CS 7545: Machine Learning Theory May 4, 2023

Projection-free Online Learning: A Review

Instructor: Jake Abernethy *Aditya Vikram*

1 Abstract

A major computational bottleneck for Online Convex Optimization (OCO) algorithms like Online Gradient Descent (OGD) is the projection step. Two approaches to circumvent that bottleneck are discussed in this paper, and they are compared along their assumptions on environment, running times and regret bounds.

2 Introduction

In online convex optimization over a domain $\mathcal{K}$ with algorithms involving the gradient, the gradient update might take us outside $\mathcal{K}$. So the gradient step is generally followed by a projection step to make the next iterate feasible. This projection onto $\mathcal{K}$ involves solving a ℓ_2-distance optimization over points in $\mathcal{K}$, generally equivalent to a Convex Quadratic Program (QCP). Solving QCPs usually requires SVD or some factorization of the matrix. For high-dimensional data (which is the norm these days), this slows down the learning process since this projection is needed for every iteration of the OCO algorithm. Two main approaches have been developed to work around these heavy projection steps.

The first approach uses the Frank-Wolfe technique (Frank and Wolfe (1956)) in the online setting. The Online Frank-Wolfe (OFW) algorithm was proposed and analyzed in Hazan and Kale (2012), and builds upon the work of Clarkson (2010), Hazan (2008) and Jaggi (2011). The OFW algorithm replaces projection with a linear optimization step instead and the next iterate stays in the domain $\mathcal{K}$ by design. Solving a linear optimization over $\mathcal{K}$ tends to be significantly faster than QCP, which gives us the speedup required. Hazan and Kale show that we can get sublinear regret bounds even by replacing the gradient and projection step with a linear optimization.

The second approach approximates the projection step for smooth convex sets, instead of replacing it. Building upon the work by Mahdavi et al. (2012), Levy and Krause (2019) propose the Fast Approximate Projection (FAstProj) algorithm. Limiting the domain to smooth convex sets, this algorithm uses gradients and geometry to approximate the projection step, while still guaranteeing sublinear regret bounds.

In this paper, we review the settings that these two approaches work in, describe the algorithms and the ideas that go into their analysis, and compare the regret bounds and running times of these alternate approaches.

3 Online Frank-Wolfe Algorithm

The OFW setting is equivalent to the general OGD setting. Given a convex compact set $\mathcal{K} \subset \mathbb{R}^n$, the learner produces points x_t for rounds $t = 1, \cdots, T$. At each round, the environment responds with convex functions $f_t : \mathcal{K} \to \mathbb{R}$ and the learner suffers $f_t(x_t)$. The goal of the learner is to minimize regret:

$$\text{Regret}_T := \sum_{t=1}^{T} f_t(x_t) - min_{x\in\mathcal{K}} \sum_{t=1}^{T} f_t(x)$$

the difference of total loss suffered to the optimal loss with a fixed point in hindsight. We aim to get sublinear ($o(T)$) regret bounds for the algorithm. Furthermore, we assume bounds on the diameter of $\mathcal{K}$: $\forall x, y \in \mathcal{K}, \|x - y\|_2 \leq D$; and that the cost functions f_t at each round are L-lipschitz. Another requirement for OFW is the availability of a linear minimization oracle: given a direction $c \in \mathbb{R}^n$, returns $x^* = \arg\min_{x \in \mathcal{K}}\{c^\top x\}$.

3.a Idea

The basic idea is same as the Frank-Wolfe algorithm: given a convex differentiable function f, first we find the minimizer x^* over $\mathcal{K}$ of the linear approximation of f given by the Taylor approximation and get the next iterate:

$$x^* = \arg\min_{x \in \mathcal{K}}\{\nabla f(x) \cdot x\} \qquad \text{and} \qquad x^{(k+1)} \leftarrow (1-\alpha)x^{(k)} + \alpha x^*$$

with an appropriate $\alpha \in [0,1]$. By definition of convexity of $\mathcal{K}$, the next iterate $x^{(k+1)}$ remains in $\mathcal{K}$ since both $x^{(k)}, x^* \in \mathcal{K}$. The convergence rate depends on an appropriate choice of the step parameter α.

In the OFW algorithm, this convex function we minimize is the cumulative average loss suffered by the learner. Define $F_t(x) := \frac{1}{t}\sum_{\tau=1}^{t} f_\tau(x)$. Note that at $t = T$, F_t is the sum term in definition of the average regret: $\frac{\text{Regret}_T}{T}$. So we can think of OFW algorithm as an approximate-greedy algorithm to minimize regret because at each round, we try to minimize the linear approximation of the average regret accumulated.

3.b Algorithm

Algorithm 1: OFW Algorithm (Hazan and Kale (2012))

```
  Input: constant $a \geq 0$
1 Initialize $x_1 \in \mathcal{K}$ arbitrarily;
2 for $t \leftarrow 1$ to $T$ do
3     Play $x_t$ and observe $f_t$;
4     Compute $F_t(x) := \frac{1}{t}\sum_{\tau=1}^{t} f_\tau(x)$;
5     Compute $v_t \leftarrow \arg\min_{x \in \mathcal{K}}\{\nabla F_t(x_t) \cdot x\}$;
6     Set $x_{t+1} \leftarrow (1 - t^{-a})\, x_t + t^{-a} v_t$;
7 end
```

As discussed, we need a linear minimization oracle for K, which is used in step 5 of Algorithm 1 to find the minimizer along the gradient of F_t. Also note that as t increases, we reduce the step size $\alpha = t^{-a}$ and $\alpha \in (0,1]$ for all $a \geq 0$.

3.c Runtime implications

Before discussing the regret bounds, let's look at the implications of running time on two online learning problems. For the online shortest path problem (Awerbuch and Kleinberg (2008)) on a DAG with n nodes, m edges and given source and sink vertices s and t, the domain $\mathcal{K}$ is a polytope in $\mathbb{R}^m$ specified with $O(m+n)$ inequalities corresponding to the set of all unit flows from s to t. The QCP for finding projection of a point onto $\mathcal{K}$ can be solved most efficiently in $O(n^{3.5})$ time, while the linear optimization oracle required in Algorithm 1 is equivalent to finding the shortest $s-t$ path in a DAG and takes $O(m+n)$ time using dynamic programming.

Another common problem in recommendation systems is online collaborative filtering, for which the domain $\mathcal{K}$ is the set of all $m \times n$ matrices with bounded trace norm. If we had to project a matrix X onto $\mathcal{K}$, it requires computing the SVD of X which takes $O(mn\min(m,n))$ time. The linear optimization oracle required here requires the computation of top singular vectors of the matrix defining the objective $C^\top X$, which can be typically done in $O(nnz(C))$, linear in the number of non-zero entries in C.

These two examples show that OFW can reduce the running time per iteration of the online learning algorithm by orders of magnitude. Next we look at the regret bounds for OFW, and see how they compare with OGD.

3.d Regret bounds

With $\Delta_t := F_t(x_t) - \min_{x\in\mathcal{K}} F_t(x)$, we have the following main theorem from Hazan and Kale (2012) that gives all the regret bounds with varying parameters.

Theorem 3.1. *For $t = 1, \cdots, T$, for some constants $B, S \geq 0$, $b \in [-1, 1/2]$ and $s \in [0, 1)$, let the cost function at each round f_t be Bt^{-b}-smooth and St^{-s}-strongly convex, in addition to being L-lipschitz. Then Algorithm 1 with input $a = d - b$ guarantees $\Delta_t \leq Ct^{-d}$ for*

$$(C, d) = \left(\max\left\{9D^2B, 3LD\right\}, \frac{1+b}{2}\right) \quad \textit{and} \quad (C, d) = \left(\max\left\{9D^2B, 3LD, \frac{36L^2}{S}\right\}, \frac{2+2b-s}{3}\right)$$

Here the two values of (C, d) depend on whether f_t is St^{-s}-strongly convex or not. The proof can be done by induction on t and uses the (strong) convexity, smoothness and lipschitzness of f_t at each step. We focus next on the regret bounds that result from Theorem 3.1.

Theorem 3.2. *For adversarial convex cost functions f_t, there is an algorithm with the following regret bound:*

$$\textit{Regret}_T \leq 57LDT^{3/4}$$

Proof Sketch. The proof of this theorem relies on defining a smooth strongly-convex function $\hat{f}_t$ using the gradient of f_t as:

$$\hat{f}_t(x) := \nabla f_t(x_t) \cdot x + \frac{L}{D} t^{-1/4} \|x - x_1\|^2$$

We run Algorithm 1 with the modified cost functions $\hat{f}_t$. It can be shown that $\hat{f}_t$ is $3L$-lipschitz, $\frac{L}{D}t^{-1/4}$-smooth and $\frac{L}{D}t^{-1/4}$-strongly convex. Substituting the constants B, S, b and s in Theorem 3.1, we get that

$$\hat{F}_t(x_t) - \min_{x\in\mathcal{K}} \hat{F}_t(x) \leq 36LDt^{-3/4}$$

At any round t, if $x_t^* = \arg\min_{x\in\mathcal{K}} \hat{F}_t(x)$, we have that $\sum_{t=1}^T \hat{f}_t(x_t^*) \leq \sum_{t=1}^T \hat{f}_t(x^*)$ for any $x^* \in \mathcal{K}$ because x_t^* at each round minimizes the cumulative sum by definition. Using the fact that $\hat{f}_t$ is $3L$-lipschitz and $\frac{L}{D}t^{-1/4}$-strongly convex, we have

$$\hat{f}_t(x_t) - \hat{f}_t(x_t^*) \leq 3L\|x_t - x_t^*\| \leq 3L\sqrt{\frac{D(\hat{F}_t(x_t) - \hat{F}_t(x_t^*))}{Lt^{-1/4}}} = 18LDt^{-1/4}$$

Finally we can substitute the definition of $\hat{f}_t$ and use the convexity of f_t and the upper bound $\sum_{t=1}^T t^{-1/4} \leq 3T^{3/4}$ to get the regret bound. ■

Thus, the OFW algorithm has an $O(T^{3/4})$ regret bound for adversarial online learning with general convex cost functions. This is worse than the OGD algorithm which has a regret bound of $O(T^{1/2})$ under the same settings. However, we save significant running time per iteration when compared to the projection step, as discussed in section 3.c. OFW algorithm is also useful in cases when the projection is much harder to compute compared to a linear minimization oracle. This could be the case when $\mathcal{K}$ can't be written succinctly but linear optimization could still be possible, as is the case with the online learning of rotations problem (Hazan et al. (2010)).

4 FAstProj Algorithm

The FAstProj algorithm proposed by Levy and Krause approximates the projection step instead of replacing it. Firstly, we restrict the domain $\mathcal{K}$ to be a smooth convex set. Specifically, we focus on sets of the form $\mathcal{K} = \{x \in \mathbb{R}^n : h(x) \leq 0\}$ where $h : \mathbb{R}^n \to \mathbb{R}$ is a smooth convex function, and we have access to the evaluation and gradient oracles of h. In addition, we also restrict ourselves to sets with an upper bound on curvature. Here curvature at any boundary point $x \in \partial\mathcal{K}$ is defined as

$\mu_x := \beta_h / \|\nabla h(x)\|$ where we assume that h is β_h-smooth (β_h could be 0). Additionally, the global curvature is the upper bound on the curvature at all boundary points: $\mu = \max_{x \in \partial\mathcal{K}} \mu_x$ and we assume that μ is upper-bounded for $\mathcal{K}$. With these assumptions on the setting, we look at the idea next.

4.a Idea and Algorithm

Instead of solving the QCP to find the projection of a given point, we approximate the projection using the gradient and geometry of $\mathcal{K}$. Assume we're given a point $x \in \mathcal{K}$ and a vector $v \in \mathbb{R}^n$, where we need to find the projection of $x + v$ onto $\mathcal{K}$. This is usually the case in OGD, where we have $x = x_t$ and $v = -\eta_t \nabla f_t(x_t)$. Figure 1 from Levy and Krause (2019) is shown on the right for reference.

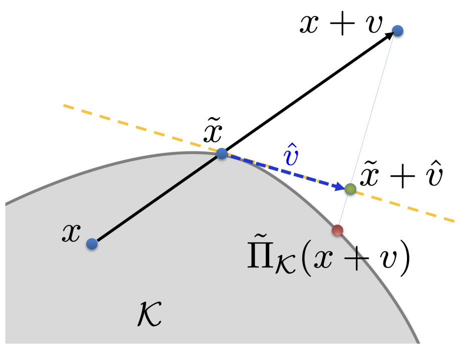

Figure 1: Approximate projection (Levy and Krause (2019))

Instead of the actual projection, we first find the point $\tilde{x}$ where the segment $(x, x+v]$ intersects with the boundary of K. Define $\tilde{v} := x + v - \tilde{x}$ and the unit normal at $\tilde{x}$ as $n = \nabla h(\tilde{x}) / \|\nabla h(\tilde{x})\|$. Then compute $\hat{v}$ (blue dashed line in Figure 1) by projecting the vector $\tilde{v}$ onto the subspace orthogonal to n: $\hat{v} = \tilde{v} - (n \cdot \tilde{v})n$ (for intuition, this subspace would be the hyperplane that separates $\tilde{x} + \epsilon n$ from $\mathcal{K}$ for $\epsilon \to 0$).

By the convexity of $\mathcal{K}$ and how the above vectors are defined, the point $\tilde{x} + \hat{v}$ is closer to $x + v$ than any point in K. For feasibility though, we need a point in K, so we trace the ray $\{\tilde{x} + \hat{v} - \alpha n : \alpha \geq 0\}$ until it intersects the boundary $\partial\mathcal{K}$, and that is our approximate projection point $\tilde{\Pi}_{\mathcal{K}}(x + v)$. Theorem 4.1 discussed later shows that when v is small enough, the approximate projection distance guarantee is close to the accurate projection guarantee.

Algorithm 2: FAstProj Algorithm (Levy and Krause (2019))

```
  Input  : constant $a \geq 0$
  Output: approximate projection of x+v onto $\mathcal{K}$: $\tilde{\Pi}_{\mathcal{K}}(x+v)$
1 if $x + v \in \mathcal{K}$ then return $x + v$ ;
2 else
3     Find $\tilde{x} \in (x, x+v] \cap \partial\mathcal{K}$, define $\tilde{v} \leftarrow x + v - \tilde{x}$;
4     Compute $n = \nabla h(\tilde{x}) / \|\nabla h(\tilde{x})\|$ and define $\hat{v} = \tilde{v} - (n \cdot \tilde{v})n$;
5     Compute $\alpha = \min\{\alpha \geq 0 : \tilde{x} + \hat{v} - \alpha n \in \mathcal{K}\}$;
6     return $\tilde{x} + \hat{v} - \alpha n$
7 end
```

The approximate projection algorithm described above outputs a point "close" (within $\mu \|v\|^2$-additive error) to the actual projection, as seen in the following theorem. To implement Algorithm 2, there are two tricky parts:

- finding $\tilde{x}$: this can be done using a binary search along the line segment $(x, x+v]$ for $h(x) = 0$. Since it is not possible to get the accurate result, we can get an ϵ-close result within $O(\log(1/\epsilon))$ evaluation oracle calls for h.
- finding α: If we knew some α' such that the point $y = \tilde{x} + \hat{v} - \alpha' n$ lies in $\mathcal{K}$, then we can again use the above bisection technique for $\alpha \in [0, \alpha')$ to get ϵ-close to the boundary $\partial\mathcal{K}$ using $O(\log(1/\epsilon))$ evaluation oracle calls for h. To get that initial point y, Levy and Krause propose solving a convex optimization problem using convex bisection, which gives an initial point along the ray inside $\mathcal{K}$ within $O(\log(1 + \|v\|))$ gradient oracle calls for h.

Using $\epsilon = O(\|v\|^2)$ still works for Theorem 4.1, so we need a total of $O(\log(1/\|v\|) + \log(1 + \|v\|))$ oracle calls to implement the approximate projection step.

4.b Analysis

Theorem 4.1. *Let $\mathcal{K}$ be a β_h-smooth convex set with diameter D and global curvature μ. Then Algorithm 2 with $x \in \mathcal{K}$ and $v \in \mathbb{R}^n$ such that $\|v\| \leq 0.5/\mu$ outputs $\tilde{\Pi}_{\mathcal{K}}(x + v) \in \mathcal{K}$ such that*

$$\forall y \in \mathcal{K} : \left\|\tilde{\Pi}_{\mathcal{K}}(x+v) - y\right\| \leq \|x + v - y\| + \mu \|v\|^2$$

In the interest of time to typeset, I'm skipping the details of the proof for this theorem. As an overview, we can use the β_h-smoothness of $\mathcal{K}$, the geometry of points and various norm triangle-inequalities to prove the above theorem. The important thing to take from this theorem is that the requirement of projections in OGD: $\forall y \in \mathcal{K} : \|\Pi_{\mathcal{K}}(x + v) - y\| \leq \|x + v - y\|$ is only off by an additive factor of $\mu \|v\|^2$. This helps us derive the following regret bounds.

Theorem 4.2. *Let $\mathcal{K}$ be a β_h-smooth convex set with diameter D and global curvature μ. Then OGD with the projection step replaced by Algorithm 2 and $\eta_t = \frac{D}{G(1+2\mu D)\sqrt{t}}$ has regret bounded by*

$$\mathit{Regret}_T \leq O\left((1 + \mu D)\, GD\sqrt{T}\right)$$

For H-strongly convex loss functions f_t, the same algorithm with $\eta_t = \frac{1}{2\mu G + Ht}$ has the bound

$$\mathit{Regret}_T \leq O\left(\frac{G^2(1 + \mu D)}{H} \log T\right)$$

The proof of both these upper bounds is similar to the proof of classic OGD regret bound along with the following inequality that follows directly from Theorem 4.1:

$$\forall x \in \mathcal{K} : \left\|\tilde{\Pi}_{\mathcal{K}}(x_t - \eta_t \nabla f(x_t)) - x\right\|^2 \leq \|(x_t - \eta_t \nabla f(x_t)) - x\|^2 + \mu^2 \eta_t^4 G^4 + (2\mu D + 1)\eta_t^2 g^2$$

Thus, using faster approximate projections, under the conditions mentioned above, we can get a $O(\sqrt{T})$ regret bound for convex loss functions and $O(\log T)$ regret bound for strongly-convex loss functions. Compared to the classic projected OGD, these are worse by just a multiplicative factor.

5 OFW vs FAstProj

The following table summarizes the assumptions on the domain and regret bounds we've seen for the OCO algorithms in this paper:

Algorithm	Assumptions on $\mathcal{K}$	Regret bound for convex f_t	Regret bound for strongly convex f_t
Projected OGD	Convex, bounded with access to a projection oracle	$O(\sqrt{T})$	$O(\log T)$
OFW	Convex, bounded with access to a linear minimization oracle	$O(T^{3/4})$	$O(T^{3/4})$
FAstProj	Convex, smooth and bounded	$O(\sqrt{T})$	$O(\log T)$

The OFW algorithm operates under the fairly general assumptions of a bounded convex domain $\mathcal{K}$ with access to a linear minimization (over $\mathcal{K}$) oracle. However the regret bounds it achieves are suboptimal compared to the classical OGD algorithm. FAstProj makes a slightly stricter assumption of smoothness on the domain $\mathcal{K}$ but gets the same order of regret bounds as classical OGD. To get the same regret as OGD, we might have to run the FAstProj algorithm a constant times more, but the running time per iteration of FAstProj makes it worth it.

Specifically comparing along running times per iteration, both OFW and FAstProj beat classical OGD by avoiding the projection step, albeit with radically different approaches while doing so. OFW reduces the running time from solving a QCP to solving a linear optimization problem over $\mathcal{K}$, while FAstProj reduces it to binary searches along line segments.

While there are cases like collaborative filtering and online shortest path problem where each step of OFW significantly beats OGD, there are also scenarios where the linear minimization might be as expensive as the QCP. For example, for the case where the domain $\mathcal{K}$ is defined by quadratic constraints: $\mathcal{K} = \left\{x \in \mathbb{R}^n : x^\top Ax \leq b\right\}$, both projection as well as linear optimization require factorization of the matrix A. Thus OFW and OGD both take $O(n^3 + n^2T)$ operations. However in this case, FAstProj has a running time of $O(n^2T \log T)$.

Thus, when it comes to choosing which projection-free online algorithm to pick, the main deciding criteria might be whether the domain is a smooth convex set or not. Since FAstProj has better regret bounds than OFW and also runs faster iterations in most scenarios, FAstProj would be a better pick. If the domain of the online learning problem is not smooth, then we would need to experiment and check whether the running time improvements from OFW are worth the suboptimal regret bounds it provides.

Additionally, a recent work, Hazan and Minasyan (2020) proposes a faster randomized projection-free online algorithm with $O(T^{2/3})$ regret bounds. It builds upon the "Follow-the-Perturbed-Leader" (FPL) online algorithm proposed by Kalai and Vempala. I read the abstract and the basic algorithm ideas, but didn't have enough time to go through the paper. So it is difficult to compare how it would fare against FAstProj in terms of running time, but it is more efficient than the OFW algorithm.

References

B. Awerbuch and R. Kleinberg. Online linear optimization and adaptive routing. *J. Comput. Syst. Sci.*, 74(1):97–114, feb 2008. ISSN 0022-0000. doi: 10.1016/j.jcss.2007.04.016. URL `https://doi.org/10.1016/j.jcss.2007.04.016`.

K. L. Clarkson. Coresets, sparse greedy approximation, and the frank-wolfe algorithm. *ACM Trans. Algorithms*, 6(4), sep 2010. ISSN 1549-6325. doi: 10.1145/1824777.1824783. URL `https://doi.org/10.1145/1824777.1824783`.

M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3(1-2):95–110, 1956. URL `https://EconPapers.repec.org/RePEc:wly:navlog:v:3:y:1956:i:1-2:p:95-110`.

E. Hazan. Sparse approximate solutions to semidefinite programs. In E. S. Laber, C. Bornstein, L. T. Nogueira, and L. Faria, editors, *LATIN 2008: Theoretical Informatics*, pages 306–316, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. ISBN 978-3-540-78773-0.

E. Hazan and S. Kale. Projection-free online learning. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, ICML'12, page 1843–1850, Madison, WI, USA, 2012. Omnipress. ISBN 9781450312851.

E. Hazan and E. Minasyan. Faster projection-free online learning. In J. Abernethy and S. Agarwal, editors, *Proceedings of Thirty Third Conference on Learning Theory*, volume 125 of *Proceedings of Machine Learning Research*, pages 1877–1893. PMLR, 09–12 Jul 2020. URL `https://proceedings.mlr.press/v125/hazan20a.html`.

E. Hazan, S. Kale, and M. K. Warmuth. Learning rotations with little regret. In *COLT*, pages 144–154. Citeseer, 2010.

M. Jaggi. Convex optimization without projection steps. *arXiv preprint arXiv:1108.1170*, 2011.

A. Kalai and S. Vempala. Efficient algorithms for online decision problems. *J. Comput. Syst. Sci.*, 71(3):291–307, oct 2005. ISSN 0022-0000. doi: 10.1016/j.jcss.2004.10.016. URL https://doi.org/10.1016/j.jcss.2004.10.016.

K. Levy and A. Krause. Projection free online learning over smooth sets. In K. Chaudhuri and M. Sugiyama, editors, *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pages 1458–1466. PMLR, 16–18 Apr 2019. URL https://proceedings.mlr.press/v89/levy19a.html.

M. Mahdavi, T. Yang, R. Jin, S. Zhu, and J. Yi. Stochastic gradient descent with only one projection. In F. Pereira, C. Burges, L. Bottou, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL https://proceedings.neurips.cc/paper_files/paper/2012/file/c52f1bd66cc19d05628bd8bf27af3ad6-Paper.pdf.