Online Optimization with Switching Cost

Minghong Lin Adam Wierman
CMS, California Institute of Technology

Alan Roytman Adam Meyerson
Dept. of Computer Science, UCLA

Lachlan L.H. Andrew
Faculty of ICT, Swinburne University of Technology

ABSTRACT

We consider algorithms for "smoothed online convex optimization (SOCO)" problems. SOCO is a variant of the class of "online convex optimization (OCO)" problems that is strongly related to the class of "metrical task systems", each of which have been studied extensively. Prior literature on these problems has focused on two performance metrics: regret and competitive ratio. There exist known algorithms with sublinear regret and known algorithms with constant competitive ratios; however no known algorithms achieve both. In this paper, we show that this is due to a fundamental incompatibility between regret and the competitive ratio – no algorithm (deterministic or randomized) can achieve sublinear regret and a constant competitive ratio, even in the case when the objective functions are linear.

1. INTRODUCTION

In an *online convex optimization (OCO)* problem, a learner interacts with an environment in a sequence of rounds. During each round t: (i) the learner must choose an action x^t from a convex decision space F; (ii) the environment reveals a cost convex function c^t, and (iii) the learner experiences cost $c^t(x^t)$.[1] The goal is to design learning algorithms that minimize the cost experiences over a (long) horizon of T rounds. OCO has a wide array of applications, e.g., portfolio management [1] and network routing [2]; and a significant literature of work developing and analyzing algorithms that can effectively learn in this environment, e.g., [3–5].

In this paper, we study a generalization of online convex optimization that we term *Smoothed Online Convex Optimization (SOCO)*. The only change in a smoothed online convex optimization compared to an online convex optimization is that the cost experienced by the learner is now $c^t(x^t) + \|x^t - x^{t-1}\|$, where $\|\cdot\|$ is an arbitrary norm. That is, the learner experiences an additional "smoothing cost" or "switching cost" associated with changing the action.

By adding switching costs to online convex optimization, SOCO captures a wide variety of important applications. In fact, a strong motivation to study SOCO is the recent development in dynamic control algorithms for data centers [6,7], where the goal is to dynamically control the number and placement of active servers (x^t) in order to minimize a combination of the delay and energy costs (captured by c^t) and the switch costs involved in cycling servers into power saving modes and migrating data ($\|x^t - x^{t-1}\|$). Other applications of recent interest include video streaming [8], where the encoding quality of a video needs (x^t) to change dynamically in response to network congestion but where large changes in encoding quality ($\|x^t - x^{t-1}\|$) are visually annoying to users, optical networking, in which there is a cost to reestablishing a light path [9], and the dynamic dispatching of electricity generators, where in addition to the time-varying operating costs of the generators there are significant 'setup' and 'ramping' costs associated with changing the electricity output (x^t) [10].

More traditionally, within the learning community, many applications typically modeled using OCO have, in reality, some cost associated with a change of action; and so may be better modeled using SOCO than OCO. For example, OCO encodes the so-called 'k-expert' problem, which has many applications where switching costs can be important, e.g., in stock portfolio management there is a cost associated with adjusting the stock portfolio owned. In fact, 'switching costs' have long been considered important in related learning problems, such as the k-armed bandit problem which has a considerable literature studying algorithms that can learn effectively despite switching costs [11, 12]. Thus, it is quite natural to also consider the addition of switching costs to OCO.

2. SMOOTHED ONLINE CONVEX OPTIMIZATION

Though the smoothed online convex optimization problems, precisely, have not been studied previously (to the best of our knowledge), there are two large literatures that have studied formulations that are very related to SOCO: (i) the online convex optimization (OCO) literature within the online learning community, e.g., [4, 5], and (ii) the metrical task system (MTS) literature within the online algorithms community, e.g., [13, 14].

Online convex optimization (OCO) is very similar to SOCO except that the cost incurred by the learner does not include a switching cost. In particular, during each round t the following happen in order: (i) the learner must choose an action x^t from a convex decision space F; (ii) the environment reveals a convex cost function c^t, and (iii) the learner experiences cost $c^t(x^t)$. The goal is to design learning algorithms that minimize the cost experiences over a (long) horizon of T rounds. Following the treatment of [4, 5], we assume that the decision space F is non-empty, bounded and

[1] Depending on the setting, (i) may happen before (ii), or vice versa.

closed, and that the Euclidean norm of gradients $\|\nabla c^t(\cdot)\|_2$ are also bounded.

Now let us consider SOCO in the OCO setting. We have the following sequence of environment revelations and learner decisions: $x^1, c^1, x^2, c^2, \ldots$, i.e., round t consists of x^t followed by c^t, the learner incurs cost $c^t(x^t) + \|x^t - x^{t-1}\|$, where $\|\cdot\|$ is a norm on $\mathbb{R}^n$. We define the cost of an algorithm A in a SOCO under OCO setting as

$$Cost_1(A) = \mathbb{E}\left[\sum_{t=1}^{T} c^t(x^t) + \|x^t - x^{t-1}\|\right].$$

where $x^1, \ldots, x^t$ are the decisions of the algorithm, $x^0 = 0$ without loss of generality, and the expectation is over any randomness used by the algorithm.

In the OCO setting, the goal of the learning algorithm is usually to minimize the *regret*, which is the difference between the cost of the algorithm and the cost of the offline optimal static solution. Formally, Algorithm A has regret $\gamma(T)$ if for any sequence of cost functions $c_1, \ldots, c_T$,

$$Cost_1(A) - \min_{\hat{x}\in F} \sum_{t=1}^{T} c^t(\hat{x}) \le \gamma(T) \tag{1}$$

A number of algorithms have been shown to provide sublinear regret for OCO. An important example of a sublinear regret learning algorithms is *online gradient descent* (OGD), which is parameterized by learning rates η_t. OGD works as follows.

Algorithm 1 (Online Gradient Descent, OGD). *Select arbitrary $x^1 \in F$. In time step $t \ge 1$, select $x^{t+1} = P(x^t - \eta_t \nabla c^t(x^t))$, where $P(y) = \arg\min_{x\in F} \|x - y\|_2$ is the projection under the Euclidean norm.*

By choosing the learning rates η_t carefully, OGD is able to achieve sub-linear regret for OCO. In particular, the variation in [4] uses $\eta_t = \Theta(1/\sqrt{t})$ and obtains $O(\sqrt{T})$-regret. Other variations are also possible. For example, [5] achieves a tighter regret bound of $O(\log T)$ by choosing $\eta_t = \Theta(1/t)$ under additional assumptions.

To extend the literature discussed above from OCO to SOCO, we need to track the switching costs incurred by the algorithms. Interestingly, the choices of η_t used by the algorithms designed for OCO also turn out to be good choices to control the switching costs of the algorithms. In particular, all of the classical gradient decent learning algorithms are still sub-linear regret for SOCO.

Corollary 1. *Consider an online gradient decent algorithm A with learning rates such that $\sum_{t=1}^{T} \eta_t = O(\gamma_1(T))$. If $Regret(A) = O(\gamma_2(T))$ for online convex optimization problems, then $Regret(A) = O(\gamma_1(T) + \gamma_2(T))$ for smoothed online convex optimization problems.*

An immediate consequence of Corollary 1 is that the online gradient decent algorithms in [4,5], which use $\eta_t = 1/\sqrt{t}$ and $\eta_t = \Theta(1/t)$, are still $O(\sqrt{T})$-regret and $O(\log T)$-regret respectively under SOCO.

Now let us consider metrical task system (MTS). A MTS is a more general variant of a SOCO, with a few slight changes. In particular, during each round the following happen in order: (i) the environment reveals a cost function c^{t-1}, (i) the learner must choose an action x^t from a decision space F; and (iii) the learner experiences cost $c^{t-1}(x^t) + d(x_{t-1}, x_t)$ for some metric $d(\cdot)$. The key differences with the class of OCO problems are (a) the environment reveals the cost function before the learner choose action, (b) the cost functions c_t are usually not assumed to be convex, (c) the decision space is typically assumed to be discrete, and (d) a switching cost metric is included in the cost incurred by the learner.

In the MTS setting, i.e., round t consists of c^{t-1} followed by x^t, the learner then incurs cost $c^{t-1}(x^t) + \|x^t - x^{t-1}\|$. We define the cost of an algorithm A in a SOCO under MTS setting as

$$Cost_2(A) = \mathbb{E}\left[\sum_{t=1}^{T} c^{t-1}(x^t) + \|x^t - x^{t-1}\|\right].$$

where $x_0 = 0$, $c^0(\cdot) = 0$ without loss of generality.

The learning problem seems "easier" under MTS setting in the sense that the cost function for a round are known *before* the action must be decided. However, the performance metric studied in the MTS literature is much "stronger" than regret. In particular, algorithms are evaluated based on their *competitive ratio*, which is the ratio between the cost of the algorithm and the cost of the offline optimal (dynamic) solution. Formally, an algorithm A is $\alpha(T)$-competitive, if for any sequence of cost functions $c_1, \ldots, c_T$,

$$Cost_2(A) \le \alpha \min_{\hat{x}^t} \sum_{t=1}^{T} \left(c^{t-1}(\hat{x}^t) + \|\hat{x}^t - \hat{x}^{t-1}\|\right) + O(1) \tag{2}$$

For the classical MTS problem, i.e., discrete decision space without convexity assumption, most results tend to be "negative". In particular, it has been proven that, given an arbitrary metric space, any deterministic algorithm must be $\Omega(n)$-competitive [13], where n is the number of states in the decision space. Further, any randomized algorithm must be $\Omega(\sqrt{\log n / \log\log n})$-competitive [15]. However, recently, results have shown that positive results are possible in SOCO setting with L1 normed space. Specifically, in the case when x^t is scalar, there exists an algorithm termed Lazy Capacity Provisioning (LCP) that is 3-competitive [6]. More generally, when x^t is not scalar an algorithm termed Averaging Fixed Horizon Control (AFHC) is $1 + O(\|1\|_\infty / w)$-competitive [7], using exact predictions of the cost functions in the next w rounds.

3. THE INCOMPATIBILITY OF REGRET AND THE COMPETITIVE RATIO

To this point, we have discussed regret and the competitive ratio independently. However, it is clear that one would ideally want to have online learning algorithms for SOCO with strong guarantees on both metrics. Specifically, we want an algorithm that is as good as the optimal static policy and nearly as good as the optimal dynamic policy.

Interestingly, it is easy to see that none of the algorithms we have discussed to this point achieve both goals. For example, though online gradient descent has sub-linear regret, its competitive ratio is infinite. Similarly, though LCP is 3-competitive, it has linear regret.

It turns out that this is no accident. We show below that the two metrics are incompatible. That is, any algorithm that has sub-linear regret *necessarily* has an infinite competitive ratio; and any algorithm that has a constant competitive ratio *necessarily* has linear regret. More formally,

we have the following:

Theorem 2. *Given an online algorithm A and an arbitrary constant $\gamma > 0$, there exists a sequence of cost functions defining a smoothed convex optimization problem on which $CR(A) + Regret(A)/T \geq \gamma$.*

The impact of this result can be stark. For example, consider an application where static control is currently being used. To justify the use of dynamic control, one would want an algorithm that is guaranteed to perform better than the best static algorithm, i.e., a sublinear regret algorithm. However, given that one is using dynamic control, one would want an algorithm that is always close to the optimal, i.e., has a constant competitive ratio.

An important remark about Theorem 2 is that the proof uses linear cost functions and a one-dimensional decision space. This highlights, that the incompatibility does not result from complexity in the cost functions or decision space, rather it is fundamental to the performance metrics.

4. BALANCING REGRET AND THE COMPETITIVE RATIO

Given the incompatibility of regret and the competitive ratio highlighted by Theorem 2, it is necessary to reevaluate the goals for algorithm design. For example, it may be enough to ensure ϵT-regret for arbitrarily small constant ϵ considered good though it is not sub-linear regret. Or, it may be enough to ensure that the algorithm is $\log \log T$-competitive even though this is not a constant guarantee.

In this section, we present an algorithm named Random Bias Greedy (RBG), which is motivated by an algorithm in [16], that can allow such tradeoffs between regret and the competitive ratio in the case when the decision space F is one-dimensional.

RBG works as follows:

Algorithm 2 (Random Bias Greedy (RBG)). *Define $w^0(x) = 0$ for all x and $w^t(x) = \min_y\{w^{t-1}(y) + c^t(y) + \|x - y\|\}$. Generate a random number $r \in (-\|1\|, \|1\|)$. For each time step t, go to the state x^t which minimizes $Y^t(x^t) = w^t(x^t) + rx^t$.*

Note that the algorithm makes use of randomness, but only in a very limited way – it parameterizes its "bias" using a random number r. Given this bias, the algorithm works by choosing actions to minimize its "work function" $w^t(x)$.

As stated above, RBG performs well for the competitive ratio, but not for regret. In particular, it is 2-competitive but has linear regret. However, one can achieve a balance between regret and the competitive ratio by parametizing the algorithm with the "wrong" norm. More specifically, recall that, the absolute value is the "only" norm on $\mathbb{R}$ in the sense that for every norm $\|\cdot\|$ on $\mathbb{R}$, $\|x\| = \|1\||x|$. However, the the norms differ in magnitude depending on the value of $\|1\|$. We take advantage of this to use different parameterizations of $\|1\|$ to "weigh" the switching cost differently. In particular, the performance of the algorithm can be bounded as follows.

Theorem 3. *Given a smoothed convex optimization problem over a one-dimensional normed space with $\|1\|_a$. Running RBG using a one-dimensional normed space with $\|1\|_b \geq \|1\|_a$, achieves a competitive ratio of $(1 + \|1\|_b/\|1\|_a)$ and $O(\max\{T/\|1\|_b, \|1\|_b\})$ regret.*

Note that if $\|1\|_b = \|1\|_a$ then we obtain that RBG is 2-competitive and has linear regret. However, if one wishes to obtain ϵT-regret, for an arbitrarily small ϵ, one can choose $\|1\|_b = 1/\epsilon$. Similarly, if one wishes to obtain sublinear regret, choosing $\|1\|_b = g(T)$ for some slowly increasing function achieves an $O(g(T))$ competitive ratio and a regret of $O(T/g(T))$. Note that because of the max operator, the optimal regret achievable by RBG is $O(\sqrt{T})$, which is equal to the optimal regret in online learning setting [4].

5. REFERENCES

[1] T. M. Cover, "Universal portfolios," *Mathematical Finance*, vol. 1, no. 1, pp. 1–29, 1991.

[2] N. Bansal, A. Blum, S. Chawla, and A. Meyerson, "Online oblivious routing," in *Proceedings of ACM symposium on Parallel algorithms and architectures (SPAA)*. ACM, 2003.

[3] A. Kalai and S. Vempala, "Efficient algorithms for universal portfolios," in *Proceedings of the 41st Annual Symposium on Foundations of Computer Science (FOCS)*, 2000.

[4] M. Zinkevich, "Online convex programming and generalized infinitesimal gradient ascent," 2003.

[5] E. Hazan, A. Agarwal, and S. Kale, "Logarithmic regret algorithms for online convex optimization," *Mach. Learn.*, vol. 69, pp. 169–192, December 2007.

[6] M. Lin, A. Wierman, L. L. H. Andrew, and E. Thereska, "Dynamic right-sizing for power-proportional data centers," in *Proc. INFOCOM*, 2011.

[7] M. Lin, Z. Liu, A. Wierman, and L. L. Andrew, "Online algorithms for geographical load balancing," in *International Green Computing Conference(IGCC)*, 2012.

[8] V. Joseph and G. de Veciana, "Variability aware network utility maximization," *CoRR*, vol. abs/1111.3728, 2011.

[9] Y. Zhang, M. Murata, H. Takagi, and Y. Ji, "Traffic-based reconfiguration for logical topologies in large-scale wdm optical networks," *Journal of lightwave technology*, vol. 23, no. 10, p. 2854, 2005.

[10] S. Kaplan, "Power plants: Characteristics and costs," *Congressional Research Service*, 2008.

[11] M. Asawa and D. Teneketzis, "Multi-armed bandits with switching penalties," *Automatic Control, IEEE Transactions on*, vol. 41, no. 3, pp. 328 –348, mar 1996.

[12] S. Guha and K. Munagala, "Multi-armed bandits with metric switching costs," in *Automata, Languages and Programming*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2009, vol. 5556, pp. 496–507.

[13] A. Borodin, N. Linial, and M. Saks, "An optimal on-line algorithm for metrical task system," *J. ACM*, vol. 39, no. 4, pp. 745–763, 1992.

[14] M. Manasse, L. McGeoch, and D. Sleator, "Competitive algorithms for on-line problems," in *Proc. ACM symposium on Theory of computing (STOC)*, 1988, pp. 322–333.

[15] A. Blum, H. Karloff, Y. Rabani, and M. Saks, "A decomposition theorem and bounds for randomized server problems," in *Foundations of Computer Science, 1992. Proceedings., 33rd Annual Symposium on*, oct 1992, pp. 197 –207.

[16] A. Coté, A. Meyerson, and L. Poplawski, "Randomized k-server on hierarchical binary trees," in *Proceedings of the 40th annual ACM Symposium on Theory of Computing (STOC)*, 2008.