Online Convex Programming and Generalized Infinitesimal Gradient Ascent (ICML 2003)

Martin Zinkevich[1]

[1]Carnegie Mellon University

Presented by Dongyun Kim (Group 11)

Table of Contents

1 Paper Review

2 Term Project Plan

Table of Contents

1. Paper Review

2. Term Project Plan

Sections of this paper

1. Introduction
2. Online Convex Programming
    1. Analyzing the Performance of the Algorithm
    2. Regret Against a Dynamic Strategy
3. Generalized Infinitesimal Gradient Ascent
    1. Repeated Games
    2. Formulating a Repeated Game as an Online Linear Program
4. Converting Old Algorithms
    1. Formal Definitions
    2. Converting an OLPA to an Online Convex Programming Algorithm
5. Related Work
6. Future Work
7. Conclusions

Sections of this paper

1. Introduction
2. Online Convex Programming
   1. Analyzing the Performance of the Algorithm
   2. Regret Against a Dynamic Strategy
3. Generalized Infinitesimal Gradient Ascent
   1. Repeated Games
   2. Formulating a Repeated Game as an Online Linear Program
4. Converting Old Algorithms
   1. Formal Definitions
   2. Converting an OLPA to an Online Convex Programming Algorithm
5. Related Work
6. Future Work
7. Conclusions

Introduction

- Convex programming consists of a convex feasible set $F \subset \mathbb{R}^n$ and a convex cost function $c : F \to \mathbb{R}$.
- Most of the methods and topics we discussed in-class were optimization for machine learning under an *offline environment*, where we have full access of the cost function, training data,..., beforehand.
- We discuss *online convex programming*, in which an algorithm faces a sequence of convex programming problems, each with the same feasible set but different cost functions.
- Each time the algorithm must choose a point before it observes the cost function.

Online Convex Programming

Definition

An **convex programming problem** consists of a convex feasible set $F \subset \mathbb{R}^n$ and a convex cost function $c : F \to \mathbb{R}$.

Online Convex Programming

Definition

An **convex programming problem** consists of a convex feasible set $F \subset \mathbb{R}^n$ and a convex cost function $c : F \rightarrow \mathbb{R}$.

Definition

An **online convex programming problem** consists of a convex feasible set $F \subset \mathbb{R}^n$ and an infinite sequence $\{c^1, c^2, ...\}$ where each $c^t : F \rightarrow \mathbb{R}$ is a convex function.

Online Convex Programming

Definition

An **convex programming problem** consists of a convex feasible set $F \subset \mathbb{R}^n$ and a convex cost function $c : F \to \mathbb{R}$.

Definition

An **online convex programming problem** consists of a convex feasible set $F \subset \mathbb{R}^n$ and an infinite sequence $\{c^1, c^2, ...\}$ where each $c^t : F \to \mathbb{R}$ is a convex function.

Definition

At each time step t, an **online convex programming algorithm** selects a vector $x^t \in F$. After the vector is selected, it receives the cost function c^t.

Online Convex Programming

- In online convex programming, all information (e.g. cost functions c^t for all time step t) is not available before decisions are made.
- Therefore, online algorithms do not reach "solutions" (e.g. minima), but instead achieve certain goals.
- A measure of performance called *regret* is considered.
- *Average regret* is the regret divided by T, the number of rounds.

Online Convex Programming

The remainder of this paper is presented under the following assumptions.

1. The feasible set F is **bounded**.
   That is, there exists $N \in \mathbb{R}$ such that $\forall x, y \in F$, $\|x - y\|_2 \leq N$.
2. The feasible F is **closed**. That is, for all sequences $\{x^1, x^2, \ldots\}$ where $x^t \in F$ for all t, if there exists a $x \in \mathbb{R}^n$ such that $x = \lim_{t \to \infty} x^t$, then $x \in F$.
3. The feasible set F is **nonempty**.
4. For all t, c^t is **differentiable**[1].

[1] We can relax this assumption in terms of the existence of subgradient as follows: Given x, there exists a vector g_x such that $c^t(y) \geq c^t(x) + g_x(y - x)$ for all y.

Online Convex Programming

The remainder of this paper is presented under the following assumptions.

5. There exists an $N \in \mathbb{R}$ such that for all t, for all $x \in F$, $\|\nabla c^t(x)\|_2 \leq N$.
6. For all t, there exists an algorithm, given x, which produces $\nabla c^t(x)$.
7. For all $y \in \mathbb{R}^n$, there exists an algorithm which can produce the projection of y onto F defined as $\text{proj}_F(y) = \arg\min_{x \in F} \|x - y\|_2$.[1]

[1] This paper uses the notation $P(y)$, but we shall use this one since we did in-class.

Online Convex Programming

Algorithm 1 (*Greedy Projection*)

Select an arbitrary $x^1 \in F$ and a sequence of learning rates $\eta_1, \eta_2, ... \in \mathbb{R}^+$. In time step t, after receiving a cost function c^t, select the next vector x^{t+1} according to:

$$x^{t+1} = \text{proj}_F(x^t - \eta_t \nabla c^t(x^t)).$$

Online Convex Programming

Algorithm 1 (*Greedy Projection*)

Select an arbitrary $x^1 \in F$ and a sequence of learning rates $\eta_1, \eta_2, ... \in \mathbb{R}^+$. In time step t, after receiving a cost function c^t, select the next vector x^{t+1} according to:

$$x^{t+1} = \text{proj}_F(x^t - \eta_t \nabla c^t(x^t)).$$

- Note that if we happened to know that cost functions c^t were actually all identical, i.e., $c^t = c$ for all time steps t, then *Greedy Projection* is exactly the same as PGD which we have learned in-class [s06-2].

Analyzing the Performance of the Algorithm

Definition

Given an algorithm A, and a convex programming problem $(F, \{c^1, c^2, ...\})$, if $\{x^1, x^2, ...,\}$ are the vectors selected by A, then the **cost** of A until time T is

$$C_A(T) = \sum_{t=1}^{T} c^t(x^t).$$

The cost of a static feasible solution $x \in F$ until time T is

$$C_x(T) = \sum_{t=1}^{T} c^t(x).$$

The **regret** of algorithm A until time T is

$$R_A(T) = C_A(T) - \min_{x \in F} C_x(T).$$

Analyzing the Performance of the Algorithm

Theorem (*Greedy Projection*'s regret)

If $\eta_t = 1/\sqrt{t}$, the regret of the Greedy Projection algorithm is

$$R_G(T) \leq \frac{\|F\|^2\sqrt{T}}{2} + (\sqrt{T} - \frac{1}{2})\|\nabla c\|_2^2$$

where $\|F\| := \max_{x,y \in F} \|x - y\|_2$ *and* $\|\nabla c\| := \max_{x \in F, t \in \{1,2,\ldots\}} \|\nabla c^t(x)\|_2$.

Analyzing the Performance of the Algorithm

Proof

- We begin with arbitrary $\{c^1, c^2, ...\}$.
  Running *Greedy Projection*, we obtain $\{x^1, x^2, ...\}$.
- Because c^t is convex, for all x:

$$c^t(x) \geq c^t(x^t) + (\nabla c^t(x^t)) \cdot (x - x^t).$$

- Set x^* to be a statically optimal vector, i.e., $x^* := \arg\min_{x \in F} C_x(T)$.
  Since $x^* \in F$, from the previous inequality we have

$$c^t(x^*) \geq c^t(x^t) + (\nabla c^t(x^t)) \cdot (x^* - x^t).$$

- subtract both sides from $c^t(x^t)$, we get

$$c^t(x^t) - c^t(x^*) \leq c^t(x^t) - \left(c^t(x^t) + (\nabla c^t(x^t)) \cdot (x^* - x^t)\right).$$

Analyzing the Performance of the Algorithm

Proof (Continued)

- Define linear function $g^t(x) := \nabla c^t(x^t) \cdot x$. If we were to change function c^t to function g^t, the behavior of the algorithm will still be the same ($\because \nabla g^t(x^t) = \nabla c^t(x^t)$).
  That is, we will select the same $\{x^1, x^2, ...\}$.
- Thus, we can rewrite the previous inequality

$$c^t(x^t) - c^t(x^*) \leq \cancel{c^t(x^t)} - \left(\cancel{c^t(x^t)} + (\nabla c^t(x^t)) \cdot (x^* - x^t)\right).$$

  as

$$c^t(x^t) - c^t(x^*) \leq g^t(x^t) - g^t(x^*) := (x^t - x^*) \cdot \nabla c^t(x^t).$$

- We will now bound the RHS of this inequality.

Analyzing the Performance of the Algorithm

Proof (Continued)

- Define for all t, $y^{t+1} := x^t - \eta_t \nabla c^t(x^t)$.
- Then, we can rewrite *Greedy Projection* as

$$x^{t+1} = \text{proj}_F(x^t - \eta_t \nabla c^t(x^t)) = \text{proj}_F(y^{t+1}).$$

- By definition of y^{t+1}, we have

$$(y^{t+1} - x^*)^2 = \left((x^t - x^*) - \eta_t \nabla c^t(x^t)\right)^2$$

$$= (x^t - x^*)^2 - 2\eta_t (x^t - x^*) \cdot \nabla c^t(x^t) + \eta_t^2 \|\nabla c^t(x^t)\|_2^2$$
$$\leq (x^t - x^*)^2 - 2\eta_t (x^t - x^*) \cdot \nabla c^t(x^t) + \eta_t^2 \|\nabla c\|_2^2.$$

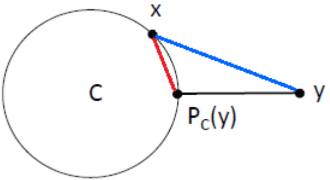
Analyzing the Performance of the Algorithm

Proof (Continued)

- Recall the following property of projection operator on convex sets, which we have discussed in-class.[1]

Property (Projection on a convex set F is contracting)

For all $y \in \mathbb{R}^n$, for all $x \in F$, $(\text{proj}_F(y) - x)^2 \leq (y - x)^2$.



2

[1] A proof is given in Schneider, R. (2013). *Convex Bodies: The Brunn–Minkowski Theory*, page 9

[2] https://wikidocs.net/22434

Analyzing the Performance of the Algorithm

Proof (Continued)

- Using the previous inequality and this property, we have,

$$(x^{t+1} - x^*)^2 = (\mathrm{proj}_F(y^{t+1}) - x^*)^2 \leq (y^{t+1} - x^*)^2$$

$$\leq (x^t - x^*)^2 - 2\eta_t(x^t - x^*) \cdot \nabla c^t(x^t) + \eta_t^2 \|\nabla c\|_2^2.$$

- Rearranging terms and dividing both sides by $2\eta_t$, we get

$$(x^t - x^*) \cdot \nabla c^t(x^t) \leq \frac{1}{2\eta_t}\left((x^t - x^*)^2 - (x^{t+1} - x^*)^2\right) + \frac{\eta_t}{2}\|\nabla c\|_2^2.$$

- We conclude the following inequality, in which taking the summation from $t = 1, ..., T$ of the LHS will give regret $R_G(T)$.

$$c^t(x^t) - c^t(x^*) \leq \frac{1}{2\eta_t}\left((x^t - x^*)^2 - (x^{t+1} - x^*)^2\right) + \frac{\eta_t}{2}\|\nabla c\|_2^2.$$

Analyzing the Performance of the Algorithm

Proof (Continued)

- By summing we get

$$
\begin{aligned}
R_G(T) &\leq \sum_{t=1}^{T} \frac{1}{2\eta_t}\left((x^t - x^*)^2 - (x^{t+1} - x^*)^2\right) \\
&\quad + \frac{\eta_t}{2}\|\nabla c\|^2 \\
&\leq \frac{1}{2\eta_1}(x^1 - x^*)^2 - \frac{1}{2\eta_T}(x^{T+1} - x^*)^2 \\
&\quad + \frac{1}{2}\sum_{t=2}^{T}\left(\frac{1}{\eta_t} - \frac{1}{\eta_{t-1}}\right)(x^t - x^*)^2 \\
&\quad + \frac{\|\nabla c\|^2}{2}\sum_{t=1}^{T}\eta_t \\
&\leq \|F\|^2\left(\frac{1}{2\eta_1} + \frac{1}{2}\sum_{t=2}^{T}\left(\frac{1}{\eta_t} - \frac{1}{\eta_{t-1}}\right)\right) \\
&\quad + \frac{\|\nabla c\|^2}{2}\sum_{t=1}^{T}\eta_t \\
&\leq \|F\|^2\frac{1}{2\eta_T} + \frac{\|\nabla c\|^2}{2}\sum_{t=1}^{T}\eta_t
\end{aligned}
$$

- Now, if we define $\eta_t = 1/\sqrt{t}$

$$
\begin{aligned}
\sum_{t=1}^{T}\eta_t &= \sum_{t=1}^{T}\frac{1}{\sqrt{t}} \\
&\leq 1 + \int_{t=1}^{T}\frac{dt}{\sqrt{t}} \\
&\leq 1 + \left[2\sqrt{t}\right]_1^T \\
&\leq 2\sqrt{T} - 1
\end{aligned}
$$

- Plugging this to the previous inequality finishes the proof. □

Analyzing the Performance of the Algorithm

Theorem

If $\eta_t = 1/\sqrt{t}$, the regret of the Greedy Projection algorithm is

$$R_G(T) \leq \frac{\|F\|^2\sqrt{T}}{2} + (\sqrt{T} - \frac{1}{2})\|\nabla c\|_2^2$$

where $\|F\| := \max_{x,y \in F} \|x - y\|_2$ *and* $\|\nabla c\| := \max_{x \in F, t \in \{1,2,\ldots\}} \|\nabla c^t(x)\|_2$.

- Therefore, the average regret of *Greedy Projection* approaches to 0

$$\limsup_{T \to \infty} \frac{R_G(T)}{T} = 0.$$

- The first term of the bound is because we might begin on the wrong side of F.
- The second part is a result of the fact that we always respond (x^{t+1}) after we see the cost function (c^t).

Lazy Projection

- This section is only in the full version of the paper.[1]

Algorithm 2 (*Lazy Projection*)

Select an arbitrary $x^1 \in F$ and a sequence of learning rates $\eta_1, \eta_2, \ldots \in \mathbb{R}^+$. Define $y^1 = x^1$. In time step t, after receiving a cost function c^t, define y^{t+1}:

$$y^{t+1} = y^t - \eta_t \nabla c^t(x^t)$$

and select the vector

$$x^{t+1} = \text{proj}_F(y^{t+1}).$$

[1] *Online convex programming and generalized infinitesimal gradient ascent* (Technical Report CMU-CS-03-110). CMU

Lazy Projection

Theorem (*Lazy Projection*'s regret)

Given a constant learning rate η, Lazy Projection's regret is

$$R_L(T) \leq \frac{\|F\|^2}{2\eta} + \frac{\eta\|\nabla c\|^2 T}{2}.$$

The proof given is in the appendix of the full version of this paper.[1]

[1] *Online convex programming and generalized infinitesimal gradient ascent* (Technical Report CMU-CS-03-110). CMU

Greedy vs. Lazy

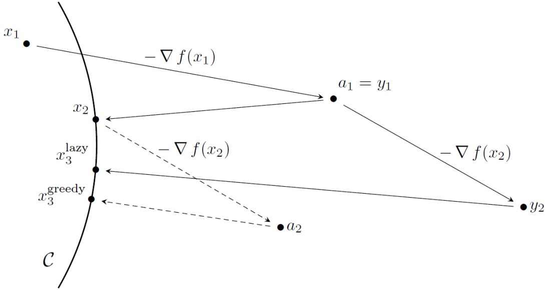


FIGURE 1. Graphical illustration of the greedy (dashed) and lazy (solid) branches of the projected subgradient (PSG) method. [1]

- *Greedy variant*: adds $-\nabla f(x_n)$ to x_n and projects back to $\mathcal{C}$ if needed.
- *Lazy variant*: the gradient term $-\nabla f(x_n)$ is **not** added to x_n, but to the "unprojected" iterate y_n. We only project to $\mathcal{C}$ in order to obtain the algorithm's next iterate.

[1] Kwon, J.,& Mertikopoulos, P. (2014). A continuous-time approach to online optimization. *Journal of Dynamics and Games*, 4(2):125–148, 2017

Conclusion and Relevant Papers

- **This paper presents an online form of the standard gradient descent from offline optimization**: Online Gradient Descent (OGD)
- This algorithm can guarantee $\mathcal{O}(\sqrt{T})$ regret for an arbitrary sequence of differentiable convex functions.
- **Note:** A sequence defined by algorithm A has "no-regret" if the regret is sublinear as a function of T, i.e., $R_A(T) = o(T)$.
- **How to improve the regret bound for strongly-convex losses?** E. Hazan, A. Agarwal, and S. Kale. Logarithmic regret algorithms for online convex optimization. *Machine Learning*, 69:169-192, 2007

Table of Contents

1 Paper Review

2 Term Project Plan

Term Project Plan

- Week 7-8: Study Online Convex Optimization (OCO) framework
- Week 9-10: Review recent papers related to OCO algorithms and its applications
- Week 11-12: Implement basic OCO algorithms via Pytorch/Tensorflow
- Week 13-14: Experiment regret convergence via datasets and apply OCO algorithms as optimizers for specific machine learning problems (e.g. SVM classification of MNIST dataset)
- Week 15: Project presentation

Term Project Plan

- Week 7-8: Study Online Convex Optimization (OCO) framework
- Week 9-10: Review recent papers related to OCO algorithms and its applications e.g. Follow-The-Leader
- Week 11-12: Implement basic OCO algorithms via Pytorch/Tensorflow
- Week 13-14: Experiment regret convergence via datasets and apply OCO algorithms as optimizers for specific machine learning problems (e.g. SVM classification of MNIST dataset)
- Week 15: Project presentation

References

Martin Zinkevich.
Online convex programming and generalized infinitesimal gradient ascent.
In *Proceedings of the 20th international conference on machine learning (ICML-2003)*, pages 928–936, 2003.

Daniel Golovin.
Lecture notes in cs 253: Advanced topics in machine learning.
http://courses.cms.caltech.edu/cs253/slides/cs253-lec3-convex.pdf.

Joon Kwon and Panayotis Mertikopoulos.
A continuous-time approach to online optimization.
Journal of Dynamics and Games, 4(2):125–148, 2017.