Consistent Online Optimization: Convex and Submodular

Mohammad Reza Karimi ETH Zürich

Andreas Krause ETH Zürich

Silvio Lattanzi Google Research

Sergei Vassilvitskii Google Research

Abstract

Modern online learning algorithms achieve low (sublinear) regret in a variety of diverse settings. These algorithms, however, update their solution at every time step. While these updates are computationally efficient, the very requirement of frequent updates makes the algorithms untenable in some practical applications. In this work, we develop online learning algorithms that update a sublinear number of times. We give a meta-algorithm based on non-homogeneous Poisson Processes that gives a smooth trade-off between regret and frequency of updates. Empirically, we show that in many cases, we can significantly reduce updates at a minimal increase in regret.

1 INTRODUCTION

Bandit, expert learning, and convex optimization algorithms have revolutionized online learning, and low regret algorithms are now known for a multitude of diverse settings. Whether the examples are drawn from a distribution, are chosen by an adversary, come annotated with additional context, or are selected from arbitrary convex domains, there are efficient algorithms that achieve sublinear, $o(T)$, regret after T timesteps.

One characteristic of these algorithms is that they rarely stick with playing the same action repeatedly, instead continually exploring new decisions. While this exploration is necessary to achieve low regret, the constant switching between actions can have adverse effects in practice both from a systems standpoint, and user interface design. On the systems side, a lot of switching can wreak havoc on caches, and incur additional latency in the processing of results. At the same time, most users prefer a sense of predictability, or *consistency* in their interactions with the system, and overly dynamic systems add to the overall cognitive load.

In this work, we investigate the trade-off between the consistency cost (defined as the number of times the algorithm changes its action), and the regret achieved by the algorithm. We show that a simple modification of classic online gradient descent approaches leads to a smooth trade-off between the two goals. At a high level, we achieve this by probabilistically deciding whether to keep playing the same action, or perform an update step. Importantly, we can do this with *constant* additional overhead—the additional computation needed to decide the probability of whether to update takes only constant time.

Finally we note that our algorithm can be easily extended to more complex settings. For instance, in Section 4 we show how to extended our technique to solve the consistent online submodular maximization problem.

1.1 Related Work

Online Convex Optimization is a very active research topic in online learning with many beautiful results (Merhav et al., 2002). Perhaps the closest to our setting is the work on learning with memory (Merhav et al., 2002). In this problem, the adversary is oblivious to the algorithm's choices but the loss that the algorithm incurs depends on the current and the recent choices of the algorithm. Interestingly, although the setting is different from ours, the algorithms introduced to solve this problem have non-trivial consistency guarantees. Often, the algorithms are based on a blocking technique (Merhav et al., 2002) that divides the rounds in blocks and allows only a constant number of switches per block. In turn, this technique automatically guarantees also to have a limited consistency cost, but with higher regret[1]. This result was later improved by György and Neu (2014) using the Shrink-

Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS) 2019, Naha, Okinawa, Japan. PMLR: Volume 89. Copyright 2019 by the author(s).

[1]The regret of this technique is $O(T^{2/3})$ and the consistency cost $O(T^{1/3})$.

ing Dartboard framework introduced by Geulen et al. (2010), but unfortunately their technique is limited to the expert setting. Recently, Anava et al. (2015) proposed a new algorithm that obtains near optimal bounds for the Online Convex Optimization setting in the counterfactual feedback model where the algorithm knows the loss that it would have incurred had it played any sequence of m decisions in the previous rounds. Our results are incomparable with these because our algorithm does not assume to have access to counterfactual feedback. Furthermore, it is simpler and can be easily extended to handle more complex settings like the consistent online submodular maximization problem.

A related area to ours is metrical task systems (MTS). The player's goal is to minimize the movement (in a metric space) plus the costs she recieves, while holding a plausible *competitive ratio*, *i.e.*, the ratio of the cost of the algorithm relative to the cost of the optimal offline algorithm on a worst-case sequence. Important problems in this field include the k-server problem (Manasse et al., 1990; Bubeck et al., 2017) and convex chasing problem (Argue et al., 2019).

Another area of work that is closely related to ours is research in online algorithms with recourse. In this setting, one seeks better online algorithms to compute optimal or approximate solutions for combinatorial problems by allowing the algorithm to make a limited number of changes to the online solution. This concept is very close to the notion of consistency that we consider in this paper. The first problem that received a lot of attention in this area is the classic online Steiner tree problem introduced by Imase and Waxman (1991) for which it is possible to design better algorithms by allowing a small recourse as shown in several papers (Gu et al., 2016; Gupta and Kumar, 2014; Lacki et al., 2015; Megow et al., 2016). The concept of recourse has been applied to other classic optimization problems as online scheduling (Andrews et al., 1999; Epstein and Levin, 2014; Phillips and Westbrook, 1998; Sanders et al., 2009; Skutella and Verschae, 2010), online flow (Gupta et al., 2014; Westbrook, 2000), online matching Bernstein et al. (2018) and online set cover (Gupta et al., 2017). Very recently, Lattanzi and Vassilvitskii (2017) introduced the notion of consistency in online algorithms for machine learning and studied the consistent online clustering problem.

2 PRELIMINARIES

We consider the online learning framework, which can be seen as a game between a player and an adversary. The game proceeds in rounds. In round $t \in \{1, 2, \ldots, T\}$, the player selects an action x_t from some set $\mathcal{X}$, and the adversary reveals the loss function f_t. The goal of the player is to adaptively select $x_1, x_2, \ldots, x_T$ to compete with the best single action in hindsight, $x^\star$. Formally, the goal of the player is to minimize her regret:

$$\mathcal{R}_T := \sum_{t=1}^{T} f_t(x_t) - \min_{x^\star \in \mathcal{X}} \sum_{t=1}^{T} f_t(x^\star).$$

Our goal will be to simultaneously achieve sublinear regret, and minimize the number of times the player changes her decision, *i.e.*, the *consistency* cost:

$$\kappa_T := \sum_{t=2}^{T} \mathbf{1}_{x_t \neq x_{t-1}}.$$

Convex Setting We begin with a restriction to the setting where $\mathcal{X}$ is a compact convex subset of $\mathbb{R}^d$. Let $\|\cdot\|$ be a norm on $\mathbb{R}^d$ and denote by $C^1_L(\mathcal{X})$ the set of all real-valued continuously differentiable functions over $\mathcal{X}$ that are Lipschitz continuous with constant L, *i.e.*, $\|\nabla f(x)\|_\star \le L$ for all $x \in \mathcal{X}$ and all $f \in C^1_L(\mathcal{X})$.

Let $\text{Proj}_{\mathcal{X}}(z)$ be the unique orthogonal projection of z onto the convex set $\mathcal{X}$. If the gradient of f_t is revealed to the player at the end of round t, then sublinear regret can be achieved by a simple gradient descent algorithm, OGD, which repeatedly takes a gradient step and projects it to the feasible set of actions.

Algorithm 1 Online Gradient Descent (OGD)

```
Select $x_1 \in \mathcal{X}$ arbitrarily
for $t \in 1, 2, \ldots, T$ do
    Play $x_t$ and receive loss $f_t(x_t)$
    Set $x_{t+1} := \text{Proj}_{\mathcal{X}}(x_t - \eta_t \nabla f_t(x_t))$.
end for
```

Theorem 1 (Zinkevich (2003))**.** *Suppose $f_t \in C^1_L(\mathcal{X})$ for all $t \in [T]$, and $\mathcal{X}$ has diameter D (w.r.t. Euclidean norm). Then Online Gradient Descent with step size $\eta_t = \frac{D}{L\sqrt{t}}$ guarantees*

$$\mathcal{R}_T \le \frac{3}{2} LD\sqrt{T} = O(\sqrt{T}). \tag{1}$$

Note that while the OGD algorithm is guaranteed to achieve low regret, it is easy to set up scenarios where it will pick a new point x_t at every time step, suffering $\kappa_T = \Omega(T)$.

Decision Path In our study, the notion of a *decision path* turns out to be useful. For a sequence of points $(x_t)_{t\in[T]} \subset \mathbb{R}^d$ we define the decision path to be the

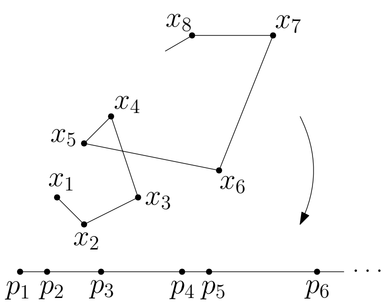

Figure 1: The Decision Path

sequence of non-negative real numbers $(p_t)_{t\in[T]}$ such that $p_1 = 0$ and for all $t > 1$,

$$p_t - p_{t-1} = \|x_t - x_{t-1}\|.$$

Intuitively, if we had to move a pebble from x_{t-1} to x_t as the algorithm went on, p_t would represent the total distance traveled by the pebble after t steps, see Figure 1 for an illustration.

For analytic reasons, we also introduce the *δ-padded decision path* as follows.

Definition 2. Let $\delta = (\delta_t)_{t\in\mathbb{N}}$ be an increasing sequence of positive numbers and $p_1, \ldots, p_T$ be a decision path. Then the corresponding δ-padded decision path is the sequence $r_1, \ldots, r_T$ with $r_1 = p_1$ and $r_t = p_t + \delta_t$ for $t > 1$.

When using the OGD algorithm, the following Lemma on the length of the decision paths will be useful.

Lemma 3. *Let $f_t \in C^1_L(\mathcal{X})$ be convex for all $t \in [T]$ and (x_t) be the sequence of decisions made by OGD. Assume that (p_t) is the decision path for (x_t).*

(i) One has for all $t \in [T]$,

$$p_t \le 2D\sqrt{t}.$$

(ii) If f_t is α-strongly convex for all $t \in [T]$, then using step size $\eta_t = \frac{1}{\alpha t}$,

$$p_t \le \tfrac{L}{\alpha}(1 + \log t).$$

We give the proof in the appendix.

Poisson Point Process Recall that a Poisson process on the real line is a stochastic process for modeling random events, which defines a probability distribution on the number of points (events) within any interval $(a, b]$ with the following key properties:

- The number of points in any interval has a Poisson distribution.

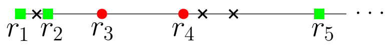

Figure 2: Let (r_t) be the padded decision path. The crosses are the Poisson process events. Red points denote rounds we do not update and green squares are updating rounds. We always update in the first round, and for round t, we update only if there is a Poisson event on the line segment ending in r_t.

- For any two disjoint intervals, the number of points in them are independent random variables.

In our work, we will consider non-homogeneous Poisson processes on the real line, with varying continuous *intensity*, $\lambda(\tau)$. In this case, the number of points on an interval $(a, b]$ is distributed as a Poisson random variable with mean $\int_a^b \lambda(\tau)\,d\tau$, *i.e.*, the intensity varies with the position of the interval. To simplify notation, we will denote by $M(\tau) = \int_0^\tau \lambda(u)\,du$.

Note on notation. We will use the variable t to index the rounds of the game, *i.e.* $t \in \mathbb{N}$, and use τ for the continuous variable indicating position on the decision path, *i.e.* $\tau \in \mathbb{R}_+$.

3 THE SOLO ALGORITHM

In order to achieve a trade-off between regret and consistency, our algorithm will make use of a basic algorithm $\mathcal{A}$, such as OGD, and decide probabilistically at each time step whether to update the current solution point, or to stick to the one from the previous round. Intuitively, we want the updates to be more likely to happen if the current solution point is far away from the proposed point, but also less likely to happen as the algorithm proceeds and starts converging to a near-optimal solution.

We will update the algorithm's solution whenever a specific point process on the padded decision path has a hit. Let $\lambda : \mathbb{R}_+ \to \mathbb{R}_+$ be an increasing intensity function we will define later. We will update the strategy at time t if and only if the non-homogeneous Poisson process with rate $\lambda(\tau)$ had a non-zero count in the interval $(r_{t-1}, r_t]$. This is equivalent to updating with probability $1 - \exp(-\int_{r_{t-1}}^{r_t} \lambda(\tau)\,d\tau) = 1 - \exp(-(M(r_t) - M(r_{t-1})))$. Figure 2 shows an example of the Poisson process and updating procedure.

Observe that this definition satisfies our desiderata from above. Suppose that the algorithm last updated the solution at time $t' < t$. As the solution drifts, *i.e.*, the distance between $x_{t'}$ and x_t increases, $r_{t'}$ and r_t get further away, and the probability of an update increases. On the other hand, as the gradient steps get smaller (and thus the distance between two con-

Algorithm 2 Sticky OnLine Optimization (SOLO)

```
$\mathcal{A}$ is a no-regret algorithm
$\lambda(\tau)$ an increasing intensity function.
$(\delta_t)$ an increasing padding sequence.
$x_1$ is the first decision proposed by $\mathcal{A}$
$r_1 \leftarrow 0$
$y_1 \leftarrow x_1$
for $t \in [T]$ do
    Play $y_t$ and receive loss $f_t(y_t)$.
    Give $\mathcal{A}$ access to the oracle of $f_t$ and set $x_{t+1}$ to be the next proposal of $\mathcal{A}$.
    $r_{t+1} \leftarrow r_t + \|x_{t+1} - x_t\| + \delta_{t+1} - \delta_t$
    With probability $1 - \exp\{-(\int_{r_t}^{r_{t+1}} \lambda(u)\,du)\}$
        Update: set $y_{t+1} \leftarrow x_{t+1}$.
    Otherwise set $y_{t+1} \leftarrow y_t$.
end for
```

secutive x's decreases), so does the probability of an update.

We formulate the above in Algorithm 2, called Sticky Online Optimization (SOLO).

Analysis The analysis of Algorithm 2 proceeds in three parts. First we relate the regret of SOLO to that of the given algorithm $\mathcal{A}$ via a triangle inequality argument, and identify the additional regret paid by Algorithm 2 (Proposition 4). Next, we prove a technical fact about the distribution of hits in non-homogeneous Poisson processes with sufficiently high intensities (Lemma 5). Finally, we specialize the choice of λ and δ to prove a consistency-regret trade-off for OGD for convex (Theorem 8) and strongly-convex functions (Theorem 9).

3.1 Regret

We analyze the regret of our algorithm as compared to $\mathcal{A}$. Assume that $f_t \in C^1_L(\mathcal{X})$ for all $t \in [T]$, and let (x_t) be the decisions proposed by $\mathcal{A}$, and (y_t) be decisions we actually played. Also denote by (r_t) the padded decision path for (x_t).

The regret of SOLO is:

$$\begin{aligned}\mathcal{R}_T &= \sum_{t=1}^{T} f_t(y_t) - \min_{x^\star\in\mathcal{X}} \sum_{t=1}^{T} f_t(x^\star) \qquad (2)\\ &= \sum_{t=1}^{T} f_t(y_t) - \sum_{t=1}^{T} f_t(x_t)\\ &\quad + \sum_{t=1}^{T} f_t(x_t) - \min_{x^\star\in\mathcal{X}} \sum_{t=1}^{T} f_t(x^\star)\\ &\le L\sum_{t=1}^{T} \|y_t - x_t\| + \mathcal{R}_T(\mathcal{A}),\end{aligned}$$

where the last line is due to the Lipschitz property of f_t and $\mathcal{R}_T(\mathcal{A})$ is the regret of $\mathcal{A}$. Letting $p(t) := \max\{m \le t : m \text{ is an updating round}\}$ to be the latest updating round before round t, it is clear that $y_t = x_{p(t)}$. Using the triangle inequality and remembering that δ is increasing:

$$\begin{aligned}\|y_t - x_t\| &= \|x_t - x_{p(t)}\|\\ &\le \sum_{j=p(t)}^{t-1} \|x_{j+1} - x_j\|\\ &= \sum_{j=p(t)}^{t-1} (r_{j+1} - r_j - [\delta(j+1) - \delta(j)])\\ &\le r_t - r_{p(t)}.\end{aligned}$$

This gives rise to the definition of *extra regret*: we define the extra regret to be

$$\mathcal{E}_T = L \cdot \sum_{t=1}^{T} (r_t - r_{p(t)}).$$

The following proposition is now immediate:

Proposition 4.

$$\mathcal{R}_T(\text{SOLO}) \le \mathcal{E}_T + \mathcal{R}_T(\mathcal{A}).$$

Observe that the extra regret depends directly on the length of the padded decision path. Thus by changing the intensity of the sampling process, we can get a trade-off between the number of changes, and the extra regret.

3.2 Gap Distribution

As we saw above, the additional regret of the algorithm depends on the length of the padded decision path between updates. In this section, we present a technical lemma bounding this quantity for non-homogeneous Poisson processes with sufficiently fast increasing intensities.

Lemma 5. *Assume $M(\tau) = \omega(\log \tau)$ and $\lambda(\tau) = \omega(1)$ to be increasing. Also assume that $(1/\lambda(\tau))' = o(1)$. Then, for all $t \in [T]$ one has $\mathbb{E}[r_t - r_{p(t)}] \le C/\lambda(r_t)$, where C is a constant independent of t and r_t.*

At a high level, Lemma 5 implies that we only need to look at the intensity at the last point of the interval (where it is the highest) to bound the expected length of the gap.

First we bring two supplementary lemmas which are crucial in the proof.

Lemma 6. *Assuming conditions of Lemma 5, one has*

$$\lim_{\tau\to\infty} \frac{M(\tau)}{\log \int_0^\tau e^{M(u)}\,du} = 1.$$

Proof. Using integration by parts and knowing that $\frac{d}{du}M(u) = \lambda(u)$ we see that

$$\int_0^\tau e^{M(u)}\,du = \tau e^{M(\tau)} - \int_0^\tau u\lambda(u)e^{M(u)}\,du.$$

Now, since the last term is positive, we get

$$\log \int_0^\tau e^{M(u)}\,du < \log(\tau e^{M(\tau)}) = M(\tau) + \log \tau. \quad (3)$$

Also, since $\lambda(\tau)$ is increasing, we can write

$$\int_0^\tau u\lambda(u)e^{M(u)}\,du \le \tau\lambda(\tau)\int_0^\tau e^{M(u)}\,du,$$

and using the first equality, we arrive at

$$(1+\tau\lambda(\tau))\int_0^\tau e^{M(u)}\,du \ge \tau e^{M(\tau)},$$

which gives

$$\log \int_0^\tau e^{M(u)}\,du \ge M(\tau) - \log\frac{1+\tau\lambda(\tau)}{\tau}. \quad (4)$$

Dividing both sides of (3) by $M(\tau)$ and taking the limsup gives

$$\limsup_{\tau\to\infty}\frac{\log\int_0^\tau e^{M(u)}\,du}{M(\tau)} \le 1 + \limsup_{\tau\to\infty}\frac{\log\tau}{M(\tau)} = 1,$$

and doing the same for (4), and taking the liminf gives

$$\liminf_{\tau\to\infty}\frac{\log\int_0^\tau e^{M(u)}\,du}{M(\tau)} \ge 1 - \limsup_{\tau\to\infty}\frac{\log\frac{1+\tau\lambda(\tau)}{\tau}}{M(\tau)}$$
$$= 1 - \limsup_{\tau\to\infty}\frac{\log\lambda(\tau)}{M(\tau)} = 1,$$

since by L'Hôpital's rule the last limit is equal to $1 - \lim_{\tau\to\infty}\frac{\lambda'(\tau)}{\lambda(\tau)^2} = 1$. Combining these two inequalities, we can observe that the following limit exists and the equality holds:

$$\lim_{\tau\to\infty}\frac{\log\int_0^\tau e^{M(u)}\,du}{M(\tau)} = 1.$$

□

Lemma 7. *With the same conditions as Lemma 5, we have*

$$e^{-M(\tau)}\int_0^\tau e^{M(u)}\,du = \Theta(1/\lambda(\tau)) \quad \textit{(as } \tau\to\infty\textit{)}$$

Also it also holds that

$$\lim_{\tau\to 0} e^{-M(\tau)}\int_0^\tau e^{M(u)}\,du = 0.$$

Proof. Let us introduce for $\tau > 0$

$$f(\tau) := \log\int_0^\tau e^{M(u)}\,du.$$

Note that the equation in the lemma is equal to $1/f'(\tau)$. Clearly $\lim_{\tau\to\infty} f(\tau) = +\infty$, as well as $\lim_{\tau\to\infty} M(\tau) = +\infty$. So by L'Hôpital's rule and Lemma 6 above, one gets

$$\lim_{\tau\to\infty}\frac{M(\tau)}{f(\tau)} = \lim_{\tau\to\infty}\frac{\lambda(\tau)}{f'(\tau)} = 1,$$

which yields

$$\lim_{\tau\to\infty}\frac{e^{-M(\tau)}\int_0^\tau e^{M(u)}\,du}{1/\lambda(\tau)} = 1,$$

which is exactly what we wanted.

For the second argument, observe that as $\tau\to 0$, $e^{-M(\tau)}\to 1$ and $\int_0^\tau e^{M(u)}\,du \to 0$, which completes the proof of this lemma. □

Proof of Lemma 5. We begin by deriving the probability distribution of $p(t)$. For all $1 < t \le T$ and $m \le t$

$$\begin{aligned}
&\mathbb{P}(p(t) = m)\\
&= \mathbb{P}(\text{update at } m)\cdot\mathbb{P}(\text{no updates from } m \text{ to } t)\\
&= (1 - e^{-(M(r_m)-M(r_{m-1}))})\cdot e^{-(M(r_t)-M(r_m))}\\
&= e^{-(M(r_t)-M(r_m))} - e^{-(M(r_t)-M(r_{m-1}))}
\end{aligned}$$

Then

$$\begin{aligned}
&\mathbb{E}[r_t - r_{p(t)}]\\
&= \sum_{m=1}^{t-1}(r_t - r_m)(e^{-(M(r_t)-M(r_m))} - e^{-(M(r_t)-M(r_{m-1}))})\\
&= \sum_{m=1}^{t-1}(r_{m+1} - r_m)e^{-(M(r_t)-M(r_m))}\\
&= e^{-M(r_t)}\sum_{m=1}^{t-1}(r_{m+1}-r_m)e^{M(r_m)}\\
&\le e^{-M(r_t)}\int_0^{r_t} e^{M(u)}\,du.
\end{aligned}$$

Now by Lemma 7, one gets the desired result. □

3.3 Regret vs. Consistency Trade-offs

We are now ready to specialize the general algorithm above to obtain our main result, namely a regret-consistency trade-off for OGD for convex and strongly convex functions.

By linearity of expectation, and using Lemma 5 we have:

$$\mathbb{E}[\mathcal{E}_T] = L\cdot\sum_{t=1}^T \mathbb{E}[r_t - r_{p(t)}] \le C\sum_{t=1}^T\frac{1}{\lambda(r_t)}. \quad (5)$$

It remains to specialize δ and λ to achieve our desired bounds.

Theorem 8 (Consistent OGD for Convex functions)**.** *Let $f_t \in C^1_L(\mathcal{X})$ be convex for all $t \in [T]$. Let $\delta_t = \sqrt{t}$, and denote by (r_t) the padded decision path. For a fixed $\varepsilon \in (0,1]$ set*

$$\lambda(\tau) := (1+\varepsilon)\,\tau^{\varepsilon}, \quad M(\tau) = \tau^{1+\varepsilon}.$$

Then:

$$\mathbb{E}[\kappa_T] = O(T^{\frac{1}{2}+\frac{\varepsilon}{2}}), \quad \mathbb{E}[\mathcal{R}_T] = O(T^{1-\frac{\varepsilon}{2}}).$$

Proof. It is easy to check that $\lambda(\tau)$ and $M(\tau)$ satisfy the conditions of Lemma 5. Therefore:

$$\mathbb{E}[r_t - r_{p(t)}] \leq \frac{C}{\lambda(r_t)} \leq \frac{C}{\lambda(\sqrt{t})} = \frac{C}{(1+\varepsilon)} t^{-\frac{\varepsilon}{2}} = C' t^{-\frac{\varepsilon}{2}}.$$

Then,

$$\mathbb{E}[\mathcal{E}_T] \leq C' \sum_{t=1}^{T} t^{-\frac{\varepsilon}{2}} \leq C'' T^{1-\frac{\varepsilon}{2}} = O(T^{1-\frac{\varepsilon}{2}}).$$

By Theorem 1, we know that $\mathcal{R}_T(\mathcal{A}) = O(\sqrt{T})$. Hence

$$\mathbb{E}[\mathcal{R}_T] = O(T^{1-\frac{\varepsilon}{2}}).$$

Finally, to bound κ_T, recall that the total path length from Lemma 3 (i), and the final padding of $\delta(T) = \sqrt{T}$. Since $M(\tau)$ is increasing, we get

$$\begin{aligned}\mathbb{E}[\kappa_T] &\leq M(r_T) \leq M((2D+1)\sqrt{T}) \\ &= (2D+1)^{1+\varepsilon} T^{\frac{1}{2}+\frac{\varepsilon}{2}} = O(T^{\frac{1}{2}+\frac{\varepsilon}{2}}),\end{aligned} \tag{6}$$

where the first inequality is due to the fact that the number of poisson points is always greater than the number of rounds that we update. □

We can use a similar analysis in the strongly convex case.

Theorem 9 (Consistent OGD for Strongly Convex functions)**.** *Let $f_t \in C^1_L(\mathcal{X})$ be α-strongly convex, for all $t \in [T]$. Let $\delta_t = \log t$, and denote by (r_t) the padded decision path. Finally, let $\gamma := 1 + \frac{L}{\alpha}$. For a fixed $\varepsilon \in (0,1)$ set:*

$$\lambda(\tau) := \exp\left(\varepsilon \frac{\tau}{\gamma}\right), \quad M(\tau) = \frac{\gamma}{\varepsilon}(\exp\left(\varepsilon\frac{\tau}{\gamma}\right) - 1).$$

Then one has

$$\mathbb{E}[\kappa_T] = O(T^{\varepsilon}), \quad \mathbb{E}[\mathcal{R}_T] = O(T^{1-\varepsilon\frac{1}{\gamma}}).$$

Proof. By Lemma 3 and the choice of δ., since $M(\tau)$ is increasing:

$$\mathbb{E}[\kappa_T] \leq M(r_T) \leq M(\gamma(1+\log T)) = O(T^{\varepsilon}). \tag{7}$$

Also, by Lemma 5 and integral approximation we get

$$\mathbb{E}[\mathcal{E}_T] \leq C + C\int_1^T \frac{1}{\lambda(\log u)}\, du = O(T^{1-\varepsilon\frac{1}{\gamma}}). \tag{8}$$

□

The reader should be convinced by now that for any online algorithm that has well-behaved asymptotic decision path length, one can use the SOLO algorithm using the right intensity function λ, and get a consistency/regret trade-off. We show this for the case of Online Submodular Maximization in the next section.

4 EXTENSIONS TO CONSISTENT SUBMODULAR MAXIMIZATION

In the previous section we described a meta-algorithm that provides a consistency vs. regret trade-off for online gradient descent. Here we show that a similar set of ideas apply to the problem of online submodular function optimization, a problem that appears naturally in many diverse areas. For instance many problems in clustering, result diversification, feature selection, and data summarization, can be phrased as optimizing a submodular function subject to a set of constraints. Many of these have natural online variants where the loss function is a *set function* defined on subsets of items, and can be reformulated as a submodular utility, see for instance the survey by Krause and Golovin (2014).

We follow a similar analysis as in Section 3. However, the notion of regret no longer suffices: in the online submodular maximization setting it is NP-hard to find an optimal solution in hindsight, thus no poly-time algorithm is going to achieve sublinear regret (unless P = NP). Instead, we consider approximate no-regret algorithms: For a maximization task, we call an algorithm $\mathcal{A}$ a *no β-regret algorithm* if for any sequence of functions $(f_t)_{t\in\mathbb{N}} \subset \mathcal{H}$ inside a function class, $f_t : \mathcal{X} \to \mathbb{R}$, $\mathcal{A}$ produces a sequence of points $(x_t)_{t\in\mathbb{N}} \subset \mathcal{X}$ such that for any $T \in \mathbb{N}$, the β-regret defined as

$$\mathcal{R}_T := \beta \cdot \max_{x^\star \in \mathcal{X}} \sum_{t=1}^{T} f_t(x^\star) - \sum_{t=1}^{T} f_t(x_t), \tag{9}$$

is sublinear. In this setting, since the task is maximization, we regard f_t as a *utility* rather than a loss.

Our goal is then to find trade-offs between approximate β-regret and consistency for different classes of submodular functions. We give examples for two special cases: monotone DR-submodular functions (Bian et al., 2017) and monotone submodular set functions (Nemhauser et al., 1978), see the references for the exact definitions.

For DR-submodular functions, Chen et al. (2018) give an Online Gradient Ascent algorithm that achieves sublinear $\frac{1}{2}$-regret. The algorithm produces points $x_1, \dots, x_T \in \mathcal{X}$ satisfying

$$\frac{1}{2} \max_{x^\star \in \mathcal{X}} \sum_{t=1}^{T} f_t(x^\star) - \sum_{t=1}^{T} f_t(x_t) \leq \frac{3}{4} DL\sqrt{T}, \qquad (10)$$

where D is the diameter of $\mathcal{X}$.

By selectively deciding when to update the solution, we transform it to an algorithm that suffers an extra regret of $O(T^{1-\frac{\varepsilon}{2}})$ in expectation, while updating at most $O(T^{\frac{\varepsilon+1}{2}})$ times.

We further extend the analysis to the discrete case, where the utility functions are submodular set functions and the optimization domain is a matroid $\mathcal{M}$. Using the Swap-Rounding technique by Chekuri et al. (2009) to round the solutions of online gradient ascent algorithm, we get an algorithm that achieves sublinear $1/2$-regret. We can then be selective on when to apply the updates to achieve additional additive regret of $T^{1-\frac{\varepsilon}{2}}$ using at most $O(T^{\frac{\varepsilon+1}{2}})$ updates. We defer the details to Appendix B.

5 EXPERIMENTS

We evaluate the performance of our algorithm for different parameters of ε, demonstrating different points on the consistency vs. regret trade-off curve. Overall, we find that while the base algorithm $\mathcal{A}$ updates the solution at every time step, overwhelmingly we suffer very little additional regret when we choose to update significantly fewer times.

We focus on two cases:

Online Convex Optimization We perform Online Logistic Regression on the MNIST dataset of handwritten digits (LeCun, 1998), to distinguish digit 3 from 5 over the unit ℓ_1 ball. In this setting, the expected regret is:

$$\mathcal{R}_T = \sum_{t=1}^{T} f_t(x_t) - T \cdot \min_{\|x\|_1 \leq 1} \mathbb{E}[f(x)],$$

where the expectation is with respect to the training set.

Online Submodular Maximization We investigate the problem introduced in the Battle of the Water Sensor Networks challenge (Avi Ostfeld et al., 2008), where the goal is to find the best set of sensor locations to detect a contamination event, minimizing detection time.

In our experiment, we select 20 sensors from among 12 527 possible locations, based on performance on a set of contamination events. As Leskovec et al. (2007) showed, this problem can be formulated as an instance of the facility location problem: For any subset of sensor locations S, every contamination event is a submodular function $f_e(S) := \max_{s \in S} w_{e,s}$, where $w_{e,s}$ is the reduction in penalty for sensor s in event e. Given that the events are sampled from some distribution $\mathcal{D}$, the optimization problem can be written as

$$\max_{S \subseteq V, |S| \leq 20} \mathbb{E}_{f \sim \mathcal{D}}[f(S)].$$

We perform Online Gradient Ascent on the multilinear extension of f_e's, which can be computed easily (*c.f.*, Appendix E.2 of Karimi et al. (2017)). The optimization domain is the matroid base polytope for the cardinality constraint.

In this setting we cannot compute the exact $1/2$-regret. Instead, we compute the extra regret, as well as cumulative utility. We also report results in the case that one rounds the continuous solutions to discrete sets in each round.

Metrics To evaluate the performance of the algorithms we plot the following quantities:

1. Consistency cost, κ_t, denoting the number of switches in the solution.
2. Extra regret, $\mathcal{E}_t$, denoting the loss in regret over using the best low-regret algorithm.
3. For the MNIST dataset, total regret $\mathcal{R}_t$.

Additionally, for consistency and extra regret, we evaluate the growth of the two quantities as a function of t. To do so, we plot the functions $\log \kappa_t / \log t$ and $\log \mathcal{E}_t / \log t$. Observe that if $x(t) = O(t^\alpha)$ then $\limsup_{t\to\infty} {}^{\log x(t)}/_{\log t} \leq \alpha$. This allows us to estimate how close our empirical results are to the theoretical bounds.

Results In the top row of Figure 3 we show the results for the MNIST dataset. In the first panel we plot the consistency cost, showing both the raw cost κ_t, as well as the scaled version $\log \kappa_t / \log t$. The results closely track those predicted by the theory, as we see κ grow as $T^{\frac{1}{2}+\frac{\varepsilon}{2}}$. In the second panel we plot the extra regret. Here we see that the theoretical results are overly pessimistic, and the actual extra regret that is achieved is far lower than that predicted by the theory. This leads us to very favorable trade-offs, for instance by taking $\varepsilon = 0.3$, the regret increase is less than 2%, but the number of updates drops by 30x over the baseline, as we update on approximately 3% of the

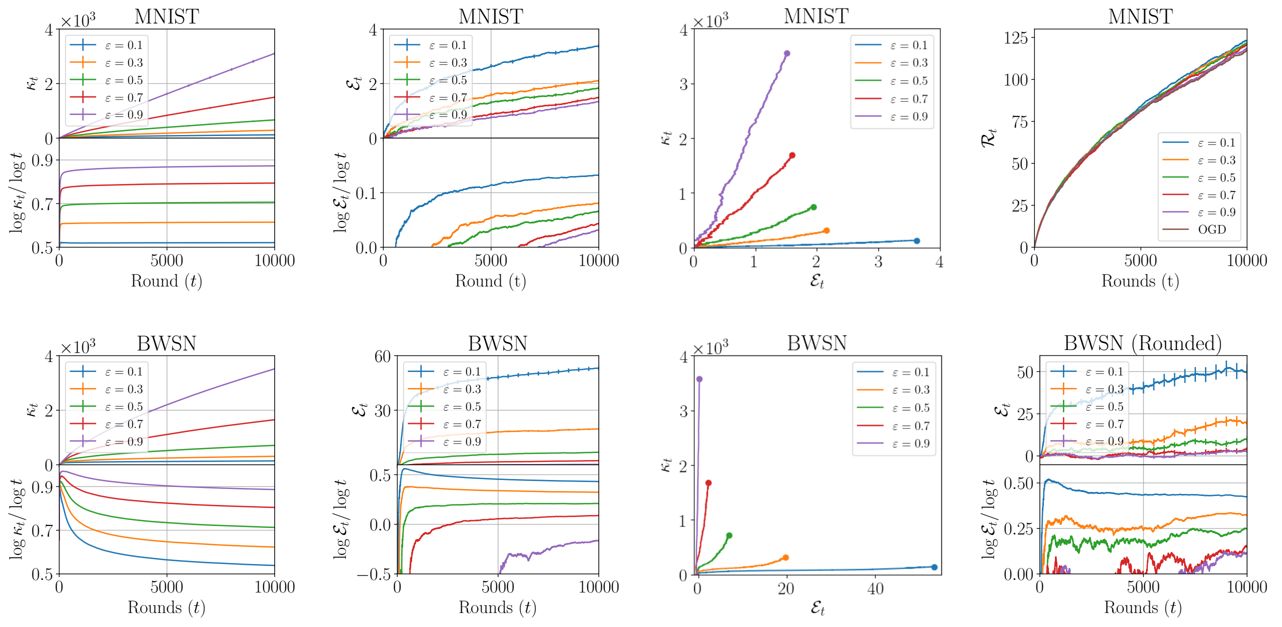

Figure 3: Top row are experimental results for Online Convex Optimization on MNIST dataset. The bottom row are experimental results for Online Submodular Maximization for the BWSN problem.

rounds. This is further demonstrated in the fourth panel, which shows the total regret of the different parameter settings. The fact that the lines are bunched close together again demonstrates that the additional regret incurred by not updating during the majority of the timesteps is minimal.

In the bottom row of Figure 3 we plot the results for the BWSN problem. Here again, in the first panel we investigate the consistency cost where we observe it be sublinear in the number of rounds. In the second panel we plot the extra regret, $\mathcal{E}_t$. Again, the extra regret is far lower than the pessimistic theoretical bound. Moreover, we see that sufficiently high, but bounded away from 1, values of ε, e.g., $\varepsilon = 0.9$, appear to lead to an *improvement* in results, suggesting that the lack of updating acts as a regularizer. (We note that the improvement is statistically significant.) This improvement is especially pronounced in the fourth plot, where we show the regret due to rounded solutions. Moreover, one can verify the analysis in Appendix B, that the regret bound for the multilinear extension upper bounds the regret bound for discrete submodular function. Concretely, by updating in 3% of the rounds, we only suffer 0.3% additional regret, when using $\varepsilon = 0.3$ and rounding the solution.

6 CONCLUSION

We presented the first truly online learning algorithms, capable of dealing with large action sets, that trade-off the frequency with which the solution is updated and the final regret achieved. Our methods are based on the realization of a specific non-homogeneous Poisson process, which allows us to balance the two opposing objectives. An immediate open question is whether the bounds derived in this work can be further improved. A broader direction is to better understand update/performance trade-offs in other online applications.

Acknowledgements

This research was supported in part by SNSF NFP 75 grant no. 407540-167212.

References

O. Anava, E. Hazan, and S. Mannor. Online learning for adversaries with memory: Price of past mistakes. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 784–792, 2015.

M. Andrews, M. X. Goemans, and L. Zhang. Improved bounds for on-line load balancing. *Algorithmica*, 23 (4):278–301, 1999. doi: 10.1007/PL00009263. URL `https://doi.org/10.1007/PL00009263`.

C. Argue, S. Bubeck, M. B. Cohen, A. Gupta, and Y. T. Lee. A nearly-linear bound for chasing nested convex bodies. In *Proceedings of the Thirtieth*

Annual ACM-SIAM Symposium on Discrete Algorithms, pages 117–122. SIAM, 2019.

Avi Ostfeld et al. The battle of water sensor networks (bwsn): A design challenge for engineers and algorithms. *Journal of Water Resources Planning and Management*, 134(6):556–568, 2008. doi: 10.1061/(ASCE)0733-9496(2008)134:6(556).

A. Bernstein, J. Holm, and E. Rotenberg. Online bipartite matching with amortized replacements. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 947–959, 2018. doi: 10.1137/1.9781611975031.61. URL `https://doi.org/10.1137/1.9781611975031.61`.

A. A. Bian, B. Mirzasoleiman, J. M. Buhmann, and A. Krause. Guaranteed non-convex optimization: Submodular maximization over continuous domains. *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2017.

S. Bubeck, M. B. Cohen, J. R. Lee, Y. T. Lee, and A. Madry. k-server via multiscale entropic regularization. *CoRR*, abs/1711.01085, 2017.

C. Chekuri, J. Vondrák, and R. Zenklusen. Dependent Randomized Rounding for Matroid Polytopes and Applications. *Computer*, 2009.

L. Chen, H. Hassani, and A. Karbasi. Online continuous submodular maximization. In *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84, pages 1896–1905. PMLR, 2018.

L. Epstein and A. Levin. Robust algorithms for preemptive scheduling. *Algorithmica*, 69(1):26–57, 2014. doi: 10.1007/s00453-012-9718-3. URL `https://doi.org/10.1007/s00453-012-9718-3`.

S. Geulen, B. Vöcking, and M. Winkler. Regret minimization for online buffering problems using the weighted majority algorithm. In *COLT 2010 - The 23rd Conference on Learning Theory, Haifa, Israel, June 27-29, 2010*, pages 132–143, 2010. URL `http://colt2010.haifa.il.ibm.com/papers/COLT2010proceedings.pdf\#page=140`.

A. Gu, A. Gupta, and A. Kumar. The power of deferral: Maintaining a constant-competitive steiner tree online. *SIAM J. Comput.*, 45(1):1–28, 2016. doi: 10.1137/140955276. URL `https://doi.org/10.1137/140955276`.

A. Gupta and A. Kumar. Online steiner tree with deletions. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 455–467, 2014. doi: 10.1137/1.9781611973402.34. URL `https://doi.org/10.1137/1.9781611973402.34`.

A. Gupta, A. Kumar, and C. Stein. Maintaining assignments online: Matching, scheduling, and flows. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 468–479, 2014. doi: 10.1137/1.9781611973402.35. URL `https://doi.org/10.1137/1.9781611973402.35`.

A. Gupta, R. Krishnaswamy, A. Kumar, and D. Panigrahi. Online and dynamic algorithms for set cover. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 537–550, 2017. doi: 10.1145/3055399.3055493. URL `http://doi.acm.org/10.1145/3055399.3055493`.

A. György and G. Neu. Near-optimal rates for limited-delay universal lossy source coding. *IEEE Trans. Information Theory*, 60(5):2823–2834, 2014. doi: 10.1109/TIT.2014.2307062. URL `https://doi.org/10.1109/TIT.2014.2307062`.

M. Imase and B. M. Waxman. Dynamic steiner tree problem. *SIAM J. Discrete Math.*, 4(3):369–384, 1991. doi: 10.1137/0404033. URL `https://doi.org/10.1137/0404033`.

M. R. Karimi, M. Lucic, H. Hassani, and A. Krause. Stochastic Submodular Maximization: The Case of Coverage Functions. *NIPS*, 2017.

A. Krause and D. Golovin. Submodular function maximization., 2014.

J. Lacki, J. Ocwieja, M. Pilipczuk, P. Sankowski, and A. Zych. The power of dynamic distance oracles: Efficient dynamic algorithms for the steiner tree. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 11–20, 2015. doi: 10.1145/2746539.2746615. URL `http://doi.acm.org/10.1145/2746539.2746615`.

S. Lattanzi and S. Vassilvitskii. Consistent k-clustering. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 1975–1984, 2017. URL `http://proceedings.mlr.press/v70/lattanzi17a.html`.

Y. LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance. Cost-effective outbreak detection in networks. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 420–429, August 2007.

M. S. Manasse, L. A. McGeoch, and D. D. Sleator. Competitive algorithms for server problems. *Journal of Algorithms*, 11(2):208–230, 1990.

N. Megow, M. Skutella, J. Verschae, and A. Wiese. The power of recourse for online MST and TSP. *SIAM J. Comput.*, 45(3):859–880, 2016. doi: 10.1137/130917703. URL `https://doi.org/10.1137/130917703`.

N. Merhav, E. Ordentlich, G. Seroussi, and M. J. Weinberger. On sequential strategies for loss functions with memory. *IEEE Trans. Information Theory*, 48(7):1947–1958, 2002. doi: 10.1109/TIT.2002.1013135. URL `https://doi.org/10.1109/TIT.2002.1013135`.

G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions—i. *Mathematical programming*, 14(1):265–294, 1978.

S. J. Phillips and J. Westbrook. On-line load balancing and network flow. *Algorithmica*, 21(3):245–261, 1998. doi: 10.1007/PL00009214. URL `https://doi.org/10.1007/PL00009214`.

P. Sanders, N. Sivadasan, and M. Skutella. Online scheduling with bounded migration. *Math. Oper. Res.*, 34(2):481–498, 2009. doi: 10.1287/moor.1090.0381. URL `https://doi.org/10.1287/moor.1090.0381`.

M. Skutella and J. Verschae. A robust PTAS for machine covering and packing. In *Algorithms - ESA 2010, 18th Annual European Symposium, Liverpool, UK, September 6-8, 2010. Proceedings, Part I*, pages 36–47, 2010. doi: 10.1007/978-3-642-15775-2_4. URL `https://doi.org/10.1007/978-3-642-15775-2\_4`.

J. Westbrook. Load balancing for response time. *J. Algorithms*, 35(1):1–16, 2000. doi: 10.1006/jagm.2000.1074. URL `https://doi.org/10.1006/jagm.2000.1074`.

M. Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 928–936, 2003.