with the changing environment, where the optimal decision may vary over time. It is defined as the difference between the loss of the online player and a sequence of any comparators $\mathbf{u}_1, ..., \mathbf{u}_T \in \mathcal{X}$, typically formulated as $\sum_{t=1}^{T} f_t(\mathbf{x}_t) - \sum_{t=1}^{T} f_t(\mathbf{u}_t)$. Zinkevich [2003] first establishes an $O(\sqrt{T}(1+P_T))$ dynamic regret bound for OGD, where P_T is defined in (2). To further reduce the dynamic regret, Zhang *et al.* [2018a] propose a two-layer algorithm, named Ader, which achieves the optimal $O(\sqrt{T(P_T+1)})$ dynamic regret bound. Specifically, Ader runs $O(\log T)$ OGD instances with different learning rates simultaneously and combines them using Hedge [Freund and Schapire, 1997]. In recent years, several studies have achieved tighter dynamic regret bounds by leveraging the curvature of loss functions, such as exponential concavity [Baby and Wang, 2021] and strong convexity [Baby and Wang, 2022]. This two-layer structure has gained widespread adoption in recent years [Zhang *et al.*, 2020; Zhang *et al.*, 2021; Wang *et al.*, 2024; Yang *et al.*, 2024a].

While $\mathbf{u}_1, ..., \mathbf{u}_T \in \mathcal{X}$ can be an arbitrary sequence in dynamic regret, several studies [Jadbabaie *et al.*, 2015; Mokhtari *et al.*, 2016; Zhang *et al.*, 2017; Zhang *et al.*, 2018b; Wan *et al.*, 2021; Wang *et al.*, 2021] investigate the restricted dynamic regret where the comparator sequence for dynamic regret is minimizers of the online functions, i.e., $\mathbf{u}_t = \arg\min_{\mathbf{x}\in\mathcal{X}} f_t(\mathbf{x})$. However, as mentioned in Zhang *et al.* [2018a], the restricted dynamic regret is too pessimistic and may lead to overfitting in some problems.

2.2 Smoothed Online Convex Optimization

Due to the fact that changes of decisions may introduce additional cost in numerous real-world scenarios, SOCO has received extensive attention in the machine learning community. In the literature, there are two performance metrics designed for SOCO: competitive ratio and dynamic regret with switching cost. While various algorithms have been proposed to reduce the competitive ratio, all of them are limited to the lookahead setting where the learner can observe the hitting cost $f_t(\cdot)$ before making the decision, and may rely on strict conditions, such as the strong convexity and quadratic growth [Lin *et al.*, 2011; Bansal *et al.*, 2015; Chen *et al.*, 2015; Li *et al.*, 2018; Goel and Wierman, 2019; Li *et al.*, 2020; Zhang *et al.*, 2021; Wang *et al.*, 2021]. For this reason, our subsequent discussion will primarily focus on the dynamic regret with switching cost, which requires weaker assumptions.

Li *et al.* [2018] consider a setting where the online player has access to the next W hitting costs and propose receding horizon gradient descent (RHGD) to minimize the dy-

Preprint – IJCAI 2025: This is the accepted version made available for conference attendees.
Do not cite. The final version will appear in the IJCAI 2025 proceedings.

namic regret with switching cost. However, they choose the switching cost in the form of a quadratic function, i.e., $\|\mathbf{x}_t - \mathbf{x}_{t-1}\|^2/2$, which may not be well-suited for general convex functions. Moreover, their analysis relies on the stringent conditions, like the strong convexity and smoothness, which are difficult to satisfy in practical applications. Online balanced descent (OBD) [Chen *et al.*, 2018], a classic algorithm in SOCO, achieves an $O(\sqrt{TL})$ dynamic regret bound with switching cost, where L is the upper bound of the path-length, i.e., $P_T = \sum_{t=1}^{T} \|\mathbf{u}_t - \mathbf{u}_{t-1}\| \leq L$. Notably, this upper bound is nonadaptive as it depends on L, rather than the actual path-length P_T. Later, regularized OBD (R-OBD) [Goel *et al.*, 2019] improves the dynamic regret bound with switching cost to $O(\sqrt{TP_{T,\ell_2}})$, where $P_{T,\ell_2} = \sum_{t=1}^{T} \|\mathbf{u}_t - \mathbf{u}_{t-1}\|^2$. Although the regret bound of R-OBD is adaptive, R-OBD chooses the squared ℓ_2-norm as the switching cost. To overcome this limitation, Zhang *et al.* [2021] propose Smoothed Ader (SAder), which is a smoothed variant of Ader. Similar to Ader [Zhang *et al.*, 2018a], SAder employs a two-layer structure and is proven to attain an $O(\sqrt{T(P_T+1)})$ dynamic regret bound with switching cost.

2.3 OCO with Delayed Feedback

Weinberger and Ordentlich [2002] first consider the setting in which the feedback of each decision $\mathbf{x}_t$ will be received at end of round $t + d' - 1$, where d' is a fixed delay, and develop a method for adapting classic OCO algorithms to delayed setting. They choose the static regret as the performance metric and demonstrate that if a vanilla OCO algorithm achieves a regret bound of $R(T)$ in non-delayed setting, this method can obtain a regret bound of $d'R(T/d')$. However, in practice, the delays are typically arbitrary, which limits the applicability of this method.

To deal with OCO with arbitrary delays, Joulani *et al.* [2013] extend the above technique, which can attain a regret bound of $dR(T/d)$ for traditional OCO algorithm with $R(T)$ regret, where d is the maximum delay. Later, Quanrud and Khashabi [2015] propose a variant of OGD to handle the OCO with delays, which achieves a static regret bound of $O(\sqrt{S})$, where $S = \sum_{t=1}^{T} d_t \geq T$ is the sum of delays over T rounds. In recent years, Korotin *et al.* [2020] consider the problem of prediction with expert advice and propose delayed Hedge, achieving an $O(\sqrt{S})$ static regret bound. However, all the aforementioned studies focus on the static regret, which is only meaningful for environments where at least one fixed decision can minimize the cumulative loss, and is not suitable for non-stationary environments in which the optimal decision varies over time.

To address this issue, Wan *et al.* [2024] investigate the dynamic regret under delayed OCO and develop DOGD, a variant of OGD, which can achieve a dynamic regret bound of $O(\sqrt{dT}(P_T+1))$. DOGD queries the gradient $\nabla f_t(\mathbf{x}_t)$ at round t and because of delayed feedback, it only uses each gradient received at round t to perform a gradient descent step. To further reduce the dynamic regret bound, Wan *et al.* [2024] design Mild-OGD, which runs several DOGD instances at the same time and uses delayed Hedge [Korotin *et al.*, 2020] to track the best one. While Mild-OGD is proven to obtain an optimal $O(\sqrt{dT(P_T+1)})$ dynamic regret bound, it is inefficient as it performs $O(\log T)$ projection operations in each round.

3 Main results

In this section, we first start with necessary assumptions and then present our Smelt-DOGD with theoretical guarantees. Finally, we develop an efficient version of Smelt-DOGD.

3.1 Assumptions

We adopt the common assumptions of online convex optimization (OCO) [Shalev-Shwartz, 2012].

Assumption 1. ***(Convexity)*** *All loss functions $f_t(\cdot)$ are convex in the domain $\mathcal{X}$.*

Assumption 2. ***(Bounded gradient norms)*** *The norm of the gradients of all loss functions over the domain $\mathcal{X}$ is bounded by G, i.e., $\|\nabla f_t(\cdot)\| \leq G, \forall t \in [T]$.*

Assumption 3. ***(Bounded domain)*** *The diameter of the domain $\mathcal{X}$ is at most D, i.e., $\max_{\mathbf{x},\mathbf{y}\in\mathcal{X}} \|\mathbf{x}-\mathbf{y}\| \leq D$, and $\mathbf{0} \in \mathcal{X}$.*

While the above assumptions is enough to handle SOCO with delayed feedback, previous work [McMahan and Streeter, 2014; Wan *et al.*, 2024] additionally introduces the following assumption to achieve a tighter regret bound.

Assumption 4. ***(In-Order property)*** *The delay of queries does not alter the arrival order of the gradients, i.e., for any $1 \leq i < j \leq T$, $\nabla f_i(\mathbf{x}_i)$ arrives before $\nabla f_j(\mathbf{x}_j)$.*

Although Assumption 4 may appear overly stringent, it is indeed satisfied in certain applications. For instance, in parallel and distributed optimization, the delay d_t in each round tends to gradually increase, i.e., $d_t \leq d_{t+1}$, meaning that the gradient queried first is more likely to arrive first [McMahan and Streeter, 2014; Zhou *et al.*, 2018]. Therefore, it is reasonable to assume that the delay enjoys the In-Order assumption. Notably, we need to emphasize that Assumption 4 is not always required and we will explicitly state the requirement in subsequent theorems.

3.2 Smelt-DOGD

Due to the lack of consideration for switching cost, Mild-OGD cannot be directly applied to SOCO with delayed feedback. To overcome this limitation, we first provide a novel analysis of its expert-algorithm, DOGD, and then modify the meta-algorithm to account for the switching cost. Our proposed method is outlined below.

Expert-algorithm. As described in Algorithm 2, we choose DOGD as the expert-algorithm. The input of expert-algorithm is the learning rate η. Due to the delayed feedback, we receive a set of gradients $\{\nabla f_k(\mathbf{x}_k) \mid k \in \mathcal{F}_t\}$ in each round t, where $\mathcal{F}_t = \{k \in [T] \mid k + d_k - 1 = t\}$ represents the index set of queried gradients arriving in round t and the elements in this set are sorted in the ascending order. For each gradient received in round t, the expert-algorithm performs a

Preprint – IJCAI 2025: This is the accepted version made available for conference attendees.
Do not cite. The final version will appear in the IJCAI 2025 proceedings.

Algorithm 1 Smelt-DOGD: Expert-algorithm

Require: A learning rate η
1: Initialization $\mathbf{z}_1^\eta = 0$ and $\tau = 1$
2: **for** time step $t = 1$ to T **do**
3: Submit $\mathbf{x}_t^\eta = \mathbf{z}_\tau^\eta$ to the meta-algorithm
4: Receive gradients $\{\nabla f_k(\mathbf{x}_k)|k \in \mathcal{F}_t\}$ from the meta-algorithm
5: **for** $k \in \mathcal{F}_t$ **do**
6: Compute $\mathbf{z}_{\tau+1}^\eta = \Pi_{\mathcal{X}}\left[\mathbf{z}_\tau^\eta - \eta \nabla f_k(\mathbf{x}_k)\right]$
7: Set $\tau = \tau + 1$
8: **end for**
9: **end for**

gradient descent update based on its arrival order and then projects the result onto the feasible domain, i.e.,

$$\mathbf{z}_{\tau+1}^\eta = \Pi_{\mathcal{X}}\left[\mathbf{z}_\tau^\eta - \eta \nabla f_k(\mathbf{x}_k)\right], \tag{3}$$

where $\Pi_{\mathcal{X}}[\cdot]$ is the projection onto the domain $\mathcal{X}$.

Although Wan *et al.* [2024] have established the dynamic regret bound for DOGD, they do not consider the switching cost. In the following, we derive the dynamic regret bound with switching cost for DOGD.

Theorem 1. *Under Assumptions 1, 2 and 3, for any comparator sequence $\mathbf{u}_1, ..., \mathbf{u}_T \in \mathcal{X}$, by setting $\eta = \frac{D}{G\sqrt{\sum_{t=1}^T m_t}}$, DOGD ensures*

$$\begin{aligned}\text{D-R-SC}(\mathbf{u}_1, ..., \mathbf{u}_T) &\leq \frac{D^2 + DP_T}{\eta} + \eta G^2 \sum_{t=1}^T m_t \\ &\quad + C + \eta G T \\ &\leq O(\sqrt{S}(P_T + 1) + C),\end{aligned}$$

where $m_t = t - \sum_{i=1}^t |\mathcal{F}_i|$, $S = \sum_{t=1}^T d_t \leq dT$ is the sum of delays, d is the maximum delay and

$$C = \begin{cases} 0, \textit{ if Assumption 4 holds,} \\ \min\{DGT, 2dGP_T\}, \textit{ otherwise.} \end{cases} \tag{4}$$

Remark 1. From the analysis of Wan *et al.* [2024], it is direct to verify that $\min\{DGT, 2dGP_T\} \leq G\sqrt{2dTDP_T}$ and $\sum_{t=1}^T m_t \leq \sum_{t=1}^T d_t = S \leq dT$. Therefore, Theorem 1 implies that the dynamic regret with switching cost of DOGD is $O(\sqrt{dT}(P_T+1))$ in the worst case and can achieve a better regret bound of $O(\sqrt{S}(P_T + 1))$ when Assumption 4 holds.

Remark 2. It is worth noting that if we set the learning rate as $\eta = \frac{\sqrt{D(D+P_T)}}{G\sqrt{\sum_{t=1}^T m_t}}$, DOGD enjoys an $O(\sqrt{dT(P_T+1)})$ regret bound, which is better than the bound in Theorem 1. This indicates that if we know the path-length beforehand, we can tune the learning rate to obtain an improved bound. To deal with the uncertainty of the path-length, we choose to run multiple instances of DOGD with different learning rates and track the best one using the meta-algorithm outlined below.

Meta-algorithm. As summarized in Algorithm 2, the meta-algorithm takes two input parameters: the hyper-parameter step size α and the set of learning rates for experts $\mathcal{H}$. We

Algorithm 2 Smelt-DOGD: Meta-algorithm

Require: A parameter α and a set $\mathcal{H}$ containing learning rates for experts
1: Activate a set of experts $\{E^{\eta_i}|\eta_i \in \mathcal{H}\}$ by invoking the expert-algorithm for each learning rate $\eta_i \in \mathcal{H}$
2: Sort learning rates in the ascending order and set $w_1^{\eta_i} = \frac{|\mathcal{H}|+1}{i(i+1)|\mathcal{H}|}$, let $\mathbf{x}_1, \{\mathbf{x}_1^{\eta_i}\}_{i=1}^N$ be any point in $\mathcal{X}$
3: **for** time step $t = 1$ to T **do**
4: Receive $\mathbf{x}_t^{\eta_i}$ from each expert E^{η_i}
5: Play the decision $\mathbf{x}_t = \sum_{\eta_i \in \mathcal{H}} w_t^{\eta_i} \mathbf{x}_t^{\eta_i}$
6: Query $\nabla f_t(\mathbf{x}_t)$ and receive $\{\nabla f_k(\mathbf{x}_k)|k \in \mathcal{F}_t\}$
7: Update the weight of each expert according to (6)
8: Send $\{\nabla f_k(\mathbf{x}_k)|k \in \mathcal{F}_t\}$ to each expert E^{η_i}
9: **end for**

first active a set of experts $\{E^{\eta_i}|\eta_i \in \mathcal{H}\}$ by invoking the expert-algorithm for each $\eta_i \in \mathcal{H}$ in Step 1. It's worth noting that the learning rates η_i are arranged in the ascending order, meaning $\eta_1 \leq ... \leq \eta_{|\mathcal{H}|}$. In Step 2, we initialize the weight $w_1^{\eta_i}$ of each expert as

$$w_1^{\eta_i} = \frac{|\mathcal{H}| + 1}{i(i+1)|\mathcal{H}|}. \tag{5}$$

In each round $t \in [T]$, the meta algorithm receives the decision $\mathbf{x}_t^{\eta_i}$ from each expert-algorithm E^{η_i} and then computes a weighted decision $\mathbf{x}_t$ based on these decisions: $\mathbf{x}_t = \sum_{\eta_i \in \mathcal{H}} w_t^{\eta_i} \mathbf{x}_t^{\eta_i}$. Subsequently, the meta-algorithm queries the gradient $\nabla f_t(\mathbf{x}_t)$ of the current decision $\mathbf{x}_t$. Due to the delayed feedback of the gradient, it only receives the results of previous queries $\{\nabla f_k(\mathbf{x}_k)|k \in \mathcal{F}_t\}$. The meta-algorithm then updates the weight of each expert algorithm $w_{t+1}^{\eta_i}$ according to:

$$w_{t+1}^{\eta_i} = \frac{w_t^{\eta_i} e^{-\alpha\left(\sum_{k \in \mathcal{F}_t} s_k(\mathbf{x}_k^{\eta_i}) + \left\|\mathbf{x}_t^{\eta_i} - \mathbf{x}_{t-1}^{\eta_i}\right\|\right)}}{\sum_{\mu \in \mathcal{H}} w_t^{\mu} e^{-\alpha\left(\sum_{k \in \mathcal{F}_t} s_k(\mathbf{x}_k^{\mu}) + \left\|\mathbf{x}_t^{\mu} - \mathbf{x}_{t-1}^{\mu}\right\|\right)}}, \tag{6}$$

where α is a parameter and $s_k(\mathbf{x}_k^{\eta_i}) = \langle \nabla f_k(\mathbf{x}_k), \mathbf{x}_k^{\eta_i} - \mathbf{x}_k \rangle$, in which we use the surrogate loss to avoid the inconsistent delay between the meta-algorithm and the expert-algorithm. Diverging from the Mild-OGD, Smelt-DOGD incorporates the switching cost into the loss of each expert-algorithm to measure its performance in (6). In the last, the meta-algorithm sends the queried gradient $\{\nabla f_k(\mathbf{x}_k)|k \in \mathcal{F}_t\}$ to each expert E^{η_i}. Then we present the following theorem to establish the theoretical guarantee for Smelt-DOGD.

Theorem 2. *Under Assumptions 1, 2 and 3, for any comparator sequence $\mathbf{u}_1, ..., \mathbf{u}_T \in \mathcal{X}$, by setting*

$$\mathcal{H} = \left\{\eta_i = \frac{2^{i-1}D}{G\sqrt{\beta}} | i = 1, ..., N\right\}, \alpha = \frac{1}{(G+1)D\sqrt{\beta}},$$

where $N = \lceil \frac{1}{2}\log_2(T+1) \rceil + 1$, $\beta = \sum_{t=1}^T m_t, m_t = t - \sum_{i=1}^t |\mathcal{F}_i|$, Smelt-DOGD ensures

$$\text{D-R-SC}(\mathbf{u}_1, ..., \mathbf{u}_T) \leq O(\sqrt{S(P_T + 1)} + C),$$

where $S = \sum_{t=1}^T d_t$ and C is defined in (4).

Preprint – IJCAI 2025: This is the accepted version made available for conference attendees.
Do not cite. The final version will appear in the IJCAI 2025 proceedings.

Remark 3. *Theorem 2 implies that Smelt-DOGD attains an $O(\sqrt{S(P_T+1)}+C)$ bound for dynamic regret with switching cost, which is on the same order as Mild-OGD. If Assumption 4 holds, Smelt-DOGD can achieve an regret bound of $O(\sqrt{S(P_T+1)})$. Conversely, if Assumption 4 is not satisfied, Smelt-DOGD obtains an $O(\sqrt{dT(P_T+1)})$ dynamic regret bound, where d is the maximum delay.*

3.3 Efficient Smelt-DOGD

In Smelt-DOGD, each expert-algorithm is required to project its decision point $\mathbf{z}_t^\eta$ onto the feasible domain $\mathcal{X}$ in (3) per round. Since Smelt-DOGD simultaneously runs $O(\log T)$ expert-algorithm instances, it requires $O(\log T)$ projections on average per round, which may be computationally expensive, especially when the feasible domain $\mathcal{X}$ is complex. To overcome the above-mentioned issue, we develop an efficient version of Smelt-DOGD by using a black-box technique [Cutkosky and Orabona, 2018; Cutkosky, 2020; Zhao *et al.*, 2022; Yang *et al.*, 2024b], which reduces the projection complexity from $O(\log T)$ to 1. The theoretical analysis demonstrates that the efficient variant achieves the same order of dynamic regret bound as Smelt-DOGD. Although this reduction method has been previously utilized to reduce the projection complexity under the standard OCO, our results demonstrate that it is also applicable to SOCO with delayed feedback for the first time.

The key idea of our reduction technique is to replace the expensive projection onto the feasible domain $\mathcal{X}$ in the expert-algorithm with a cheaper rescaling operation. Firstly, we introduce a simpler domain $\mathcal{Y}$ to replace the feasible domain $\mathcal{X}$. In particular, we choose $\mathcal{Y}$ as the minimum Euclidean ball containing the feasible domain, i.e., $\mathcal{Y} = \{\mathbf{x} \mid \|\mathbf{x}\| \leq D\} \supseteq \mathcal{X}$. Then, following the approach of the previous work [Cutkosky, 2020], we construct a surrogate loss function $g_t : \mathcal{Y} \mapsto \mathbb{R}$ over the domain $\mathcal{Y}$:

$$\begin{aligned} g_t(\mathbf{y}_t^\eta) = &\langle \nabla f_t(\mathbf{x}_t), \mathbf{y}_t^\eta - \mathbf{y}_t \rangle \\ &- \mathbb{I}_{\{\langle \nabla f_t(\mathbf{x}_t), \mathbf{v}_t \rangle < 0\}} \cdot \langle \nabla f_t(\mathbf{x}_t), \mathbf{v}_t \rangle \cdot M_{\mathcal{X}}(\mathbf{y}_t^\eta), \end{aligned} \tag{7}$$

where $M_{\mathcal{X}}(\mathbf{y}) = \inf_{\mathbf{x}\in\mathcal{X}} \|\mathbf{y} - \mathbf{x}\|$ is the distance between $\mathbf{y}$ and domain $\mathcal{X}$, $\mathbf{v}_t = (\mathbf{y}_t - \mathbf{x}_t) / \|\mathbf{y}_t - \mathbf{x}_t\|$ is a unit vector indicating the projection direction and $\mathbb{I}_{\{\cdot\}}$ is an indicator function defined as

$$\mathbb{I}_{\{A\}} := \begin{cases} 1 & \text{if } A \text{ holds,} \\ 0 & \text{else.} \end{cases}$$

Our expert-algorithm will minimize the surrogate loss $g_t(\mathbf{y}_t^\eta)$ so that it only performs a simpler rescaling operation to the surrogate domain $\mathcal{Y}$. In each round, we project the weighted decision point $\mathbf{y}_t$ back onto the feasible domain $\mathcal{X}$ only once in the meta-algorithm. Applying this reduction technique to Smelt-DOGD, we develop the efficient version of Smelt-DOGD, as described below.

Expert-algorithm. As shown in Algorithm 3, our expert-algorithm performs the gradient descent step using the gradient $\nabla g_k(\mathbf{y}_k)$ of the surrogate loss over the domain $\mathcal{Y}$. In each round, after receiving the gradient set $\{\nabla g_k(\mathbf{y}_k) | k \in \mathcal{F}_t\}$ from the meta-algorithm, the expert algorithm utilizes the gradient of the surrogate loss to update its own decisions. After we update the decision $\hat{\mathbf{z}}_{\tau+1}^\eta$ in Step 6, we can just use a scaling operation to project this decision into the surrogate domain $\mathcal{Y}$:

$$\mathbf{z}_{\tau+1}^\eta = \hat{\mathbf{z}}_{\tau+1}^\eta \left(\mathbb{I}_{\{\|\hat{\mathbf{z}}_{\tau+1}^\eta\| \leq D\}} + \frac{D}{\|\hat{\mathbf{z}}_{\tau+1}^\eta\|} \cdot \mathbb{I}_{\{\|\hat{\mathbf{z}}_{\tau+1}^\eta\| > D\}} \right), \tag{8}$$

which is more efficient than the traditional projection operation.

Algorithm 3 Efficient Smelt-DOGD: Expert-algorithm

Require: A learning rate η
1: Initialize $\mathbf{z}_1^\eta = \mathbf{0}$ and $\tau = 1$
2: **for** time step $t = 1$ to T **do**
3: Submit $\mathbf{y}_t^\eta = \mathbf{z}_\tau^\eta$ to the meta-algorithm
4: Receive gradients $\{\nabla g_k(\mathbf{y}_k) | k \in \mathcal{F}_t\}$ from the meta-algorithm
5: **for** $k \in \mathcal{F}_t$ **do**
6: Compute $\hat{\mathbf{z}}_{\tau+1}^\eta = \mathbf{z}_\tau^\eta - \eta \nabla g_k(\mathbf{y}_k)$
7: Project $\hat{\mathbf{z}}_{\tau+1}^\eta$ to the domain $\mathcal{Y}$ according to (8)
8: Set $\tau = \tau + 1$
9: **end for**
10: **end for**

Algorithm 4 Efficient Smelt-DOGD: Meta-algorithm

Require: A parameter α and a set $\mathcal{H}$ containing learning rates for experts
1: Activate a set of experts $\{E^{\eta_i} | \eta_i \in \mathcal{H}\}$ by invoking the expert-algorithm for each learning rate $\eta_i \in \mathcal{H}$
2: Set $w_1^{\eta_i}$ of each expert according to (5) and initialize $\mathbf{x}_1, \{\mathbf{y}_1^{\eta_i}\}_{i=1}^N$ to be any point in $\mathcal{X}$
3: **for** time step $t = 1$ to T **do**
4: Receive $\mathbf{y}_t^{\eta_i}$ from each expert E^{η_i}
5: Compute the decision $\mathbf{y}_{t+1} = \sum_{\eta_i \in \mathcal{H}} w_t^{\eta_i} \mathbf{y}_t^{\eta_i}$
6: Submit $\mathbf{x}_{t+1} = \Pi_{\mathcal{X}}[\mathbf{y}_{t+1}]$
7: Query $\nabla f_t(\mathbf{x}_t)$ and receive $\{\nabla f_k(\mathbf{x}_k) | k \in \mathcal{F}_t\}$
8: For each gradient in $\{\nabla f_k(\mathbf{x}_k) | k \in \mathcal{F}_t\}$, calculate $\{\nabla g_k(\mathbf{y}_k) | k \in \mathcal{F}_t\}$
9: Update the weight of each expert according to (9)
10: Send $\{\nabla g_k(\mathbf{x}_k) | k \in \mathcal{F}_t\}$ to expert E^{η_i}
11: **end for**

Meta-algorithm. Our meta-algorithm is presented in Algorithm 4. Similar to Smelt-DOGD, we first activate a set of experts $\{E^{\eta_i} | \eta_i \in \mathcal{H}\}$ with different learning rates. In each round t, the meta-algorithm receives decisions $\mathbf{y}_t^{\eta_i}$ from the expert algorithms and computes a weighted sum of these decisions:

$$\mathbf{y}_{t+1} = \sum_{\eta_i \in \mathcal{H}} w_t^{\eta_i} \mathbf{y}_t^{\eta_i}.$$

Since these decisions $\mathbf{y}_t^{\eta_i}$ lie within the domain $\mathcal{Y}$, the result $\mathbf{y}_{t+1}$ needs to be projected onto the feasible domain $\mathcal{X}$, which is the only projection performed in each round. After receiving the gradient information, we calculate the gradient $\{\nabla g_k(\mathbf{y}_k) | k \in \mathcal{F}_t\}$ using $\{\nabla f_k(\mathbf{x}_k) | k \in \mathcal{F}_t\}$. According to the Lemma 1 in Zhao *et al.* [2022],

Preprint – IJCAI 2025: This is the accepted version made available for conference attendees.
Do not cite. The final version will appear in the IJCAI 2025 proceedings.

it is not hard to verify that for any $\mathbf{y} \in \mathcal{Y}$, when $\langle \nabla f_t(\mathbf{x}_t), \mathbf{v}_t \rangle \geq 0$, we ensure $\nabla g_t(\mathbf{y}) = \nabla f_t(\mathbf{x}_t)$, where $\mathbf{v}_t = (\mathbf{y} - \Pi_{\mathcal{X}}[\mathbf{y}]) / \|\mathbf{y} - \Pi_{\mathcal{X}}[\mathbf{y}]\|$. When $\langle \nabla f_t(\mathbf{x}_t), \mathbf{v}_t \rangle < 0$, we have $\nabla g_t(\mathbf{y}) = \nabla f_t(\mathbf{x}_t) - \langle \nabla f_t(\mathbf{x}_t), \mathbf{v}_t \rangle \cdot \mathbf{v}_t$. Notably, when deriving the gradient $\nabla g_t(\mathbf{y}_t)$, we encounter the need to calculate $\Pi_{\mathcal{X}}[\mathbf{y}_t]$. However, we can simply utilize the previous results $\mathbf{x}_t$ from Step 6 in Algorithm 4. We also incorporate the switching cost of each expert algorithm over the domain $\mathcal{Y}$ into the surrogate loss function:

$$w_{t+1}^{\eta_i} = \frac{w_t^{\eta_i} e^{-\alpha\left(\sum_{k\in\mathcal{F}_t} h_k(\mathbf{y}_k^{\eta_i}) + \left\|\mathbf{y}_t^{\eta_i} - \mathbf{y}_{t-1}^{\eta_i}\right\|\right)}}{\sum_{\mu\in\mathcal{H}} w_t^{\mu} e^{-\alpha\left(\sum_{k\in\mathcal{F}_t} h_k(\mathbf{y}_k^{\mu}) + \left\|\mathbf{y}_t^{\mu} - \mathbf{y}_{t-1}^{\mu}\right\|\right)}}, \quad (9)$$

where

$$h_k(\mathbf{y}_k^{\eta_i}) = \langle \nabla g_k(\mathbf{y}_k), \mathbf{y}_k^{\eta_i} - \mathbf{y}_k \rangle.$$

From the above discussion, it is evident that our efficient Smelt-DOGD reduces the projection complexity from $O(\log T)$ to 1. Next, we present the dynamic regret bound with switching cost for the efficient Smelt-DOGD.

Theorem 3. *Under Assumptions 1, 2 and 3, for any comparator sequence $\mathbf{u}_1, ..., \mathbf{u}_T \in \mathcal{X}$, by setting*

$$\mathcal{H} = \left\{ \eta_i = \frac{2^i D}{G\sqrt{\beta}} | i = 1, ..., N \right\}, \alpha = \frac{1}{2(G+2)D\sqrt{\beta}},$$

where $N = \lceil \frac{1}{2}\log_2(T+1) \rceil + 1, \beta = \sum_{t=1}^{T} m_t, m_t = t - \sum_{i=1}^{t} |\mathcal{F}_i|$, efficient Smelt-DOGD ensures

$$\text{D-R-SC}(\mathbf{u}_1, ..., \mathbf{u}_T) \leq O(\sqrt{S(P_T+1)} + C),$$

where $S = \sum_{t=1}^{T} d_t$ and C is defined in (4).

Remark 4. Theorem 3 demonstrates that the efficient Smelt-DOGD can achieve an $O(\sqrt{S(P_T+1)} + C)$ dynamic regret with switching cost, which implies that it reduces the projection complexity of Smelt-DOGD from $O(\log T)$ to 1 while preserving the order of its regret bound.

4 Experiments

In this section, we evaluate the performance of the proposed Smelt-DOGD and efficient Smelt-DOGD methods against an existing algorithm for OCO with delayed feedback, Mild-OGD [Wan *et al.*, 2024], through numerical experiments. For the parameters, we set those of Smelt-DOGD and efficient Smelt-DOGD according to Theorem 2 and Theorem 3, respectively. As for Mild-OGD, we follow recommendations from Theorem 3 in Wan *et al.* [2024].

4.1 Online Classification

We implement the online classification on ijcnn1 dataset from LIBSVM Data [Chang and Lin, 2011; Prokhorov, 2001]. Let T denote the number of total rounds. In each round $t \in [T]$, a batch of training examples $\{(\mathbf{x}_{t,1}, y_{t,1}), \ldots, (\mathbf{x}_{t,m}, y_{t,m})\}$ arrive, where $(\mathbf{x}_{t,i}, y_{t,i}) \in [-1,1]^n \times \{-1, 1\}, i = 1, \ldots, m$. The online learner aims to predict a linear model $\mathbf{w}_t \in \mathcal{W}$ and suffers both a hitting cost $f_t(\mathbf{w}_t) = \frac{1}{m}\sum_{i=1}^{m} \left|\mathbf{w}_t^\top \mathbf{x}_{t,i} - y_{t,i}\right|$ and a switching cost $\|\mathbf{w}_t - \mathbf{w}_{t-1}\|$. In practical applications, there is often a trade-off in the switching cost, i.e., $\lambda\|\mathbf{w}_t - \mathbf{w}_{t-1}\|$, thus the total loss in each round is $H_t(\mathbf{w}_t) = \frac{1}{m}\sum_{i=1}^{m} \left|\mathbf{w}_t^\top \mathbf{x}_{t,i} - y_{t,i}\right| + \lambda\|\mathbf{w}_t - \mathbf{w}_{t-1}\|$. Although Smelt-DOGD only considers the setting where $\lambda = 1$, we can rescale the loss function to satisfy the requirement. Specifically, we redefine the hitting cost as $\frac{f_t(\mathbf{x}_t)}{\lambda}$, and our methods will minimize the following loss function

$$h_t(\mathbf{w}_t) = \frac{1}{\lambda m}\sum_{i=1}^{m} \left|\mathbf{w}_t^\top \mathbf{x}_{t,i} - y_{t,i}\right| + \|\mathbf{w}_t - \mathbf{w}_{t-1}\|.$$

In this experiment, we set domain diameter as $D = 10$ and follow Zhao *et al.* [2022] to choose the domain $\mathcal{W}$ as an ellipsoid $\mathcal{W} = \left\{\mathbf{w} \in \mathbb{R}^n \mid \mathbf{w}^\top \mathbf{E}\mathbf{w} \leq \lambda_{\min}(\mathbf{E}) \cdot (D/2)^2\right\}$, where $\mathbf{E}$ is a certain diagonal matrix and $\lambda_{\min}$ denotes its minimum eigenvalue. To simulate the changing environment, we flip the labels of samples every 1000 iterations. For this dataset, dimensionality $n = 22$ and we set $T = 4000$, batch size $m = 256, G = \sqrt{22}, \lambda = 10$ and delay d_t is selected uniformly at random from $[1, 5]$.

The results, including the cumulative loss $\sum_{t=1}^{T} h_t(\mathbf{w}_t)$, the instantaneous loss $h_t(\mathbf{w}_t)$ and the running time (in seconds), are presented in Figure 1 with error bars. All curves are averaged over 10 runs under different random seeds. As can be seen, both of our proposed methods suffer less cumulative loss than Mild-OGD. Moreover, our efficient Smelt-DOGD achieves a comparable performance to Smelt-DOGD, while achieving approximately 4 times speedup due to the improved projection complexity.

4.2 Online Regression

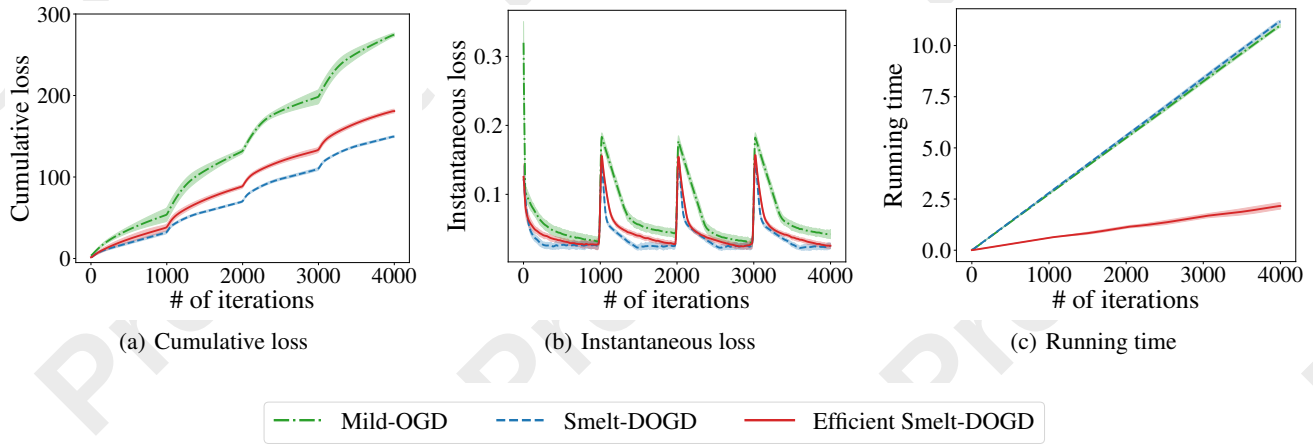
In this experiment, we consider a least mean square regression problem. In each round t, a small batch of training examples $\{(\mathbf{x}_{t,1}, y_{t,1}), \ldots, (\mathbf{x}_{t,m}, y_{t,m})\}$ arrive, and simultaneously, the learner makes a prediction $\mathbf{w}_t$ of the unknown parameter. The learner suffers a loss, defined as $f_t(\mathbf{w}_t) = \frac{1}{m}\sum_{i=1}^{m}(\mathbf{w}_t^\top \mathbf{x}_{t,i} - y_{t,i})^2 + \lambda\|\mathbf{w}_t - \mathbf{w}_{t-1}\|$, where m is the batch size and λ is the trade-off parameter. We also aim to minimize the rescaled function

$$h_t(\mathbf{w}_t) = \frac{1}{\lambda m}\sum_{i=1}^{m}\left(\mathbf{w}_t^\top \mathbf{x}_{t,i} - y_{t,i}\right)^2 + \|\mathbf{w}_t - \mathbf{w}_{t-1}\|.$$

We conduct the experiments on a synthetic dataset, which is constructed through the following process. We first sample the ground truth vector $\mathbf{w}_\star$ from an ellipsoid $\mathcal{W} = \left\{\mathbf{w} \in \mathbb{R}^n \mid \mathbf{w}^\top \mathbf{E}\mathbf{w} \leq \lambda_{\min}(\mathbf{E}) \cdot (D/2)^2\right\}$, where D is set to be 10, $\mathbf{E}$ is a certain diagonal matrix and $\lambda_{\min}$ denotes its minimum eigenvalue. As for the feature vector $\mathbf{x}_{t,i}$, we sample them uniformly at random from $[-1, 1]^n$. Afterwards, we generate $y_{t,i}$ according to a linear model: $y_{t,i} = \mathbf{w}_*^\top \mathbf{x}_{t,i} + \epsilon_t$, where the noise ϵ_t is drawn from a normal distribution with a mean of 0 and a standard deviation of 0.1. We set the dimensionality $n = 500$, $T = 4000$, batch size $m = 128$, trade-off parameter $\lambda = 10$, $D = 200$ and delay d_t is selected uniformly at random from $[1, 5]$. To simulate the changing environment, we flip $\mathbf{w}_*$ every 1000 iterations.

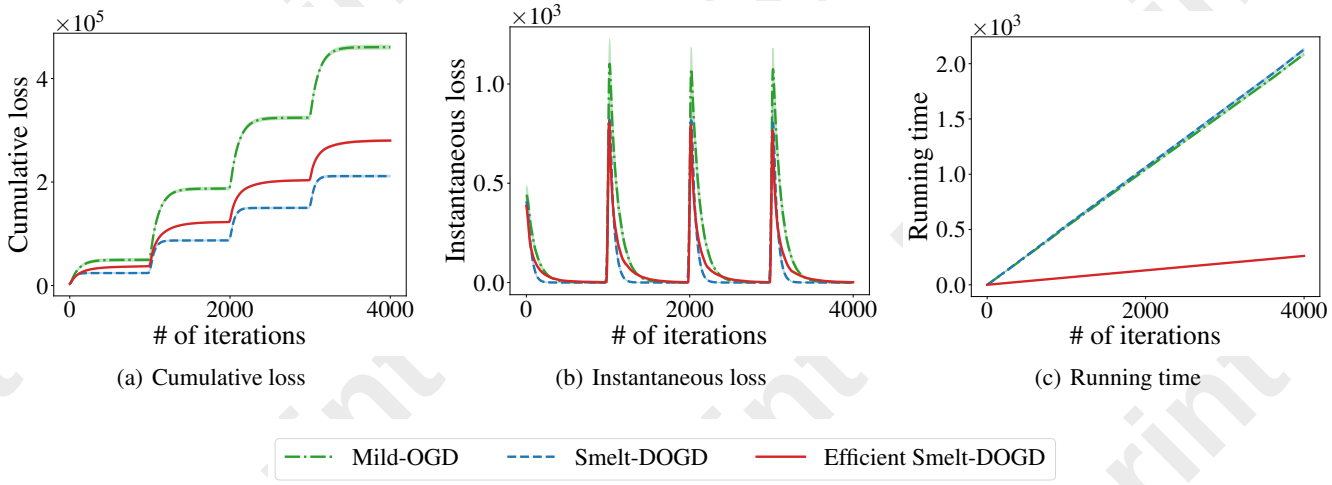
We report the cumulative loss $\sum_{t=1}^{T} h_t(\mathbf{w}_t)$, the instantaneous loss $h_t(\mathbf{w}_t)$ and the running time (in seconds) in Figure 2 with error bars. All curves averaged over 5 runs under

Preprint – IJCAI 2025: This is the accepted version made available for conference attendees.
Do not cite. The final version will appear in the IJCAI 2025 proceedings.



(a) Cumulative loss (b) Instantaneous loss (c) Running time

Figure 1: Results for the online classification.



(a) Cumulative loss (b) Instantaneous loss (c) Running time

Figure 2: Results for the online regression.

different random seeds. As shown in Figure 2, our Smelt-DOGD and efficient Smelt-DOGD suffer less loss than Mild-OGD. Additionally, efficient Smelt-DOGD is significantly more efficient compared with other methods, albeit with a slight compromise on the cumulative loss.

To summarize, the empirical results demonstrate the effectiveness of our methods in addressing the switching cost, as well as efficient Smelt-DOGD's superiority in terms of the running time.

5 Conclusion

In this paper, we investigate SOCO with arbitrary delays. We extend Mild-OGD, a two-layer algorithm for OCO with delayed feedback, to account for the switching cost. We first provide a novel analysis of its expert-algorithm, and take the switching cost into consideration when assigning the weight to each expert in the meta-algorithm. Our proposed method, Smelt-DOGD, can achieve an $O(\sqrt{dT(P_T+1)})$ dynamic regret bound with switching cost. Moreover, to reduce the computational overhead, we employ a black-box technique to develop an efficient version of Smelt-DOGD, which only performs a single projection per round, yet still keeps the order of its regret bound. Finally, the results of experiments further validate the superiority of our methods.

Acknowledgments

This work was partially supported by NSFC (U23A20382, 62361146852), and the Collaborative Innovation Center of Novel Software Technology and Industrialization. The authors would like to thank the anonymous reviewers for their constructive suggestions.

References

[Baby and Wang, 2021] Dheeraj Baby and Yu-Xiang Wang. Optimal dynamic regret in exp-concave online learning. In

Preprint – IJCAI 2025: This is the accepted version made available for conference attendees.
Do not cite. The final version will appear in the IJCAI 2025 proceedings.

Proceedings of the 34th Annual Conference On Learning Theory, pages 359–409, 2021.

[Baby and Wang, 2022] Dheeraj Baby and Yu-Xiang Wang. Optimal dynamic regret in proper online learning with strongly convex losses and beyond. In *Proceedings of International Conference on Artificial Intelligence and Statistics*, pages 1805–1845, 2022.

[Bansal *et al.*, 2015] Nikhil Bansal, Anupam Gupta, Ravishankar Krishnaswamy, Kirk Pruhs, Kevin Schewior, and Cliff Stein. A 2-competitive algorithm for online convex optimization with switching costs. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, 2015.

[Chang and Lin, 2011] Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):1–27, 2011.

[Chen *et al.*, 2015] Niangjun Chen, Anish Agarwal, Adam Wierman, Siddharth Barman, and Lachlan LH Andrew. Online convex optimization using predictions. In *Proceedings of the 2015 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, pages 191–204, 2015.

[Chen *et al.*, 2018] Niangjun Chen, Gautam Goel, and Adam Wierman. Smoothed online convex optimization in high dimensions via online balanced descent. In *Proceedings of the 31st Annual Conference On Learning Theory*, pages 1574–1594, 2018.

[Cutkosky and Orabona, 2018] Ashok Cutkosky and Francesco Orabona. Black-box reductions for parameter-free online learning in banach spaces. In *Proceedings of the 31st Annual Conference On Learning Theory*, pages 1493–1529, 2018.

[Cutkosky, 2020] Ashok Cutkosky. Parameter-free, dynamic, and strongly-adaptive online learning. In *Proceedings of the 37th International Conference on Machine Learning*, pages 2250–2259, 2020.

[Freund and Schapire, 1997] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of online learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[Goel and Wierman, 2019] Gautam Goel and Adam Wierman. An online algorithm for smoothed regression and lqr control. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2504–2513, 2019.

[Goel *et al.*, 2019] Gautam Goel, Yiheng Lin, Haoyuan Sun, and Adam Wierman. Beyond online balanced descent: An optimal algorithm for smoothed online optimization. In *Advances in Neural Information Processing Systems 32*, pages 1875–1885, 2019.

[He *et al.*, 2014] Xinran He, Junfeng Pan, Ou Jin, Tianbing Xu, Bo Liu, Tao Xu, Yanxin Shi, Antoine Atallah, Ralf Herbrich, Stuart Bowers, et al. Practical lessons from predicting clicks on ads at facebook. In *Proceedings of the 8th International Workshop on Data mining for Online Advertising*, pages 1–9, 2014.

[Jadbabaie *et al.*, 2015] Ali Jadbabaie, Alexander Rakhlin, Shahin Shahrampour, and Karthik Sridharan. Online optimization: Competing with dynamic comparators. In *Proceedings of the 18th International Conference on Artificial Intelligence and Statistics*, pages 398–406, 2015.

[Joseph and de Veciana, 2012] Vinay Joseph and Gustavo de Veciana. Jointly optimizing multi-user rate adaptation for video transport over wireless systems: Mean-fairness-variability tradeoffs. In *Proceedings of the 31st Annual IEEE International Conference on Computer Communications*, pages 567–575, 2012.

[Joulani *et al.*, 2013] Pooria Joulani, Andras Gyorgy, and Csaba Szepesvari. Online learning under delayed feedback. In *Proceedings of the 30th International Conference on Machine Learning*, pages 1453–1461, 2013.

[Kim *et al.*, 2015] Taehwan Kim, Yisong Yue, Sarah Taylor, and Iain Matthews. A decision tree framework for spatiotemporal sequence prediction. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 577–586, 2015.

[Korotin *et al.*, 2020] Alexander Korotin, Vladimir V'yugin, and Evgeny Burnaev. Adaptive hedging under delayed feedback. *Neurocomputing*, 397:356–368, 2020.

[Li *et al.*, 2018] Yingying Li, Guannan Qu, and Na Li. Using predictions in online optimization with switching costs: A fast algorithm and a fundamental limit. In *2018 Annual American Control Conference*, pages 3008–3013, 2018.

[Li *et al.*, 2020] Yingying Li, Guannan Qu, and Na Li. Online optimization with predictions and switching costs: Fast algorithms and the fundamental limit. *IEEE Transactions on Automatic Control*, pages 4761–4768, 2020.

[Lin *et al.*, 2011] Minghong Lin, Adam Wierman, Lachlan LH Andrew, and Eno Thereska. Dynamic right-sizing for power-proportional data centers. In *Proceedings of the 30th IEEE International Conference on Computer Communications*, pages 1098–1106, 2011.

[McMahan and Streeter, 2014] Brendan McMahan and Matthew Streeter. Delay-tolerant algorithms for asynchronous distributed online learning. In *Advances in Neural Information Processing Systems 27*, pages 2915–2923, 2014.

[McMahan *et al.*, 2013] H Brendan McMahan, Gary Holt, David Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, et al. Ad click prediction: a view from the trenches. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1222–1230, 2013.

[Mokhtari *et al.*, 2016] Aryan Mokhtari, Shahin Shahrampour, Ali Jadbabaie, and Alejandro Ribeiro. Online optimization in dynamic environments: Improved regret rates for strongly convex problems. In *Proceedings of 55th IEEE Conference on Decision and Control*, pages 7195–7201, 2016.

Preprint – IJCAI 2025: This is the accepted version made available for conference attendees.
Do not cite. The final version will appear in the IJCAI 2025 proceedings.

[Prokhorov, 2001] Danil Prokhorov. Ijcnn 2001 neural network competition. *Slide presentation in IJCNN*, 1(97):38, 2001.

[Quanrud and Khashabi, 2015] Kent Quanrud and Daniel Khashabi. Online learning with adversarial delays. In *Advances in Neural Information Processing Systems 28*, pages 1270–1278, 2015.

[Shalev-Shwartz, 2012] Shalev-Shwartz. Online learning and online convex optimization. *Foundations and Trends in Machine Learning*, 4(2):107–194, 2012.

[Wan *et al.*, 2021] Yuanyu Wan, Bo Xue, and Lijun Zhang. Projection-free online learning in dynamic environments. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence*, volume 35, pages 10067–10075, 2021.

[Wan *et al.*, 2022a] Yuanyu Wan, Wei-Wei Tu, and Lijun Zhang. Online frank-wolfe with arbitrary delays. In *Advances in Neural Information Processing Systems 35*, pages 19703–19715, 2022.

[Wan *et al.*, 2022b] Yuanyu Wan, Wei-Wei Tu, and Lijun Zhang. Online frank-wolfe with arbitrary delays. In *Advances in Neural Information Processing Systems 35*, pages 19703–19715, 2022.

[Wan *et al.*, 2022c] Yuanyu Wan, Wei-Wei Tu, and Lijun Zhang. Online strongly convex optimization with unknown delays. *Machine Learning*, 111(3):871–893, 2022.

[Wan *et al.*, 2023] Yuanyu Wan, Yibo Wang, Chang Yao, Wei-Wei Tu, and Lijun Zhang. Projection-free online learning with arbitrary delays. *arXiv preprint arXiv:2204.04964*, 2023.

[Wan *et al.*, 2024] Yuanyu Wan, Chang Yao, Mingli Song, and Lijun Zhang. Non-stationary online convex optimization with arbitrary delays. In *41th International Conference on Machine Learning*, 2024.

[Wang *et al.*, 2021] Juncheng Wang, Ben Liang, Min Dong, Gary Boudreau, and Hatem Abou-Zeid. Delay-tolerant constrained oco with application to network resource allocation. In *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*, pages 1–10, 2021.

[Wang *et al.*, 2024] Yibo Wang, Wenhao Yang, Wei Jiang, Shiyin Lu, Bin Wang, Haihong Tang, Yuanyu Wan, and Lijun Zhang. Non-stationary projection-free online learning with dynamic and adaptive regret guarantees. In *Proceedings of the 38th AAAI Conference on Artificial Intelligence*, pages 15671–15679, 2024.

[Weinberger and Ordentlich, 2002] Marcelo J Weinberger and Erik Ordentlich. On delayed prediction of individual sequences. *IEEE Transactions on Information Theory*, 48(7):1959–1976, 2002.

[Yang *et al.*, 2024a] Wenhao Yang, Wei Jiang, Yibo Wang, Ping Yang, Yao Hu, and Lijun Zhang. Small-loss adaptive regret for online convex optimization. In *Proceedings of the 41st International Conference on Machine Learning*, pages 56156–56195, 2024.

[Yang *et al.*, 2024b] Wenhao Yang, Yibo Wang, Peng Zhao, and Lijun Zhang. Universal online convex optimization with 1 projection per round. In *Advances in Neural Information Processing Systems 37*, pages 31438–31472, 2024.

[Yang *et al.*, 2025] Sifan Yang, Yuanyu Wan, and Lijun Zhang. Online nonsubmodular optimization with delayed feedback in the bandit setting. In *Proceedings of the 39th AAAI Conference on Artificial Intelligence*, pages 21992–22000, 2025.

[Zanini *et al.*, 2010] Francesco Zanini, David Atienza, Giovanni De Micheli, and Stephen P Boyd. Online convex optimization-based algorithm for thermal management of mpsocs. In *Proceedings of the 20th Symposium on Great Lakes Symposium on VLSI*, pages 203–208, 2010.

[Zhang *et al.*, 2017] Lijun Zhang, Tianbao Yang, Jinfeng Yi, Rong Jin, and Zhi-Hua Zhou. Improved dynamic regret for non-degenerate functions. In *Advances in Neural Information Processing Systems 30*, pages 732–741, 2017.

[Zhang *et al.*, 2018a] Lijun Zhang, Shiyin Lu, and Zhi-Hua Zhou. Adaptive online learning in dynamic environments. In *Advances in Neural Information Processing Systems 31*, pages 2250–2259, 2018.

[Zhang *et al.*, 2018b] Lijun Zhang, Tianbao Yang, Rong Jin, and Zhi-Hua Zhou. Dynamic regret of strongly adaptive methods. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pages 5882–5891, 2018.

[Zhang *et al.*, 2020] Lijun Zhang, Shiyin Lu, and Tianbao Yang. Minimizing dynamic regret and adaptive regret simultaneously. In *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 309–319, 2020.

[Zhang *et al.*, 2021] Lijun Zhang, Wei Jiang, Shiyin Lu, and Tianbao Yang. Revisiting smoothed online learning. In *Advances in Neural Information Processing Systems 34*, pages 13599–13612, 2021.

[Zhang *et al.*, 2022] Lijun Zhang, Wei Jiang, Jinfeng Yi, and Tianbao Yang. Smoothed online convex optimization based on discounted-normal-predictor. In *Advances in Neural Information Processing Systems 35*, pages 4928–4942, 2022.

[Zhao *et al.*, 2022] Peng Zhao, Yan-Feng Xie, Lijun Zhang, and Zhi-Hua Zhou. Efficient methods for non-stationary online learning. In *Advances in Neural Information Processing Systems 35*, pages 11573–11585, 2022.

[Zhou *et al.*, 2018] Zhengyuan Zhou, Panayotis Mertikopoulos, Nicholas Bambos, Peter Glynn, Yinyu Ye, Li-Jia Li, and Li Fei-Fei. Distributed asynchronous optimization with unbounded delays: How slow can you go? In *Proceedings of the 35th International Conference on Machine Learning*, pages 5970–5979, 2018.

[Zinkevich, 2003] Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th International Conference on Machine Learning*, pages 928–936, 2003.