Online Learning with Local Permutations and Delayed Feedback

Ohad Shamir [*1] **Liran Szlak** [*1]

Abstract

We propose an Online Learning with Local Permutations (OLLP) setting, in which the learner is allowed to slightly permute the *order* of the loss functions generated by an adversary. On one hand, this models natural situations where the exact order of the learner's responses is not crucial, and on the other hand, might allow better learning and regret performance, by mitigating highly adversarial loss sequences. Also, with random permutations, this can be seen as a setting interpolating between adversarial and stochastic losses. In this paper, we consider the applicability of this setting to convex online learning with delayed feedback, in which the feedback on the prediction made in round t arrives with some delay τ. With such delayed feedback, the best possible regret bound is well-known to be $O(\sqrt{\tau T})$. We prove that by being able to permute losses by a distance of at most M (for $M \geq \tau$), the regret can be improved to $O(\sqrt{T}(1 + \sqrt{\tau^2/M}))$, using a Mirror-Descent based algorithm which can be applied for both Euclidean and non-Euclidean geometries. We also prove a lower bound, showing that for $M < \tau/3$, it is impossible to improve the standard $O(\sqrt{\tau T})$ regret bound by more than constant factors. Finally, we provide some experiments validating the performance of our algorithm.

1. Introduction

Online learning is traditionally posed as a repeated game where the learner has to provide predictions on an arbitrary sequence of loss functions, possibly even generated adversarially. Although it is often possible to devise algorithms with non-trivial regret guarantees, these have to cope with arbitrary loss sequences, which makes them conservative and in some cases inferior to algorithms not tailored to cope with worst-case behavior. Indeed, an emerging line of work considers how better online learning can be obtained on "easy" data, which satisfies some additional assumptions. Some examples include losses which are sampled i.i.d. from some distribution, change slowly in time, have a consistently best-performing predictor across time, have some predictable structure, mix adversarial and stochastic losses, etc. (e.g. (Sani et al., 2014; Karnin and Anava, 2016; Bubeck and Slivkins, 2012; Seldin and Slivkins, 2014; Hazan and Kale, 2010; Chiang et al., 2012; Steinhardt and Liang, 2014; Hazan and Kale, 2011; Rakhlin and Sridharan, 2013; Seldin and Slivkins, 2014)).

In this paper, we take a related but different direction: Rather than explicitly excluding highly adversarial loss sequences, we consider how slightly perturbing them can mitigate their worst-case behavior, and lead to improved performance. Conceptually, this resembles smoothed analysis (Spielman and Teng, 2004), in which one considers the worst-case performance of some algorithm, after performing some perturbation to their input. The idea is that if the worst-case instances are isolated and brittle, then a perturbation will lead to easier instances, and better reflect the attainable performance in practice.

Specifically, we propose a setting, in which the learner is allowed to slightly *reorder* the sequence of losses generated by an adversary: Assuming the adversary chooses losses $h_1, \ldots, h_T$, and before any losses are revealed, the learner may choose a permutation σ on $\{1, \ldots, T\}$, satisfying $\max_t |t - \sigma(t)| \leq M$ for some parameter M, and then play a standard online learning game on losses $h_{\sigma(1)}, \ldots, h_{\sigma(T)}$. We denote this as the *Online Learning with Local Permutations* (OLLP) setting. Here, M controls the amount of power given to the learner: $M = 0$ means that no reordering is performed, and the setting is equivalent to standard adversarial online learning. At the other extreme, $M = T$ means that the learner can reorder the losses arbitrarily. For example, the learner may choose to order the losses uniformly at random, making it a quasi-stochastic setting (the only difference compared to i.i.d. losses is that they are sampled without-replacement rather than with-replacement).

We argue that allowing the learner some flexibility in the

[*]Equal contribution [1]Weizmann Institute of Science, Rehovot, Israel. Correspondence to: Liran Szlak <liran.szlak@weizmann.ac.il>.

Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017. Copyright 2017 by the author(s).

order of responses is a natural assumption. For example, when the learner needs to provide rapid predictions on a high-frequency stream of examples, it is often immaterial if the predictions are not provided in the exact same order at which the examples arrived. Indeed, by buffering examples for a few rounds before being answered, one can simulate the local permutations discussed earlier.

We believe that this setting can be useful in various online learning problems, where it is natural to change a bit the order of the loss functions. In this paper, we focus on one well-known problem, namely online learning with delayed feedback. In this case, rather than being provided with the loss function immediately after prediction is made, the learner only receives the loss function after a certain number $\tau \geq 1$ of rounds. This naturally models situations where the feedback comes much more slowly than the required frequency of predictions: To give a concrete example, consider a web advertisement problem, where an algorithm picks an ad to display, and then receives a feedback from the user in the form of a click. It is likely that the algorithm will be required to choose ads for new users while still waiting for the feedback from the previous user. Web advertisement, answering user queries and many other regimes where the feedback come from a user, are all relevant to this setting. It is also plausible that the distribution of examples in the above scenarios is not stochastic neither is it adversarial, and thus the proposed setting of Online Learning With Local Permutation is relevant to these cases.

For convex online learning with delayed feedback, in a standard adversarial setting, it is known that the attainable regret is on the order of $O(\sqrt{\tau T})$, and this is also the best possible in the worst case (Weinberger and Ordentlich, 2002; Mesterharm, 2005; Langford et al., 2009; Joulani et al., 2013; Quanrud and Khashabi, 2015). On the other hand, in a stochastic setting where the losses are sampled i.i.d. from some distribution, (Agarwal and Duchi, 2011) show that the attainable regret is much better, on the order of $O(\sqrt{T}+\tau)$. This gap between the worst-case adversarial setting, and the milder i.i.d. setting, hints that this problem is a good fit for our OLLP framework.

Thus, in this paper, we focus on online learning with feedback delayed up to τ rounds, in the OLLP framework where the learner is allowed to locally permute the loss functions (up to a distance of M). First, we devise an algorithm, denoted as Delayed Permuted Mirror Descent, and prove that it achieves an expected regret bound of order $O(\sqrt{T(\tau^2/M+1)})$ assuming $M \geq \tau$. As M increases compared to τ, this regret bound interpolates between the standard adversarial $\sqrt{\tau T}$ regret, and a milder $\sqrt{T}$ regret, typical of i.i.d. losses. As its name implies, the algorithm is based on the well-known online mirror descent (OMD) algorithm (see (Hazan et al., 2016; Shalev-Shwartz et al., 2012)), and works in the same generality, involving both Euclidean and non-Euclidean geometries. The algorithm is based on dividing the entire sequence of functions into blocks of size M and performing a random permutation within each block. Then, two copies of OMD are ran on different parts of each block, with appropriate parameter settings. A careful analysis, mixing adversarial and stochastic elements, leads to the regret bound.

In addition, we provide a lower bound complementing our upper bound analysis, showing that when M is significantly smaller than τ (specifically, $\tau/3$), then even with local permutations, it is impossible to obtain a worse-case regret better than $\Omega(\sqrt{\tau T})$, matching (up to constants) the attainable regret in the standard adversarial setting where no permutations are allowed. Finally, we provide some experiments validating the performance of our algorithm.

The rest of the paper is organized as follows: in section 2 we formally define the Online Learning with Local Permutation setting, section 3 describes the Delayed Permuted Mirror Descent algorithm and outlines its regret analysis, section 4 discusses a lower bound for the delayed setting with limited permutation power, section 5 shows experiments, and finally section 6 provides concluding remarks, discussion, and open questions. Appendix A contains most of the proofs.

2. Setting and Notation

Convex Online Learning. Convex online learning is posed as a repeated game between a learner and an adversary (assumed to be oblivious in this paper). First, the adversary chooses T convex losses $h_1, \ldots, h_T$ which are functions from a convex set $\mathcal{W}$ to $\mathbb{R}$. At each iteration $t \in \{1, 2, \ldots, T\}$, the learner makes a prediction w_t, and suffers a loss of $h_t(w_t)$. To simplify the presentation, we use the same notation $\nabla h_t(w)$ to denote either a gradient of h_t at w (if the loss is differentiable) or a subgradient at w otherwise, and refer to it in both cases as a gradient. We assume that both $w \in \mathcal{W}$ and the gradients of any function h_t in any point $w \in \mathcal{W}$ are bounded w.r.t. some norm: Given a norm $\|\cdot\|$ with a dual norm $\|\cdot\|_*$, we assume that the diameter of the space $\mathcal{W}$ is bounded by B^2 and that $\forall w \in \mathcal{W}, \forall h \in \{h_1, h_2, ..., h_T\} : \|\nabla h(w)\|_* \leq G$. The purpose of the learner is to minimize her (expected) regret, i.e.

$$R(T) = \mathbb{E}\left[\sum_{t=1}^{T} h_t(w_t) - \sum_{t=1}^{T} h_t(w^*)\right]$$

$$\text{where } w^* = \underset{w \in \mathcal{W}}{\operatorname{argmin}} \sum_{t=1}^{T} h_t(w)$$

where the expectation is with respect to the possible randomness of the algorithm.

Learning with Local Permutations. In this paper, we introduce and study a variant of this standard setting, which gives the learner a bit more power, by allowing her to slightly modify the *order* in which the losses are processed, thus potentially avoiding highly adversarial but brittle loss constructions. We denote this setting as the Online Learning with Local Permutations (OLLP) setting. Formally, letting M be a permutation window parameter, the learner is allowed (at the beginning of the game, and before any losses are revealed) to permute $h_1, \dots, h_T$ to $h_{\sigma^{-1}(1)}, \dots, h_{\sigma^{-1}(T)}$, where σ is a permutation from the set $Perm := \{\sigma : \forall t, |\sigma(t) - t| \le M\}$. After this permutation is performed, the learner is presented with the permuted sequence as in the standard online learning setting, with the same regret as before. To simplify notation, we let $f_t = h_{\sigma^{-1}(t)}$, so the learner is presented with the loss sequence $f_1, \dots, f_T$, and the regret is the same as the standard regret, i.e. $R(T) = \mathbb{E}\left[\sum_{t=1}^T f_t(w_t) - \sum_{t=1}^T f_t(w^*)\right]$. Note that if $M = 0$ then we are in the fully adversarial setting (no permutation is allowed). At the other extreme, if $M = T$ and σ is chosen uniformly at random, then we are in a stochastic setting, with a uniform distribution over the set of functions chosen by the adversary (note that this is close but differs a bit from a setting of i.i.d. losses). In between, as M varies, we get an interpolation between these two settings.

Learning with Delayed Feedback. The OLLP setting can be useful in many applications, and can potentially lead to improved regret bounds for various tasks, compared to the standard adversarial online learning. In this paper, we focus on studying its applicability to the task of learning from delayed feedback.

Whereas in standard online learning, the learner gets to observe the loss f_t immediately at the end of iteration t, here we assume that at round t, she only gets to observe $f_{t-\tau}$ for some delay parameter $\tau < T$ (and if $t < \tau$, no feedback is received). For simplicity, we focus on the case where τ is fixed, independent of t, although our results can be easily generalized (as discussed in subsection 3.3). We emphasize that this is distinct from another delayed feedback scenario sometimes studied in the literature (Agarwal and Duchi, 2011; Langford et al., 2009), where rather than receiving $f_{t-\tau}$ the learner only receives a (sub)gradient of $f_{t-\tau}$ at $w_{t-\tau}$. This is a more difficult setting, which is relevant for instance when the delay is due to the time it takes to compute the gradient.

3. Algorithm and Analysis

Our algorithmic approach builds on the well-established online mirror descent framework. Thus, we begin with a short reminder of the Online Mirror Descent algorithm. For a more extensive explanation refer to (Hazan et al., 2016). Readers who are familiar with the algorithm are invited to skip to Subsection 3.1.

The online mirror descent algorithm is a generalization of online gradient descent, which can handle non-Euclidean geometries. The general idea is the following: we start with some point $w_t \in \mathcal{W}$, where $\mathcal{W}$ is our primal space. We then map this point to the dual space using a (strictly convex and continuously differentiable) mirror map ψ, i.e. $\nabla\psi(w_t) \in \mathcal{W}^*$, then perform the gradient update in the dual space, and finally map the resulting new point back to our primal space $\mathcal{W}$ again, i.e. we want to find a point $w_{t+1} \in \mathcal{W}$ s.t. $\nabla\psi(w_{t+1}) = \nabla\psi(w_t) - \eta \cdot g_t$ where g_t denotes the gradient. Denoting by $w_{t+\frac{1}{2}}$ the point satisfying $\nabla\psi(w_{t+\frac{1}{2}}) = \nabla\psi(w_t) - \eta \cdot g_t$, it can be shown that $w_{t+\frac{1}{2}} = (\nabla\psi^*)(\nabla\psi(w_t) - \eta \cdot g_t)$, where ψ^* is the dual function of ψ. This point, $w_{t+\frac{1}{2}}$, might lie outside our hypothesis class $\mathcal{W}$, and thus we might need to project it back to our space $\mathcal{W}$. We use the Bregman divergence associated to ψ to do this:

$$w_{t+1} = \underset{w \in \mathcal{W}}{argmin} \triangle_\psi(w, w_{t+\frac{1}{2}}),$$

where the Bregman divergence Δ_ψ is defined as

$$\triangle_\psi(x, y) = \psi(x) - \psi(y) - \langle \nabla\psi(y), x - y \rangle.$$

Specific choices of the mirror map ψ leads to specific instantiations of the algorithms for various geometries. Perhaps the simplest example is $\psi(x) = \frac{1}{2}\|x\|_2^2$, with associated Bregman divergence $\triangle_\psi(x, y) = \frac{1}{2} \cdot \|x - y\|^2$. This leads us to the standard and well-known online gradient descent algorithm, where w_{t+1} is the Euclidean projection on the set $\mathcal{W}$ of

$$w_t - \eta \cdot g_t.$$

Another example is the negative entropy mirror map $\psi(x) = \sum_{i=1}^n x_i \cdot \log(x_i)$, which is 1-strongly convex with respect to the 1-norm on the simplex $\mathcal{W} = \left\{x \in \mathbb{R}_+^n : \sum_{i=1}^n x_i = 1\right\}$. In that case, the resulting algorithm is the well-known multiplicative updates algorithm, where

$$w_{t+1,i} = w_{t,i} \cdot \exp(-\eta g_{t,i}) / \sum_{j=1}^n w_{t,i} \cdot \exp(-\eta g_{t,j}).$$

Instead of the 1-norm on the simplex, one can also consider arbitrary p-norms, and take $\psi(x) = \frac{1}{2} \cdot \|x\|_q^2$, where q is the dual norm (satisfying $1/p + 1/q = 1$).

3.1. The Delayed Permuted Mirror Descent Algorithm

Before describing the algorithm, we note that we will focus here on the case where the permutation window parameter

M is larger than the delay parameter τ. If $M < \tau$, then our regret bound is generally no better than the $O(\sqrt{\tau T})$ obtainable by a standard algorithm without any permutations, and this is actually tight as shown in Section 4.

We now turn to present our algorithm, denoted as The Delayed Permuted Mirror Descent algorithm (see algorithm 1 below as well as figure 1 for a graphical illustration). First, the algorithm splits the time horizon T into M consecutive blocks, and performs a uniformly random permutation on the loss functions within each block. Then, it runs two online mirror descent algorithms in parallel, and uses the delayed gradients in order to update two separate predictors – w^f and w^s, where w^f is used for prediction in the first τ rounds of each block, and w^s is used for prediction in the remaining $M-\tau$ rounds (here, f stands for "first" and s stands for "second"). The algorithm maintaining w^s crucially relies on the fact that the gradient of any two functions in a block (at some point w) is equal, in expectation over the random permutation within each block. This allows us to avoid most of the cost incurred by delays within each block, since the expected gradient of a delayed function and the current function are equal. A complicating factor is that at the first τ rounds of each block, no losses from the current block has been revealed so far. To tackle this, we use another algorithm (maintaining w^f), specifically to deal with the losses at the beginning of each block. This algorithm does not benefit from the random permutation, and its regret scales the same as standard adversarial online learning with delayed feedback. However, as the block size M increases, the *proportion* of losses handled by w^f decreases, and hence its influence on the overall regret diminishes.

The above refers to how the blocks are divided for purposes of *prediction*. For purposes of *updating* the predictor of each algorithm, we need to use the blocks a bit differently. Specifically, we let T_1 and T_2 be two sets of indices. T_1 includes all indices from the first τ time points of every block, and is used to update w^f. T_2 includes the first $M-\tau$ indices of every block, and is used to update w^s (see figure 1). Perhaps surprisingly, note that T_1 and T_2 are not disjoint, and their union does not cover all of $\{1,\ldots,T\}$. The reason is that due to the random permutation in each block, the second algorithm only needs to update on some of the loss functions in each block, in order to obtain an expected regret bound on all the losses it predicts on.

Algorithm 1 Delayed Permuted Mirror Descent

```
**Input:** $M, \eta_f, \eta_s$
Init: $w_1^f = 0, w_1^s = 0, j_f = j_s = 1$
Divide $T$ to consecutive blocks of size $M$, and permute the losses uniformly at random within each block. Let $f_1, \ldots, f_T$ denote the resulting permuted losses.
**for** $t = 1..., T$ **do**
  **if** $t \in$ first $\tau$ rounds of the block **then**
    Predict using $w_{j_f}^f$
    Receive a loss function from $\tau$ places back: $f_{t-M} = f_{T_1(j_f-\tau)}$. If none exists (in the first $\tau$ iterations), take the 0 function.
    Compute: $\nabla f_{T_1(j_f-\tau)}\left(w_{j_f-\tau}^f\right)$
    Update: $w_{j_f+\frac{1}{2}}^f = (\nabla\psi^*)\left(\nabla\psi\left(w_{j_f}^f\right) - \eta_f \nabla f_{T_1(j_f-\tau)}\left(w_{j_f-\tau}^f\right)\right)$
    Project: $w_{j_f+1} = \underset{w\in\mathcal{W}}{argmin}\triangle_\psi\left(w, w_{j_f+\frac{1}{2}}^f\right)$
    $j_f = j_f + 1$
  **else**
    Predict using $w_{j_s}^s$
    Receive a loss function from $\tau$ places back: $f_{t-\tau} = f_{T_2(j_s)}$
    Compute: $\nabla f_{t-\tau}\left(w_{j_s}^s\right) = \nabla f_{T_2(j_s)}\left(w_{j_s}^s\right)$
    Update: $w_{j_s+\frac{1}{2}}^s = (\nabla\psi^*)\left(\nabla\psi\left(w_{j_s}^s\right) - \eta_s \cdot \nabla f_{T_2(j_s)}\left(w_{j_s}^s\right)\right)$
    Project: $w_{j_s+1} = \underset{w\in\mathcal{W}}{argmin}\triangle_\psi\left(w, w_{j_s+\frac{1}{2}}^s\right)$
    $j_s = j_s + 1$
  **end if**
**end for**
```

3.2. Analysis

The regret analysis of the Delayed Permuted Mirror Descent algorithm is based on a separate analysis of each of the two mirror descent sub-algorithms, where in the first sub-algorithm the delay parameter τ enters multiplicatively, but doesn't play a significant role in the regret of the second sub-algorithm (which utilizes the stochastic nature of the permutations). Combining the regret bound of the two sub-algorithms, and using the fact that the portion of losses predicted by the second algorithm increases with M, leads to an overall regret bound improving in M.

In the proof, to analyze the effect of delay, we need a bound on the distance between any two consequent predictors w_t, w_{t+1} generated by the sub-algorithm. This depends on the mirror map and Bregman divergence used for the update, and we currently do not have a bound holding in full generality. Instead, we let $\Psi_{(\eta_f,G)}$ be some upper bound on $\|w_{t+1} - w_t\|$, where the update is using step-size η_f and gradients of norm $\leq G$. Using $\Psi_{(\eta_f,G)}$ we prove a general bound for all mirror maps. In Lemmas 3 and 4 in Appendix A.1, we show that for two common mirror maps (corresponding to online gradient descent and multiplicative weights), $\Psi_{(\eta_f,G)} \leq c \cdot \eta_f G$ for some numerical constant c, leading to a regret bound of $O(\sqrt{T(\tau^2/M+1)})$. Also, we prove theorem 1 for 1-strongly convex mirror

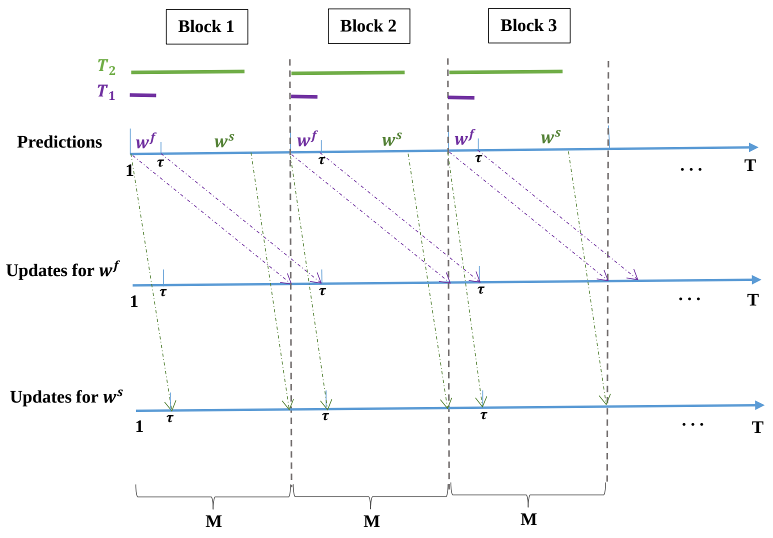

Figure 1. Scheme of predictions and updates of both parallel algorithms (best viewed in color; see text for details). Top color bars mark which iterations are in T_1 (purple lines) and which are in T_2 (green lines). Top timeline shows which predictor, w^f or w^s, is used to predict in each iteration. Middle timeline shows where gradients for updating w^f come from (first τ iterations of the previous block), and lower timeline shows where gradients for updating w^s come from (first $M - \tau$ iterations of the same block, each gradient from exactly τ rounds back).

maps, although it can be generalized to any λ-strongly convex mirror map by scaling.

Theorem 1. *Given a norm $\|\cdot\|$, suppose that we run the Delayed Permuted Mirror Descent algorithm using a mirror map ψ which is 1-strongly convex w.r.t. $\|\cdot\|$, over a domain $\mathcal{W}$ with diameter B^2 w.r.t the bregman divergence of ψ: $\forall w, v \in \mathcal{W} : \triangle_\psi(w,v) \leq B^2$, and such that the (sub)-gradient g of each loss function on any $w \in \mathcal{W}$ satisfies $\|g\|_* \leq G$ (where $\|\cdot\|_*$ is the dual norm of $\|\cdot\|$). Then the expected regret, given a delay parameter τ and step sizes η_f, η_s satisfies:*

$$\mathbb{E}\left[\sum_{t=1}^{T} f_t(w_t) - f_t(w^*)\right] \leq \frac{B^2}{\eta_f} + \eta_f \cdot \frac{T\tau}{M} \cdot \frac{G^2}{2} + \frac{T\tau^2}{M} \cdot G \cdot \Psi_{(\eta_f, G)} + \frac{B^2}{\eta_s} + \eta_s \cdot \frac{T \cdot (M-\tau)}{M} \cdot \frac{G^2}{2}$$

Furthermore, if $\Psi_{(\eta_f,G)} \leq c \cdot \eta_f G$ for some constant c, and $\eta_f = \frac{B \cdot \sqrt{M}}{G \cdot \sqrt{T \cdot \tau \cdot \left(\frac{1}{2} + c \cdot \tau\right)}}$, $\eta_s = \frac{B \cdot \sqrt{2M}}{G \cdot \sqrt{T \cdot (M-\tau)}}$, the regret is bounded by

$$c\sqrt{\frac{T\tau}{M}} \cdot BG\sqrt{\frac{1}{2} + c \cdot \tau} + \sqrt{\frac{2T(M-\tau)}{M}} \cdot BG = \mathcal{O}\left(\sqrt{T} \cdot \left(\sqrt{\frac{\tau^2}{M}} + 1\right)\right)$$

When $M = \mathcal{O}(\tau)$, this bound is $O(\sqrt{\tau T})$. similar to the standard adversarial learning case. However, as M increases, the regret gradually improves to $O(\sqrt{T} + \tau)$, which is the regret attainable in a purely stochastic setting with i.i.d. losses. The full proof can be found in appendix A.1.1, and we sketch below the main ideas.

First, using the definition of regret, we show that it is enough to upper-bound the regret of each of the two sub-algorithms separately. Then, by a standard convexity argument, we reduce this to bounding sums of terms of the form $\mathbb{E}[\langle w_t^f - w_f^*, \nabla f_t(w_t^f)\rangle]$ for the first sub-algorithm, and $\mathbb{E}\left[\langle w_t^s - w_s^*, \nabla f_t\left(w_t^s\right)\rangle\right]$ for the second sub-algorithm (where w_f^* and w_s^* are the best fixed points in hindsight for the losses predicted on by the first and second sub-algorithms, respectively, and where for simplicity we assume the losses are differentiable). In contrast, we can use the standard analysis of mirror descent, using delayed gradients, to get a bound for the somewhat different terms $\mathbb{E}[\langle w_t^f - w_f^*, \nabla f_{t-\tau}(w_{t-\tau}^f)\rangle]$ for the first sub-algorithm, and $\mathbb{E}\left[\langle w_t^s - w_s^*, \nabla f_{t-\tau}\left(w_t^s\right)\rangle\right]$ for the second sub-algorithm. Thus, it remains to bridge between these terms.

Starting with the second sub-algorithm, we note that since we performed a random permutation within each block, the expected value of all loss functions within a block (in expectation over the block, and evaluated at a fixed point) is equal. Moreover, at any time point, the predictor w^s maintained by the second sub-algorithm does not depend on the delayed nor the current loss function. Therefore, conditioned on w_t^s, and in expectation over the random permutation in the block, we have that

$$\mathbb{E}[\nabla f_t(w_t^s)] = \mathbb{E}[f_{t-\tau}(w_t^s)]$$

from which it can be shown that

$$\mathbb{E}\left[\langle w_t^s - w_s^*, \nabla f_t(w_t^s)\rangle\right] = \mathbb{E}\left[\langle w_t^s - w_s^*, \nabla f_{t-\tau}(w_t^s)\rangle\right]$$

Thus, up to a negligible factor having to do with the first few rounds of the game, the second sub-algorithm's expected regret does not suffer from the delayed feedback.

For the first sub-algorithm, we perform an analysis which does not rely on the random permutation. Specifically, we first show that since we care just about the sum of the losses, it is sufficient to bound the difference between $\mathbb{E}[\langle w_t^f - w_f^*, \nabla f_t(w_t^f)\rangle]$ and $\mathbb{E}[\langle w_{t+\tau}^f - w_f^*, \nabla f_t(w_t^f)\rangle]$. Using Cauchy-Shwartz, this difference can be upper bounded by $\|w_t^f - w_{t+\tau}^f\| \cdot \|\nabla f_t(w_t^f)\|$, which in turn is at most $c \cdot \tau \cdot \eta_f \cdot G^2$ using our assumptions on the gradients of the losses and the distance between consecutive predictors produced by the first sub-algorithm.

Overall, we get two regret bounds, one for each sub-algorithm. The regret of the first sub-algorithm scales with τ, similar to the no-permutation setting, but the sub-algorithm handles only a small fraction of the iterations (the first τ in every block of size M). In the rest of the iterations, where we use the second sub-algorithm, we get a bound that resembles more the stochastic case, without such dependence on τ. Combining the two, the result stated in Theorem 1 follows.

3.3. Handling Variable Delay Size

So far, we discussed a setting where the feedback arrives with a fixed delay of size τ. However, in many situations the feedback might arrive with a variable delay size τ_t at any iteration t, which may raise a few issues.
First, feedback might arrive in an asynchronous fashion, causing us to update our predictor using gradients from time points further in past after already using more recent gradients. This complicates the analysis of the algorithm. A second, algorithmic problem, is that we could also possibly receive multiple feedbacks simultaneously, or no feedback at all, in certain iterations, since the delay is of variable size. One simple solution is to use buffering and reduce the problem to a constant delay setting. Specifically, we assume that all delays are bounded by some maximal delay size τ. We would like to use one gradient to update our predictor at every iteration (this is mainly for ease of analysis, practically one could update the predictor with multiple loss functions in a single iteration). In order to achieve this, we can use a buffer to store loss functions that were received but have not been used to update the predictors yet. We define $Grad_f$ and $Grad_s$, two buffers that will contain gradients from time points in T_1 or T_2, correspondingly. Each buffer is of size τ. If we denote by $\mathcal{F}_t$ the set of function that have arrived in time t, we can simply store loss functions that have arrived asynchronously in the buffers defined above, sort them in ascending order, and take the delayed loss function from exactly τ iterations back in the update step. This loss function must be in the appropriate buffer since the maximal delay size is τ. From this moment on, the algorithm can proceed as usual and its analysis still applies.

4. Lower Bound

In this section, we give a lower bound in the setting where $M < \frac{\tau}{3}$ with all feedback having delay of exactly τ. We will show that for this case, the regret bound cannot be improved by more than a constant factor over the bound of the adversarial online learning problem with a fixed delay of size τ, namely $\Omega\left(\sqrt{\tau T}\right)$ for a sequence of length T. We hypothesize that this regret bound also cannot be significantly improved for any $M = O(\tau)$ (and not just $\tau/3$). However, proving this remains an open problem.

Theorem 2. *For every (possible randomized) algorithm A with a permutation window of size $M \leq \frac{\tau}{3}$, there exists a choice of linear, 1-Lipschitz functions over $[-1,1] \subset \mathbb{R}$, such that the expected regret of A after T rounds (with respect to the algorithm's randomness), is*

$$\mathbb{E}\left[\sum_{t=1}^{T} f_t(w_t) - \sum_{t=1}^{T} f_t(w^*)\right] = \Omega\left(\sqrt{\tau T}\right)$$

$$\textit{where } w^* = \underset{w \in \mathcal{W}}{\textit{argmin}} \sum_{t=1}^{T} f_t(w)$$

For completeness, we we also provide in appendix A.2 a proof that when $M = 0$ (i.e. no permutations allowed), then the worst-case regret is no better than $\Omega(\sqrt{\tau T})$. This is of course a special case of Theorem 2, but applies to the standard adversarial online setting (without any local permutations), and the proof is simpler. The proof sketch for the setting where no permutation is allowed was already provided in (Langford et al., 2009), and our contribution is in providing a full formal proof.

The proof in the case where $M = 0$ is based on linear losses of the form $f_t = \alpha_t \cdot w_t$ over $[-1,+1]$, where $\alpha_t \in \{-1,+1\}$. Without permutations, it is possible to prove a $\Omega(\sqrt{\tau T})$ lower bound by dividing the T iterations into blocks of size τ, where the α values of all losses at each block is the same and randomly chosen to equal either $+1$ or -1. Since the learner does not obtain any information about this value until the block is over, this reduces to adversarial online learning over T/τ rounds, where the regret at each round scales linearly with τ, and overall regret at least $\Omega(\tau\sqrt{T/\tau}) = \Omega(\sqrt{\tau T})$.

In the proof of theorem 2, we show that by using a similar construction, even with permutations, having a permutation window less than $\tau/3$ still means that the α values would still be unknown until all loss functions of the block are processed, leading to the same lower bound up to constants.

The formal proof appears in the appendix, but can be sketched as follows: first, we divide the T iterations into blocks of size $\tau/3$. Loss functions within each block are identical, of the form $f_t = \alpha_t \cdot w_t$, and the value of α per block is chosen uniformly at random from $\{-1, +1\}$, as before. Since here, the permutation window M is smaller than $\tau/3$, then even after permutation, the time difference between the first and last time we encounter an α that originated from a single block is less than τ. This means that by the time we get any information on the α in a given block, the algorithm already had to process all the losses in the block, which leads to the same difficulty as the no-permutation setting. Specifically, since the predictors chosen by the algorithm when handling the losses of the block do not depend on the α value in that block, and that α is chosen randomly, we get that the expected loss of the algorithm at any time point t equals 0. Thus, the cumulative loss across the entire loss sequence is also 0. In contrast, for w^*, the optimal predictor in hindsight over the entire sequence, we can prove an expected accumulated loss of $-\Omega(\sqrt{\tau T})$ after T iterations, using Khintchine inequality and the fact that the α's were randomly chosen per block. This leads us to a lower bound of expected regret of order $\sqrt{\tau T}$, for any algorithm with a local permutation window of size $M < \tau/3$.

5. Experiments

We consider the adversarial setting described in section 4, where an adversary chooses a sequence of functions such that every τ functions are identical, creating blocks of size τ of identical loss functions, of the form $f_t(w_t) = \alpha_t \cdot w_t$ where α_t is chosen randomly in $\{-1, +1\}$ for each block. In all experiments we use $T = 10^5$ rounds, a delay parameter of $\tau = 200$, set our step sizes according to the theoretical analysis, and report the mean regret value over 1000 repetitions of the experiments.

In our first experiment, we considered the behavior of our Delayed Permuted Mirror Descent algorithm, for window sizes $M > \tau$, ranging from $\tau + 1$ to T. In this experiment, we chose the α values randomly, while ensuring a gap of 200 between the number of blocks with $+1$ values and the number of blocks with -1 values (this ensures that the optimal w^* is a sufficiently strong competitor, since otherwise the setting is too "easy" and the algorithm can attain negative regret in some situations). The results are shown in Figures 2 and 3, where the first figure presents the accumulated regret of our algorithm over time, whereas the second figure presents the overall regret after T rounds, as a function of the window size M.

When applying our algorithm in this setting with different values of $M > \tau$, ranging from $M = \tau + 1$ and up to $M = T$, we get a regret that scales from the order of the adversarial bound to the order of the stochastic bound depending on the window size, as expected by our analysis. For all window sizes greater than $5 \cdot \tau$, we get a regret that is in the order of the stochastic bound - this is not surprising, since after the permutation we get a sequence of functions that is very close to an i.i.d. sequence, in which case any algorithm can be shown to achieve $O(\sqrt{T})$ regret in expectation. Note that this performance is better than that predicted by our theoretical analysis, which implies an $O(\sqrt{T})$ behavior only when $M \geq \Omega(\tau^2)$. It is an open and interesting question whether it means that our analysis can be improved, or whether there is a harder construction leading to a tighter lower bound.

In our second experiment, we demonstrate the brittleness of the lower bound construction for standard online learning with delayed feedback, focusing on the $M < \tau$ regime. Specifically, we create loss functions with blocks as before (where following the lower bound construction, the α values in each block of size $\tau = 200$ is chosen uniformly at random). Then, we perform a random permutation over consecutive windows of size M (ranging from $M = 0$ up to $M = \frac{9}{10}\tau$ in intervals of $\frac{1}{10}\tau$). Finally, we run standard Online Gradient Descent with delayed gradients (and fixed step size $1/\sqrt{T}$), on the permuted losses. The results are presented in Figure 4.

For window sizes $M < \frac{\tau}{2}$ we see that the regret is close to the adversarial bound, whereas as we increase the window size the regret decreases towards the stochastic bound. This experiment evidently shows that this hardness construction is indeed brittle, and easily breaks in the face of local permutations, even for window sizes $M < \tau$.

6. Discussion

We presented the OLLP setting, where a learner can locally permute the sequence of examples from which she learns. This setting can potentially allow for improved learning in many problems, where the worst-case regret is based on highly adversarial yet brittle constructions. In this paper, we focused on the problem of learning from delayed feedback in the OLLP setting, and showed how it is possible to improve the regret by allowing local permutations. Also,

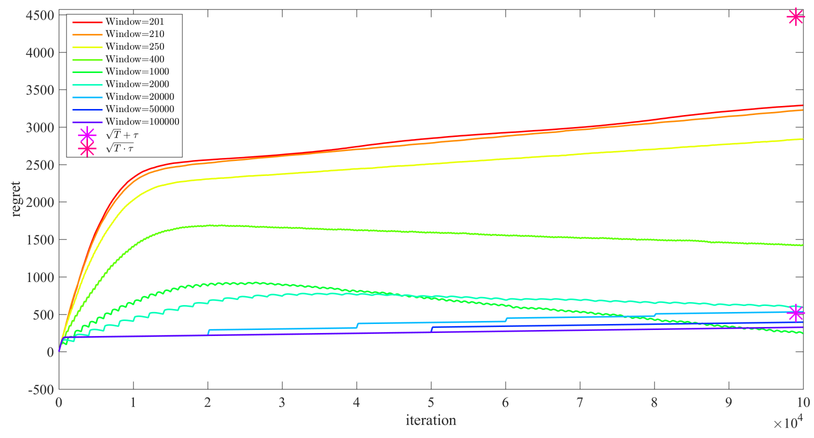

Figure 2. Regret of the Delayed Permuted Mirror Descent algorithm, with local permutation in window sizes ranging from $M = \tau + 1$ to $M = T$. A pink $*$ indicates the order of the stochastic bound ($\sqrt{T} + \tau$), and a red $*$ indicates the order of the adversarial bound ($\sqrt{\tau T}$). Regret is averaged over 1000 repetitions. Best viewed in color.

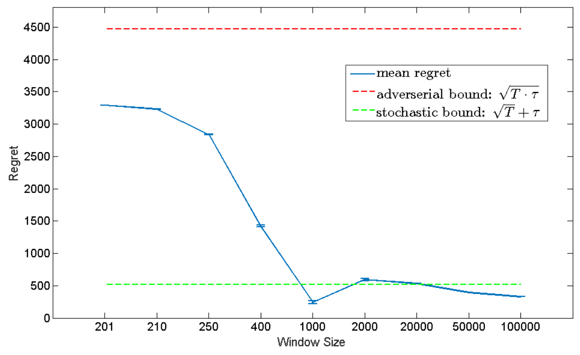

Figure 3. Regret of the Delayed Permuted Mirror Descent algorithm for different window sizes, after $T = 10^5$ iterations, with local permutation window sizes ranging from $M = \tau + 1$ to $M = T$. Red dashed line (top) indicates the order of the adversarial bound ($\sqrt{\tau T}$) and green dashed line (bottom) indicates the order of the stochastic bound ($\sqrt{T} + \tau$). Regret is averaged over 1000 repetitions, error bars indicate standard error of the mean. Best viewed in color.

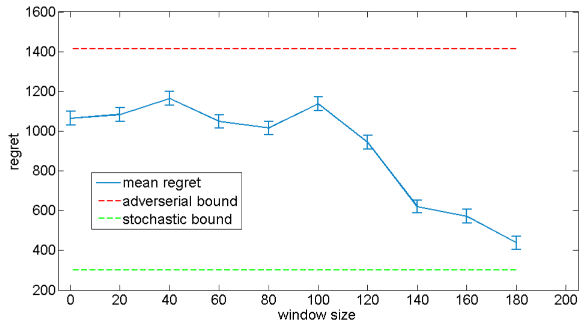

Figure 4. Regret of the standard Online Gradient Descent algorithm, in a adversarialy designed setting as described in 4, and with local permutation in window sizes ranging from $M = 0$ to $M = \frac{9}{10}\tau$. Red dashed line (top) indicates the order of the adversarial bound ($\sqrt{\tau T}$) and green dashed line (bottom) indicates the order of the stochastic bound ($\sqrt{T} + \tau$). Regret is averaged over 1000 repetitions, error bars indicate standard error of the mean. Best viewed in color.

we proved a lower bound in the situation where the permutation window is significantly smaller than the feedback delay, and showed that in this case, permutations cannot allow for a better regret bound than the standard adversarial setting. We also provided some experiments, demonstrating the power of the setting as well as the feasibility of the proposed algorithm. An interesting open question is what minimal permutation size allows non-trivial regret improvement, and whether our upper bound in Theorem 1 is tight. As suggested by our empirical experiments, it is possible that even small local permutations are enough to break highly adversarial sequences and improve performance in otherwise worst-case scenarios. Another interesting direction is to extend our results to a partial feedback (i.e. bandit) setting. Finally, it would be interesting to study other cases where local permutations allow us to interpolate between fully adversarial and more benign online learning scenarios.

Acknowledgements

OS is supported in part by an FP7 Marie Curie CIG grant, the Intel ICRI-CI Institute, and Israel Science Foundation grant 425/13. LS is an ISEF fellow.

References

Alekh Agarwal and John C Duchi. Distributed delayed stochastic optimization. In *Advances in Neural Information Processing Systems*, pages 873–881, 2011.

Sébastien Bubeck and Aleksandrs Slivkins. The best of both worlds: Stochastic and adversarial bandits. In *COLT*, pages 42–1, 2012.

Chao-Kai Chiang, Tianbao Yang, Chia-Jung Lee, Mehrdad Mahdavi, Chi-Jen Lu, Rong Jin, and Shenghuo Zhu. Online optimization with gradual variations. In *COLT*, pages 6–1, 2012.

Elad Hazan and Satyen Kale. Extracting certainty from uncertainty: Regret bounded by variation in costs. *Machine learning*, 80(2-3):165–188, 2010.

Elad Hazan and Satyen Kale. Better algorithms for benign bandits. *Journal of Machine Learning Research*, 12 (Apr):1287–1311, 2011.

Elad Hazan et al. Introduction to online convex optimization. *Foundations and Trends® in Optimization*, 2(3-4): 157–325, 2016.

Pooria Joulani, András György, and Csaba Szepesvári. Online learning under delayed feedback. In *ICML (3)*, pages 1453–1461, 2013.

Zohar S Karnin and Oren Anava. Multi-armed bandits: Competing with optimal sequences. In *NIPS*, 2016.

John Langford, Alexander Smola, and Martin Zinkevich. Slow learners are fast. *arXiv preprint arXiv:0911.0491*, 2009.

Ishai Menache, Ohad Shamir, and Navendu Jain. On-demand, spot, or both: Dynamic resource allocation for executing batch jobs in the cloud. In *11th International Conference on Autonomic Computing (ICAC 14)*, pages 177–187, 2014.

Chris Mesterharm. On-line learning with delayed label feedback. In *International Conference on Algorithmic Learning Theory*, pages 399–413. Springer, 2005.

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

Alexander Rakhlin and Karthik Sridharan. Online learning with predictable sequences. In *COLT*, pages 993–1019, 2013.

Amir Sani, Gergely Neu, and Alessandro Lazaric. Exploiting easy data in online optimization. In *Advances in Neural Information Processing Systems*, pages 810–818, 2014.

Yevgeny Seldin and Aleksandrs Slivkins. One practical algorithm for both stochastic and adversarial bandits. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1287–1295, 2014.

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

Daniel A Spielman and Shang-Hua Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *Journal of the ACM (JACM)*, 51 (3):385–463, 2004.

Jacob Steinhardt and Percy Liang. Adaptivity and optimism: An improved exponentiated gradient algorithm. In *ICML*, pages 1593–1601, 2014.

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