Simple Projection-Free Algorithm for Contextual Recommendation with Logarithmic Regret and Robustness

Shinsaku Sakaue
CyberAgent, National Institute of Informatics, RIKEN AIP
shinsaku.sakaue@gmail.com

Abstract

Contextual recommendation is a variant of contextual linear bandits in which the learner observes an (optimal) action rather than a reward scalar. Recently, Sakaue et al. (2025) developed an efficient Online Newton Step (ONS) approach with an $O(d \log T)$ regret bound, where d is the dimension of the action space and T is the time horizon. In this paper, we present a simple algorithm that is more efficient than the ONS-based method while achieving the same regret guarantee. Our core idea is to exploit the improperness inherent in contextual recommendation, leading to an update rule akin to the second-order perceptron from online classification. This removes the Mahalanobis projection step required by ONS, which is often a major computational bottleneck. More importantly, the same algorithm remains robust to possibly suboptimal action feedback, whereas the prior ONS-based method required running multiple ONS learners with different learning rates for this extension. We describe how our method works in general Hilbert spaces (e.g., via kernelization), where eliminating Mahalanobis projections becomes even more beneficial.

1 Introduction

A pervasive challenge in data-driven decision-making is learning from observed *actions* rather than explicit utility signals: a system records what a user or expert *chooses*—a purchased item, a selected route, a clinical treatment—without directly observing the hidden preference that motivated the choice. This action-only feedback regime arises in various fields, e.g., inverse optimization [2, 13], inverse reinforcement learning [33], and learning from revealed preference [8]. *Contextual recommendation* (a.k.a. *online inverse linear optimization*) is a fundamental online learning problem that formalizes such action-only feedback scenarios, in which the learner sequentially recommends actions and infers a user's hidden preference from the user's actions. This problem is a variant of (contextual) linear bandits [1, 16]—in the standard setting, the learner observes a scalar reward, whereas here the learner observes an (optimal) action—and has been extensively studied [3, 4, 6, 7, 20, 34, 37, 38].

Specifically, let $\mathcal{V}$ be a real Hilbert space with inner product $\langle \cdot, \cdot \rangle$ (one may think of $\mathcal{V} = \mathbb{R}^d$ with the standard dot product as a canonical example). At each round t, the learner observes a nonempty feasible set $\mathcal{X}_t \subseteq \mathcal{V}$, recommends an action $\hat{x}_t \in \arg\max_{x \in \mathcal{X}_t} \langle \hat{w}_t, x \rangle$ for some prediction $\hat{w}_t \in \mathcal{V}$, and then observes a user action $x_t \in \mathcal{X}_t$, which is optimal (or approximately optimal) with respect to the user's unknown preference vector $u \in \mathcal{V}$. In a purchasing example, $\mathcal{X}_t$ consists of bundles of goods affordable at the current prices and budget, u represents the customer's fixed preferences, and x_t is the observed purchase [4, Section 4.1]. The learner's performance is measured by the regret, $R_T(u) \coloneqq \sum_{t=1}^{T} \langle u, x_t - \hat{x}_t \rangle$, which quantifies the cumulative gap between the utilities of the user's actions and the learner's recommendations over T rounds. In what follows, we refer to $\hat{x}_t - x_t$ as a *residual* at round t; the regret is the inner product of u with the negative cumulative residual.

Sakaue et al. [38] recently proposed an algorithm based on the Online Newton Step (ONS) with a regret bound of $O(d \log T)$ under $\mathcal{V} = \mathbb{R}^d$. This is the first *efficient* logarithmic-regret algorithm in

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

Table 1: Total arithmetic-operation bounds over T rounds under $\mathcal{V} = \mathbb{R}^d$, with the other problem parameters fixed. For the ONS-based methods, the learner's domain is the unit ball. Let $\tau_{\mathrm{opt}}(t)$ be the arithmetic cost of solving $\hat{x}_t \in \arg\max_{x \in \mathcal{X}_t} \langle \hat{w}_t, x \rangle$, and let $\omega \in (2, 3]$ be an admissible matrix multiplication exponent. All methods enjoy the $R_T(u) = O(d \log T)$ regret bound under optimal action feedback. See Section 6 and Appendix H for details of these complexity bounds.

Method	Total time complexity
ONS-based method [38]	$O\left(Td^\omega \log T + \sum_{t=1}^T \tau_{\mathrm{opt}}(t)\right)$
ONS-based method [38] with LightONS [42]	$O\left(Td^2 + d^\omega \sqrt{T \log T} + \sum_{t=1}^T \tau_{\mathrm{opt}}(t)\right)$
CoRectron (Algorithm 1)	$O\left(Td^2 + \sum_{t=1}^T \tau_{\mathrm{opt}}(t)\right)$

the sense that its per-round update cost, excluding linear optimization, depends at most logarithmically on T when projection accuracy is taken into account.[1] However, an ONS update involves a *Mahalanobis projection* step at each round. This is often the computational bottleneck, especially in high-dimensional or kernelized settings. Specifically, when $\mathcal{V} = \mathbb{R}^d$ and the learner's domain is the unit ball, the projection step solves $\min_{w:\|w\|_2 \le 1} (w' - w)^\top A (w' - w)$ for some $w' \in \mathbb{R}^d$ and a positive-definite matrix $A \in \mathbb{R}^{d \times d}$. Even in this unit-ball case, known implementations compute a feasible approximate Mahalanobis projection in $O(d^\omega n)$ or $O(d^3 + dn)$ arithmetic operations, where n is the number of bisection steps and $\omega \in (2, 3]$ is an admissible matrix multiplication exponent; see Wang et al. [42, Algorithm 4 and Lemma 7]. The ONS-based approach of Sakaue et al. [38] involves such a projection step at every round, thereby incurring a substantial computational overhead. Therefore, the core question we address is: *can we avoid Mahalanobis projections while retaining logarithmic regret guarantees?*

1.1 Our contributions

We summarize our contributions at a high level and then outline our core idea and analysis techniques.

1. **Simple projection-free algorithm.** We present a simple and efficient algorithm (Algorithm 1) that avoids Mahalanobis projections by exploiting the improperness arising in contextual recommendation. Our algorithm removes the projection overhead in the ONS-based methods [38] (Table 1). Compared with the LightONS-based method [42], its time complexity bound is asymptotically smaller when the additional projection term dominates the shared computation costs (Appendix H).
2. **Logarithmic regret bound.** We prove a regret bound controlled by a log-determinant potential (Theorem 1), which yields the same bound of $O(d \log T)$ as Sakaue et al. [38] when $\mathcal{V} = \mathbb{R}^d$.
3. **Robustness to suboptimal feedback.** The same algorithm achieves a regret bound relative to the revealed actions that scales with the square root of their cumulative suboptimality (Theorem 2).
4. **Contextual models.** We study the problem in a Hilbert space and show how contextual models can be handled in a unified manner, where eliminating Mahalanobis projections is even more beneficial.
5. **Empirical support.** Synthetic experiments in Appendix I show that our algorithm achieves lower runtime and regret than ONS-based methods. It is also stable across hyperparameter choices.

Our core idea: exploiting improperness. In contextual recommendation, the learner recommends actions $\hat{x}_t \in \arg\max_{x \in \mathcal{X}_t} \langle \hat{w}_t, x \rangle$ based on predicted utility vectors $\hat{w}_t \in \mathcal{V}$. Therefore, positive rescaling of $\hat{w}_t$ preserves the set of maximizing actions. This leads to an "improper" viewpoint: we do not need any scale constraint on $\hat{w}_t$; in particular, $\|\hat{w}_t\|$ is allowed to exceed $\|u\|$, the (unknown) scale of the user's preference vector. Building on this observation, our algorithm performs a projection-free second-order update built from the past residuals. This update is reminiscent of the classical second-order perceptron [12] from online classification. Motivated by this connection, we name our method CoRectron (contextual recommendation via second-order perceptron) for ease of reference.

[1] Sakaue et al. [38, Section 3] state that their per-round cost is independent of T. Their ONS update uses an exact projection, but their analysis does not specify how the projection cost depends on numerical accuracy.

Regret analysis. Our analysis rests on a one-sided optimality inequality, $\langle \hat{w}_t, \hat{x}_t - x_t \rangle \geq 0$, that follows from the optimality of $\hat{x}_t$ for $\hat{w}_t$. This yields a sign condition that enables us to control the growth of a potential of the cumulative residuals (Lemma 2). Combined with the celebrated elliptical potential lemma, this leads to a template regret bound controlled by the log-determinant of the Gram matrix of residuals (Theorem 1), which specializes to the $O(d \log T)$ regret bound when $\mathcal{V} = \mathbb{R}^d$. Interestingly, this clean logarithmic regret guarantee contrasts with existing mistake bounds of the second-order perceptron for online classification, which involve margin-dependent terms that may be arbitrarily large in the worst case [12, 35]. We elaborate on this difference in Section 3.1.

Robustness. Importantly, our template bound (3) is derived without assuming that the observed actions are optimal. Consequently, without changing the algorithm, we can establish a regret bound relative to the revealed actions. The bound scales with the square root of their cumulative suboptimality (Theorem 2). This result contrasts with the approach of Sakaue et al. [38] for obtaining the same robustness guarantee: they use MetaGrad [41], which runs $O(\log T)$ ONS learners with different learning rates to adapt to an unknown suboptimality level, thereby increasing the cost of the ONS updates by a factor of $O(\log T)$.

Organization. The rest of this paper is organized as follows. Section 1.2 reviews related work. Section 2 formalizes the problem. Section 3 presents CoRectron and its template regret bound. Section 4 analyzes the regret under possibly suboptimal feedback actions. Section 5 describes a unified view for handling contextual models. Section 6 discusses implementation and major computational factors.

1.2 Related work

Bärmann et al. [3, 4] studied contextual recommendation as an approach to inverse optimization and proposed a method based on online convex optimization (OCO) with an $O(\sqrt{T})$ regret bound. Later, cutting-plane-based methods improved the dependence on T: Besbes et al. [6, 7] gave an $O(d^4 \log T)$ regret bound, and Gollapudi et al. [20] established $O(d \log T)$ and $\exp(O(d \log d))$ regret bounds. These cutting-plane methods require per-round update time polynomial in T, whereas Sakaue et al. [38] achieved an $O(d \log T)$ regret bound via ONS with per-round update time depending at most logarithmically on T when projection accuracy is taken into account. These update costs exclude linear optimization. Under additional structural assumptions, finite regret bounds smaller than $\exp(O(d \log d))$ were also obtained [34, 37], while we focus on the general setting without them.

ONS is a well-established OCO algorithm [22], and its Mahalanobis projection step is recognized as a major computational bottleneck, leading to a line of work on faster online exp-concave optimization [27, 30, 31, 42]. LightONS [42] reduces projection overhead and can accelerate the ONS-based method of Sakaue et al. [38]. Its bound on total time complexity retains an additional projection term that CoRectron eliminates; whether this changes the asymptotic order depends on the dimension, time horizon, and cost of linear optimization (Table 1 and Appendix H). Moreover, CoRectron remains robust to suboptimal feedback without changing its update rule, and our analysis applies to general Hilbert spaces.

Kernelized second-order online learning has also been widely studied, including sketching/embedding methods for acceleration [10, 11]. These approaches typically involve an efficiency–regret trade-off. Hence, when such tools are plugged into the ONS approach for contextual recommendation, they generally improve runtime at the price of weaker regret guarantees. Similar techniques may also be useful for accelerating our method, and exploring this is an interesting direction for future work.

In a separate line of work, Shivaswamy and Joachims [39] studied *coactive learning*, whose optimal-feedback case includes finite-dimensional contextual recommendation. Predating the work of Bärmann et al. [3, 4], their results include an $O(\sqrt{T})$ regret bound using the projection-free preference perceptron and algorithms based on OCO. Their second-order preference perceptron uses Mahalanobis projections and has a logarithmic regret bound for strongly convex losses. They also observed logarithmic growth of the cumulative utility regret empirically, but did not prove this bound. CoRectron achieves an $O(d \log T)$ regret bound with projection-free second-order updates.

For smooth exp-concave losses, Garber and Kretzu [19] obtain $\tilde{O}(d^{2/3}T^{2/3})$ regret using $O(T)$ linear-optimization oracle calls, where the tilde hides logarithmic factors. More generally, projection-free OCO with a separation oracle can achieve $\tilde{O}(\sqrt{dT} + d^2)$ [29]. These are much larger than the $O(d \log T)$ bound we obtain by exploiting the improper structure of contextual recommendation.

Finally, our perspective is different from unconstrained/parameter-free OCO [14, 15, 23, 28]. That line of work aims to compete with comparators of unknown scale with minimal regret degradation from that achievable when the comparator scale is known. By contrast, we exploit the improperness induced by scale invariance in contextual recommendation: we may impose any normalizing assumption on the scale of the user preference (e.g., $\|u\| \le 1$) while allowing the learner's predictions to exceed that scale, and we use this asymmetry to reduce the computation cost without worsening regret guarantees.

Subsequent work. After this paper appeared, Dewasurendra [18] and Cai et al. [9] obtained $O(d)$ regret for unit-ball actions and preferences under optimal feedback. Sakaue [36] improved this to $O\left(\sqrt{d}\right)$ in expectation. Gupta et al. [21] attained the same bound deterministically in polynomial time. Kitaoka [24] obtained finite regret and finitely many mistakes for uniformly bounded integer action sets with unique optima.

2 Problem setting and background

Let $\mathcal{V}$ be a real Hilbert space with inner product $\langle \cdot, \cdot \rangle$ and induced norm $\|\cdot\|$. For rounds $t = 1, \dots, T$, a user arrives with contextual information encoded as a feasible action set $\mathcal{X}_t \subseteq \mathcal{V}$, which is nonempty and weakly compact.[2] The user takes an action $x_t \in \mathcal{X}_t$, and its utility is given by $\langle u, x_t \rangle$, where $u \in \mathcal{V}$ represents the user's hidden preference and defines the linear functional $x \mapsto \langle u, x \rangle$. This u is unknown and fixed across rounds. Unless otherwise stated, we assume that x_t is optimal for u over $\mathcal{X}_t$, as formalized below.

Assumption 1 (Optimal-action feedback)**.** It holds that $x_t \in \arg\max_{x \in \mathcal{X}_t} \langle u, x \rangle$ for every t.

At each round t, the learner observes $\mathcal{X}_t$ and, before observing x_t, computes a utility-vector prediction $\hat{w}_t \in \mathcal{V}$ and recommends an action $\hat{x}_t \in \arg\max_{x \in \mathcal{X}_t} \langle \hat{w}_t, x \rangle$; here, the learner has access to a linear-optimization oracle over $\mathcal{X}_t$. Then, the learner observes the user's action x_t (but not u nor the realized utility value $\langle u, x_t \rangle$). The learner's performance is measured by the *regret*, $R_T(u) \coloneqq \sum_{t=1}^{T} \langle u, x_t - \hat{x}_t \rangle$, which quantifies the cumulative suboptimality of the learner's recommendations $\hat{x}_t$ compared to the user's actions x_t. We use $r_t \coloneqq \langle u, x_t - \hat{x}_t \rangle$ to denote the instantaneous regret at round t. Throughout this paper, we make the following standard boundedness assumption.

Assumption 2. There exists $B > 0$ such that, for all t and all $x, x' \in \mathcal{X}_t$, $\langle u, x - x' \rangle \le B$ holds.

In particular, the instantaneous regret satisfies $r_t \in [-B, B]$. Under Assumption 1, $r_t \in [0, B]$ holds.

Hilbert-space operator notation. We allow $\mathcal{V}$ to be infinite-dimensional. We use I to denote the identity operator on $\mathcal{V}$ and I_m to denote the identity matrix on $\mathbb{R}^m$. For a bounded positive-definite self-adjoint operator A on $\mathcal{V}$, we write $\|v\|_A \coloneqq \sqrt{\langle v, Av \rangle}$ for $v \in \mathcal{V}$. For $a \in \mathcal{V}$, define $a \otimes a \colon \mathcal{V} \to \mathcal{V}$ by $(a \otimes a)(v) = \langle a, v \rangle a$ for all $v \in \mathcal{V}$; in finite dimensions this coincides with $aa^\top$. For a bounded linear map $G \colon \mathbb{R}^m \to \mathcal{V}$, let $G^* \colon \mathcal{V} \to \mathbb{R}^m$ denote its adjoint, defined by $\langle Ga, v \rangle = a^\top (G^* v)$ for all $a \in \mathbb{R}^m$ and $v \in \mathcal{V}$ (transpose in finite dimensions). For $\lambda > 0$ and $a_1, \dots, a_m \in \mathcal{V}$, let $A \colon \mathcal{V} \to \mathcal{V}$ be a linear operator of the form $A = \lambda \mathrm{I} + \sum_{i=1}^{m} a_i \otimes a_i$. Then, for all $v \in \mathcal{V}$, we have $\langle v, Av \rangle = \lambda \|v\|^2 + \sum_{i=1}^{m} \langle a_i, v \rangle^2 \ge \lambda \|v\|^2$. Hence $A \succeq \lambda \mathrm{I}$ holds and, in particular, A is invertible by the Lax–Milgram theorem [25]. See Appendix A for additional details on Hilbert-space operators.

Online learning approach. We briefly review the online learning approach taken in prior work, which is relevant to our idea. A standard approach [3, 4, 39] is to use $\ell_t \colon w \mapsto \langle w, \hat{x}_t - x_t \rangle$ as loss functions and compute predictions $\hat{w}_t$ via online convex optimization (OCO) on a bounded domain $\mathcal{W} \subseteq \mathbb{R}^d$ that contains u. Then, as shown below, the standard OCO regret bound for ℓ_t against the comparator u serves as an upper bound on the contextual recommendation regret $R_T(u)$:

$$R_T(u) = \sum_{t=1}^{T} \langle u, x_t - \hat{x}_t \rangle \le \sum_{t=1}^{T} \langle u, x_t - \hat{x}_t \rangle + \sum_{t=1}^{T} \langle \hat{w}_t, \hat{x}_t - x_t \rangle = \sum_{t=1}^{T} \ell_t(\hat{w}_t) - \sum_{t=1}^{T} \ell_t(u),$$

where the first inequality follows from the optimality of $\hat{x}_t$ for $\hat{w}_t$ over $\mathcal{X}_t$. Therefore, simply applying a standard OCO algorithm (e.g., the online gradient descent) to the losses ℓ_t yields an $O(\sqrt{T})$ regret

[2] This standard condition ensures that the maximizer set $\arg\max_{x \in \mathcal{X}_t} \langle w, x \rangle$ is nonempty for all $w \in \mathcal{V}$.

Algorithm 1 CoRectron: contextual recommendation via second-order perceptron

Require: Regularization parameter $\lambda > 0$, $A_0 = \lambda \mathrm{I}$, $\zeta_0 = 0 \in \mathcal{V}$

1: **for** $t = 1, 2, \ldots, T$ **do**
2: Compute $\hat{w}_t \leftarrow -A_{t-1}^{-1}\zeta_{t-1}$
3: Observe context $\mathcal{X}_t \subseteq \mathcal{V}$
4: Recommend $\hat{x}_t \in \arg\max_{x \in \mathcal{X}_t} \langle \hat{w}_t, x \rangle$
5: Observe $x_t \in \mathcal{X}_t$ and set $g_t \leftarrow \hat{x}_t - x_t$
6: Update $A_t \leftarrow A_{t-1} + g_t \otimes g_t$ and $\zeta_t \leftarrow \zeta_{t-1} + g_t$

bound for contextual recommendation.[3] Sakaue et al. [38] improved this to $O(d \log T)$ by applying ONS to modified exp-concave losses, but the high-level idea of using OCO on a bounded domain remains the same. These OGD and ONS updates require projections to ensure $\hat{w}_t \in \mathcal{W}$.

Improper viewpoint. In contextual recommendation, the learner's recommendation is obtained through linear optimization as $\hat{x}_t \in \arg\max_{x \in \mathcal{X}_t} \langle \hat{w}_t, x \rangle$, and the regret $R_T(u) = \sum_{t=1}^T \langle u, x_t - \hat{x}_t \rangle$ is evaluated on the recommended actions $\hat{x}_t$ (rather than on $\hat{w}_t$). Thus, $\hat{x}_t$ remains a maximizing action after multiplying $\hat{w}_t$ by any positive scalar. Also, the user's actions x_t carry information only about the direction of u, and the learner cannot hope to infer the scale $\|u\|$ from feedback alone. In light of the above, we may fix the scale of u (e.g., $\|u\| = 1$) while allowing the learner to output utility vectors $\hat{w}_t$ of arbitrary scale. This situation is in contrast to the OCO approach above, which constrains the learner's predictions $\hat{w}_t$ to a bounded domain containing u. This *improper* viewpoint—$\hat{w}_t$ is not constrained to the same domain as u—is central to eliminating projections on the learner's iterates. The algorithm and analysis that follow do not require prior knowledge of $\|u\|$.

3 CoRectron

This section presents our algorithm and its template regret bound, which we instantiate in Section 5.

3.1 Algorithm description

For the recommended actions $\hat{x}_t$ and the observed actions x_t, we define the *residual* at round t by $g_t \coloneqq \hat{x}_t - x_t$,[4] and we use $\zeta_t \coloneqq \sum_{s=1}^t g_s$ to denote the cumulative residual up to round t. Then, the regret is expressed as $R_T(u) = -\langle u, \zeta_T \rangle$. Our algorithm, CoRectron (Algorithm 1), maintains the cumulative residual ζ_t and the curvature matrix/operator A_t. The utility-vector prediction for round t is computed as $\hat{w}_t = -A_{t-1}^{-1}\zeta_{t-1}$. This algorithm is free of any projection step.

Relation to second-order perceptron. Algorithm 1 resembles the second-order perceptron for online classification [12]. In that setting, the context information is $\mathcal{X}_t = \{y z_t/2 : y \in \{-1, +1\}\}$ for the t-th input vector z_t. A label prediction $\hat{y}_t = \mathrm{sign}(\langle \hat{w}_t, z_t \rangle) \in \{-1, +1\}$, with $\mathrm{sign}(0) = +1$, induces the recommended action $\hat{x}_t = \hat{y}_t z_t/2$, and the true label $y_t \in \{-1, +1\}$ reveals the correct action $x_t = y_t z_t/2$ as feedback. The algorithm performs the second-order update $\hat{w}_{t+1} = -A_t^{-1}\zeta_t$ only when a mistake occurs, i.e., $g_t = \hat{x}_t - x_t = -y_t z_t$ if $\hat{y}_t \neq y_t$ and $g_t = 0$ otherwise. Our CoRectron can be seen as an extension of this classification algorithm to the contextual recommendation setting, where the context $\mathcal{X}_t$ is a set of feasible actions rather than a single input vector with two possible labels. Notably, our analysis for contextual recommendation yields clean logarithmic bounds on the regret $R_T(u)$, as we will see below. This is in contrast to the classification setting, where mistake bounds of the second-order perceptron involve margin-dependent terms that can be arbitrarily large in the worst case [12, 35].[5] In this sense, our analysis sheds light on the power of the second-order perceptron in contextual recommendation.

[3] By the above argument, the OCO-based approach also yields an upper bound on $\sum_{t=1}^T \langle \hat{w}_t, \hat{x}_t - x_t \rangle$, the online counterpart of the cumulative *suboptimality loss* $w \mapsto \max_{x \in \mathcal{X}_t} \langle w, x \rangle - \langle w, x_t \rangle$, which is a common loss function in inverse optimization [32]. As an evaluation metric, however, this quantity is often less informative because the learner can make it arbitrarily small by shrinking the scale of $\hat{w}_t$ (although informative to some extent when the learner's domain is bounded away from the origin [38]). Thus, our work focuses on bounding the regret $R_T(u)$.

[4] The residual g_t coincides with a subgradient of the suboptimality loss at $\hat{w}_t$.

[5] For binary classification linearly separable by a unit vector u, the number of mistakes is at most $R_T(u)/\gamma$ for the margin $\gamma \coloneqq \min_t |\langle u, z_t \rangle| > 0$. Thus, even when the regret bound is margin-free, the mistake bound typically depends on the margin.

3.2 Template regret bound

We analyze the regret of Algorithm 1. For later use, define the Gram matrix of the residuals as

$$K_t \coloneqq (\langle g_i,\, g_j\rangle)_{i,j=1}^{t} \in \mathbb{R}^{t\times t} \qquad \text{for } t = 1,\dots,T. \tag{1}$$

We also define the *cumulative potential* as

$$\Phi_t \coloneqq \langle \zeta_t,\, A_t^{-1}\zeta_t\rangle = \|\zeta_t\|_{A_t^{-1}}^2 \qquad \text{for } t = 0,\dots,T. \tag{2}$$

This can be seen as a variant of the well-known *elliptical potential* $\sum_{s=1}^{t}\langle g_s,\, A_s^{-1}g_s\rangle$, where we here use the cumulative residual ζ_t instead of taking the sum of the per-round quadratic forms $\langle g_s,\, A_s^{-1}g_s\rangle$.

Our goal is to prove the following template regret bound; Section 5 instantiates this in specific settings.

Theorem 1 (Main regret bound)**.** *For any $u \in \mathcal{V}$, Algorithm 1 with $\lambda > 0$ satisfies the following template regret bound:*

$$R_T(u) \le \|u\|_{A_T}\sqrt{\log\det(\mathrm{I}_T + \lambda^{-1}K_T)}. \tag{3}$$

In particular, if the observed actions $x_t \in \mathcal{X}_t$ are optimal for u (i.e., under Assumption 1) and the boundedness assumption (Assumption 2) holds, we have

$$R_T(u) \le B\log\det\bigl(\mathrm{I}_T + \lambda^{-1}K_T\bigr) + \|u\|\sqrt{\lambda\log\det(\mathrm{I}_T + \lambda^{-1}K_T)}. \tag{4}$$

Proof. For the first claim, we bound $R_T(u) = -\langle u,\, \zeta_T\rangle$ from above as follows:

$$-\langle u,\, \zeta_T\rangle \le \|u\|_{A_T}\sqrt{\Phi_T} \overset{\text{CEI}}{\le} \|u\|_{A_T}\sqrt{\sum_{t=1}^{T}\langle g_t,\, A_t^{-1}g_t\rangle} \overset{\text{EPL}}{\le} \|u\|_{A_T}\sqrt{\log\det(\mathrm{I}_T + \lambda^{-1}K_T)}.$$

The first inequality uses the Cauchy–Schwarz inequality with respect to the A_T-inner product. The remaining two inequalities are shown later; the second one, $\Phi_T \le \sum_{t=1}^{T}\langle g_t,\, A_t^{-1}g_t\rangle$, is the cumulative-potential–elliptical-potential inequality (CEI); the third, $\sum_{t=1}^{T}\langle g_t,\, A_t^{-1}g_t\rangle \le \log\det\bigl(\mathrm{I}_T + \lambda^{-1}K_T\bigr)$, is the well-known elliptical-potential lemma (EPL). Assuming these, we obtain the first claim (3).

For the second claim, let $r_t = \langle u,\, x_t - \hat{x}_t\rangle \in [0, B]$, where $0 \le r_t$ and $r_t \le B$ follow from the optimality of x_t for u (Assumption 1) and the boundedness assumption (Assumption 2), respectively. From $A_T = \lambda\mathrm{I} + \sum_{t=1}^{T} g_t \otimes g_t$ and $\langle u,\, g_t\rangle = -r_t$, we have

$$\|u\|_{A_T}^2 = \lambda\|u\|^2 + \sum_{t=1}^{T}\langle u,\, g_t\rangle^2 = \lambda\|u\|^2 + \sum_{t=1}^{T} r_t^2 \le \lambda\|u\|^2 + B\sum_{t=1}^{T} r_t = \lambda\|u\|^2 + BR_T(u).$$

Combining this with (3), with $H_T \coloneqq \log\det\bigl(\mathrm{I}_T + \lambda^{-1}K_T\bigr)$, gives

$$R_T(u) \le \sqrt{\lambda\|u\|^2 + BR_T(u)}\sqrt{H_T},$$

equivalently $R_T(u)^2 - BH_TR_T(u) - \lambda\|u\|^2H_T \le 0$. Solving this quadratic inequality for $R_T(u)$ gives

$$R_T(u) \le \frac{1}{2}\left(BH_T + \sqrt{B^2H_T^2 + 4\lambda\|u\|^2H_T}\right) \le BH_T + \|u\|\sqrt{\lambda H_T},$$

where we used the subadditivity of the square root in the last step. This establishes (4). □

3.2.1 Cumulative-potential–elliptical-potential inequality

We present the proof of the cumulative-potential–elliptical-potential inequality (CEI) used in the proof of Theorem 1. A similar argument is used in the analysis of the second-order perceptron [12], but it focuses on the classification setting. First, we present a key sign condition.

Lemma 1 (Sign condition)**.** *For each iteration t of Algorithm 1, we have $\langle g_t,\, A_{t-1}^{-1}\zeta_{t-1}\rangle \le 0$.*

Proof. The case of $t = 1$ is trivial since $\zeta_0 = 0$. Let $t \ge 2$. Recall $g_t = \hat{x}_t - x_t$ and $\hat{w}_t = -A_{t-1}^{-1}\zeta_{t-1}$. Since $\hat{x}_t$ maximizes $x \mapsto \langle \hat{w}_t,\, x\rangle$ over $\mathcal{X}_t$ and $x_t \in \mathcal{X}_t$, it holds that $\langle \hat{w}_t,\, \hat{x}_t\rangle \ge \langle \hat{w}_t,\, x_t\rangle$, hence $\langle g_t,\, A_{t-1}^{-1}\zeta_{t-1}\rangle = -\langle \hat{w}_t,\, g_t\rangle = -\langle \hat{w}_t,\, \hat{x}_t - x_t\rangle \le 0$. Thus, the sign condition holds for every t. □

This sign condition enables us to control the potential Φ_t in (2), leading to the following proof of CEI.

Lemma 2 (CP–EP inequality)**.** *Let g_t, A_t, and ζ_t be generated by Algorithm 1 with $\lambda > 0$. Then we have*

$$\Phi_T \le \sum_{t=1}^{T} \langle g_t,\, A_t^{-1} g_t \rangle. \tag{CEI}$$

Proof. Fix t. By the Sherman–Morrison formula (see Appendix A for details), we have

$$A_t^{-1} = A_{t-1}^{-1} - \frac{A_{t-1}^{-1} g_t \otimes A_{t-1}^{-1} g_t}{1 + \langle g_t,\, A_{t-1}^{-1} g_t \rangle}. \tag{5}$$

Defining $\mu_t \coloneqq \langle g_t,\, A_{t-1}^{-1} g_t \rangle$ and $\nu_t \coloneqq \langle g_t,\, A_{t-1}^{-1} \zeta_{t-1} \rangle$, and using the above identity, we obtain

$$\Phi_t - \Phi_{t-1} = \langle \zeta_{t-1} + g_t, A_t^{-1}(\zeta_{t-1} + g_t) \rangle - \langle \zeta_{t-1}, A_{t-1}^{-1} \zeta_{t-1} \rangle = 2\nu_t + \mu_t - \frac{(\nu_t + \mu_t)^2}{1 + \mu_t} = \frac{\mu_t + 2\nu_t - \nu_t^2}{1 + \mu_t}.$$

By Lemma 1, we have $\nu_t \le 0$, which implies $2\nu_t - \nu_t^2 \le 0$ and hence $\Phi_t - \Phi_{t-1} \le \mu_t/(1+\mu_t)$. Also, by using the Sherman–Morrison formula (5) again, we have

$$\langle g_t,\, A_t^{-1} g_t \rangle = \mu_t - \frac{\mu_t^2}{1 + \mu_t} = \frac{\mu_t}{1 + \mu_t}. \tag{6}$$

Thus, $\Phi_t - \Phi_{t-1} \le \langle g_t,\, A_t^{-1} g_t \rangle$ holds. Summing over t and using $\Phi_0 = 0$ yield (CEI). □

3.2.2 Elliptical potential lemma

The remaining ingredient is the elliptical potential lemma (EPL), which is a standard tool in the analysis of linear bandits [1, 16] and ONS [22]. For completeness, we restate it here and present a proof adapted to our Hilbert-space setting in Appendix B.

Lemma 3 (Elliptical potential lemma)**.** *Let $\lambda > 0$, $A_t = \lambda \mathrm{I} + \sum_{s=1}^{t} g_s \otimes g_s$ for $t = 1, \dots, T$, and $K_T = (\langle g_i,\, g_j \rangle)_{i,j=1}^{T} \in \mathbb{R}^{T \times T}$ as in (1). Then, it holds that*

$$\sum_{t=1}^{T} \langle g_t,\, A_t^{-1} g_t \rangle \le \log\det\bigl(\mathrm{I}_T + \lambda^{-1} K_T\bigr). \tag{EPL}$$

4 Robustness to suboptimal feedback actions

We now allow the feedback action x_t to merely lie in the feasible set $\mathcal{X}_t$, not necessarily optimal for u, and quantify how the main regret bound degrades with the cumulative suboptimality. The regret is still defined with respect to the revealed actions x_t, i.e., $R_T(u) = \sum_{t=1}^{T} \langle u,\, x_t - \hat{x}_t \rangle$, even though x_t may be suboptimal. Here, the per-round regret $r_t = \langle u,\, x_t - \hat{x}_t \rangle$ can be negative if x_t is suboptimal for u, hindering the derivation of (4) in Theorem 1. Still, $r_t \in [-B, B]$ holds by Assumption 2.

We define the per-round suboptimality and its cumulative version as follows:

$$\delta_t(u) \coloneqq \max_{x \in \mathcal{X}_t} \langle u,\, x \rangle - \langle u,\, x_t \rangle \ge 0 \qquad \text{and} \qquad \Delta_T(u) \coloneqq \sum_{t=1}^{T} \delta_t(u).$$

The following lemma, essentially contained in the proof of Sakaue et al. [38, Theorem 4.1], is useful for handling suboptimal feedback. We give the proof in Appendix C for completeness.

Lemma 4 (cf. [38, Theorem 4.1])**.** *Under Assumption 2, we have $\sum_{t=1}^{T} r_t^2 \le B R_T(u) + 2B\Delta_T(u)$.*

Combining this with the template bound in Theorem 1 gives the following robust guarantee.

Theorem 2 (Suboptimality-robust bound)**.** *Assume only that $x_t \in \mathcal{X}_t$ holds for every t (not necessarily optimal) and run Algorithm 1 with $\lambda > 0$. Under Assumption 2, for any $u \in \mathcal{V}$, we have*

$$R_T(u) \le B H_T + \|u\| \sqrt{\lambda H_T} + \sqrt{2B\Delta_T(u) H_T}, \tag{7}$$

where we let $H_T = \log\det\bigl(\mathrm{I}_T + \lambda^{-1} K_T\bigr)$, as in the proof of Theorem 1. When $\Delta_T(u) = 0$ (i.e., under Assumption 1), this recovers (4) in Theorem 1.

Proof. If $R_T(u) \le 0$, (7) holds trivially as the right-hand side is nonnegative. Assume $R_T(u) > 0$. From $A_T = \lambda \mathrm{I} + \sum_{t=1}^{T} g_t \otimes g_t$, $r_t = \langle u,\, x_t - \hat{x}_t \rangle = -\langle u,\, g_t \rangle$, and Lemma 4, we have

$$\|u\|_{A_T}^2 = \lambda \|u\|^2 + \sum_{t=1}^{T} \langle u,\, g_t \rangle^2 = \lambda \|u\|^2 + \sum_{t=1}^{T} r_t^2 \le \lambda \|u\|^2 + B R_T(u) + 2B\Delta_T(u). \tag{8}$$

Note that the template bound (3) in Theorem 1 does not require optimality of x_t, and hence it remains valid here. Squaring both sides of (3) and applying (8) yield

$$R_T(u)^2 \le \|u\|_{A_T}^2 H_T \le H_T\big(\lambda \|u\|^2 + B R_T(u) + 2B\Delta_T(u)\big).$$

Solving the quadratic inequality and using the subadditivity of the square root give the bound (7). □

Comparison with the prior approach. Sakaue et al. [38, Theorem 4.1] obtained a similar robust guarantee by running $\Theta(\log T)$ ONS instances, based on MetaGrad [41]; this increases the cost of the ONS updates by a factor of $\log T$. In contrast, our CoRectron itself is suboptimality-robust without changing the algorithm. This is enabled by our algorithm design and analysis, which make explicit the role of the data-dependent norm $\|u\|_{A_T}$ arising from the second-order update with the cumulative residuals.

Regret relative to optimal actions. While the regret $R_T(u)$ measures the learner's performance relative to the revealed actions, prior work [34, 39] also considers the regret relative to optimal actions, which equals $R_T(u) + \Delta_T(u)$ in our setting. Its bound therefore includes an additional $\Delta_T(u)$ term. This linear dependence is unavoidable in general (see Appendix D).

5 Application to contextual models

This section instantiates the template regret bound under the Hilbert-space formulation for various contextual settings. Prior work also studies contextual settings in finite dimensions [6, 7, 38], but the kernelized setting, described below, has not been explicitly covered. In offline inverse optimization, kernelized formulations have served as a useful approach for handling non-linearity [5, 26]; therefore, the applicability of our online method to the kernelized setting constitutes a meaningful contribution.

5.1 Generic lifting template

Let $n \in \mathbb{Z}_{>0}$. Suppose each round gives a base action set $\mathcal{X}_t^{\mathrm{base}} \subseteq \mathbb{R}^n$ and contextual information, encoded as a bounded linear map $\Psi_t \colon \mathbb{R}^n \to \mathcal{V}$. We lift $\mathcal{X}_t^{\mathrm{base}}$ to $\mathcal{X}_t$ as the image of $\mathcal{X}_t^{\mathrm{base}}$ under Ψ_t, i.e.,

$$\mathcal{X}_t \coloneqq \Psi_t(\mathcal{X}_t^{\mathrm{base}}) = \{\Psi_t(x) : x \in \mathcal{X}_t^{\mathrm{base}}\} \subseteq \mathcal{V}.$$

Let $\Psi_t^* \colon \mathcal{V} \to \mathbb{R}^n$ be the adjoint, given by $\langle w,\, \Psi_t(x) \rangle_{\mathcal{V}} = \langle \Psi_t^* w,\, x \rangle_{\mathbb{R}^n}$ for all $w \in \mathcal{V}$ and $x \in \mathbb{R}^n$.[6] Then, maximizing $\langle w,\, \cdot \rangle_{\mathcal{V}}$ over $\mathcal{X}_t$ is equivalent to maximizing $\langle \Psi_t^* w,\, \cdot \rangle_{\mathbb{R}^n}$ over $\mathcal{X}_t^{\mathrm{base}}$. In particular, a linear-optimization oracle over $\mathcal{X}_t^{\mathrm{base}}$ implements the oracle over $\mathcal{X}_t$ required by Algorithm 1.

Base-action feedback and residuals. In contextual settings, the interaction is naturally phrased in the base action space: the learner observes $\mathcal{X}_t^{\mathrm{base}}$ and recommends an action in $\mathcal{X}_t^{\mathrm{base}}$, after which a user action $x_t^{\mathrm{base}} \in \mathcal{X}_t^{\mathrm{base}}$ is revealed. Given a learner's prediction $\hat{w}_t \in \mathcal{V}$, let $\hat{w}_t^{\mathrm{base}} \coloneqq \Psi_t^* \hat{w}_t$ and

$$\hat{x}_t^{\mathrm{base}} \in \arg\max\big\{ \langle \hat{w}_t^{\mathrm{base}},\, x \rangle_{\mathbb{R}^n} \;:\; x \in \mathcal{X}_t^{\mathrm{base}} \big\}.$$

Then, $\hat{x}_t = \Psi_t(\hat{x}_t^{\mathrm{base}})$ belongs to $\arg\max_{x \in \mathcal{X}_t} \langle \hat{w}_t,\, x \rangle_{\mathcal{V}}$. Moreover, the revealed base action x_t^{base} induces the revealed lifted action $x_t = \Psi_t(x_t^{\mathrm{base}}) \in \mathcal{X}_t$, and hence we obtain the residual as

$$g_t = \hat{x}_t - x_t = \Psi_t(g_t^{\mathrm{base}}) \in \mathcal{V}, \qquad \text{where} \qquad g_t^{\mathrm{base}} \coloneqq \hat{x}_t^{\mathrm{base}} - x_t^{\mathrm{base}} \in \mathbb{R}^n,$$

thanks to the linearity of Ψ_t. Therefore, base-action feedback is sufficient to compute the residuals required by Algorithm 1. In what follows, we assume the diameter of the base-action sets $\mathcal{X}_t^{\mathrm{base}}$ is uniformly bounded by some $X > 0$ in the Euclidean norm, so that $\|g_t^{\mathrm{base}}\|_2 \le X$ holds for all t.

[6] We use subscripts to distinguish the inner products in $\mathcal{V}$ and $\mathbb{R}^n$.

5.2 Examples

Non-contextual model. Take $\mathcal{V} = \mathbb{R}^n$ and $\Psi_t = \mathrm{I}_n$. Then, we have $\mathcal{X}_t = \mathcal{X}_t^{\mathrm{base}}$ and $\Psi_t^* = \mathrm{I}_n$.

Linear contextual model. Consider context vectors $z_t \in \mathbb{R}^p$ representing users' features and linear utilities $\mathcal{X}_t^{\mathrm{base}} \ni x \mapsto \langle Uz_t,\, x\rangle_{\mathbb{R}^n}$, where $U \in \mathbb{R}^{n\times p}$ is an unknown mapping from features to preferences. Take $\mathcal{V} = \mathbb{R}^{n\times p}$ with the Frobenius inner product $\langle A,\, B\rangle_{\mathrm{F}} = \mathrm{tr}(A^\top B)$ and let $\Psi_t(x) = xz_t^\top$. Then, the adjoint is $\Psi_t^*(W) = Wz_t$ since we have $\langle W,\, \Psi_t(x)\rangle_{\mathrm{F}} = \mathrm{tr}(W^\top xz_t^\top) = \langle Wz_t,\, x\rangle_{\mathbb{R}^n}$.

Kernelized contextual model. Let $\mathcal{Z}$ be a context space and $\mathcal{V} = \mathcal{H}$ an RKHS of vector-valued functions $h\colon \mathcal{Z} \to \mathbb{R}^n$ with kernel $\mathcal{K}$. For each context $z \in \mathcal{Z}$, let $\mathcal{K}_z\colon \mathbb{R}^n \to \mathcal{H}$ be the canonical feature operator $(\mathcal{K}_z x)(\cdot) = \mathcal{K}(\cdot, z)x$ for $x \in \mathbb{R}^n$. By the reproducing property, we have

$$\langle h,\, \mathcal{K}_z x\rangle_{\mathcal{H}} = \langle h(z),\, x\rangle_{\mathbb{R}^n}, \qquad \forall h \in \mathcal{H},\ \forall x \in \mathbb{R}^n. \tag{9}$$

Given a context $z_t \in \mathcal{Z}$ at round t, an unknown function $h^\star \in \mathcal{H}$ induces the utility $\mathcal{X}_t^{\mathrm{base}} \ni x \mapsto \langle h^\star(z_t),\, x\rangle_{\mathbb{R}^n}$. Let $\Psi_t = \mathcal{K}_{z_t}$. Then, the adjoint is given by $\Psi_t^*(h) = h(z_t)$ for any $h \in \mathcal{H}$ by (9).

5.3 Instantiating regret bounds

Our template regret bounds, (4) and (7), depend on the structure of the action space $\mathcal{X}_t$ through the log-determinant potential $H_T = \log\det\bigl(\mathrm{I}_T + \frac{1}{\lambda}K_T\bigr)$, where K_T is the Gram matrix (1) of residuals g_t. Below, we bound H_T to derive specific regret bounds in the three models of Section 5.2. For later use, we define the *effective dimension* of K_T for $\lambda > 0$ as follows, which is commonly used in kernelized online learning [10, 11]:

$$d_{\mathrm{eff}}^T(\lambda) \coloneqq \mathrm{tr}\bigl(K_T(K_T + \lambda \mathrm{I}_T)^{-1}\bigr) = \sum_{i=1}^{m} \frac{\sigma_i}{\sigma_i + \lambda},$$

where $\sigma_1, \dots, \sigma_m > 0$ are the non-zero eigenvalues of K_T. We have $d_{\mathrm{eff}}^T(\lambda) \le \mathrm{rank}(K_T)$; in finite-dimensional models this is at most $\dim(\mathcal{V})$. In infinite-dimensional models, $d_{\mathrm{eff}}^T(\lambda)$ can be finite even if $\dim(\mathcal{V})$ is infinite. The following lemma upper bounds H_T in terms of the effective dimension.

Lemma 5. *For any positive semidefinite matrix $K_T \in \mathbb{R}^{T\times T}$ and $\lambda > 0$, we have*

$$\log\det\bigl(\mathrm{I}_T + \lambda^{-1}K_T\bigr) \le d_{\mathrm{eff}}^T(\lambda)\bigl(1 + \log\bigl(1 + \lambda^{-1}\|K_T\|_{\mathrm{op}}\bigr)\bigr),$$

where $\|K_T\|_{\mathrm{op}}$ is the operator norm of K_T.

See Appendix E for the proof of Lemma 5. Using this lemma in our regret bounds, we obtain the following specific regret bounds for the three models. We present the detailed derivations in Appendix F.

Corollary 1 (Instantiated regret bounds)**.** *Under Assumption 2, assuming $\|x - x'\|_2 \le X$ for all $x, x' \in \mathcal{X}_t^{\mathrm{base}}$ and all t for some $X > 0$, we have the following regret bounds.*

(i) ***Non-contextual.*** *Run Algorithm 1 with $\lambda = B^2 n$. For any $u \in \mathbb{R}^n$ with $\|u\|_2 \le 1$, we have*

$$\textstyle\sum_{t=1}^T \langle u,\, x_t - \hat{x}_t\rangle = O\Bigl(Bn\log\bigl(2 + \frac{TX}{Bn}\bigr) + \sqrt{B\Delta_T(u)\, n\log\bigl(2 + \frac{TX}{Bn}\bigr)}\Bigr).$$

(ii) ***Linear contextual.*** *Assume additionally $\|z_t\|_2 \le Z$ for all t for some $Z > 0$. Run Algorithm 1 with $\lambda = B^2 np$. Then, for any $U \in \mathbb{R}^{n\times p}$ with $\|U\|_{\mathrm{F}} \le 1$, we have*

$$\textstyle\sum_{t=1}^T \langle Uz_t,\, x_t^{\mathrm{base}} - \hat{x}_t^{\mathrm{base}}\rangle = O\Bigl(Bnp\log\bigl(2 + \frac{TXZ}{Bnp}\bigr) + \sqrt{B\Delta_T(U)\, np\log\bigl(2 + \frac{TXZ}{Bnp}\bigr)}\Bigr).$$

(iii) ***Kernelized contextual.*** *Assume additionally $\|\mathcal{K}(z,z)\|_{\mathrm{op}} \le \kappa^2$ for all $z \in \mathcal{Z}$ for some $\kappa > 0$. Run Algorithm 1 with an arbitrary $\lambda \in (0, T]$. Then, for any $h^\star \in \mathcal{H}$ with $\|h^\star\|_{\mathcal{H}} \le 1$, we have*

$$\textstyle\sum_{t=1}^T \langle h^\star(z_t), x_t^{\mathrm{base}} - \hat{x}_t^{\mathrm{base}}\rangle = O\Bigl(Bd_{\mathrm{eff}}^T(\lambda)\log(2 + \frac{TX\kappa}{\lambda}) + \sqrt{(\lambda + B\Delta_T(h^\star))d_{\mathrm{eff}}^T(\lambda)\log(2 + \frac{TX\kappa}{\lambda})}\Bigr).$$

In particular, under optimal feedback (Assumption 1), the above bounds hold with $\Delta_T(\cdot) = 0$.

The non-contextual result recovers the $O(n\log T)$ regret bound of the existing studies [20, Theorem 4.2] [38, Theorem 3.1], as well as its suboptimality-robust version [38, Theorem 4.1].[7] By contrast, the extension to the infinite-dimensional kernelized setting is a new result of this work.

[7] As in the prior work, we state the bounds with λ tuned under the assumption that B is known, for direct comparison. For implementation, however, it suffices to use any known upper bound on B, or one may simply take $\lambda = n$ and $\lambda = np$ in the non-contextual and linear contextual cases, respectively; the resulting regret bound follows from Theorem 2.

6 Implementation and computational considerations

We discuss how to implement Algorithm 1 and the dominant per-round computational factors. In contextual models, the update proceeds as follows: solve the linear system for $\hat{w}_t = -A_{t-1}^{-1}\zeta_{t-1}$, compute $\hat{w}_t^{\mathrm{base}} = \Psi_t^* \hat{w}_t$, solve linear optimization to obtain $\hat{x}_t^{\mathrm{base}} \in \arg\max_{x \in \mathcal{X}_t^{\mathrm{base}}} \langle \hat{w}_t^{\mathrm{base}}, x \rangle$, compute $g_t = \Psi_t(\hat{x}_t^{\mathrm{base}} - x_t^{\mathrm{base}})$, and update $A_t = A_{t-1} + g_t \otimes g_t$ and $\zeta_t = \zeta_{t-1} + g_t$. Among these steps, major computational factors are (i) solving the linear system for $\hat{w}_t$, (ii) feature map evaluations, and (iii) linear optimization to find $\hat{x}_t^{\mathrm{base}}$. Writing the computational costs of these steps at round t as $\tau_{\mathrm{solve}}(t)$, $\tau_{\mathrm{feat}}(t)$, and $\tau_{\mathrm{opt}}(t)$, respectively, we express the per-round computational cost as

$$\tau_{\mathrm{solve}}(t) + \tau_{\mathrm{feat}}(t) + \tau_{\mathrm{opt}}(t).$$

Finite-dimensional models. Consider finite-dimensional cases with $\mathcal{V} = \mathbb{R}^d$, where $d = n$ in the non-contextual model and $d = np$ in the linear contextual model. Evaluating feature maps for computing g_t takes $\tau_{\mathrm{feat}}(t) = O(d)$. With the rank-one-update structure of $A_t = A_{t-1} + g_t \otimes g_t$, solving the linear system for $\hat{w}_{t+1}$ can be done in $O(d^2)$ time, hence $\tau_{\mathrm{solve}}(t) = O(d^2)$. Therefore, the per-round time complexity is $\tau_{\mathrm{feat}}(t) + \tau_{\mathrm{opt}}(t) + \tau_{\mathrm{solve}}(t) = O\big(d^2 + \tau_{\mathrm{opt}}(t)\big)$, where $\tau_{\mathrm{opt}}(t)$ depends on the structure of $\mathcal{X}_t^{\mathrm{base}}$. Summing over t yields the total time complexity in the bottom row of Table 1.

Kernelized contextual model. We present the implementation details for the kernelized model in Appendix G, and here we summarize the main computational factors. Based on the representer form, we work with the Gram system $K_t + \lambda \mathrm{I}_t$. Letting τ_{Kv} denote the cost of one kernel-vector product $\mathcal{K}(z, z')v \in \mathbb{R}^n$, we can implement the update so that $\tau_{\mathrm{solve}}(t) = O(t^2)$ and $\tau_{\mathrm{feat}}(t) = O(t\,\tau_{Kv})$ hold. Hence, the per-round cost is $\tau_{\mathrm{solve}}(t) + \tau_{\mathrm{feat}}(t) + \tau_{\mathrm{opt}}(t) = O\big(t^2 + t\,\tau_{Kv} + \tau_{\mathrm{opt}}(t)\big)$.

7 Conclusion

We have presented an efficient projection-free algorithm with logarithmic regret bounds for contextual recommendation. Moreover, the same algorithm is also shown to be robust to suboptimal feedback actions. Conceptually, our contribution is to shift the perspective on contextual recommendation from the OCO approach to an improper approach, specifically one based on the second-order perceptron. We have also provided a unified view for handling linear and kernelized contextual models. Synthetic experiments in Appendix I show lower runtime and regret than the evaluated ONS baselines.

Regarding limitations, our guarantees rely on access to an exact linear-optimization oracle over each feasible set. In addition, the exact kernelized implementation still incurs $O(t^2)$ per-round cost, and thus further approximation or compression is needed for huge-scale instances with very long horizons. Future work includes addressing these limitations and evaluating the method on real-world data.

Acknowledgments and Disclosure of Funding

Shinsaku Sakaue was supported by JST BOOST Program Japan Grant Number JPMJBY24D1. The author declares no competing interests.

References

[1] Y. Abbasi-Yadkori, D. Pál, and C. Szepesvári. Improved algorithms for linear stochastic bandits. In *Advances in Neural Information Processing Systems*, volume 24, pages 2312–2320. Curran Associates, Inc., 2011 (cited on pages 1, 7).

[2] R. K. Ahuja and J. B. Orlin. Inverse optimization. *Operations Research*, 49(5):771–783, 2001 (cited on page 1).

[3] A. Bärmann, A. Martin, S. Pokutta, and O. Schneider. An online-learning approach to inverse optimization. *arXiv:1810.12997*, 2020 (cited on pages 1, 3, 4).

[4] A. Bärmann, S. Pokutta, and O. Schneider. Emulating the expert: Inverse optimization through online learning. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 400–410. PMLR, 2017 (cited on pages 1, 3, 4).

[5] D. Bertsimas, V. Gupta, and I. C. Paschalidis. Data-driven estimation in equilibrium using inverse optimization. *Mathematical Programming*, 153(2):595–633, 2015 (cited on page 8).

[6] O. Besbes, Y. Fonseca, and I. Lobel. Online learning from optimal actions. In *Proceedings of the 34th Conference on Learning Theory*, volume 134, pages 586–586. PMLR, 2021 (cited on pages 1, 3, 8).

[7] O. Besbes, Y. Fonseca, and I. Lobel. Contextual inverse optimization: Offline and online learning. *Operations Research*, 73(1):424–443, 2025 (cited on pages 1, 3, 8).

[8] J. R. Birge, X. Li, and C. Sun. Learning from stochastically revealed preference. In *Advances in Neural Information Processing Systems*, volume 35, pages 35061–35071. Curran Associates, Inc., 2022 (cited on page 1).

[9] Y. Cai, A. Gupta, V. Gupta, G. Guruganesh, Y. Jiang, C. Liaw, A. Mehta, R. Paes Leme, G. Velegkas, and D. Wang. Efficient online inverse optimization with $O(d)$ regret. *arXiv:2609.13440v1*, 2026 (cited on page 4).

[10] D. Calandriello, A. Lazaric, and M. Valko. Efficient second-order online kernel learning with adaptive embedding. In *Advances in Neural Information Processing Systems*, volume 30, pages 6140–6150. Curran Associates, Inc., 2017 (cited on pages 3, 9).

[11] D. Calandriello, A. Lazaric, and M. Valko. Second-order kernel online convex optimization with adaptive sketching. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 645–653. PMLR, 2017 (cited on pages 3, 9).

[12] N. Cesa-Bianchi, A. Conconi, and C. Gentile. A second-order perceptron algorithm. *SIAM Journal on Computing*, 34(3):640–668, 2005 (cited on pages 2, 3, 5, 6).

[13] T. C. Y. Chan, R. Mahmood, and I. Y. Zhu. Inverse optimization: Theory and applications. *Operations Research*, 73(2):1046–1074, 2025 (cited on page 1).

[14] A. Cutkosky and Z. Mhammedi. Fully unconstrained online learning. In *Advances in Neural Information Processing Systems*, volume 37, pages 10148–10201. Curran Associates, Inc., 2024 (cited on page 4).

[15] A. Cutkosky and F. Orabona. Black-box reductions for parameter-free online learning in Banach spaces. In *Proceedings of the 31st Conference on Learning Theory*, volume 75, pages 1493–1529. PMLR, 2018 (cited on page 4).

[16] V. Dani, T. P. Hayes, and S. M. Kakade. Stochastic linear optimization under bandit feedback. In *Proceedings of the 21st Conference on Learning Theory*, pages 355–366. Omnipress, 2008 (cited on pages 1, 7).

[17] C. Y. Deng. A generalization of the Sherman–Morrison–Woodbury formula. *Applied Mathematics Letters*, 24(9):1561–1564, 2011 (cited on page 13).

[18] P. Dewasurendra. Multiscale reward hedging from correct demonstrations. *arXiv:2608.06825v1*, 2026. Withdrawn (cited on page 4).

[19] D. Garber and B. Kretzu. Projection-free online exp-concave optimization. In *Proceedings of the 36th Conference on Learning Theory*, volume 195, pages 1259–1284. PMLR, 2023 (cited on page 3).

[20] S. Gollapudi, G. Guruganesh, K. Kollias, P. Manurangsi, R. Paes Leme, and J. Schneider. Contextual recommendations and low-regret cutting-plane algorithms. In *Advances in Neural Information Processing Systems*, volume 34, pages 22498–22508. Curran Associates, Inc., 2021 (cited on pages 1, 3, 9).

[21] A. Gupta, G. Guruganesh, H. Lin, V. Mirrokni, R. Paes Leme, and D. P. Woodruff. Optimal and efficient online inverse optimization. *arXiv:2610.08735v1*, 2026 (cited on page 4).

[22] E. Hazan, A. Agarwal, and S. Kale. Logarithmic regret algorithms for online convex optimization. *Machine Learning*, 69(2):169–192, 2007 (cited on pages 3, 7).

[23] A. Jacobsen and A. Cutkosky. Unconstrained online learning with unbounded losses. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pages 14590–14630. PMLR, 2023 (cited on page 4).

[24] A. Kitaoka. Online inverse integer linear optimization via small-gradient skipping: Constant regret and finite mistakes. *arXiv:2609.09809v2*, 2026 (cited on page 4).

[25] P. D. Lax and A. N. Milgram. Parabolic equations. *Annals of Mathematics Studies*, 33:167–190, 1954 (cited on page 4).

[26] Y. Long, T. Ok, P. Zattoni Scroccaro, and P. Mohajerin Esfahani. Scalable kernel inverse optimization. In *Advances in Neural Information Processing Systems*, volume 37, pages 99464–99487. Curran Associates, Inc., 2024 (cited on page 8).

[27] H. Luo, A. Agarwal, N. Cesa-Bianchi, and J. Langford. Efficient second order online learning by sketching. In *Advances in Neural Information Processing Systems*, volume 29, pages 902–910, 2016 (cited on page 3).

[28] H. B. McMahan and M. Streeter. No-regret algorithms for unconstrained online convex optimization. In *Advances in Neural Information Processing Systems*, volume 25, pages 2402–2410. Curran Associates, Inc., 2012 (cited on page 4).

[29] Z. Mhammedi. An ellipsoid algorithm for online convex optimization. In *Advances in Neural Information Processing Systems*, volume 38, pages 804–844. Curran Associates, Inc., 2025 (cited on page 3).

[30] Z. Mhammedi and K. Gatmiry. Quasi-Newton steps for efficient online exp-concave optimization. In *Proceedings of the 36th Conference on Learning Theory*, volume 195, pages 4473–4503. PMLR, 2023 (cited on page 3).

[31] Z. Mhammedi, W. M. Koolen, and T. van Erven. Lipschitz adaptivity with multiple learning rates in online learning. In *Proceedings of the 32nd Conference on Learning Theory*, volume 99, pages 2490–2511. PMLR, 2019 (cited on page 3).

[32] P. Mohajerin Esfahani, S. Shafieezadeh-Abadeh, G. A. Hanasusanto, and D. Kuhn. Data-driven inverse optimization with imperfect information. *Mathematical Programming*, 167:191–234, 2018 (cited on page 5).

[33] A. Y. Ng and S. J. Russell. Algorithms for inverse reinforcement learning. In *Proceedings of the 17th International Conference on Machine Learning*, pages 663–670. Morgan Kaufmann Publishers Inc., 2000 (cited on page 1).

[34] T. Oki and S. Sakaue. Finite and corruption-robust regret bounds in online inverse linear optimization under M-convex action sets. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306, pages 94550–94563. PMLR, 2026 (cited on pages 1, 3, 8).

[35] F. Orabona, N. Cesa-Bianchi, and C. Gentile. Beyond logarithmic bounds in online learning. In *Proceedings of the 15th International Conference on Artificial Intelligence and Statistics*, volume 22, pages 823–831. PMLR, 2012 (cited on pages 3, 5).

[36] S. Sakaue. Tight regret bound for online inverse linear optimization via multiscale matrix weights. *arXiv:2609.26978v1*, 2026 (cited on page 4).

[37] S. Sakaue, H. Bao, and T. Tsuchiya. Revisiting online learning approach to inverse linear optimization: A Fenchel–Young loss perspective and gap-dependent regret analysis. In *Proceedings of the 28th International Conference on Artificial Intelligence and Statistics*, volume 258, pages 46–54. PMLR, 2025 (cited on pages 1, 3).

[38] S. Sakaue, T. Tsuchiya, H. Bao, and T. Oki. Online inverse linear optimization: Efficient logarithmic-regret algorithm, robustness to suboptimality, and lower bound. In *Advances in Neural Information Processing Systems*, volume 38, pages 85189–85217. Curran Associates, Inc., 2025 (cited on pages 1–3, 5, 7–9, 14, 17–20, 23).

[39] P. Shivaswamy and T. Joachims. Coactive learning. *Journal of Artificial Intelligence Research*, 53:1–40, 2015 (cited on pages 3, 4, 8).

[40] T. van Erven and W. M. Koolen. MetaGrad: Multiple learning rates in online learning. In *Advances in Neural Information Processing Systems*, volume 29, pages 3666–3674. Curran Associates, Inc., 2016 (cited on page 23).

[41] T. van Erven, W. M. Koolen, and D. van der Hoeven. MetaGrad: Adaptation using multiple learning rates in online learning. *Journal of Machine Learning Research*, 22(161):1–61, 2021 (cited on pages 3, 8).

[42] Y.-H. Wang, P. Zhao, and Z.-H. Zhou. A simple, optimal and efficient algorithm for online exp-concave optimization. In *Proceedings of the 39th Conference on Learning Theory*, volume 336, pages 6651–6691. PMLR, 2026 (cited on pages 2, 3, 18, 20).

A Additional preliminaries on operators

Most of the fundamental tools used in this paper are covered in Section 2. Here, we present additional preliminaries on operators that are used in the proofs.

In the proof of Lemma 2, we use the Sherman–Morrison identity of the form

$$(A + g \otimes g)^{-1} = A^{-1} - \frac{A^{-1}g \otimes A^{-1}g}{1 + \langle g,\, A^{-1}g\rangle},$$

where $A\colon \mathcal{V} \to \mathcal{V}$ is a bounded self-adjoint operator satisfying $A \succeq \lambda \mathrm{I}$ for some $\lambda > 0$, and $g \in \mathcal{V}$. Also, in the proof of Lemma 3 in Appendix B, we use the Woodbury identity of the form

$$(\lambda \mathrm{I} + GG^*)^{-1} = \lambda^{-1}\mathrm{I} - \lambda^{-2}G\big(\mathrm{I}_m + \lambda^{-1}G^*G\big)^{-1}G^*,$$

where $G\colon \mathbb{R}^m \to \mathcal{V}$ is a bounded linear map, $G^*\colon \mathcal{V} \to \mathbb{R}^m$ is its adjoint, and $\lambda > 0$. These are special cases of the following Sherman–Morrison–Woodbury identity. While the finite-dimensional version is well-known, it extends to the general Hilbert-space setting (e.g., Deng [17, Theorem 1.1]). We present the statement and proof here for completeness.

Proposition 1. *Let $A\colon \mathcal{V} \to \mathcal{V}$ be a bounded invertible linear operator and $G\colon \mathbb{R}^m \to \mathcal{V}$ be a bounded linear map. Assume that $\mathrm{I}_m + G^*A^{-1}G$ is invertible. Then, we have*

$$(A + GG^*)^{-1} = A^{-1} - A^{-1}G(G^*A^{-1}G + \mathrm{I}_m)^{-1}G^*A^{-1}.$$

Proof. Let $M \coloneqq G^*A^{-1}G \in \mathbb{R}^{m\times m}$ and assume $\mathrm{I}_m + M$ is invertible. Define

$$B \coloneqq (\mathrm{I}_m + M)^{-1} \qquad \text{and} \qquad X \coloneqq A^{-1} - A^{-1}GBG^*A^{-1}.$$

We first check that X is a right inverse of $A + GG^*$:

$$\begin{aligned}(A + GG^*)X &= \mathrm{I} + GG^*A^{-1} - GBG^*A^{-1} - GG^*A^{-1}GBG^*A^{-1}\\ &= \mathrm{I} + G(\mathrm{I}_m - B - MB)G^*A^{-1} = \mathrm{I},\end{aligned}$$

where the last equality follows from $(\mathrm{I}_m + M)B = \mathrm{I}_m$. Likewise, X is a left inverse:

$$\begin{aligned}X(A + GG^*) &= \mathrm{I} + A^{-1}GG^* - A^{-1}GBG^* - A^{-1}GBG^*A^{-1}GG^*\\ &= \mathrm{I} + A^{-1}G(\mathrm{I}_m - B - BM)G^* = \mathrm{I},\end{aligned}$$

where we used $B(\mathrm{I}_m + M) = \mathrm{I}_m$. Hence, $X = (A + GG^*)^{-1}$, i.e.,

$$(A + GG^*)^{-1} = A^{-1} - A^{-1}G(G^*A^{-1}G + \mathrm{I}_m)^{-1}G^*A^{-1},$$

as claimed. □

B Proof of Lemma 3

Lemma 3 (Elliptical potential lemma)**.** *Let $\lambda > 0$, $A_t = \lambda \mathrm{I} + \sum_{s=1}^{t} g_s \otimes g_s$ for $t = 1, \ldots, T$, and $K_T = (\langle g_i,\, g_j\rangle)_{i,j=1}^{T} \in \mathbb{R}^{T\times T}$ as in (1). Then, it holds that*

$$\sum_{t=1}^{T} \langle g_t,\, A_t^{-1}g_t\rangle \le \log\det\big(\mathrm{I}_T + \lambda^{-1}K_T\big). \tag{EPL}$$

Proof. As shown in (6), $\langle g_t,\, A_t^{-1}g_t\rangle = \frac{\mu_t}{1+\mu_t}$ holds, where $\mu_t \coloneqq \langle g_t,\, A_{t-1}^{-1}g_t\rangle$. Using $\mu/(1+\mu) \le \log(1+\mu)$ for all $\mu \ge 0$ and summing over t, we obtain

$$\sum_{t=1}^{T} \langle g_t,\, A_t^{-1}g_t\rangle \le \sum_{t=1}^{T} \log(1 + \mu_t). \tag{10}$$

It remains to show

$$\sum_{t=1}^{T} \log(1 + \mu_t) = \log\det\big(\mathrm{I}_T + \lambda^{-1}K_T\big). \tag{11}$$

For each t, define

$$K_t = (\langle g_i,\, g_j\rangle)_{i,j=1}^{t} \in \mathbb{R}^{t\times t}, \qquad k_t = (\langle g_i,\, g_t\rangle)_{i=1}^{t-1} \in \mathbb{R}^{t-1}, \qquad \text{and} \qquad M_t = \mathrm{I}_t + \lambda^{-1}K_t.$$

Let $G_{t-1}\colon \mathbb{R}^{t-1} \to \mathcal{V}$ be given by $G_{t-1}e_i = g_i$ for $i = 1, \dots, t-1$, where e_i is the i-th standard basis vector in $\mathbb{R}^{t-1}$. Its adjoint $G_{t-1}^*\colon \mathcal{V} \to \mathbb{R}^{t-1}$ is defined by $\langle G_{t-1}a,\, v\rangle = a^\top(G_{t-1}^* v)$ for all $a \in \mathbb{R}^{t-1}$ and $v \in \mathcal{V}$, so that $(G_{t-1}^* v)_i = \langle g_i,\, v\rangle$ holds. Thus, for all $v \in \mathcal{V}$, we have

$$G_{t-1}G_{t-1}^* v = \sum_{s=1}^{t-1} \langle g_s,\, v\rangle g_s = \sum_{s=1}^{t-1} (g_s \otimes g_s)v,$$

hence $G_{t-1}G_{t-1}^* = \sum_{s=1}^{t-1} g_s \otimes g_s$. Also, for all $a \in \mathbb{R}^{t-1}$, we have

$$G_{t-1}^* G_{t-1}a = \sum_{s=1}^{t-1} a_s G_{t-1}^* g_s = \sum_{s=1}^{t-1} a_s (\langle g_1,\, g_s\rangle, \dots, \langle g_{t-1},\, g_s\rangle)^\top = K_{t-1}a,$$

hence $G_{t-1}^* G_{t-1} = K_{t-1}$. Therefore, we have

$$A_{t-1} = \lambda \mathrm{I} + G_{t-1}G_{t-1}^* \quad \text{and} \quad M_{t-1} = \mathrm{I}_{t-1} + \lambda^{-1} G_{t-1}^* G_{t-1}.$$

The Woodbury identity (see Appendix A) yields

$$A_{t-1}^{-1} = \lambda^{-1}\mathrm{I} - \lambda^{-2} G_{t-1} M_{t-1}^{-1} G_{t-1}^*.$$

Hence, we have

$$\mu_t = \langle g_t,\, A_{t-1}^{-1} g_t\rangle = \lambda^{-1}\langle g_t,\, g_t\rangle - \lambda^{-2} k_t^\top M_{t-1}^{-1} k_t. \tag{12}$$

Moreover, M_t has the block decomposition:

$$M_t = \begin{pmatrix} M_{t-1} & \lambda^{-1}k_t \\ \lambda^{-1}k_t^\top & 1 + \lambda^{-1}\langle g_t,\, g_t\rangle \end{pmatrix}.$$

By the Schur complement formula and (12), we have

$$\det(M_t) = \det(M_{t-1})(1 + \mu_t).$$

Iterating this identity (with $\det(M_0) = 1$) gives $\det(M_T) = \prod_{t=1}^{T}(1 + \mu_t)$, which is equivalent to the desired equality (11). Combining (10) and (11) proves (EPL). □

C Proof of Lemma 4

Lemma 4 (cf. [38, Theorem 4.1])**.** *Under Assumption 2, we have* $\sum_{t=1}^{T} r_t^2 \le BR_T(u) + 2B\Delta_T(u)$.

Proof. Fix t and note that $|r_t| \le B$ holds by Assumption 2. Since $\hat{x}_t \in \mathcal{X}_t$ is feasible, the definition of $\delta_t(u) \ge 0$ gives

$$\delta_t(u) = \max_{x\in\mathcal{X}_t} \langle u,\, x\rangle - \langle u,\, x_t\rangle \ge \langle u,\, \hat{x}_t\rangle - \langle u,\, x_t\rangle = -r_t. \tag{13}$$

If $r_t \ge 0$, then $0 \le r_t \le B$ implies $r_t^2 - Br_t \le 0$. If $r_t < 0$, then $0 \le -r_t \le B$ and (13) imply

$$r_t^2 - Br_t \le B(-r_t) - Br_t \le 2B\delta_t(u).$$

Thus, in any case, we have $r_t^2 - Br_t \le \max\{0, 2B\delta_t(u)\} = 2B\delta_t(u)$. Summing over t gives $\sum_{t=1}^{T} r_t^2 \le BR_T(u) + 2B\Delta_T(u)$, as desired. □

D Regret relative to optimal actions

As discussed in Section 4, we may also consider the regret relative to optimal actions. Specifically, for each $t \in \{1, \dots, T\}$, choose $x_t^\star \in \arg\max_{x\in\mathcal{X}_t}\langle u,\, x\rangle$, and define $R_T^\star(u) \coloneqq \sum_{t=1}^{T}\langle u,\, x_t^\star - \hat{x}_t\rangle$. Under the conditions of Theorem 2, the definition of $\Delta_T(u)$ gives

$$R_T^\star(u) = R_T(u) + \Delta_T(u) \le BH_T + \|u\|\sqrt{\lambda H_T} + \sqrt{2B\Delta_T(u)H_T} + \Delta_T(u).$$

The linear dependence on $\Delta_T(u)$ is unavoidable in general. To see this, fix any learner, an integer $T \geq 1$, and $K \in [0, T]$, and let $a := K/T$. Consider two one-dimensional instances with $\mathcal{X}_t = \{-a, 0, a\}$ and $x_t = 0$ for every $t \in \{1, \ldots, T\}$, with hidden preferences $u = +1$ and $u = -1$. Both satisfy $\Delta_T(u) = K$ and Assumption 2 with $B = 2$. The learner observes the same action sets and feedback in both instances. With the same random choices, it therefore makes the same recommendations. Their per-round regrets relative to optimal actions sum to $(a - \hat{x}_t) + (a + \hat{x}_t) = 2a$. Writing $\mathbb{E}_+$ and $\mathbb{E}_-$ for expectations over the learner's randomness in the two respective instances, we obtain

$$\mathbb{E}_+[R_T^\star(+1)] + \mathbb{E}_-[R_T^\star(-1)] = 2K.$$

Hence, for at least one fixed preference, the expected regret is at least $K = \Delta_T(u)$, precluding a uniform sublinear guarantee when $K = \Theta(T)$.

E Proof of Lemma 5

Lemma 5. *For any positive semidefinite matrix $K_T \in \mathbb{R}^{T \times T}$ and $\lambda > 0$, we have*

$$\log\det\big(\mathrm{I}_T + \lambda^{-1} K_T\big) \leq d_{\text{eff}}^T(\lambda)\big(1 + \log\big(1 + \lambda^{-1}\|K_T\|_{\text{op}}\big)\big),$$

where $\|K_T\|_{\text{op}}$ is the operator norm of K_T.

Proof. Let $\sigma_1, \ldots, \sigma_m > 0$ be the non-zero eigenvalues of K_T. Then, we have

$$\log\det\big(\mathrm{I}_T + \lambda^{-1} K_T\big) = \sum_{i=1}^{m} \log\Big(1 + \frac{\sigma_i}{\lambda}\Big).$$

Since $\log(1 + x) \leq x$ for $x \geq 0$, we have

$$(1 + x)\log(1 + x) = \log(1 + x) + x\log(1 + x) \leq x(1 + \log(1 + x)),$$

that is,

$$\log(1 + x) \leq \frac{x}{1 + x}(1 + \log(1 + x)).$$

Applying this with $x = \sigma_i/\lambda$ and using $\log(1 + \sigma_i/\lambda) \leq \log(1 + \|K_T\|_{\text{op}}/\lambda)$ yield

$$\log\Big(1 + \frac{\sigma_i}{\lambda}\Big) \leq \frac{\sigma_i}{\sigma_i + \lambda}\big(1 + \log\big(1 + \lambda^{-1}\|K_T\|_{\text{op}}\big)\big).$$

Summing over i and using $\sum_{i=1}^m \frac{\sigma_i}{\sigma_i + \lambda} = d_{\text{eff}}^T(\lambda)$ prove the claim. □

F Proof of Corollary 1

Corollary 1 (Instantiated regret bounds)**.** *Under Assumption 2, assuming $\|x - x'\|_2 \leq X$ for all $x, x' \in \mathcal{X}_t^{\text{base}}$ and all t for some $X > 0$, we have the following regret bounds.*

(i) ***Non-contextual.*** *Run Algorithm 1 with $\lambda = B^2 n$. For any $u \in \mathbb{R}^n$ with $\|u\|_2 \leq 1$, we have*

$$\textstyle\sum_{t=1}^T \langle u,\, x_t - \hat{x}_t\rangle = O\Big(Bn\log\big(2 + \frac{TX}{Bn}\big) + \sqrt{B\Delta_T(u)\, n\log\big(2 + \frac{TX}{Bn}\big)}\Big).$$

(ii) ***Linear contextual.*** *Assume additionally $\|z_t\|_2 \leq Z$ for all t for some $Z > 0$. Run Algorithm 1 with $\lambda = B^2 np$. Then, for any $U \in \mathbb{R}^{n \times p}$ with $\|U\|_{\text{F}} \leq 1$, we have*

$$\textstyle\sum_{t=1}^T \langle Uz_t,\, x_t^{\text{base}} - \hat{x}_t^{\text{base}}\rangle = O\Big(Bnp\log\big(2 + \frac{TXZ}{Bnp}\big) + \sqrt{B\Delta_T(U)\, np\log\big(2 + \frac{TXZ}{Bnp}\big)}\Big).$$

(iii) ***Kernelized contextual.*** *Assume additionally $\|\mathcal{K}(z, z)\|_{\text{op}} \leq \kappa^2$ for all $z \in \mathcal{Z}$ for some $\kappa > 0$. Run Algorithm 1 with an arbitrary $\lambda \in (0, T]$. Then, for any $h^\star \in \mathcal{H}$ with $\|h^\star\|_{\mathcal{H}} \leq 1$, we have*

$$\textstyle\sum_{t=1}^T \langle h^\star(z_t), x_t^{\text{base}} - \hat{x}_t^{\text{base}}\rangle = O\Big(Bd_{\text{eff}}^T(\lambda)\log(2 + \frac{TX\kappa}{\lambda}) + \sqrt{(\lambda + B\Delta_T(h^\star))d_{\text{eff}}^T(\lambda)\log(2 + \frac{TX\kappa}{\lambda})}\Big).$$

In particular, under optimal feedback (Assumption 1), the above bounds hold with $\Delta_T(\cdot) = 0$.

Proof of Corollary 1. Given Lemma 5, it remains to bound $\|K_T\|_{\mathrm{op}}$ for the three models (and $d_{\mathrm{eff}}^T(\lambda)$ for the finite-dimensional models). From $K_T = (\langle g_i,\, g_j\rangle)_{i,j=1}^T$, $g_t = \Psi_t(g_t^{\mathrm{base}})$, and $\|g_t^{\mathrm{base}}\|_2 \le X$, we have the following bound on $\|K_T\|_{\mathrm{op}}$, which we will use later:

$$\|K_T\|_{\mathrm{op}} \le \mathrm{tr}(K_T) = \sum_{t=1}^T \langle g_t^{\mathrm{base}},\, \Psi_t^*\Psi_t(g_t^{\mathrm{base}})\rangle_{\mathbb{R}^n} \le \sum_{t=1}^T \|\Psi_t^*\Psi_t\|_{\mathrm{op}}\|g_t^{\mathrm{base}}\|_2^2 \le X^2 \sum_{t=1}^T \|\Psi_t^*\Psi_t\|_{\mathrm{op}}.$$

Non-contextual model ($\mathcal{V} = \mathbb{R}^n$). From $\Psi_t = \Psi_t^* = \mathrm{I}_n$, we have $\|K_T\|_{\mathrm{op}} \le TX^2$. Also, we have $d_{\mathrm{eff}}^T(\lambda) \le \dim(\mathcal{V}) = n$. Therefore, we obtain

$$H_T \le n\Big(1 + \log\Big(1 + \tfrac{TX^2}{\lambda}\Big)\Big).$$

Linear contextual model ($\mathcal{V} = \mathbb{R}^{n\times p}$). Assume additionally $\|z_t\|_2 \le Z$ for all t. Then, we have $\|\Psi_t^*\Psi_t\|_{\mathrm{op}} = \|z_t z_t^\top\|_{\mathrm{op}} = \|z_t\|_2^2 \le Z^2$, hence $\|K_T\|_{\mathrm{op}} \le TX^2Z^2$. Also, $d_{\mathrm{eff}}^T(\lambda) \le \dim(\mathcal{V}) = np$ holds. Therefore, we obtain

$$H_T \le np\Big(1 + \log\Big(1 + \tfrac{TX^2Z^2}{\lambda}\Big)\Big).$$

Kernelized contextual model (RKHS). Assume additionally $\|\mathcal{K}(z,z)\|_{\mathrm{op}} \le \kappa^2$ for all $z \in \mathcal{Z}$. Then, $\|\Psi_t^*\Psi_t\|_{\mathrm{op}} = \|\mathcal{K}(z_t,z_t)\|_{\mathrm{op}} \le \kappa^2$ holds, hence $\|K_T\|_{\mathrm{op}} \le TX^2\kappa^2$. Therefore, we obtain

$$H_T \le d_{\mathrm{eff}}^T(\lambda)\Big(1 + \log\Big(1 + \tfrac{TX^2\kappa^2}{\lambda}\Big)\Big).$$

Below, we simplify those bounds on H_T. Note that the bounds take the form $H_T \le d(1+\log(1+a))$ for some positive d and a. For $a \ge 0$, it holds that $1 + \log(1+a) \le 2\log(2+a)$. Therefore, we have $H_T = O(d\log(2+a))$. Let us take a closer look at the argument a in each model, with $\log(2+bc^2) \le 2\log(2+bc)$ for $b \ge 1$ and $c > 0$ in mind. In the non-contextual case, suppose $T > n$; otherwise $R_T(u) \le Bn$ trivially holds. Then, setting $\lambda = B^2 n$ yields $H_T = O\left(n\log\left(2 + \frac{TX}{Bn}\right)\right)$. Similarly, in the linear contextual case, we can assume $T > np$ (otherwise $R_T(u) \le Bnp$ trivially holds), and setting $\lambda = B^2np$ yields $H_T = O\left(np\log\left(2 + \frac{TXZ}{Bnp}\right)\right)$. In the kernelized case, we keep $\lambda \in (0, T]$ arbitrary, hence $H_T = O\left(d_{\mathrm{eff}}^T(\lambda)\log\left(2 + \frac{TX\kappa}{\lambda}\right)\right)$.

Combining these bounds with (7) in Theorem 2 and using that u, U, and $h^\star$ each have norm at most 1 in their respective spaces yield the displayed regret bounds. □

G Kernelized implementation details

We present the implementation details for the kernelized model discussed in Section 6. At the beginning of every round t, we maintain

$$L_{t-1}L_{t-1}^\top = K_{t-1} + \lambda \mathrm{I}_{t-1} \qquad \text{and} \qquad c_{t-1} = (K_{t-1} + \lambda \mathrm{I}_{t-1})^{-1}\mathbf{1}_{t-1},$$

where $L_{t-1} \in \mathbb{R}^{(t-1)\times(t-1)}$ is the Cholesky factor of $K_{t-1} + \lambda\mathrm{I}_{t-1}$ and $\mathbf{1}_{t-1} \in \mathbb{R}^{t-1}$ is the all-ones vector. Define $G_{t-1}\colon \mathbb{R}^{t-1} \to \mathcal{H}$ by $G_{t-1}e_s = g_s$, so that we have

$$\zeta_{t-1} = G_{t-1}\mathbf{1}_{t-1}, \quad A_{t-1} = \lambda\mathrm{I} + G_{t-1}G_{t-1}^*, \quad \text{and} \quad K_{t-1} = G_{t-1}^*G_{t-1}.$$

Using $\hat{w}_t = -A_{t-1}^{-1}\zeta_{t-1}$ and $(\lambda\mathrm{I} + GG^*)^{-1}G = G(G^*G + \lambda\mathrm{I}_m)^{-1}$, which holds for any bounded linear map $G\colon \mathbb{R}^m \to \mathcal{H}$ and $\lambda > 0$, we have

$$\hat{w}_t = -\big(\lambda\mathrm{I} + G_{t-1}G_{t-1}^*\big)^{-1}G_{t-1}\mathbf{1}_{t-1} = -G_{t-1}(K_{t-1} + \lambda\mathrm{I}_{t-1})^{-1}\mathbf{1}_{t-1} = -\sum_{s=1}^{t-1}(c_{t-1})_s g_s.$$

Using these maintained quantities, we proceed as follows (for convenience, we put the computation of $\hat{w}_{t+1}$ at the end of round t, but it is equivalent to computing $\hat{w}_t$ at the beginning of round t):

1. Compute the predicted objective vector

$$\hat{w}_t^{\mathrm{base}} = \Psi_t^* \hat{w}_t = -\sum_{s=1}^{t-1} (c_{t-1})_s \, \Psi_t^* g_s.$$

In the kernelized model, we have $\Psi_t^* g_s = \Psi_t^* \Psi_s(g_s^{\mathrm{base}}) = \mathcal{K}(z_t, z_s) g_s^{\mathrm{base}}$, and thus this computation requires $t-1$ kernel-vector products.

2. Call the base linear optimization oracle to compute

$$\hat{x}_t^{\mathrm{base}} \in \arg\max\{ \langle \hat{w}_t^{\mathrm{base}}, x \rangle : x \in \mathcal{X}_t^{\mathrm{base}} \},$$

observe x_t^{base}, and set $g_t = \Psi_t\big(\hat{x}_t^{\mathrm{base}} - x_t^{\mathrm{base}}\big)$.

3. Compute the new Gram entries

$$k_t = (\langle g_s, g_t \rangle_{\mathcal{H}})_{s=1}^{t-1} \qquad \text{and} \qquad \rho_t = \langle g_t, g_t \rangle_{\mathcal{H}},$$

where $\langle g_s, g_t \rangle_{\mathcal{H}} = \langle g_s^{\mathrm{base}}, \mathcal{K}(z_s, z_t) g_t^{\mathrm{base}} \rangle_{\mathbb{R}^n}$.

4. Update the Cholesky factor from L_{t-1} to L_t for $K_t + \lambda \mathrm{I}_t$ using the block update

$$K_t + \lambda \mathrm{I}_t = \begin{bmatrix} K_{t-1} + \lambda \mathrm{I}_{t-1} & k_t \\ k_t^\top & \rho_t + \lambda \end{bmatrix}.$$

Specifically, given $L_{t-1} L_{t-1}^\top = K_{t-1} + \lambda \mathrm{I}_{t-1}$, we first solve

$$L_{t-1} y_t = k_t.$$

Then we set

$$\beta_t = \sqrt{\rho_t + \lambda - \|y_t\|_2^2} \qquad \text{and} \qquad L_t = \begin{bmatrix} L_{t-1} & 0 \\ y_t^\top & \beta_t \end{bmatrix}.$$

Since $K_t + \lambda \mathrm{I}_t \succ 0$, β_t is well-defined, and $L_t L_t^\top = K_t + \lambda \mathrm{I}_t$ holds.

5. Solve

$$(K_t + \lambda \mathrm{I}_t) c_t = \mathbf{1}_t$$

by forward substitution with L_t and backward substitution with $L_t^\top$, and set

$$\hat{w}_{t+1} = -\sum_{s=1}^{t} (c_t)_s g_s.$$

In each round, computing $\hat{w}_t^{\mathrm{base}}$ and updating the new Gram row/column each require $O(t\, \tau_{Kv})$ time. Thus, the feature-evaluation cost is

$$\tau_{\mathrm{feat}}(t) = O(t\, \tau_{Kv}).$$

From Steps 4–5, solving the linear system for $\hat{w}_{t+1}$ consists of (i) one forward substitution for $L_{t-1} y_t = k_t$ and (ii) one forward and one backward substitution for $(K_t + \lambda \mathrm{I}_t) c_t = \mathbf{1}_t$ via L_t and $L_t^\top$. Thus, the cost of solving the linear system is

$$\tau_{\mathrm{solve}}(t) = O\big((t-1)^2\big) + O\big(t^2\big) = O(t^2).$$

Therefore, the total per-round computational cost is

$$\tau_{\mathrm{solve}}(t) + \tau_{\mathrm{feat}}(t) + \tau_{\mathrm{opt}}(t) = O\big(t^2 + t\, \tau_{Kv} + \tau_{\mathrm{opt}}(t)\big).$$

H Detailed comparisons with ONS-based methods

We discuss the time complexity bounds of the ONS-based methods shown in Table 1 in the finite-dimensional setting, i.e., $\mathcal{V} = \mathbb{R}^d$.

The ONS-based method of Sakaue et al. [38] shares the same $\tau_{\mathrm{feat}}(t)$ and $\tau_{\mathrm{opt}}(t)$ terms, but each round additionally solves $\hat{w}_t \in \arg\min_{w : \|w\| \le 1} \|w_t' - w\|_{A_{t-1}}$ for Mahalanobis projection, where w_t'

is the unconstrained point. FastProj [42, Algorithm 4 and Lemma 7] computes a feasible approximate projection in $O(d^\omega n_t)$ arithmetic operations, where n_t is the number of bisection steps. With the surrogate-gradient bound, curvature parameter, and positive initial regularization fixed, the ONS curvature matrix A_{t-1} has operator norm and condition number $O(t)$, while $\|w'_t\| = O(1)$. Setting the Euclidean error tolerance to $(t+1)^{-3}$ adds only $O(1)$ to the regret bound, since the projection errors contribute $O(\|A_{t-1}\|_{\mathrm{op}}(t+1)^{-3}) = O(t^{-2})$ per round. This tolerance requires $n_t = O(\log(t+1))$, giving the total cost in the top row of Table 1. The tridiagonalization-based implementation instead uses $O(d^3 + d\log(t+1))$ arithmetic operations per round. CoRectron avoids these projection computations.

LightONS, proposed by Wang et al. [42], is an efficient variant of ONS. It achieves a lower total time complexity of $O\big(Td^2 + d^\omega\sqrt{T\log T}\big)$ by reducing the number of projection steps, while the $\tau_{\mathrm{opt}}(t)$ part remains the same since $\hat{x}_t$ is needed to construct a subgradient. Thus, using LightONS instead of standard ONS in Sakaue et al. [38] yields the total time complexity in the middle row of Table 1. To compare the time complexity bounds for the LightONS-based method and CoRectron, we regard the dimension d as a function of the time horizon T, with $\omega > 2$ and the other problem parameters fixed. As $T \to \infty$, CoRectron's total time complexity bound has strictly smaller asymptotic order when

$$\frac{d}{(T/\log T)^{1/(2\omega-4)}} \longrightarrow \infty \quad \text{and} \quad \sum_{t=1}^{T} \tau_{\mathrm{opt}}(t) = o\big(d^\omega\sqrt{T\log T}\big).$$

If $d^\omega\sqrt{T\log T} = O\big(Td^2 + \sum_{t=1}^{T}\tau_{\mathrm{opt}}(t)\big)$, the two upper bounds have the same asymptotic order.

In kernelized ONS, adapting the tridiagonalization method of Wang et al. [42, Algorithm 4 (Choice 2) and Lemma 7] yields a feasible approximate projection in $O(t^3 + tn_t)$ arithmetic operations whenever a nontrivial projection is needed. Here, n_t is the number of bisection steps required for the chosen RKHS error tolerance. CoRectron avoids this projection cost and performs its linear-system update in $O(t^2)$ arithmetic operations. This practical advantage is also reflected in the experiments in Appendix I, where CoRectron typically achieves lower runtime than ONS while maintaining competitive or smaller regret.

I Numerical experiments

We conducted the experiments on a MacBook Air equipped with an Apple M4 chip (10 cores) and 32 GB RAM, running macOS 26.5.2. The experiments used Python 3.11.16 with NumPy 2.4.3, SciPy 1.17.1, pandas 3.0.1, and Matplotlib 3.10.8; NumPy and SciPy used Apple Accelerate for linear algebra. Across the 1,920 experimental runs, the summed wall-clock time for the timed algorithm operations was approximately 33 minutes, excluding data and user-action generation, saving, and plotting. Preliminary experiments and implementation checks required additional computation.

I.1 Experiments with optimal-action feedback

This section presents experiments with optimal-action feedback. Appendix I.2 considers suboptimal-action feedback.

I.1.1 Experimental setup

Problem setting: contextual m-out-of-n. At each round t, a context $z_t \in \mathbb{R}^p$ is observed, and we define the (unnormalized) action set

$$\mathcal{X}_{\mathrm{raw}} = \left\{ x \in \{0,1\}^n \colon \sum_{i=1}^{n} x_i = m \right\}.$$

This corresponds to selecting m items from n candidates. In our experiments, we fix $n = 10$, $m = 5$, and $p = 10$, and the user and learner choose actions from the normalized action set

$$\mathcal{X}_t^{\mathrm{base}} = \frac{1}{\sqrt{10}}\,\mathcal{X}_{\mathrm{raw}}.$$

That is, each action vector is scaled by $1/\sqrt{10}$, thereby ensuring that any $x, x' \in \mathcal{X}_t^{\mathrm{base}}$ satisfies $\|x - x'\|_2 \le 1$. We use the same base action set $\mathcal{X}_t^{\mathrm{base}}$ for all t for simplicity, but the contexts z_t

vary over time; therefore, the lifted action sets $\mathcal{X}_t$ vary over time. Let $u_t^{\mathrm{base}} \in \mathbb{R}^n$ denote the user's base utility vector (specified later for linear and kernel models), and let

$$x_t^{\mathrm{base}} \in \arg\max\{\, \langle u_t^{\mathrm{base}},\, x\rangle \,:\, x \in \mathcal{X}_t^{\mathrm{base}} \,\} \quad \text{and} \quad R_T = \sum_{t=1}^{T} \langle u_t^{\mathrm{base}},\, x_t^{\mathrm{base}} - \hat{x}_t^{\mathrm{base}} \rangle.$$

Here, x_t^{base} is the user's selected base action, $\hat{x}_t^{\mathrm{base}}$ is the learner's recommended base action, and R_T is the cumulative regret.

Data generation. At each round, we sample $\tilde{z}_t \sim \mathcal{N}(0, \mathrm{I}_p)$ (standard normal) and set

$$z_t = \begin{cases} \tilde{z}_t & \|\tilde{z}_t\|_2 \le 1 \\ \tilde{z}_t / \|\tilde{z}_t\|_2 & \|\tilde{z}_t\|_2 > 1 \end{cases}$$

so that $\|z_t\|_2 \le 1$ always holds. To generate the true utility u_t^{base}, we consider two settings, referred to below as Linear and Kernel, where Kernel uses the radial basis function (RBF) kernel.

- Linear: $u_t^{\mathrm{base}} = U^\star z_t$, where $U^\star \in \mathbb{R}^{n\times p}$ is obtained by sampling independent standard normal entries and normalizing the Frobenius norm to 1.
- Kernel: $u_t^{\mathrm{base}} = h^\star(z_t)$, where

  $$h^\star(z) = \sum_{j=1}^{J} k_{\mathrm{rbf}}(z, c_j; \vartheta_{\mathrm{rbf}})\, a_j \qquad \text{and} \qquad k_{\mathrm{rbf}}(z, z'; \vartheta_{\mathrm{rbf}}) = \exp\!\left(-\frac{\|z - z'\|_2^2}{2\vartheta_{\mathrm{rbf}}^2}\right).$$

  Here, $\vartheta_{\mathrm{rbf}} > 0$ is the RBF bandwidth (length scale), $c_j \in \mathbb{R}^p$ are kernel centers, $a_j \in \mathbb{R}^n$ are coefficient vectors, and $J = 16$ is the number of centers. The centers c_j are independently drawn from the standard normal distribution and then projected onto the context support, which is the unit ball in our setting. The coefficients are first sampled from the standard normal distribution, $a_j^{\mathrm{raw}} \sim \mathcal{N}(0, \mathrm{I}_n)$, and we set $A_{\mathrm{raw}} = [(a_1^{\mathrm{raw}})^\top; \ldots; (a_J^{\mathrm{raw}})^\top] \in \mathbb{R}^{J\times n}$ and $K_{ij} = k_{\mathrm{rbf}}(c_i, c_j; \vartheta_{\mathrm{rbf}})$. We then rescale the coefficients as

  $$A = \frac{1}{\sqrt{\mathrm{tr}(A_{\mathrm{raw}}^\top K A_{\mathrm{raw}})}} A_{\mathrm{raw}},$$

  where $a_j^\top$ is row j of A for $j = 1, \ldots, J$, thereby normalizing the RKHS norm of $h^\star$ to 1. The learners use the vector-valued kernel $\mathcal{K}(z, z') = k_{\mathrm{rbf}}(z, z'; \vartheta_{\mathrm{rbf}})\mathrm{I}_n$ with the same bandwidth as the data generator.

Evaluation scenarios. In the Linear setting, we set the time horizon to $T = 10000$. For the Kernel setting, we evaluate $\vartheta_{\mathrm{rbf}} \in \{0.5, 1.0, 2.0\}$ and set $T = 1000$ to save computational cost. For each scenario, we run experiments with 10 random seeds; in each seed, all algorithms share the same round sequence (i.e., the same z_t, u_t^{base}). Results are averaged over the 10 seeds.

Compared methods. We compare CoRectron-L (CoRectron for the linear-contextual setting), ONS (Online Newton Step), OGD (Online Gradient Descent), CoRectron-K (CoRectron for the RBF-kernel setting), and KONS (Kernelized ONS). In the Linear scenario, we evaluate the first three methods; in the Kernel scenario, we evaluate all five methods.

Below, we describe the methods other than CoRectron; CoRectron itself was introduced in the main text. For convenience, we adopt the vectorized form for OGD and ONS. All methods are initialized at the origin and iteratively perform updates using base-action residual gradients $g_t^{\mathrm{base}} = \hat{x}_t^{\mathrm{base}} - x_t^{\mathrm{base}}$.

- OGD (Linear): Let $g_t = \mathrm{vec}(g_t^{\mathrm{base}} z_t^\top) \in \mathbb{R}^{np}$, where $\mathrm{vec}(\cdot)$ is the vectorization operator. With the step size $\eta_{\mathrm{OGD}} > 0$, it performs the update as

  $$\hat{w}_{t+1} = \arg\min\{\, \|w - (\hat{w}_t - \eta_{\mathrm{OGD}} g_t)\|_2 \,:\, w \in \mathbb{R}^{np},\ \|w\|_2 \le 1 \,\}.$$

- ONS (Linear): We use the same g_t. Following Sakaue et al. [38], we apply ONS to exp-concave surrogate losses. In this case, the surrogate-loss parameter $\eta_{\mathrm{sur}} > 0$ is fixed, and the ONS gradient is written as $\tilde{g}_t = \eta_{\mathrm{sur}} g_t$. Let $A_0 = \epsilon \mathrm{I}_{np}$ and perform the following update:

  $$A_t = A_{t-1} + \tilde{g}_t \tilde{g}_t^\top \qquad \text{and} \qquad \hat{y}_{t+1} = \hat{w}_t - \gamma_{\mathrm{ons}}^{-1} A_t^{-1} \tilde{g}_t.$$

  We then compute the Mahalanobis projection of $\hat{y}_{t+1}$ onto the unit ball:

  $$\hat{w}_{t+1} = \arg\min\{\, \|w - \hat{y}_{t+1}\|_{A_t} \,:\, w \in \mathbb{R}^{np},\ \|w\|_2 \le 1 \,\}.$$

- KONS (Kernel): In the RKHS, we use the coefficient representation $\hat{w}_t = \sum_{i=1}^{t-1} (\hat{c}_t)_i \phi_i$, where $\hat{c}_t \in \mathbb{R}^{t-1}$ and $\phi_i = \mathcal{K}_{z_i} g_i^{\mathrm{base}}$. Define the zero-padded coefficient vector

$$\hat{c}_t^{\mathrm{ext}} := (\hat{c}_t^\top, 0)^\top \in \mathbb{R}^t.$$

Let G_t be the Gram matrix of $\phi_1, \ldots, \phi_t$, define $\tilde{G}_t := \eta_{\mathrm{sur}}^2 G_t$, and let $e_t \in \mathbb{R}^t$ be the basis vector whose last component is 1. We also define

$$B_t := \tilde{G}_t + \epsilon \mathrm{I}_t.$$

Then, the update is written as

$$q_t = B_t^{-1} e_t \qquad \text{and} \qquad \hat{y}_{t+1} = \hat{c}_t^{\mathrm{ext}} - \gamma_{\mathrm{ons}}^{-1} (\eta_{\mathrm{sur}} q_t).$$

We then compute the RKHS Mahalanobis projection through its coefficient representation:

$$\hat{c}_{t+1} \in \arg\min \left\{ (c - \hat{y}_{t+1})^\top G_t B_t (c - \hat{y}_{t+1}) \,:\, c \in \mathbb{R}^t,\ \left\| \sum_{i=1}^{t} c_i \phi_i \right\|_{\mathcal{H}} \le 1 \right\}.$$

The projection error tolerance is 10^{-10} in the Euclidean norm for ONS and in the RKHS norm for KONS.

Hyperparameter setting. Recall that the diameter of the base action set is 1, and the utility vectors are normalized so that $\|u_t^{\mathrm{base}}\|_2 \le 1$. Therefore, we set the bound B that appears in Assumption 2 to 1. OGD, ONS, and KONS, described above, use projection onto the unit ball, and we set the radius parameter to 1 for these methods. Given these choices, the analysis of Sakaue et al. [38] suggests using the surrogate-loss scaling parameter $\eta_{\mathrm{sur}} = 1/10$ and the common step-size parameter $\gamma_{\mathrm{ons}} = 1/2$ for both ONS and KONS. From $n = p = 10$, we define $d = np = 100$ in the Linear setting. In the Kernel setting, the effective dimension that appears in the theory is not available in advance. Thus, we use the same value $d = 100$ as its proxy. The remaining hyperparameters are η_{OGD} for OGD, ϵ for ONS and KONS, and λ for CoRectron-L and CoRectron-K. These act as regularization-strength parameters: larger ϵ or λ, and smaller η_{OGD}, make the update more conservative. Because these parameters often strongly affect empirical performance, we introduce reference values $\bar{\eta}_{\mathrm{OGD}}$, $\bar{\epsilon}$, and $\bar{\lambda}$, and sweep them over the common coefficient grid $c \in \{10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 10^2, 10^3\}$ as follows:

- OGD: As the domain radius and the gradient norm are at most 1, we set the reference step size to $\bar{\eta}_{\mathrm{OGD}} = \frac{2}{\sqrt{T}}$ and sweep $\eta_{\mathrm{OGD}} \leftarrow \bar{\eta}_{\mathrm{OGD}}/c$. Therefore, larger c corresponds to a smaller step size.
- ONS and KONS: As the domain radius is 1, and the dimension and the step-size coefficient are $d = 100$ and $\gamma_{\mathrm{ons}} = 1/2$, respectively, we set the reference regularization strength to $\bar{\epsilon} = \frac{d}{4\gamma_{\mathrm{ons}}^2} = 100$ based on Sakaue et al. [38], and sweep $\epsilon \leftarrow c\bar{\epsilon}$.
- CoRectron-L and -K: As $B = 1$ and $d = 100$, we set the reference regularization to $\bar{\lambda} = B^2 d = d = 100$ and sweep $\lambda \leftarrow c\,\bar{\lambda}$.

Remark 1 (On LightONS)**.** Our experiments compare CoRectron with standard ONS. Although LightONS [42] was not implemented, no nontrivial Mahalanobis projection occurred at the tested ONS setting with the lowest mean final regret. Thus, standard ONS provides a useful baseline here, since its measured runtime does not include the nontrivial projection costs that LightONS aims to reduce. Still, the LightONS-based method may follow different iterates and incur other costs, so this observation does not establish its runtime relative to CoRectron.

I.1.2 Results

We report the following metrics, all averaged over the 10 seeds:

- Final regret: the cumulative regret R_T at the end of the horizon T.
- Total runtime: wall-clock time in seconds for predictions, linear optimization for recommendations, and updates, measured by `time.perf_counter`; data and user-action generation are excluded. To standardize runtime comparisons, we fix thread counts for BLAS/OpenMP-related libraries to 1.
- Total number of Mahalanobis projections: For ONS and KONS, we monitor the total number of rounds in which nontrivial Mahalanobis projections are performed, to examine their relation to runtime. We do not count trivial projections where the input is already within the unit ball. For CoRectron and OGD, this count is identically zero as they do not perform such projections.

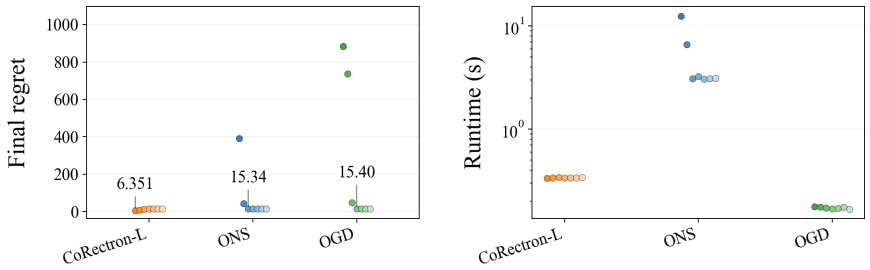


Figure 1: Overall comparison in the Linear setting. Final regret and runtime over the coefficient grid are shown. All values are means over 10 seeds. In each algorithm block, darker points correspond to smaller c, and c increases from left to right. For final regret, the annotated value is the best over the coefficient sweep.

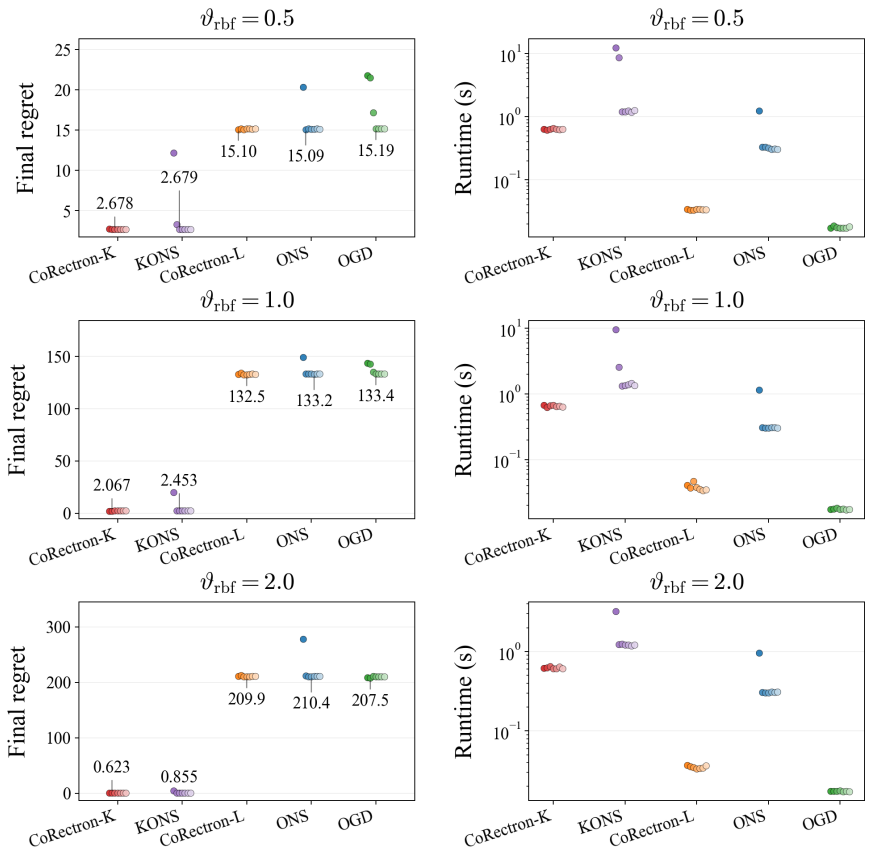


Figure 2: Overall comparison in the Kernel-RBF setting. Final regret and runtime over the coefficient grid are shown. All values are means over 10 seeds. In each algorithm block, darker points correspond to smaller c, and c increases from left to right. For final regret, the annotated value is the best over the coefficient sweep.

Overall comparison. Figures 1 and 2 show the final regret and total runtime of each method over the seven coefficient values in the sweep, reported separately for the Linear and Kernel settings as averages over 10 random seeds. The darker points correspond to smaller c, and c increases from left to right within each algorithm block. For final regret, the best value within the sweep over the scaling coefficient c is also annotated in the figures.

We first examine Figure 1, which leads to the following observations:

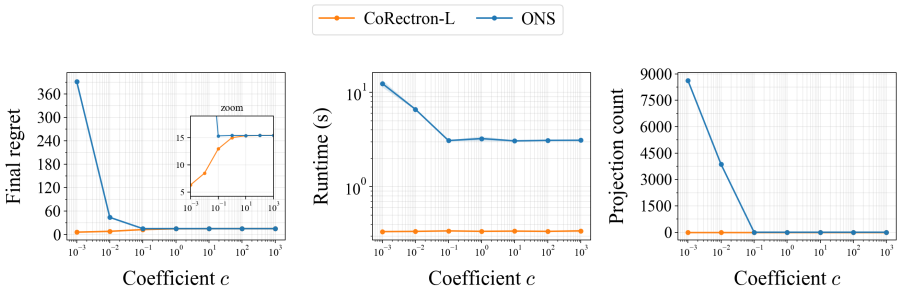


Figure 3: Comparison of CoRectron-L and ONS in the Linear setting. Final regret, runtime, and total number of Mahalanobis projections across coefficient values c are shown. Shaded bands indicate pointwise 95% confidence intervals over 10 seeds, computed by using Student's t distribution, namely mean $\pm t_{0.975,n_{\text{seeds}}-1}\, s/\sqrt{n_{\text{seeds}}}$, where s is the sample standard deviation and n_{seeds} is the number of seeds.

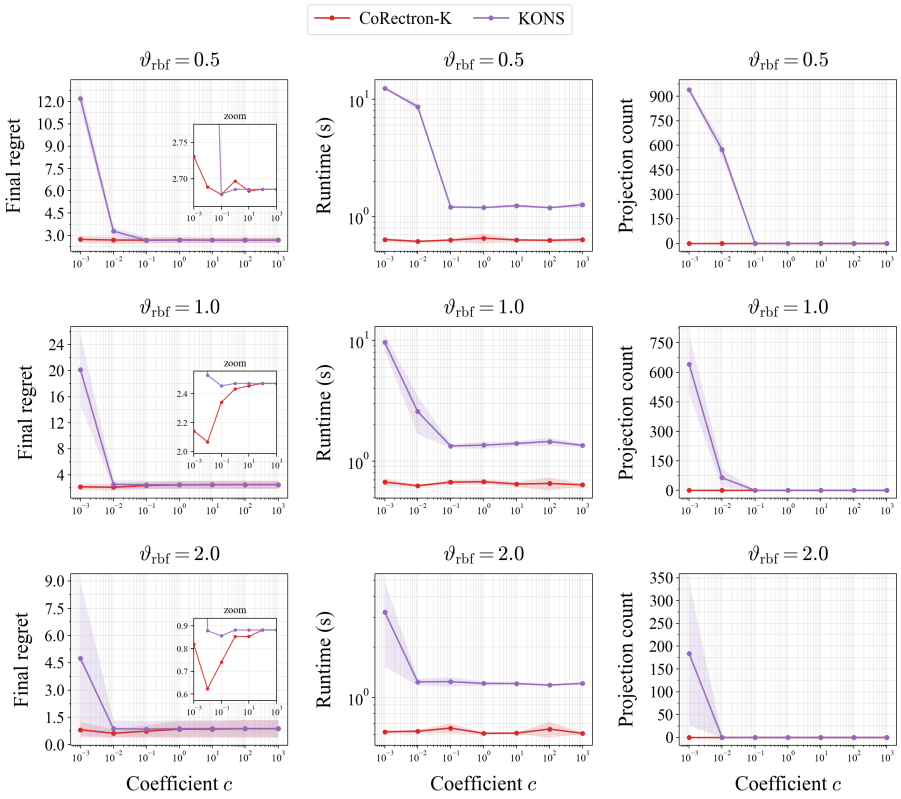


Figure 4: Comparison of CoRectron-K and KONS in the Kernel-RBF setting. Final regret, runtime, and total number of Mahalanobis projections across coefficient values c are shown. Shaded bands indicate pointwise 95% confidence intervals over 10 seeds, computed as in Figure 3.

- CoRectron-L with $c = 10^{-3}$ achieves the best final regret. The best values of ONS and OGD are roughly tied, and both are more than twice as large as the best value of CoRectron-L.
- CoRectron-L exhibits low final regret and stable runtime throughout the sweep, indicating stability to hyperparameter tuning. In contrast, for ONS and OGD, making the regularization too weak (that is, taking c too small) causes final regret to increase substantially; for ONS, runtime also increases. Moreover, the best final regret of these methods is attained at the smallest value of c that avoids such deterioration, suggesting that they require relatively delicate hyperparameter tuning.
- In terms of runtime, OGD is the fastest, while CoRectron-L is faster than ONS across all c values.

We next turn to Figure 2. For reference, this figure also plots the results of the methods designed for the Linear setting, namely CoRectron-L, ONS, and OGD. The following observations can be made:

- CoRectron-K achieves the best final regret, slightly outperforming KONS. In addition, the final regret of all methods designed for the Linear setting is substantially worse than that of the methods designed for the Kernel setting, demonstrating the importance of using a contextual model that matches the underlying data generation.
- As in the Linear case, KONS also exhibits deteriorated final regret and runtime when c is too small. In contrast, CoRectron-K, like CoRectron-L in the Linear setting, achieves consistently low final regret and runtime across the tuning of the regularization parameter.

Detailed comparisons of CoRectron-L/-K and ONS/KONS. Figures 3 and 4 provide a closer comparison between CoRectron-L/-K and ONS/KONS. The horizontal axis is the coefficient c, and the vertical axes report final regret, runtime, and the total number of Mahalanobis projections. The shaded bands indicate the 95% confidence intervals around the means over 10 seeds.

- Consistent with the overall comparison above, CoRectron achieves the best final regret, maintaining high performance and efficiency stably across changes in c.
- The runtime curves of ONS/KONS track their projection-count curves, suggesting that repeated Mahalanobis projections are a major contributor to the increased runtime at small c. CoRectron avoids this additional cost and maintains stable runtime across the coefficient sweep, highlighting the practical benefit of eliminating Mahalanobis projections.

I.2 Experiments with suboptimal-action feedback

This section examines the robustness of the compared algorithms when the observed action can be suboptimal, as discussed in Section 4. We use the Linear setting of Appendix I.1 throughout; refer to Appendix I.1 for the problem setting, data generation, and descriptions of the compared methods (CoRectron-L, ONS, OGD).

I.2.1 Experimental setup

Hyperparameter setting. We fix each method's regularization coefficient at the best-performing value identified in the sweep of Appendix I.1: $c = 10^{-3}$ for CoRectron-L, $c = 10^{-1}$ for ONS, and $c = 1$ for OGD.

Suboptimal-action models. We consider two models of possibly suboptimal user actions. In both models, the true utility vector u_t^{base} is generated as in Appendix I.1.

- One-swap (swap probability $\alpha \in [0, 1]$): With probability α, the item ranked m-th under u_t^{base} (the marginal item in the optimal selection) is replaced by the item ranked $(m+1)$-th (the best excluded item); with probability $1 - \alpha$, the optimal action $x_t^{\text{base}} \in \arg\max_{x \in \mathcal{X}_t^{\text{base}}} \langle u_t^{\text{base}}, x \rangle$ is revealed unchanged. This yields per-round suboptimality $\delta_t = \big(u_t^{\text{base}}(m) - u_t^{\text{base}}(m+1)\big)/\sqrt{10} \geq 0$ whenever the swap occurs, where $u_t^{\text{base}}(k)$ denotes the k-th largest component. At $\alpha = 0$ this coincides with the optimal-feedback setting. We sweep $\alpha \in \{0.0, 0.1, \ldots, 1.0\}$.
- Score-perturb (relative noise level $\xi \geq 0$): The revealed action is taken as the argmax of the perturbed utility $u_t^{\text{base}} + \xi \|u_t^{\text{base}}\|_2 \, \varepsilon_t$, where $\varepsilon_t \sim \mathcal{N}(0, \mathrm{I}_n)$ is drawn independently each round. Scaling the noise magnitude by $\|u_t^{\text{base}}\|_2$ makes ξ a relative noise level that is invariant to the scale of u_t^{base}. At $\xi = 0$ this recovers the optimal-feedback setting. We sweep $\xi \in \{0.0, 0.1, \ldots, 1.0\}$.

Remark 2 (On MetaGrad-based method)**.** We leave an experimental comparison with the MetaGrad-based method [38, 40] to future work, since implementing and tuning that baseline would require substantial additional engineering effort. It also needs to maintain $\Theta(\log T)$ ONS instances, increasing the cost of ONS updates by a factor of $O(\log T)$.

I.2.2 Results

Figure 5 shows the final regret as a function of α and ξ. The following observations can be made:

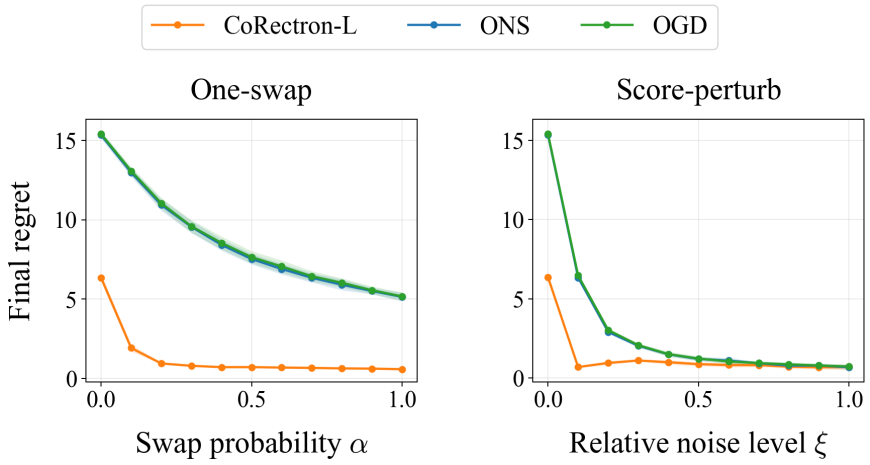


Figure 5: Robustness to suboptimal-action feedback in the Linear setting. Left: final regret R_T against swap probability α (one-swap model). Right: final regret R_T against relative noise level ξ (score-perturb model). Each method uses the best-performing coefficient c from Appendix I.1. All values are means over 10 seeds; shaded bands indicate pointwise 95% confidence intervals computed as in Figure 3.

- CoRectron-L achieves the lowest mean final regret across all noise levels in both settings, except at $\xi = 1$ in the score-perturb model, where ONS achieves a slightly lower value.
- The final regret of all methods tends to decrease as α or ξ increases. This reflects the definition of regret $R_T = \sum_{t=1}^{T} \langle u_t^{\text{base}}, x_t^{\text{base}} - \hat{x}_t^{\text{base}} \rangle$: as x_t^{base} becomes more suboptimal, the target the learner must compete against weakens, which can reduce R_T. This trend is not directly reflected by the worst-case regret upper bound in Theorem 2 involving the $\sqrt{2B\Delta_T(u)H_T}$ term, which is increasing in the cumulative suboptimality $\Delta_T(u)$. Bridging this gap—developing a theoretical guarantee that explains the empirical decrease in regret as the user action becomes more suboptimal—is an interesting open problem.

Declaration of LLM usage

The author used OpenAI's ChatGPT and Codex for manuscript editing, literature checks, and proof checking. The author takes responsibility for all content.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction accurately summarize the main contributions and scope of the paper, including the projection-free algorithm, logarithmic regret guarantees, robustness properties, and contextual extensions.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 7 discusses the main limitations of this work, including oracle assumptions and computational costs.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best

judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: The assumptions and proofs accompany Theorems 1 and 2 and Corollary 1; auxiliary proofs are provided in Appendices B, C, E and F.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Appendix I specifies the synthetic environment, compared methods, hyperparameters, metrics, and evaluation over 10 random seeds.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The code, synthetic-data generators, and reproduction instructions are available at https://github.com/ssakaue/projection-free-cr-public.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Appendices I.1 and I.2 specify the model classes, action spaces, hyperparameter grids, selection of coefficients, evaluation metrics, and baseline configurations.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Figures 3–5 show 95% confidence intervals for means over 10 random seeds. The caption of Figure 3 specifies the calculation using Student's t distribution.

   Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Appendix I specifies the hardware, memory, software versions, thread settings, runtime measurements, and aggregate timed runtime, and discloses additional computation for preliminary experiments and implementation checks.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: The work is a theoretical and synthetic experimental study and does not involve sensitive data, human subjects, or other aspects raising a specific ethics concern beyond standard scientific conduct.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [N/A]

Justification: This work studies computational efficiency and regret guarantees for general online learning algorithms. We have not identified specific societal impacts arising directly from these algorithmic contributions.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The paper does not release a model, dataset, or other asset that creates a recognizable high-risk misuse concern requiring safeguards.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: The experiments use NumPy, pandas, and SciPy under BSD-3-Clause licenses, Matplotlib under its PSF-based license, and PyYAML under the MIT license; dependency versions are listed in the code package. No third-party datasets or pretrained models are used.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The accompanying code package includes a README, reproduction instructions, configuration files, dependency versions, and an MIT license.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The work does not involve crowdsourcing or human-subject research.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The work does not involve human subjects, so IRB approval is not applicable.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: The use of ChatGPT and Codex is disclosed in Declaration of LLM usage.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.