ALGORITHMS FOR ONLINE CONVEX PROGRAMMING

AHMED JAMOUSSI

1 Introduction

Convex programming is a generalization of linear programming and has many application in machine learning nowadays, such as finding a hypothesis h in a hypothesis space H that minimizes some cost function or maximizes a utility function. A convex programming model requires a feasible convex set $F \subseteq R^n$ and a convex cost function $c : F \to R$ and its goal is to find a point $x \in F$ that minimizes c. It also has many other application, such as network routing problems, nonlinear facility location problems and consumer optimization problem.

In online convex programming an algorithm faces a sequence of convex programming problems. At each time-step the algorithm must output a point in F that minimizes a cost function before observing it. Formally, at each iteration $t \in \{0, 1, ..., T\}$ (T is the unknown number of iterations) the algorithm chooses a point $x_t \in F$. Once the choice is made the cost function $c_t : F \to R$ is revealed and the cost induced is $c_t(x_t)$. The acknowledgment of the algorithm falls into two classes: in the "bandit" model, only the loss $c_t(x_t)$ itself is observable, whereas in the full information model the whole cost functions c_t are given to the algorithm. We are mostly interested in the full information model. This method of programming models various engineering and scientific problems in which the decision has to be made before knowing the values or the costs, such as industrial production for example.
In this project we describe and discuss basic tools and algorithms used in recent framework of online convex programming.

We give a few definitions that will be used throughout the paper.

Definition 1.1. *A set $S \subseteq \mathbb{R}^n$ is **convex** if for all $x, y \in S$ and $\lambda \in (0, 1)$, we have $\lambda x + (1 - \lambda)y \in S$.*

Definition 1.2. *For a convex set S, a function $f : S \to \mathbb{R}$ is said to be **convex** if for all $x, y \in S$ and $\lambda \in (0, 1)$*

$$f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y)$$

*f is said to be σ-**strongly convex** if for all $x, y \in S$ and $\lambda \in (0, 1)$*

$$f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y) - \frac{\sigma}{2}\lambda(1 - \lambda) \|x - y\|^2$$

Definition 1.3. *An **online convex programming problem** consists of a convex set $F \subseteq \mathbb{R}^n$ and an infinite sequence $\{c_1, c_2, \dots\}$ where each $c_t : F \to \mathbb{R}$ is a convex function.*

At each step t, an online convex programming algorithm selects a vector $x_t \in F$, and then receives the cost function c_t. As the cost function is revealed after the vector x_t is chosen, there is no hope of finding the minimum of each function. Instead, algorithms often try to minimize the notion of regret, which is defined as follows.

Definition 1.4. *Given an algorithm A, and a convex programming problem $(F, \{c_1, c_2, \dots\})$, if $\{x_1, x_2, \dots\}$ are the vectors selected by A, then the **regret** of A until time T is*

$$r_A(T) := \sum_{t=1}^{T} c_t(x_t) - \min_{x \in F} \sum_{t=1}^{T} c_t(x)$$

Intuitively, the regret compares the online algorithm to the best possible choice for an offline algorithm. We are interested in finding bounds on the regret as small as possible that guarantees that

$$\lim_{T \to \infty} \frac{r_A(T)}{T} = 0$$

Having such a property would imply that algorithm A performs, on average, as well as the best offline algorithm.

Assumption 1.5. *Throughout this paper the following will be assumed:*

- *The feasible set F is **convex**.*
- *The feasible set F is **bounded**: There exists $M \in \mathbb{R}$ such that for all $x, y \in F, \quad d(x, y) \leq M$.*
- *The feasible set F is **closed**: If $\{x_n\}_{n=1}^{\infty}$ is a sequence in F that converges to $x^* \in \mathbb{R}^n$, then $x^* \in F$.*
- *The feasible set F is **non-empty**: $F \neq \emptyset$.*
- *There exists $N \in \mathbb{R}^n$ such that for all t and for all $x \in F$, we have $||\nabla c_t(x)|| \leq N$.*
- *For all t, c_t is continuously differentiable.*
- *For all t, there exists an algorithm, given x, which produces $\nabla c_t(x)$.*
- *For all $y \in \mathbb{R}^n$, we denote the projection of y onto F as:*

$$P_F(y) = \operatorname*{argmin}_{x \in F} d(x, y)$$

Observe that $P_F(y)$ is always well-defined and unique as F is nonempty, convex and closed. However, we do not necessarily assume that the projection can be obtained easily. Sometimes projections are not computable analytically or they are too costly to obtain.

We start with a key observation that tells us that we can restrict our study to cost functions that are linear. Although we will not present most of the proofs, this a key argument used in all of them. Observe that since c_t is convex and differentiable, we have that for all $x, y \in F$,

$$c_t(x) - c_t(y) \leq \nabla c_t(x)^\top (x - y)$$

In particular, for $x^* \in \operatorname{argmin}_{x \in F} \sum_{t=1}^{T} c_t(x)$, we get that for all t,

$$c_t(x_t) - c_t(x^*) \leq \nabla c_t(x_t)^\top (x_t - x^*)$$

Thus, if we define $\hat{c}_t(x) = \nabla c_t(x_t)^T x$, we get that

$$\sum_{t=1}^{T} c_t(x_t) - c_t(x^*) \leq \sum_{t=1}^{T} \hat{c}_t(x_t) - \hat{c}_t(x^*)$$

Therefore we can simply simply study the functions $\hat{c}_t$ instead, since the left-hand side is the definition of the regret. Thus, from now on, we will only consider linear functions, i.e. we assume that that for all t, we can write

$$c_t(x) = g_t^\top x \quad \text{for some } g_t \in \mathbb{R}^n$$

Observe that we then have

$$\nabla c_t(x) = g_t \quad (x \in \mathbb{R}^n)$$

2 RFTL algorithm

The first and somewhat natural idea that comes to mind when dealing with online convex programming is the following. When at step t, choose x_t such that it minimizes the sum of the cost functions from 1 up to $t-1$. Although this approach is seducing by its simplicity and the fact that the sum of convex functions is convex, making the subproblem convex, it can lead to totally wrong results, as we saw in class.
A way to avoid that is to introduce a regularization term, which gives the classic RFTL (Regularized Follow The Leader) algorithm.

2.1 Algorithm definition

Let $R : F \to \mathbb{R}$ be a σ-strongly convex function twice continuously differentiable. We recall that a twice continuously differentiable function is σ-strongly convex if and only if the eigenvalues of its hessians are bounded away from 0 by σ. We give the full definition of RTFL in algorithm 1.

Algorithm 1 RFTL

1: Input: $\mu > 0$, strongly convex regularizer function R
2: Let $x_1 = \operatorname{argmin}_{x \in F} R(x)$
3: **for** $t = 1$ to T **do**
4: Predict x_t
5: Observe c_t
6: Update $x_{t+1} = \operatorname*{argmin}_{x \in F} \left[\mu \sum_{s=1}^{t} c_s(x) + R(x) \right]$
7: **end for**

2.2 Algorithm performance

Denote

$$\lambda = \max_{t, x \in F} g_t^\top [\nabla^2 R(x)]^{-1} g_t$$
$$D = \max_{y \in F} R(y) - R(x_1)$$

observe that λ and D depend on the regularization function $R(x)$, the convex set F and the magnitude of the cost function c_t.

Since $x_1 = \underset{x \in F}{\operatorname{argmin}} R(x)$, we can think of D as the diameter of the set F as measured by R. On the other hand, λ is the squared magnitude of c_t measured according to a derived norm from R. We get a $O(\sqrt{T})$ for this algorithm from [1].

Theorem 2.1. *The RFTL algorithm achieves the following bound on regret:*

$$r(T) \leq 2\sqrt{2\lambda DT}$$

2.3 Discussion

First observe that step 6 in Algorithm 1 is well-defined. Indeed, as R is strongly convex and c_t is convex for all t, the minimizer over F exists and is unique. The main point of interest lies in solving step 6. Luckily, it has a nice and well-studied form, namely a convex function over a convex set. To solve it, different approaches can be used. If projection onto F is available and easily computable, then a projected gradient (or even subgradient) can be used.
In the case where the c_t are twice continuously differentiable, then a projected Newton method can be used, as suggested in [3]. Such an approach can be particularly interesting as it has nice convergence properties. For example, in the case of F being a closed hyper-rectangle, then projected Newton guarantees quadratic convergence under mild assumptions, i.e. it guarantees that

$$\left\|x^{k+1} - \bar{x}\right\| \leq c \left\|x^k - \bar{x}\right\|^2$$

for $\bar{x} \in \underset{x \in F}{\operatorname{argmin}} \left[\mu \sum_{s=1}^{t} c_s(x) + R(x)\right]$ and $c \in (0,1)$. Although appealing, such an approach is often not usable in practice, as it requires the calculations of Hessians. If n and/or T are large, it becomes computationally impossible to obtain the Hessians.

If projecting is not possible, or computationally costly, then other methods are possible. A popular one is the Frank-Wolfe algorithm (and variants of it), also known as conditional gradient. The main idea of the Frank-Wolfe algorithm is the following. When minimizing a function f over a convex set F, if the current iterate is x_k, select

$$p_k \in \underset{p \in F}{\operatorname{argmin}} \nabla f(x_k)^\top p \tag{1}$$

then set $x_{k+1} = x_k + t_k(p_k - x_k)$ for some $t_k > 0$. The minimization of the linear form (1) might be easier than the projection on F.

3 The primal-dual approach

Definition 3.1. *The **Bregman divergence** with respect to R is defined as*

$$B_R(x||y) = R(x) - R(y) - (x-y)^\top \nabla R(y)$$

Observe that since R is convex, we have that for all $x, y \in F$, $B_R(x||y) \geq 0$.

3.1 Algorithm definiton

The full definition of the primal-dual approach is given in algorithm 2.

Algorithm 2 Primal-Dual

1: Input: $\mu > 0$, strongly convex regularizer function R
2: Choose y_1 such that $\nabla R(y_1) = 0$
3: Compute $x_1 = \operatorname{argmin}_{x \in F} B_R(x||y_1)$
4: **for** $t = 1$ to T **do**
5: Predict x_t
6: Observe c_t
7: Choose y_{t+1} such that

$$\nabla R(y_{t+1}) = \nabla R(y_t) - \mu \nabla c_t(x_t). \quad \text{(Lazy version)}$$
$$\nabla R(y_{t+1}) = \nabla R(x_t) - \mu \nabla c_t(x_t). \quad \text{(Active version)}$$

8: Compute

$$x_{t+1} = \operatorname*{argmin}_{x \in F} B_R(x||y_{t+1})$$

9: **end for**

3.2 Algorithm performance

In order to analyse the performance of our algorithm, we need to introduce a few notations.
For H a symmetric positive definite matrix, $\|x\|_H := \sqrt{x^\top H x}$ defines a norm on $\mathbb{R}^n$. In that case, H^{-1} is also symmetric positive definite and thus also induces a norm. We call this norm the dual norm and write $\|x\|_{H^{-1}} = \|x\|_H^*$. We then get the following convergence result from [1], assuring again a $O(\sqrt{T})$ convergence.

Theorem 3.2. *Suppose that for all $x, y \in F$, we have $B_R(x||y) \geq \frac{1}{2}\|x - y\|_H^2$. Let G_* and D such that $\|\nabla c_t(x_t)\|_H^* \leq G_*$ for all t, and $B_R(x||x_1) \leq D^2$ for all $x \in F$. Then, with $\mu = \frac{D}{2G_*\sqrt{T}}$, we get the following bound on the regret:*

$$r(T) \leq DG_*\sqrt{T}$$

Observe that the requirement $B_R(x||y) \geq \frac{1}{2}\|x - y\|^2$ always holds true if R is σ-strongly convex with $\sigma \geq 1$.

3.3 Discussion

First observe that $B_R(x||y)$ is strongly convex as a function of x, for fixed y. This is because

$$\frac{\partial^2 B_R(x||y)}{\partial x^2} = \frac{\partial^2 R(x)}{\partial x^2}$$

Because R is σ-strongly convex, the eigenvalues of $\frac{\partial^2 R(x)}{\partial x^2}$ are bounded away from 0 by σ, and thus so are the eigenvalues of $\frac{\partial^2 B_R(x||y)}{\partial x^2}$. This shows that $B_R(x||y)$ is σ-strongly convex as a function of x. Thus step 8 of Algorithm 2 is well-defined and the argmin is unique.
It is hard to say whether step 7 is easily done. It depends greatly on what the function ∇R looks like. In [1], they suggest to take $R(x) = \|x\|^2$ or $R(x) = x \log x$. In that case, the gradients have a nice form and step 7 can be solved analytically.

Interestingly enough, RTFL and the lazy primal-dual approach are exactly the

same in the case of linear cost functions if the μ chosen in step 1 of both algorithms are the same. Indeed, observe that in the lazy version of algorithm 2, we have

$$\begin{aligned}\nabla R(y_{t+1}) &= \nabla R(y_t) - \mu g_t \\ &= \nabla R(y_{t-1}) - \mu(g_{t-1} + g_t) \\ &\vdots \\ &= \nabla R(y_1) - \mu \sum_{s=1}^{t} g_s \\ &= -\mu \sum_{s=1}^{t} g_s\end{aligned}$$

Now, the update rule in algorithm 1 solves

$$x_{t+1}^* \in \operatorname*{argmin}_{x \in F} \left\{ \sum_{s=1}^{t} c_s(x) + \frac{1}{\mu} R(x) \right\} \tag{2}$$

In the linear case, this implies that

$$\sum_{s=1}^{t} g_s + \frac{1}{\mu} \nabla R(x_{t+1}^*) = 0$$

which we can rewrite

$$\nabla R(x_{t+1}^*) = -\mu \sum_{s=1}^{t} g_s$$

Since R is strongly convex, it is in particular strictly convex, thus the argmin in (2) is a singleton. Hence the solution to the above equation must be unique. Thus we must have $y_{t+1} = x_{t+1}^*$. We then have

$$\begin{aligned}B_R(x||y_{t+1}) &= R(x) - R(y_{t+1}) - \nabla R(y_{t+1})^\top (x - y_{t+1}) \\ &= R(x) - R(y_{t+1}) + \mu \sum_{s=1}^{t} g_s^\top (x - y_{t+1}) \\ &= R(x) + \mu \sum_{s=1}^{t} g_s^\top x - R(y_{t+1}) - \mu \sum_{s=1}^{t} g_s^\top y_{t+1}\end{aligned}$$

Observe that the last two terms do not depend on x, hence the update rule of algorithm 2 comes down to

$$\begin{aligned}x_{t+1} = \operatorname*{argmin}_{x \in F} B_R(x||y_{t+1}) &= \operatorname*{argmin}_{x \in F} \left\{ R(x) + \mu \sum_{s=1}^{t} g_s^\top x \right\} \\ &= \operatorname*{argmin}_{x \in F} \left\{ \sum_{s=1}^{t} c_s(x) + \frac{1}{\mu} R(x) \right\}\end{aligned}$$

Hence the updates are the same as in (2) and that shows equivalence of the two algorithms in the linear case.

4 Online Gradient Descent

4.1 Algorithm definition

The full definition of the online gradient descent algorithm is given in algorithm 3.

Algorithm 3 Online Gradient Descent

1: Input: set F
2: Choose $x_1 \in F$.
3: **for** $t = 1$ to T **do**
4: Predict x_t
5: Observe c_t
6: Choose a step-size $\mu_t > 0$ and compute y_{t+1} as

$$y_{t+1} = x_t - \mu_t \nabla c_t(x_t)$$

7: Compute

$$x_{t+1} = P_F(y_{t+1})$$

8: **end for**

4.2 Algorithm performance

We provide two convergence results for Algorithm 3, the first one is due to [2] and the second is due to [1]. Both are interesting as the first one has weaker assumptions, but the second one achieves a better convergence rate.
We denote

$$\|\nabla c\| := \max_{t=1,\ldots,T} \|\nabla c_t(x_t)\|$$

and

$$\|F\| := \max_{x,y \in F} \|x - y\|$$

Observe that $\|\nabla c\|$ exists by our assumption in section 1, and $\|F\|$ exists as F is compact and the norm function is continuous.

Theorem 4.1. *If $\mu_t = t^{-\frac{1}{2}}$, we get the following bound on the regret:*

$$r(T) \leq \frac{\|F\|^2 \sqrt{T}}{2} + \left(\sqrt{T} - \frac{1}{2}\right) \|\nabla c\|^2$$

As with previous the theorems, we obtain a $O(\sqrt{T})$ bound for the regret. The following theorem gives a better bound, under stronger assumptions.

Theorem 4.2. *Suppose that c_t is σ-strongly convex for all t. Then, with $\mu_t = \frac{1}{\sigma t}$, we get the following bound on the regret:*

$$r(T) \leq \frac{\|\nabla c\|^2}{2\sigma}(1 + \log T)$$

We give a quick argument as to why strong convexity allows for a better bound. First, define

$$x^* \in \operatorname*{argmin}_{x \in F} \sum_{t=1}^{T} c_t(x)$$

Then we have

$$r(T) = \sum_{t=1}^{T} c_t(x_t) - \sum_{t=1}^{T} c_t(x^*)$$

Now, the way the proofs proceed is by trying to bound $c_t(x_t) - c_t(x^*)$. It is not hard to prove, using simple projection results and basic algebra, that

$$\nabla c_t(x_t)^\top (x_t - x^*) \leq \frac{1}{2\mu_t}\big(\|x_t - x^*\|^2 - \|x_{t+1} - x^*\|^2 \big) + \frac{\mu_t}{2} \|\nabla c\|^2 \tag{3}$$

Now, if we only assume that the c_t are convex, then we get the bound

$$c_t(x_t) - c_t(x^*) \leq \nabla c_t(x_t)^\top (x_t - x^*) \tag{4}$$

Summing from 1 to T, using (3) and (4), we get that

$$r(T) \leq \sum_{t=1}^{T} \frac{1}{2\mu_t}\big(\|x_t - x^*\|^2 - \|x_{t+1} - x^*\|^2 \big) + \frac{\|\nabla c\|^2}{2} \sum_{t=1}^{T} \mu_t$$

A few simple manipulations then show that

$$r(T) \leq \|F\|^2 \frac{1}{2\mu_T} + \frac{\|\nabla c\|^2}{2} \sum_{t=1}^{T} \mu_t$$

Setting $\mu_t = t^{-1/2}$ yields the results in Theorem 4.1, as the sum can be bounded from above using an integral.
If, in addition, the c_t are σ-strongly convex, then the bound in (4) can be improved to

$$c_t(x_t) - c_t(x^*) \leq \nabla c_t(x_t)^\top (x_t - x^*) - \frac{\sigma}{2} \|x_t - x^*\|^2 \tag{5}$$

Summing from 1 to T and using (3) and (5), we get that

$$r(T) \leq \frac{1}{2}\left((\frac{1}{\mu_1} - \sigma) \|x_1 - x^*\|^2 + \sum_{t=2}^{T} (\frac{1}{\mu_t} - \frac{1}{\mu_{t-1}} - \sigma) \|x_t - x^*\|^2 \right) + \frac{\|\nabla c\|^2}{2} \sum_{t=1}^{T} \mu_t$$

Setting $\mu_t = \frac{1}{\sigma t}$ cancels out the first term and leaves us with

$$r(T) \leq \frac{\|\nabla c\|^2}{2} \sum_{t=1}^{T} \frac{1}{\sigma t}$$

which can be easily bounded using an integral, thus giving the result in Theorem 4.2.
It is interesting to note that under the assumptions of Theorem 4.2, the FTL algorithm can be used and achieves the same regret bound, as seen in [4].

4.3 Discussion

This approach is appealing because of its simplicity. First, it does not require a regularization term, something that might not always be easy to choose. It is simply a gradient step followed by a projection on a convex set. If the projection is available and easily computable, then this approach should probably be preferred as it guarantees the same convergence rate as the two methods presented earlier, and does not require to solve a minimization subproblem. If in addition, the cost functions are strongly convex, then this projected gradient method is extremely interesting as it guarantees a $O(\log T)$ convergence, a major improvement compared

to $O(\sqrt{T})$. It is also interesting to note that to obtain the $O(\sqrt{T})$ rate from Theorem 4.1, it is not even necessary that the cost functions be differentiable. Indeed, the algorithm would still work perfectly fine with subgradients, provided they are bounded on F.

5 Conclusion

In this project we investigated 3 different algorithms to solve online convex programming problems, namely RFTL, primal-dual and online gradient descent. For each algorithm, we gave its performance and discussed how it would be interesting to use it, depending on the structure of the problem (i.e. the feasible set, and the form of the cost functions).
Overall, we enjoyed this project as it allowed us to explore a branch of convex optimization unknown to us, and to learn common techniques used in optimization.
As a future work, we would like to explore in more detail the bandit model, where only the cost $c_t(x_t)$ is revealed, as we believe it is more representative of some problems encountered in the industry.

References

[1] E. Hazan: *The convex optimization approach to regret minimization*, in S. Sra, S. Nowozin, and S. Wright, editors, Optimization for Machine Learning, 287-303, MIT press (2011).
[2] M. Zinkevich: *Online convex programming and generalized infinitesimal gradient ascent*, Proceedings of the Twentieth International Conference (ICML), 928-936 (2003).
[3] M. Schmidt, D. Kim and S. Sra: *Projected Newton-type methods in machine learning*, Optimization for Machine Learning, MIT Press (2011).
[4] S. Shalev-shwartz, S. Kakade: *Mind the Duality Gap: Logarithmic regret algorithms for online optimization*, Advances in Neural Information Processing Systems 21, 1457-1464 (2009).

McGill University, Department of Mathematics, Student ID : 260676547
Email address: ahmed.jamoussi@mail.mcgill.ca