ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Learning &
Adaptive Systems

Consistent Online Optimisation

Mohammad Reza Karimi Jaghargh

Master Thesis

Computer Science Department
Learning and Adaptive Systems Group
ETH Zürich

Supervisor

Prof. Dr. Andreas Krause

March 29, 2019

Consistent Online Optimisation

Mohammad Reza Karimi Jaghargh

Learning and Adaptive Systems Group, Computer Science Department, ETH Zürich

E-mail address: mkarimi@ethz.ch

To Ali,
the shining star in the darkness,
and to my family, for their constant support.

Contents

Chapter 1. Introduction 1
1.1. Related Work 1
1.2. Contributions 2
1.3. Acknowledgements 2

Chapter 2. Preliminaries 3
2.1. Online Learning 3
2.1.1. Online Convex Optimisation 4
2.1.2. Strongly Convex Losses 5
2.2. Poisson Processes 6
2.3. Submodularity 7
2.3.1. Basic Notions 7
2.3.2. Continuous Extensions 8
2.3.3. Submodular Maximisation and Multilinear Extension 9
2.3.4. Rounding 9
2.3.5. Continuous Submodularity 9
2.4. Online Submodular Maximisation 11

Chapter 3. Consistent Algorithms 13
3.1. Consistency Cost 13
3.2. Decision Paths 13
3.3. Towards No-Regret Consistent Algorithms 15
3.4. Analysis of SOLO 16
3.4.1. Regret 16
3.4.2. Extra Regret 17
3.5. Online Gradient Descent Trade-offs 19
3.6. Online Submodular Maximisation Trade-offs 21

Chapter 4. Regularisation 23
4.1. Banach Spaces 23
4.2. Regularisation in Online Learning 23
4.3. Consistent RFTL 23
4.4. Example: OGD Revisited 23
4.5. Example: Experts Advice 23

Chapter 5. Experimental Results 25

Chapter 6. Conclusion 29

Appendix A. Supplementary Lemmata 31

Bibliography 33

CHAPTER 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 3.6 we show how to extended our technique to solve the consistent online submodular maximization problem, and in Section **todo: the EG algorithm...**

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 Shrinking Dartboard framework introduced by Geulen et al. (2010), but unfortunately their technique is limited to the expert setting. Recently,

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

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.

1.2. Contributions

Much of this work is previously presented in Karimi et al. (2019), specifically, sections ....

1.3. Acknowledgements

I thank Andreas Krause for his constant support and enlightening ideas and guidance. **TBC.**

CHAPTER 2

Preliminaries

Here we bring the preliminaries to this work, including some remarks about online learning, Poisson processes, and submodularity.

2.1. Online Learning

Many prediction problems can be represented as repeated games between a forecaster and the environment (Cesa-Bianchi and Lugosi, 2006), or a player and an adversary. In this representation, at round t, the player has to choose a strategy x_t from a set of possible strategies $\mathcal{X}$, and will incur the loss $f_t(x_t)$ from the adversary. This loss can be thought of as the 'prediction error', which is defined naturally based on the environment, or some hand-crafted function depending on x_t, that the adversary imposes on the player.

There could be randomness involved in the player's strategy, as well as the adversary. For example, one setting is that the adversary knows the distribution on $\mathcal{X}$ which the player is going to sample her strategy, but does not know the sample before choosing his loss. Another example would be the case that the adversary samples losses from a fixed distribution, unknown to the player, but not dependent on player's strategy. Depending on the randomness and the adversary's knowledge, some settings are closer to the situation of game playing, and some settings are closer to that of learning. We will go through the main results about game theoretic understanding of online learning in a later chapter. In this work, we only focus on the setting where the player is faced with an adversary, and her actions are deterministic.

To measure the performance of the player, one needs to introduce a baseline. A seemingly easy notion is that of *regret*, which one compares the accumulated loss up to round T with the least possible loss in hindsight, following the best fixed strategy, that is,

$$R(T) = \sum_{t=1}^{T} f_t(x_t) - \min_{x^\star \in \mathcal{X}} \sum_{t=1}^{T} f_t(x^\star). \tag{2.1}$$

Notice that, in general, the player cannot compute the regret, as she needs access to all functions $f_1, \dots, f_T$. But what one can do is to provide algorithms that provably get low regrets. Moreover, one may ask for more sophisticated baselines, rather than the ones playing with a fixed strategy. It turns out that even for this simple baseline, providing algorithms achieving low regret is nontrivial.

One accepted notion of 'low regret' is that of Hannan Consistency, that is,

$$\limsup_{T\to\infty} \frac{R(T)}{T} \leq 0, \quad \text{a.s.},$$

where the probability space is defined for the random decisions of the player (and/or the adversary). In deterministic setting, we can write the condition as $R(T) = o(T)$ as $T \to \infty$. In the online learning community, it is customary to call a Hannan consistent algorithm, a *no-regret* algorithm.

2.1.1. Online Convex Optimisation. One of the vastly studied settings in the online learning community is the convex setting, also called 'Online Convex Optimisation'. In this setting, one assumes that $\mathcal{X}$ is a compact convex subset of $\mathbb{R}^d$, and loss functions are convex.

Let us put a norm $\|\cdot\|$ on $\mathbb{R}^d$, and let $\|\cdot\|_\star$ denote its dual norm, i.e.

$$\|x\|_\star = \sup\{z^\top x : \|z\| \leq 1\}.$$

We denote by $C_L^1(\mathcal{X})$ the set of real-valued continuously differentiable functions defined over $\mathcal{X}$ that are Lipschitz continuous with constant L, i.e., $\|\nabla f(x)\|_\star \leq L$ for all $x \in \mathcal{X}$. This is equivalent to say that for all $x, y \in \mathcal{X}$, we have

$$|f(x) - f(y)| \leq L \cdot \|x - y\|.$$

Also, let $\mathrm{Proj}_{\mathcal{X}}(z)$ be the unique orthogonal projection of z onto the convex set $\mathcal{X}$. If $\mathcal{X}$ is clear from the context, we drop it and write $\mathrm{Proj}(z)$ instead.

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, known as Online Gradient Descent, demonstrated in Algorithm 1.

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 $x_{t+1} := \mathrm{Proj}(x_t - \eta_t \nabla f_t(x_t))$.
end for

The following theorem proves that OGD suffers a sublinear regret, that is, it is a no-regret algorithm.

THEOREM 2.1 (Zinkevich (2003, Theorem 1)). *Suppose $f_t \in C_L^1(\mathcal{X})$ and convex for all $t \in [T]$. Then Online Gradient Descent with step size $\eta_t = t^{-1/2}$ guarantees*

$$R(T) \leq \frac{D^2\sqrt{T}}{2} + (\sqrt{T} - \tfrac{1}{2})L^2 = O(\sqrt{T}). \tag{2.2}$$

In particular, one has $\limsup_{t\to\infty} R(T)/T \leq 0$. Therefore, OGD is a no-regret algorithm.

PROOF. This proof follows Zinkevich's, and proceeds in three steps. First, we prove that replacing convex losses with their linear approximation gives an upper bound on the regret. Next, we deal with projection step, and finally, we prove the claim of the theorem.

Assume that the algorithm produces points $x_1, x_2, \ldots$ and define $g_t = \nabla f_t(x_t)$. We claim that replacing f_t with the linear functions $g_t(x) := \langle g_t, x\rangle$ gives an upper bound on regert. This is a straightforward application of convexity. One can write for all $x \in \mathcal{X}$,

$$f_t(x) \geq f_t(x_t) + \langle g_t, x - x_t\rangle .$$

Thus,

$$f_t(x_t) - f_t(x) \leq g_t(x_t) - g_t(x),$$

which means that the regret would be at least as much with the modified sequence of functions.

Next, we define for all $t \in [T]$, the point $y_{t+1} := x_t - \eta_t g_t$. Observe that $x_{t+1} = \mathrm{Proj}(y_{t+1})$. For all $x^\star \in \mathcal{X}$ we have

$$\|y_{t+1} - x^\star\|^2 = \|x_t - x^\star\|^2 - 2\eta_t \langle g_t, x_t - x^\star\rangle + \eta_t^2 \|g_t\|^2.$$

By Projection Lemma (Lemma A.1), we have $\|x_{t+1} - x^\star\| \leq \|y_{t+1} - x^\star\|$. Thus, by reorganising terms we get

$$\langle g_t, x_t - x^\star\rangle \leq \frac{1}{2\eta_t}(\|x_t - x^\star\|^2 - \|x_{t+1} - x^\star\|^2) + \frac{\eta_t}{2}\|g_t\|^2.$$

The regret (for the functions g_t) is nothing else than the sum of left-hand-side over all t. Hence,

$$\begin{aligned}
R(T) &\leq \sum_{t=1}^{T} \langle g_t, x_t - x^\star\rangle \\
&\leq \sum_{t=1}^{T} \left(\frac{1}{2\eta_t}(\|x_t - x^\star\|^2 - \|x_{t+1} - x^\star\|^2) + \frac{\eta_t}{2}\|g_t\|^2\right) \\
&= \frac{1}{2\eta_1}\|x_1 - x^\star\|^2 - \frac{1}{2\eta_T}\|x_{T+1} - x^\star\|^2 + \frac{1}{2}\sum_{t=2}^{T}\left(\frac{1}{\eta_t} - \frac{1}{\eta_{t-1}}\right)\|x_t - x^\star\|^2 + \frac{L^2}{2}\sum_{t=1}^{T}\eta_t \\
&\leq D^2\left(\frac{1}{2\eta_1} + \frac{1}{2}\sum_{t=2}^{T}\left(\frac{1}{\eta_t} - \frac{1}{\eta_{t-1}}\right)\right) + \frac{L^2}{2}\sum_{t=1}^{T}\eta_t \\
&= D^2\frac{1}{2\eta_T} + \frac{L^2}{2}\sum_{t=1}^{T}\eta_t.
\end{aligned}$$

As we chose $\eta_t = t^{-1/2}$, we can use the fact that $\sum_{t=1}^{T} t^{-1/2} \leq 2\sqrt{T} - 1$, and get

$$R(T) \leq \frac{1}{2}D^2\sqrt{T} + L^2(\sqrt{T} - \tfrac{1}{2}),$$

as desired. □

Remark 2.2. It is worthwhile to mention that if L and D are known to the player in advance, he can choose a more accurate learning rate, $\eta_t = \frac{D}{L}t^{-1/2}$, and obtain the regret bound

$$R(T) \leq \frac{3}{2}DL\sqrt{T}.$$

The proof is essentially the same, hence we exclude it. The intersted reader is referred to Hazan (2015) for a complete exposition.

If there are more restrictions to the function class, and the player knows those restrictions, it is possible to get better regret bounds. One such example is when the functions are *strongly convex*.

2.1.2. Strongly Convex Losses. We first remind the notion of strong convexity and explain some of its properties. In the following, we assume that $\mathcal{X}$ is a closed convex subset of $\mathbb{R}^d$.

Definition 2.3 (Strongly Convex Function). A C^1 function $f : \mathcal{X} \to \mathbb{R}$ is α-strongly convex (w.r.t. the norm $\|\cdot\|$), if for all $x, y \in \operatorname{relint}\mathcal{X}$ and all $s \in (0, 1)$, one has

$$f(sx + (1-s)y) \leq sf(x) + (1-s)f(y) - \frac{\alpha}{2}s(1-s)\|x - y\|^2.$$

REMARK 2.4. If the norm is taken to be the Euclidean norm, and if f is C^1, strong convexity is equivalent to

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\alpha}{2}\|y - x\|_2^2.$$

Moreover, if f is continuously twice differentiable, then, strong convexity w.r.t. the Euclidean norm is equivalent to $D^2 f(x) \succeq \alpha I$ for all $x \in \mathcal{X}$.

The following theorem shows that if the step sizes for OGD are chosen properly, logarithmic regrets can be achieved for the case of strongly convex functions. Note that the player should be aware of the value of α.

THEOREM 2.5 (Hazan et al. (2007, Theorem 1)). *Assuming that f_t are L-Lipschitz and α-strongly convex, the OGD algorithm with step sizes $\eta_t = \frac{1}{\alpha t}$ achieves the following guarantee, for all $T > 1$.*

$$R(T) \leq \frac{L^2}{2\alpha}(1 + \log T) = O(\log T).$$

2.2. Poisson Processes

Here we bring an introduction to Poisson processes. We omit some details and refer the reader to the excellent book by Kingman (1992).

Let $(X, \mathcal{F})$ be a measurable space. A *Poisson process* on X is a random countable subset Π of X, such that

(1) For any disjoint measurable subsets $A_1, \ldots, A_n$ of X, the random variables $N(A_1)$, $N(A_2)$, $\ldots$, $N(A_n)$ are independent, where $N(A) := |\Pi \cap A|$,
(2) $N(A)$ has the Poisson distribution with mean $\mu = \mu(A)$, where $0 \leq \mu \leq \infty$. That is, for all integers $k \geq 0$,

$$\Pr[N(A) = k] = e^{-\mu(A)} \frac{\mu(A)^k}{k!}.$$

If $\mu(A)$ is finite, the intersection $\Pi \cap A$ is finite, and empty if $\mu = 0$, almost surely. Also, if $\mu(A) = \infty$, then $\Pi \cap A$ is countably infinite. It can be shown that μ is a measure on X, which is called the *mean measure*. The name comes from the fact that $\mathbf{E}|\Pi \cap A| = \mu(A)$.

In the case that X is the real line $\mathbb{R}$ with its Borel σ-algebra, we may express the Poisson process in terms of its *intensity* or *rate*, defined as a positive measurable function λ with

$$\mu(A) = \int_A \lambda(x)\, dx.$$

This is indeed the Radon-Nikodym derivative of μ with respect to the Lebesgue measure on $\mathbb{R}$ (if the derivative exists). In the case that λ is constant, we call the process a *homogeneous Poisson process*.

Recall that when μ is not infinite on bounded sets, it is determined uniquely by its values on the intervals $(a, b]$. Define

$$M(\tau) = \mu(0, \tau] = \int_0^\tau \lambda(u)\, du, \quad \tau \geq 0,$$

$$M(\tau) = -\mu(\tau, 0] = -\int_\tau^0 \lambda(u)\, du, \quad \tau < 0.$$

Thus, we observe that M is an increasing function and

$$\mu(a, b] = M(b) - M(a).$$

We finish this section with a useful lemma that we need later on.

LEMMA 2.6. *Let $\Pi = \{X_1, X_2, \ldots\}$ be a Poisson process on $(0, \infty)$ with intensity λ, and M be as above. Then, for $0 < a < b$, the probability that $\Pi \cap (a, b)$ is empty equals*

$$\Pr[\Pi \cap (a, b) = \varnothing] = e^{-(M(b)-M(a))}.$$

PROOF. As mentioned before, the number of points of Π inside (a, b) follows a Poisson distribution with mean $\mu = \mu(a, b) = \int_a^b \lambda = M(b) - M(a)$. As a Poisson random variable N has probability mass function $\Pr[N = k] = e^{-\mu}\mu^k/k!$, we get the result by setting $k = 0$. □

2.3. Submodularity

Submodular functions are discrete analogs of convex functions. They arise naturally in many areas, such as the study of graphs, matroids, covering problems, and facility location problems. These functions are extensively studied in operations research and combinatorial optimization (Krause and Golovin, 2012). Recently, submodular functions have proven to be key concepts in other areas such as machine learning, algorithmic game theory, and social sciences. As such, they have been applied to a host of important problems such as modeling valuation functions in combinatorial auctions, feature and variable selection (Krause and Guestrin, 2005), data summarization (Lin and Bilmes, 2011), and influence maximization (Kempe et al., 2003).

2.3.1. Basic Notions. In the following section, we assume $V = \{1, \ldots, n\}$ to be the ground set. A *set function* f is a real-valued function whose domain is the power set of V, i.e., $f \colon 2^V \to \mathbb{R}$. One can also look at a set function as a function defined on the discrete hypercube $C_n = \{-1, 1\}^n$. A set function f is called *submodular* if for all $A, B \subseteq V$, it holds

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B). \tag{2.3}$$

It can be shown that this condition is equivalent to the so-called *diminishing returns* (DR) property, that is, for all $A \subseteq B \subseteq V$ and $x \notin B$, it holds that

$$f(A + x) - f(A) \geq f(B + x) - f(B), \tag{2.4}$$

where $A + x$ is a shorthand for $A \cup \{x\}$. Also, a set function f is called *monotone*, if for all $A \subseteq B \subseteq V$, it holds

$$f(A) \leq f(B).$$

In optimisation problems related to set function, as in continuous optimisation, one can have different types of combinatorial constraints. One of the most popular types of constraints are *matroid* constraints. That is, one searches for the maximum (or minimum) of a set function over the sets that are from a matroid. Basically, a matroid is a structure that abstracts and generalises the notion of linear independence in vector spaces, and is defined as follows.

DEFINITION 2.7 (Matroid). Let V be a finite ground set. A collection of subsets $\mathcal{M} \subseteq 2^V$ of V is called a *matroid* if

(1) $\emptyset \in \mathcal{M}$,
(2) If $A \in \mathcal{M}$ and $B \subseteq A$, then $B \in \mathcal{M}$. That is, $\mathcal{M}$ is downwards closed.
(3) If $A, B \in \mathcal{M}$ with $|A| > |B|$, then there exists an element $x \in A \setminus B$ such that $B + x \in \mathcal{M}$.

Usually the elements of a matroid are called *independent sets*. The maximal elements of a matroid are of importance, and are called *bases* of the matroid. One can prove that the size of all basis elements of a matroid are the same (called the *rank* of the matroid).

For *any* subset of V, one can assing a number $r(A)$ called the *rank of* A, which is the size of the largest independent set $B \subseteq A$. An interesting result in matroid theory states that the rank function of a matroid is submodular. Moreover, this gives an alternative defintion of a matroid, as follows.

DEFINITION 2.8 (Matroid, Second Definition). Let V be a finite ground set, and $r\colon 2^V \to \mathbb{Z}_+$ be a function satisfying

(1) For any $A \subseteq V$, $r(A) \leq |A|$,
(2) r is submodular,
(3) For all $A \subseteq V$ and all $x \in V$, $r(A) \leq r(A + x) \leq r(A) + 1$.

Then, the set

$$\{A \subseteq V \mid r(A) = |A|\}$$

is a matroid over V.

A running example for a matroid is the *k-uniform matroid*, which is the set of all subsets of size at most k, for some $k \in \mathbb{N}$. Other important matroids, including the partition matroid, are of important use in applications, for which the reader is referred to Fujishige (2005, Chapter II.2).

2.3.2. Continuous Extensions. The idea of relaxing a combinatorial problem to a continuous optimisation problem, and translating the continuous solution back to a combinatorial solution (i.e. *rounding*) is ubiquitous in combinatorial optimisation community. Likewise, for submodular set functions, one may look for *extensions* to continuous domains. In what follows, we introduce the *Lovász extension*, and in the next section we introduce the *multilinear extension*, which both are studied extensively in submodularity community and are related to this thesis.

The Lovász extension (Lovász, 1983) creates a connection between submodular functions and convex functions, and thus, is helpful for submodular minimisation.

DEFINITION 2.9 (Lovász Extension). Let $f\colon 2^V \to \mathbb{R}$ be a set function with $f(\emptyset) = 0$. The Lovász extension of f is the function $f_L : \mathbb{R}^n \to \mathbb{R}$ defined as follows: For $x \in \mathbb{R}^n$, order the components in decreasing order $x_{i_1} \geq x_{i_2} \geq \cdots \geq x_{i_n}$, and define $f_L(x)$ to be

$$f_L(x) = \sum_{j=1}^{n} x_{i_j}[f(\{i_1, \ldots, i_j\}) - f(\{i_1, \ldots, i_{j-1}\})].$$

The importance of this extension resides in the following theorem.

THEOREM 2.10 ((Lovász, 1983, Proposition 4.1)). *Let f be any set function defined on subsets of V and let f_L be its Lovász extension on the nonnegative vectors. Then f is submodular if and only if f_L is convex.*

This theorem allows one to minimise the Lovász extension efficiently (using, say, the Ellipsoid method), and hence find the minimum of a submodular function. Thus, submodular minimisation can be done in poly-time (see Fujishige (2005) and references therein).

2.3.3. Submodular Maximisation and Multilinear Extension. ..... hardness.....

For a monotone submodular set function $f\colon 2^V \to \mathbb{R}_+$, a canonical extension to a smooth monotone submodular function (in the sense of Section 2.3.5) can be obtained as follows (Calinescu et al., 2011): For $x \in [0,1]^n$, let $\hat{x}$ denote a random vector in C_n where each coordinate j is independently rounded to 1 with probability x_j or 0 otherwise. We identify $\hat{x} \in C_n$ with a set $S \subseteq V$ whose indicator vector is $\hat{x} = \mathbf{1}_S$. Then, define

$$F(x) := \mathbf{E}[f(\hat{x})] = \sum_{S \subseteq V} f(S) \prod_{i \in S} x_i \prod_{j \notin S} (1 - x_j).$$

We call F the *multilinear extension* of f.

LEMMA 2.11. *The multilinear extension of a monotone submodular set function f is monotone and (continuous) submodular.*

PROOF. By definition of F, one has

$$\frac{\partial F}{\partial x_i}(x) = \mathbf{E}[f(\hat{x}) \mid \hat{x}_i = 1] - \mathbf{E}[f(\hat{x}) \mid \hat{x}_i = 0] \geq 0,$$

thus F is monotone, as $\nabla F(x) \geq 0$.

Also, for $i \neq j$ we can compute

$$\begin{aligned}\frac{\partial^2 F}{\partial x_i \partial x_j}(x) = \mathbf{E}[f(\hat{x}) \mid \hat{x}_i = 1, \hat{x}_j = 1] - \mathbf{E}[f(\hat{x}) \mid \hat{x}_i = 1, \hat{x}_j = 0] \\ - \mathbf{E}[f(\hat{x}) \mid \hat{x}_i = 0, \hat{x}_j = 1] + \mathbf{E}[f(\hat{x}) \mid \hat{x}_i = 0, \hat{x}_j = 0] \leq 0,\end{aligned}$$

where the last inequality follows from submodularity of f. Moreover, $\partial^2 F(x)/\partial x_i^2 = 0$. Thus, F is continuous submodular (this also shows that F is also DR-submodular, see Definition 2.15). □

TODO: explain matroid polytope and the continuous greedy algorithm.

2.3.4. Rounding. Here I shall explain the Swap-Rounding. **TODO: complete the picture and bring the theorem that one can maximise monotone submodular with matroid constraints...**

2.3.5. Continuous Submodularity. The notion of submodularity we defined in Section 2.3.1 can be extended to more general structures such as distributive lattices and continuous domains. Our goal in this section is to present some of such extensions to continuous domains (Bach, 2016).

Let $n \in \mathbb{N}$ and take $\mathcal{X}_1, \ldots, \mathcal{X}_n$ to be compact subsets of $\mathbb{R}$. Define $\mathcal{X} = \prod_i \mathcal{X}_i$ to be the product space of all $\mathcal{X}_i$. Our running examples for $\mathcal{X}_i$ is either finite sets (discrete) or intervals (continuous).

DEFINITION 2.12 (Submodular Function). A continuous function $f\colon \mathcal{X} \to \mathbb{R}$ is called *(continuous) submodular* if for all $x, y \in \mathcal{X}$, one has

$$f(x) + f(y) \geq f(x \wedge y) + f(x \vee y), \tag{2.5}$$

where $\wedge$ is component-wise minimum, and $\vee$ is component-wise maximum.

The following lemma sheds light on the connection with our previous definition of submodularity (2.3).

LEMMA 2.13. *Let $f\colon \mathcal{X} \to \mathbb{R}_+$ be a continuous function and $x \in \mathcal{X}$. Let e_i, e_j be two basis vectors of $\mathbb{R}^n$, and $a_i, a_j \in \mathbb{R}_+$, such that $x_i + a_i e_i \in \mathcal{X}_i$ and $x_j + a_j e_j \in \mathcal{X}_j$. Then f is continuous submodular (2.5) if and only if*

$$f(x + a_i e_i) + f(x + a_j e_j) \geq f(x) + f(x + a_i e_i + a_j e_j). \tag{2.6}$$

PROOF. If f is submodular in the sense of (2.5), then take x to be $x + a_i e_i$ and y to be $x + a_j e_j$. The inequality follows from the definition. For the other direction of the lemma, one can argue by induction on the number of different coordinates in $x \vee y$ and $x \wedge y$. We omit the details here. □

The lemma above is most useful if one considers the limiting cases. If $\mathcal{X}_l$ are subsets of $\mathbb{Z}$, the case where $a_i = a_j = 1$ is important, and when $\mathcal{X}_l$ are intervals, the limiting case $a_i \to 0$ is important. The special case when f is twice differentiable gives rise to the following lemma.

LEMMA 2.14. *Let $\mathcal{X}_k$ be intervals in $\mathbb{R}$ and $f\colon \mathcal{X} \to \mathbb{R}$ be a continuously twice differentiable function. Then f is submodular if and only if for all $x \in \operatorname{int} \mathcal{X}$,*

$$\frac{\partial^2 f}{\partial x_i \partial x_j}(x) \leq 0, \quad \forall i \neq j.$$

Examples of Submodular Functions. We bring some examples of submodular functions.

- Set $\mathcal{X}_i = \{0, 1\}$ for all $i \in [k]$. Then, it is easy to see that the definition (2.5) coincides with the usual definition of a submodular *set function* (2.3). Here, x is the indicator vector of a set, and $x \vee y$ is the indicator of the union of the corresponding sets, etc.
- Let $\varphi_{ij}\colon \mathbb{R} \to \mathbb{R}$ be convex. One can prove that the functions $x \mapsto \varphi_{ij}(x_i - x_j)$ are submodular. Moreover, for a concave function $g\colon \mathbb{R} \to \mathbb{R}$, the function $x \mapsto g(\sum_{i=1}^n \lambda_i x_i)$ is submodular, if all λ_i are nonnegative. These are two examples of submodular functions which are convex and concave, respectively.
- The Lovász extension of a submodular set function f (defined in Section 2.3.2) is always convex and submodular. Also, the multilinear extension of f is submodular, but neither convex nor concave in general.

The careful reader should have noticed by now that the notion of continuous submodularity only generalises (2.3). One may ask, is there a similar property implied from continuous submodularity analoguos to (2.4)? The general answer is no Soma et al. (2014). This motivated Bian et al. (2017) to define a subclass of continuous submodular functions, known as DR-submodular functions, that resemble the DR property.

DEFINITION 2.15 (DR-submodular Function). A continuous function $f\colon \mathcal{X} \to \mathbb{R}$ is called DR-submodular if for all $a \leq b \in \mathcal{X}$, all $i \in V$, and all $r \in \mathbb{R}_+$ such that $a + re_i \in \mathcal{X}$ and $b + re_i \in \mathcal{X}$, it holds

$$f(a + re_i) - f(a) \geq f(b + re_i) - f(b).$$

Moreover, if f is twice differentiable, this condition is equivalent to

$$\frac{\partial^2 f}{\partial x_i \partial x_j}(x) \leq 0, \quad \forall x \in \mathcal{X}, \ \ \forall i, j \in [k].$$

Interestingly, a similar hardness result as for submodular set function maximisation hold for DR-submodular functions, which we bring a gist in the following theorem.

Thus, in both regimes (continuous or discrete) one has the same approximation ratio and (roughly) the same hardness.

THEOREM 2.16 (Bian et al. (2017, Proposition 3)). *The problem of maximising a monotone DR-submodular continuous function subject to a general down-closed polytope constraint is NP-hard. For any $\varepsilon > 0$, it cannot be approximated in polynomial time within a ratio of $(1 - 1/e + \varepsilon)$ (up to low-order terms), unless RP = NP.*

2.4. Online Submodular Maximisation

As explained in previous sections, it is NP-hard to find an optimal solution (in hindsight) for submodular maximisation problem. Thus, the usual notion of regret (2.1) is no longer meaningful, as no poly-time algorithm is going to achieve sublinear regret (unless P = NP). Instead, we consider approximate no-regret algorithms:

DEFINITION 2.17 (No β-regret Algorithm). For a maximisation task, we call an algorithm $\mathcal{A}$ a *no β-regret algorithm* for some $\beta < 1$ if for any sequence of functions $(f_t)_{t\in\mathbb{N}} \subset \mathcal{H}$ inside a function class $\mathcal{H}$, with $f_t \colon \mathcal{X} \to \mathbb{R}$, $\mathcal{A}$ 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

$$R_\beta(T) := \beta \cdot \max_{x^\star \in \mathcal{X}} \sum_{t=1}^{T} f_t(x^\star) - \sum_{t=1}^{T} f_t(x_t), \tag{2.7}$$

is sublinear, that is, $R_\beta(T) = o(T)$.

In this setting, since the task is maximisation, we regard f_t as *utility* rather than a loss. We drop β in R_β if it is clear from the context.

For monotone DR-submodular functions, one can show that Online Gradient Ascent achieves sublinear $\frac{1}{2}$-regret.

THEOREM 2.18 (Chen, Hassani and Karbasi (2018, Theorem 2)). *Assume that the functions $f_t \colon \mathcal{X} \to \mathbb{R}_+$ are monotone and DR-submodular for $t \in [T]$. Let $x_1, \dots, x_T$ be the choices of OGD with step size $\eta_t = \frac{D}{L\sqrt{t}}$, where D is the diameter of $\mathcal{X}$ and L is an upper bound on Lipschitz constants of f_t, then we have*

$$\frac{1}{2} \max_{x^\star \in \mathcal{X}} \sum_{t=1}^{T} f_t(x^\star) - \sum_{t=1}^{T} f_t(x_t) \leq \frac{3}{4} DL\sqrt{T}. \tag{2.8}$$

We close this section with two remarks.

REMARK 2.19. Theorem 2.18 also holds when one uses stochastic gradients, and the result holds in expectation. This is especially useful when f_t are defined as expectations (thus, hard to compute), and one can easily sample from the distribution over the gradients.

REMARK 2.20. In work by (Chen, Harshaw, Hassani and Karbasi, 2018), a projection-free no $(1 - 1/e)$-regret algorithm is provided. However, in this thesis we only analyse the OGD algorithm's performance due to its simplicity. The interested reader is referred to Remark 3.10 for further discussion.

CHAPTER 3

Consistent Algorithms

Our goal in this chapter will be to simultaneously achieve sublinear regret and minimize some notion of *consistency* cost. We first ... (**TODO: complete the exposition here.**)

3.1. Consistency Cost

A possible notion for consistency cost is the number of times the player changes her decision in T rounds, i.e.,

$$\kappa_T := \sum_{t=2}^{T} \mathbf{1}_{x_t \neq x_{t-1}}. \tag{3.1}$$

We note that there are different possible notions of consistency, all dependent on the respective application. As an example, in the situation of online discrete submodular maximisation, where at each round the player plays a *subset* $S_t \subseteq V$ and the loss functions are submodular, a notion of consistency cost can be defined as

$$\kappa'_T := \sum_{t=2}^{T} |S_t \triangle S_{t-1}|, \tag{3.2}$$

where $A \triangle B$ is the symmetric difference of A and B. In other situations where the decision space is a general metric space $(\mathcal{X}, d)$, one can define the consistency cost to be

$$\kappa''_T := \sum_{t=2}^{T} d(x_t, x_{t-1}). \tag{3.3}$$

As an example, the cost (3.1) becomes a special case of (3.3), when one puts the *discrete metric* on $\mathcal{X}$. Moreover, (3.2) is a special case, when one take $d(A, B) = |A \triangle B|$ to be a metric on the discrete hypercube.

Keeping each of these costs small[1] will result in different behaviours of the online algorithm. The first cost encourages the algorithm to *stick* to the previously played decisions. The second cost makes the algorithm to change the elements of the selected set as little as possible; it keeps a bag of items and change the items in the bag as few times as possible. The third one trys to *move* as little as possible in the decision space.

In what follows, we take (3.1) as the defintion of cosistency cost.

3.2. Decision Paths

Under the assumption of Lipschitz loss functions, upper bounds on regret can be obtained by bounding the amount one *moves* in the decision space. This brings us to the notion of a *decision path*.

[1] with a correct interpretation of smallness.

DEFINITION 3.1 (Decision Path). For a sequence of points $(x_t)_{t\in[T]} \subset \mathbb{R}^d$ we define the decision path to be the 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.

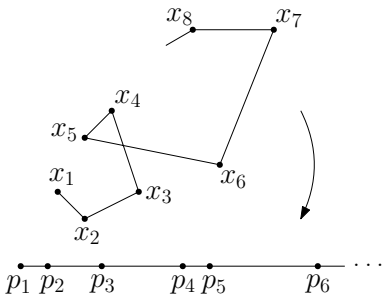

FIGURE 1. The Decision Path

For analytic reasons that become apparent in later sections, we also introduce the *δ-padded decision path* as follows.

DEFINITION 3.2 (δ-padded Decision Path). 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 $t > 1$.

Any online algorithm whose decision space is a subset of Euclidean space, naturally defines a decision path: take x_t to be the decision made at round t. For the special case of the OGD algorithm, one has the following lemma for the length of the corresponding decision path.

LEMMA 3.3. *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) Using step size $\eta_t = \frac{D}{L}t^{-1/2}$, one has for all $t \in [T]$,

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

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

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

PROOF. By the OGD update rule, the Projection Lemma A.1, and Lipschitz continuity of f_t, one gets

$$\begin{aligned} p_{t+1} - p_t &= \|x_{t+1} - x_t\| \\ &= \|\mathrm{Proj}_{\mathcal{X}}(x_t - \eta_t \nabla f_t(x_t)) - x_t\| \\ &\leq \|x_t - \eta_t \nabla f_t(x_t) - x_t\| \\ &= \eta_t \|\nabla f_t(x_t)\| \leq \eta_t L. \end{aligned}$$

Since $p_1 = 0$ one can write

$$p_t = \sum_{j=1}^{t-1}(p_{j+1} - p_j) \leq L \sum_{j=1}^{t-1} \eta_j.$$

For (i), we have $\eta_t = \frac{D}{L\sqrt{t}}$ and

$$p_t \leq L \sum_{j=1}^{t-1} \frac{D}{L\sqrt{j}} \leq D \int_0^{t-1} z^{-\frac{1}{2}}\, dz \leq 2D\sqrt{t}.$$

For (ii), we have $\eta_t = \frac{1}{\alpha t}$ and

$$p_t \leq L \sum_{j=1}^{t-1} \frac{1}{\alpha j} \leq \frac{L}{\alpha}(1 + \log t).$$

□

3.3. Towards No-Regret Consistent Algorithms

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 to $\mathcal{A}$'s proposal, 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 $\mathcal{A}$'s newly proposed point, but also less likely to happen as the algorithm proceeds and starts converging to a near-optimal solution. As in a Poisson process, the distribution of inter-arrival times is dependent on the *length* of the interval, we will update the algorithm's solution whenever a specific point process on the padded decision path has a hit.

Concretely, let $\lambda : \mathbb{R}_+ \to \mathbb{R}_+$ be an increasing intensity function we will define later. Also define $(r_t)_{t \leq T}$ to be the padded decision path of $\mathcal{A}$. 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.

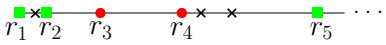

FIGURE 2. Let (r_t) be the padded decision path. The crosses are the Poisson process events. Red points denote rounds we do not update and green squares are updating rounds. We always update in the first round, and for round t, we update only if there is a Poission event on the line segment ending in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

Observe that this definition satisfies our desiderata from above. Suppose that the algorithm last updated its strategy 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 consecutive x's decreases), so does the probability of an update.

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

Algorithm 2 Sticky OnLine Optimization (SOLO)

$\mathcal{A}$ is a no-regret algorithm
λ an increasing intensity function.
(δ_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 $1 - \exp\big\{ - \big(\int_{r_t}^{r_{t+1}} \lambda(u)\, du \big) \big\}$
    Update: set $y_{t+1} \leftarrow x_{t+1}$.
  Otherwise set $y_{t+1} \leftarrow y_t$.
end for

3.4. Analysis of SOLO

The analysis of Algorithm 2 proceeds in two 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 3.4). Then, we prove a technical fact about the distribution of hits in non-homogeneous Poisson processes with sufficiently high intensities (Proposition3.5).

3.4.1. Regret. We analyze the regret of our algorithm as compared to $\mathcal{A}$. Assume that $f_t \in C_L^1(\mathcal{X})$ 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:

$$
\begin{aligned}
R^{\mathrm{SOLO}}(T) &= \sum_{t=1}^{T} f_t(y_t) - \min_{x^\star \in \mathcal{X}} \sum_{t=1}^{T} f_t(x^\star) \\
&= \sum_{t=1}^{T} f_t(y_t) - \sum_{t=1}^{T} f_t(x_t) + \sum_{t=1}^{T} f_t(x_t) - \min_{x^\star \in \mathcal{X}} \sum_{t=1}^{T} f_t(x^\star) \\
&\le L \sum_{t=1}^{T} \|y_t - x_t\| + R^{\mathcal{A}}(T),
\end{aligned}
\tag{3.4}
$$

where the last line is due to the Lipschitz continuity of f_t and $R^{\mathcal{A}}(T)$ 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

$$\mathcal{E}_T = L \cdot \sum_{t=1}^{T} (r_t - r_{p(t)}). \tag{3.5}$$

The following proposition is now immediate:

PROPOSITION 3.4.

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

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.

3.4.2. Extra Regret. 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 proposition, bounding this quantity for non-homogeneous Poisson processes with sufficiently fast increasing intensities.

PROPOSITION 3.5. *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*

$$\mathbf{E}[r_t - r_{p(t)}] \le C/\lambda(r_t),$$

where C is a constant independent of t and r_t, and expectation is with respect to the randomness of the Poisson process. Moreover,

$$\mathbf{E}[\mathcal{E}_T] \le C \sum_{t=1}^{T} \frac{1}{\lambda(r_t)}.$$

At a high level, Proposition 3.5 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 lemmata which are crucial in the proof.

LEMMA 3.6. *Assuming conditions of Proposition 3.5, 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

$$\log \int_0^\tau e^{M(u)}\,du < \log(\tau e^{M(\tau)}) = M(\tau) + \log\tau. \tag{3.6}$$

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

$$\log \int_0^\tau e^{M(u)}\,du \ge M(\tau) - \log\frac{1+\tau\lambda(\tau)}{\tau}. \tag{3.7}$$

Dividing both sides of (3.6) 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 (3.7), and taking the liminf gives

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

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 3.7. *With the same conditions as in Proposition 3.5, we have*

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

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 3.6 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 Proposition 3.5. We begin by deriving the probability distribution of $p(t)$. For all $1 < t \le T$ and $m \le t$

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

Then,

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

Now by Lemma 3.7, one gets the desired bound. Also, by linearity of expectation, we get

$$\mathbf{E}[\mathcal{E}_T] = L \cdot \sum_{t=1}^{T}\mathbf{E}[r_t - r_{p(t)}] \le C\sum_{t=1}^{T}\frac{1}{\lambda(r_t)}, \tag{3.8}$$

which finishes the proof. □

3.5. Online Gradient Descent Trade-offs

We are now ready to specialize the general algorithm above to obtain our main results. First, we bring a regret-consistency trade-off for OGD for convex and strongly convex functions. Next, we provide trade-offs for online submodular maximisation.

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

Theorem 3.8 (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:

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

PROOF. It is easy to check that $\lambda(\tau)$ and $M(\tau)$ satisfy the conditions of Proposition 3.5. Therefore:

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

Then,

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

By Theorem 2.1, we know that $R_{\mathcal{A}}(T) = O(\sqrt{T})$. Hence

$$\mathbf{E}[R^{\mathrm{SOLO}}(T)] = O(T^{1-\frac{\varepsilon}{2}}).$$

Finally, to bound κ_T, recall that the total path length from Lemma 3.3 (i), and the final padding of $\delta(T) = \sqrt{T}$. Since $M(\tau)$ is increasing, we get

$$\begin{aligned} \mathbf{E}[\kappa_T] &\leq M(r_T) \leq M((2D+1)\sqrt{T}) \\ &= (2D+1)^{1+\varepsilon} T^{\frac{1}{2}+\frac{\varepsilon}{2}} = O(T^{\frac{1}{2}+\frac{\varepsilon}{2}}), \end{aligned} \tag{3.9}$$

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 3.9 (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:*

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

Then one has

$$\mathbf{E}[\kappa_T] = O(T^{\varepsilon}), \quad \mathbf{E}[R^{\mathrm{SOLO}}(T)] = O(T^{1-\varepsilon\frac{1}{\gamma}}).$$

PROOF. By Lemma 3.3 and the choice of $\delta_\cdot$, since $M(\tau)$ is increasing:

$$\mathbf{E}[\kappa_T] \leq M(r_T) \leq M(\gamma(1+\log T)) = O(T^{\varepsilon}). \tag{3.10}$$

Also, by Proposition 3.5 and integral approximation we get

$$\mathbf{E}[\mathcal{E}_T] \leq C + C\int_1^T \frac{1}{\lambda(\log u)}\, du = O(T^{1-\varepsilon\frac{1}{\gamma}}). \tag{3.11}$$

□

REMARK 3.10. 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. Thus, a key element in using SOLO as an add-on to an online algorithm is to analyse the decision path length's asymptotics.

3.6. Online Submodular Maximisation Trade-offs

In this section we show that a similar set of ideas apply to the problem of online submodular function maximisation. Note that in this case, due to hardness (see Section **??**), our goal is 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).

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 $\frac{1}{2}$-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.

DR-Submodular case. We can use our result to devise a consistent online DR-submodular maximization algorithm, using the simple fact that our regret analysis is additive with respect to $\mathcal{E}_t$. Namely, if we add $\sum_{t=1}^{T} f_t(x_t) - \sum_{t=1}^{T} f_t(y_t)$ to (2.8), we get the notion of $\frac{1}{2}$-regret for the sequence (y_t). The rest of the analysis follows immediately: by being consistent, one suffers an extra regret of $O(T^{1-\frac{\varepsilon}{2}})$ in expectation, while updating at most $O(T^{\frac{\varepsilon+1}{2}})$ times.

Monotone Submodular Functions. We now relate the DR-Submodular result to the discrete case, where the utility functions are submodular set functions and the optimization domain is a matroid $\mathcal{M}$.

It is known (see Calinescu et al. (2011)) that for any non-negative monotone submodular function $F : 2^V \to \mathbb{R}_+$, where V is a d-element set, the *multilinear extension* $f : [0,1]^d \to \mathbb{R}_+$ defined as

is DR-submodular. Also for a monotone submodular function F with marginal values in $[0,1]$ and multilinear extension f, one can *round* a continuous point to a set: Let $\mathcal{X}$ be the matroid base polytope of $\mathcal{M}$, i.e. the convex hull of all the indicator vectors of bases of $\mathcal{M}$. Given a point $x \in \mathcal{X}$, Swap-Rounding Chekuri et al. (2009) returns a random element of $\mathcal{M}$, say $R = \text{round}_{\mathcal{M}}(x)$, such that $\mathbb{E}[F(R)] \geq f(x)$.

We now take the online stochastic setting, and also assume that in each round, the player can evaluate the multilinear extension of the submodular utility, as well as its gradients. With a few modifications of the consistent OGD algorithm, one can design an algorithm for this discrete case, as described in Algorithm 3.

It is clear that the same bounds for consistency cost in the case of DR-Submodular functions holds for this algorithm as well. To provide the regret bounds, we first take the expectation w.r.t. the distribution induced by rounding, and then, w.r.t. the probability of updating. By the property of Swap-Rounding, for any sequence $y_1, \ldots, y_T \in \mathcal{X}$ we have

$$\mathbb{E}_{S_t}[F_t(S_t)] \geq f_t(y_t)$$

Algorithm 3 Consistent Online Submodular Maximization

$x_1 \leftarrow$ a point in $\mathcal{X}$.
$S_1 \leftarrow$ rounded set of x_1.
$r_1 \leftarrow 0$.
for $t \in [T]$ **do**
  Play S_t and receive loss $F_t(S_t)$.
  $x_{t+1} \leftarrow \text{Proj}_{\mathcal{X}}(x_t + \eta_t \nabla f_t(x_t))$
  $r_{t+1} \leftarrow r_t + \|x_{t+1} - x_t\| + \sqrt{t+1} - \sqrt{t}$
  With probability $1 - \exp(-(r_{t+1}^{1+\varepsilon} - r_t^{1+\varepsilon}))$
    Update: set $S_{t+1} \leftarrow \text{round}_{\mathcal{M}}(x_t)$.
  Otherwise set $S_{t+1} \leftarrow S_t$.
end for

Also, since sum of submodular functions is submodular and the multilinear extension is an extension, we have

$$\max_{x^\star \in \mathcal{X}} \sum_{t=1}^{T} f_t(x^\star) \geq \max_{S^\star \in \mathcal{M}} \sum_{t=1}^{T} F_t(S^\star).$$

Therefore, by summing the inequalities we can get

$$\frac{1}{2} \max_{x^\star \in \mathcal{X}} \sum_{t=1}^{T} f_t(x^\star) - \sum_{t=1}^{T} f_t(y_t) \geq \mathbb{E}\left\{ \frac{1}{2} \max_{S^\star \in \mathcal{M}} \sum_{t=1}^{T} F_t(S^\star) - \sum_{t=1}^{T} F_t(S_t) \right\},$$

which means that the expected value of $\frac{1}{2}$-regret for the discrete setting is upper bounded by the $\frac{1}{2}$-regret of the DR-Submodular setting. Taking expectation w.r.t. the probabilities of updates, gives the desired result.

Remark 3.11. Note that we did not use the fact that the initial online algorithm has no $\frac{1}{2}$-regret. If the initial algorithm is in general a no β-regret algorithm, with a decision path length of $O(\sqrt{T})$, one gets the same result.

CHAPTER 4

Regularisation

4.1. Banach Spaces

4.2. Regularisation in Online Learning

4.3. Consistent RFTL

4.4. Example: OGD Revisited

4.5. Example: Experts Advice

Todo: talk about the experts problem and negative entropy regularisation...

Let Δ denote the n-simplex. Assume the sequence of functions $(f_t)_{t\in[T]}$ with $f_t\colon \Delta \to \mathbb{R}$ are Lipschitz continuous with respect to 1-norm, that is, for all $x \in \Delta$, we have $\|\nabla f(x)\|_\infty \leq G_\infty$, for some $G_\infty > 0$.

The Exponentiated Gradient algorithm, gives sublinear regret, as stated in the following theorem:

THEOREM 4.1. *Choose x_1 to be the center of the simplex. Then, at round t play x_t and update*

$$x_{t+1,i} = \frac{1}{Z} \cdot x_{t,i} \cdot e^{-\eta(\nabla f_t(x_t))_i},$$

where Z is a normalising constant such that $x_{t+1} \in \Delta$. Playing such, with $\eta = \frac{1}{G_\infty}\sqrt{\frac{\log d}{T}}$ gives

$$\text{regret}_T \leq 2G_\infty \sqrt{T \log d}.$$

We now note that the points x_t does not move that fast in Δ. By Pinsker's inequality, we have

$$2\|x_{t+1} - x_t\|_1^2 \leq \text{KL}(x_{t+1}\|x_t).$$

Thus, it suffices to bound the KL divergence between x_{t+1} and x_t. For simplicity, set $\nabla_t := \nabla f_t(x_t)$. We have

$$\begin{aligned}
\text{KL}(x_{t+1}\|x_t) &= \sum_{i=1}^{d} x_t^i \log \frac{x_t^i}{x_{t+1}^i} \\
&= \sum_{i=1}^{d} x_t^i \cdot (\log x_t^i + \log Z - \log x_t^i + \eta \nabla_t^i) \\
&= \log Z + \eta \left\langle x_t, \nabla_t \right\rangle .
\end{aligned}$$

LEMMA 4.2. *If $\eta \leq 1/G_\infty$, it holds that*

$$\log Z \leq -\eta \left\langle x_t, \nabla_t \right\rangle + \eta^2 G_\infty^2.$$

PROOF. Using the fact that for $x \leq 1$ it holds $e^x \leq 1 + x + x^2$, we get

$$\begin{aligned}\log Z &= \log \sum_{i=1}^{d} x_t^i \cdot e^{-\eta \nabla_t^i} \\ &\leq \log \sum_{i=1}^{d} x_t^i \cdot (1 - \eta \nabla_t^i + \eta^2 (\nabla_t^i)^2) \\ &\leq \log(1 - \eta \langle x_t, \nabla_t \rangle + \eta^2 G_\infty^2) \\ &\leq -\eta \langle x_t, \nabla_t \rangle + \eta^2 G_\infty^2\end{aligned}$$

□

Now by the lemma we get our desired result. In summary, we have the following.

PROPOSITION 4.3. *Let $(x_t)_{t \leq T}$ denote the sequence played by the EG algorithm. It holds*

$$\|x_{t+1} - x_t\|_1 \leq \frac{1}{\sqrt{2}} \eta \, G_\infty.$$

Note that here, η just depends on the time horizon T (and hence, lacks an asymptotic characterisation). For overcoming this issue, we apply the "doubling trick": we can apply such trick, as the regret is of $O(\sqrt{T})$. By applying the doubling trick to EG algorithm and denoting by η_t the learning rate at round t, we get $\eta_t = \Theta(1/\sqrt{t})$.

Hence, we are in the place to apply our consistent meta algorithm to EG. Note that the path lenght (in 1-norm) is bounded above by $O(\sqrt{T})$, thus the following trade-off exists for each $0 < \varepsilon < 1$:

$$\mathbf{E}(\text{regret}_T) = O(T^{1-\frac{\varepsilon}{2}}), \quad \mathbf{E}(\kappa_T) = O(T^{\frac{1}{2}+\frac{\varepsilon}{2}}).$$

This result, however, is sub-optimal, as compared to Altschuler and Talwar (2018).

CHAPTER 5

Experimental Results

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 $\frac{1}{2}$-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 $\log \mathcal{E}_t / \log t$

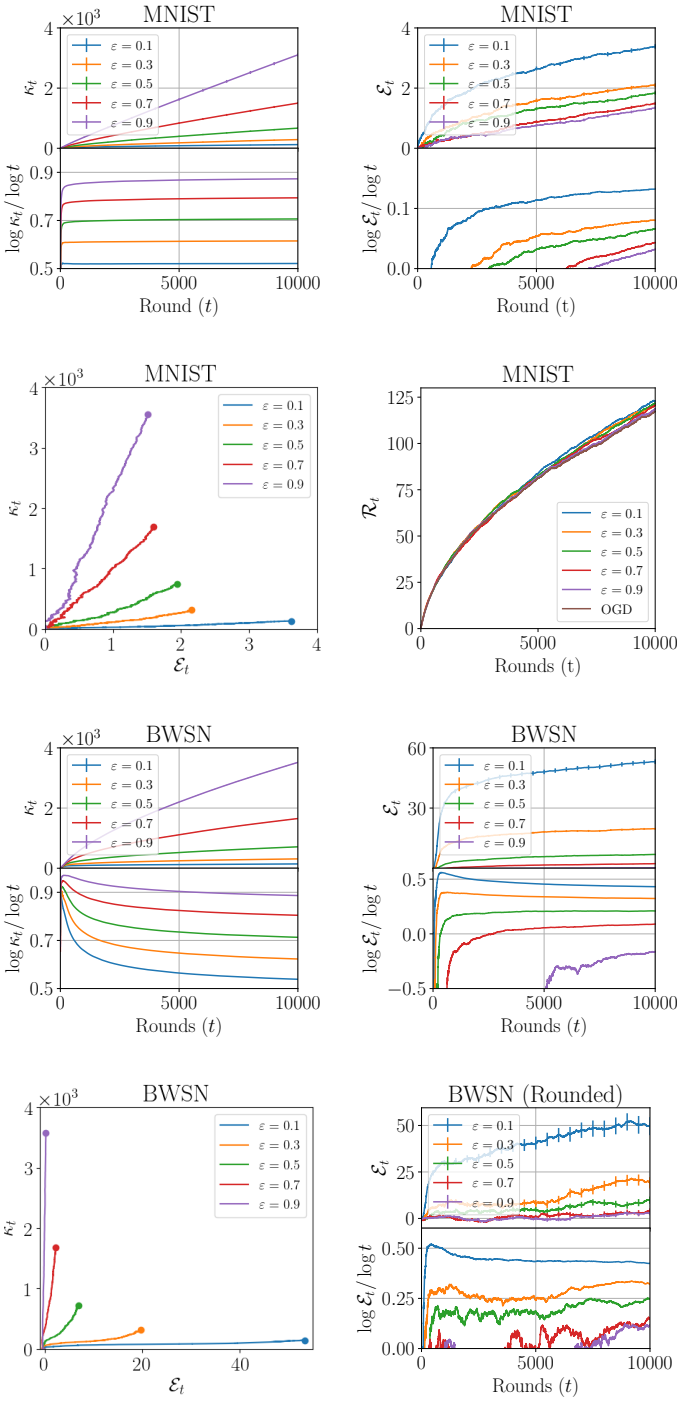

FIGURE 1. Top row are experimental results for Online Convex Optimization on MNIST dataset. The bottom row are experimental results for Online Submodular Maximization for the BWSN problem.

. Observe that if $x(t) = O(t^\alpha)$ then $\limsup_{t\to\infty} \frac{\log x(t)}{\log t} \leq \alpha$. This allows us to estimate how close our empirical results are to the theoretical bounds.

Results. In the top row of Figure 1 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 $T^{\frac{1}{2}+\frac{\epsilon}{2}}$. 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 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 1 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 Section 3.6, 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.

CHAPTER 6

Conclusion

APPENDIX A

Supplementary Lemmata

LEMMA A.1 (Projection Lemma). *Let X be a nonempty convex closed set in $\mathbb{R}^d$, $x \in \mathbb{R}^d$ be a point and z be any point in X. One has*

$$\| \operatorname{Proj}(x) - z\| \leq \|x - z\|.$$

PROOF. Let $y = \operatorname{Proj}(x)$. If $y = x$, then the claim is clear. So we assume that $x \notin X$. We first note that $\langle y - x, y - z\rangle \leq 0$. This is because y is the projection of x onto a convex set, and the hyperplane perpendicular to $x - y$ and passing through y separates z from x.

For $\alpha \in [0, 1]$ define $z_\alpha := z + \alpha(y - z)$. By convexity, $z_\alpha \in X$. Define

$$f(\alpha) := \|z_\alpha - x\|^2 - \|z_\alpha - y\|^2.$$

We wish to prove that $f(0) \leq 0$. As $z_1 = y$, we have $f(1) = 0$. Thus, it suffices to prove that $f'(\alpha) \leq 0$ for all $\alpha \in [0, 1]$. We have

$$f(\alpha) = \|z - x\|^2 - 2\alpha \langle y - z, x - z\rangle - (1 - 2\alpha)\|z - y\|^2,$$

and

$$f'(\alpha) = 2 \langle y - z, y - x\rangle \leq 0.$$

Hence, we are done. □

Bibliography

Altschuler, J. and Talwar, K. (2018), Online learning over a finite action set with limited switching, *in* 'Conference On Learning Theory', pp. 1569–1573.

Anava, O., Hazan, E. and Mannor, S. (2015), 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', pp. 784–792.

Andrews, M., Goemans, M. X. and Zhang, L. (1999), 'Improved bounds for on-line load balancing', *Algorithmica* **23**(4), 278–301.

Argue, C., Bubeck, S., Cohen, M. B., Gupta, A. and Lee, Y. T. (2019), A nearly-linear bound for chasing nested convex bodies, *in* 'Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms', SIAM, pp. 117–122.

Avi Ostfeld et al. (2008), '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.

Bach, F. (2016), 'Submodular functions: from discrete to continuous domains', *Mathematical Programming* pp. 1–41.

Bernstein, A., Holm, J. and Rotenberg, E. (2018), 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', pp. 947–959.

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

Bubeck, S., Cohen, M. B., Lee, J. R., Lee, Y. T. and Madry, A. (2017), 'k-server via multiscale entropic regularization', *CoRR* **abs/1711.01085**.

Calinescu, G., Chekuri, C., Pál, M. and Vondrák, J. (2011), 'Maximizing a Monotone Submodular Function Subject to a Matroid Constraint', *SIAM Journal on Computing* **40**(6).

Cesa-Bianchi, N. and Lugosi, G. (2006), *Prediction, learning, and games*, Cambridge university press.

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

Chen, L., Harshaw, C., Hassani, H. and Karbasi, A. (2018), Projection-free online optimization with stochastic gradient: From convexity to submodularity, *in* 'International Conference on Machine Learning', pp. 813–822.

Chen, L., Hassani, H. and Karbasi, A. (2018), Online continuous submodular maximization, *in* 'Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics', Vol. 84, PMLR, pp. 1896–1905.

Epstein, L. and Levin, A. (2014), 'Robust algorithms for preemptive scheduling', *Algorithmica* **69**(1), 26–57.

Fujishige, S. (2005), *Submodular functions and optimization*, Vol. 58, Elsevier.

Geulen, S., Vöcking, B. and Winkler, M. (2010), 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', pp. 132–143.

Gu, A., Gupta, A. and Kumar, A. (2016), 'The power of deferral: Maintaining a constant-competitive steiner tree online', *SIAM J. Comput.* **45**(1), 1–28.

Gupta, A., Krishnaswamy, R., Kumar, A. and Panigrahi, D. (2017), 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', pp. 537–550.

Gupta, A. and Kumar, A. (2014), 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', pp. 455–467.

Gupta, A., Kumar, A. and Stein, C. (2014), 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', pp. 468–479.

György, A. and Neu, G. (2014), 'Near-optimal rates for limited-delay universal lossy source coding', *IEEE Trans. Information Theory* **60**(5), 2823–2834.

Hazan, E. (2015), 'Introduction to online convex optimization', *Foundations and Trends® in Optimization* **2**(3-4), 157–325.

Hazan, E., Agarwal, A. and Kale, S. (2007), 'Logarithmic regret algorithms for online convex optimization', *Machine Learning* **69**(2-3), 169–192.

Imase, M. and Waxman, B. M. (1991), 'Dynamic steiner tree problem', *SIAM J. Discrete Math.* **4**(3), 369–384.

Karimi, M. R., Krause, A., Lattanzi, S. and Vassilvitskii, S. (2019), Consistent online optimization: Convex and submodular, *in* 'Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics', PMLR.

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

Kempe, D., Kleinberg, J. and Tardos, E. (2003), 'Maximizing the spread of influence through a social network', *9th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* pp. 137–146.

Kingman, J. F. C. (1992), *Poisson processes*, Vol. 3, Clarendon Press.

Krause, A. and Golovin, D. (2012), 'Submodular function maximization', *Tractability: Practical Approaches to Hard Problems* **3**(19), 8.

Krause, A. and Guestrin, C. (2005), Near-optimal nonmyopic value of information in graphical models, *in* 'Conference on Uncertainty in Artificial Intelligence (UAI)'.

Lacki, J., Ocwieja, J., Pilipczuk, M., Sankowski, P. and Zych, A. (2015), 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', pp. 11–20.

Lattanzi, S. and Vassilvitskii, S. (2017), Consistent k-clustering, *in* 'Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017', pp. 1975–1984.

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

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

Lin, H. and Bilmes, J. (2011), A class of submodular functions for document summarization, *in* 'Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1', Association for Computational Linguistics, pp. 510–520.

Lovász, L. (1983), Submodular functions and convexity, *in* 'Mathematical Programming The State of the Art', Springer, pp. 235–257.

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

Megow, N., Skutella, M., Verschae, J. and Wiese, A. (2016), 'The power of recourse for online MST and TSP', *SIAM J. Comput.* **45**(3), 859–880.

Merhav, N., Ordentlich, E., Seroussi, G. and Weinberger, M. J. (2002), 'On sequential strategies for loss functions with memory', *IEEE Trans. Information Theory* **48**(7), 1947–1958.

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

Phillips, S. J. and Westbrook, J. (1998), 'On-line load balancing and network flow', *Algorithmica* **21**(3), 245–261.

Sanders, P., Sivadasan, N. and Skutella, M. (2009), 'Online scheduling with bounded migration', *Math. Oper. Res.* **34**(2), 481–498.

Skutella, M. and Verschae, J. (2010), 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', pp. 36–47.

Soma, T., Kakimura, N., Inaba, K. and Kawarabayashi, K.-i. (2014), Optimal budget allocation: Theoretical guarantee and efficient algorithm, *in* 'International Conference on Machine Learning', pp. 351–359.

Westbrook, J. (2000), 'Load balancing for response time', *J. Algorithms* **35**(1), 1–16.

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

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

Authored by (in block letters):
For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.