Implement in PyTorch the parameter-free algorithm in "Black-Box reductions for parameter-free online learning in Banach Spaces"

Ping Hu
Department of Computer Science
Stony Brook University peggyhu0315@cs.stonybrook.edu

Abstract

The abstract paragraph should be indented ½ inch (3 picas) on both the left- and right-hand margins. Use 10 point type, with a vertical spacing (leading) of 11 points. The word **Abstract** must be centered, bold, and in point size 12. Two line spaces precede the abstract. The abstract must be limited to one paragraph.

1 Online Learning

Online learning refers to the optimization problem according to a series of online input during the optimization process, which is different from the off-line optimization where all of the information in the objective function or in the problem is already known. Online learning is a well-established framework for understanding iterative optimization algorithms, including stochastic optimization algorithms or algorithms operating on large data streams. The online learning process is modeled as a sequence of consecutive optimization rounds where in each iteration $t \in T$, an online learning algorithm picks a point w_t in some space W, observes a loss function $l_t : W \to R$, and suffers loss $l_t(w_t)$. Here, the loss function describes how "bad" the optimization in the current iteration is. For the entire online learning process, the concept of "regret", R_T, is proposed to describe how good this sequence of optimization is.

$$R_T(\mathring{w}) = \sum_{t=1}^{T} l_t(w_t) - l_t(\mathring{w}) \tag{1}$$

In Eq. 1, the $\mathring{w}$ represents the solution from the learner's competitor who makes the decision after knowing the information in all of the iterations. Intuitively, the competitor's solution, $\mathring{w}$, may be better (which means less regret) than the online learner's solution in each iteration because the competitor is looking for a "global" minimum. However, it is not guaranteed that the competitor's solution achieves less loss in each iteration than the online learner's solution. The target in the online learning process is to guarantee the minimal regret which is the cumulative loss suffered along the process.

1.1 Motivation

To make the problem easier to solve, we assume W is a convex set and each l_t is convex (this is called Online Convex Optimization). The problem analogous to OLO where $\langle g_t, w_t \rangle$ is generalized to an arbitrary convex function l_t is solved through a reduction to OLO[5]. With this assumption, we can further reduce the problem to online linear optimization (OLO) in which each l_t must be a linear function. In other words, the $l_t = \langle g_t, w_t \rangle$ where $g_t \in \partial l_t(w_t)$.

Lower bounds for unconstrained online linear optimization [17; 20] imply that when l_t are L-Lipschitz, no algorithm can guarantee regret better than $\Omega(\|\mathring{w}\| L\sqrt{Tln(\|\mathring{w}\| LT + 1)})$[1]. In following sections, we assume $L = 1$.

Preprint. Work in progress.

In online convex optimization, an online player chooses a point in a convex set W. After the point is chosen, a concave payoff function is revealed, and the online player receives payoff which is the concave function applied to the point she chose. This scenario is repeated for many iterations.

1.2 The State of Art

To achieve optimal regret, most of the existing online algorithms require the user to set the learning rate (step size) η to an unknown/oracle value, $e.g$, Online Gradient Descent [5] $etc.$ Although these online optimization methods update the w_t according to the online input in an adaptive way, when put into practical optimization problem, the tuning of learning rate can lead to a tedious manual work which runs in an opposite direction against the essential motivation in optimization. Recently, several novel parameter-free algorithms have been proposed, among which, a series of research works of reducing online linear optimization problems into coin-betting problems are proposed[3, 1]. The Coin-Betting-ONS is one of the method reducing the OLO problem into the coin betting problem.

The contributions of proposed algorithm includes simplifying the design of parameter-free algorithms, proving that algorithms for online exp-concave optimization imply parameter-free algorithms for OLO, and further, it is proved that the online learning in arbitrary dimensions with any norm can be reduced into one-dimensional online learning.

2 Reduction from Online Linear Learning to Coin-Betting Algorithms

The proposed method reduces the stochastic subgradient descent procedure to betting on a number of coins. Considering a scenario of a gambler making repeated bets on the outcomes of adversarial coin flips, the gambler's goal is to earn the most of the money from his or her bets. The definitions in the coin betting process are described as follow.

Definitions The rule of coin betting is that: initially, the gambler has some initial fund, ϵ; before the gambler knows the result of the current round, the gambler makes a bet on the face of the coin as well as bet a fraction, v, of his or her owned money, w;once the coin's face is shown, the gambler will lose the corresponding fraction of the money, w, if the guess is not the same as the coin, and the gambler will win the corresponding fraction of the money, w, if the guess is the same as the coin. Therefore, according to the abstraction of coin betting in [3], the gambler's "reward" can be defined as

$$Reward_t = \sum_{t=1}^{T} -g_t w_t$$

where $w_t = v_t * Wealth_t$ and v_t is the betting fraction. v_t is a signed value, the sign represents either of the two faces of the coin. Also, we define $Wealth$ as

$$Wealth_t = \epsilon - \sum_{t=1}^{T} g_t w_t$$

where g_t represents the result of every trial of flipping the coin, $\in \{-1, 1\}$. To generalize the problem slightly, we assume $g_t \in [-1, 1]$, with $Reward$ and $Regret$ are still the same.

2.1 Online Newton Step to Online Linear Optimization via Betting Algorithms

The algorithm is shown in Algorithm 1.

In this algorithm, the loss function is $l_t(v_t) = -ln(1 - g_t v_t)$. The beauty of this algorithm lies in casting the coin-betting problem as an online learning problem whose loss function is of exp-concave. It is possible to $ln(T)$ regret when using exp-concave functions as the loss function[2], while for a linear loss function the regret is only guaranteed to be $\sqrt{T}$. The algorithm describes a coin betting strategy making use of Online Newton Step (COCOB-ONS) for each iteration's optimization. The Online Newton Step can reach regret in $O(log(T))$ for an arbitrary sequence of strictly convex functions (with bounded first and second derivatives) and exp-concave functions[2].

```
Result: betting fraction $v_t$
1  Initial $Wealth_0 = \epsilon$, initial betting fraction $v_0 = -0.5$;
2  while $t \leq T$ do
3      Bet $w_t = v_t Wealth_{t-1}$;
4      Receive $g_t \in [-1, 1]$;
5      Update $Wealth_t = Wealth_{t-1} - g_t w_t$;
6      //compute the new betting fraction $v_{t+1} \in [-\frac{1}{2}, \frac{1}{2}]$ via
7      //Online Newton Step update on the loss function $-ln(1 - g_t v_t)$;
8      Set $z_t = \frac{d}{dv_t}(-ln(1 - g_t v_t))$;
9      Set $A_t = 1 + \sum_{i=1}^{t} z^i$;
10     $v_{t+1} = \max(\min(v_t - \frac{2}{2-ln(3)} \frac{z_t}{A_t}), \frac{1}{2}, -\frac{1}{2})$
11 end
```

Algorithm 1: Coin-Betting through ONS

3 Implementation and Analysis

The algorithm is implemented in $Numpy$ and $PyTorch$. Specifically, the algorithm for 1D function is implemented in $Numpy$ and the algorithm for nD function is implemented in $PyTorch$ (can also be used in 1D functions).

Both versions can be found in `https://github.com/Ping-Hu/COCOB-Implementation`

In addition, the compared online algorithm, AdaGrad, is also implemented which is included in the git repository as well.

This repository will still be updated after the report deadline because I probably may find some bugs in my code. To be of integrity, please refer to the commit before the deadline.

3.1 COCOB-ONS Evaluation in 1D non-smooth function

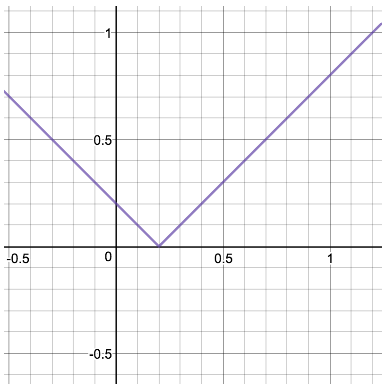

Figure 1: $f(x) = |x - 0.2|$

The tested function is

$$f(x) = |x - 0.2| \tag{2}$$

shown in 1.

The online optimized result (in the first 128 iterations) of 2 is shown in 2. The left figure illustrates the wealth in each iteration. The right figure illustrates the betting fraction in each iteration.

To compare the convergence rate of COCOB-ONS and the convergence of another online optimization algorithm, AdaGrad, both algorithms are applied in solving the minimum value of function 2.

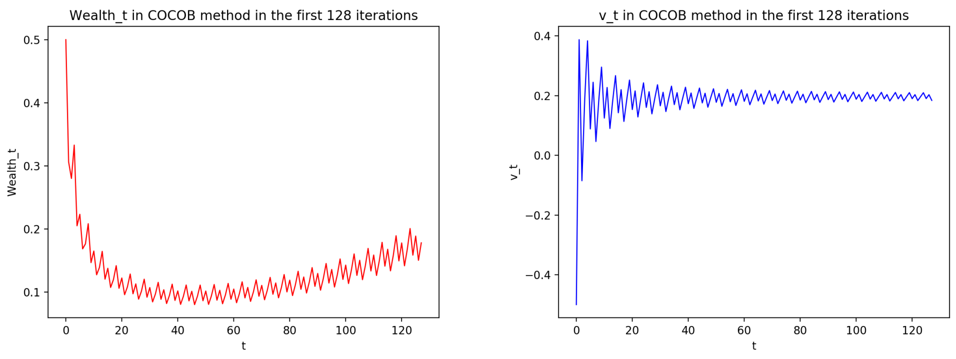

Figure 2: COCOB-ONS Evaluation in function 2

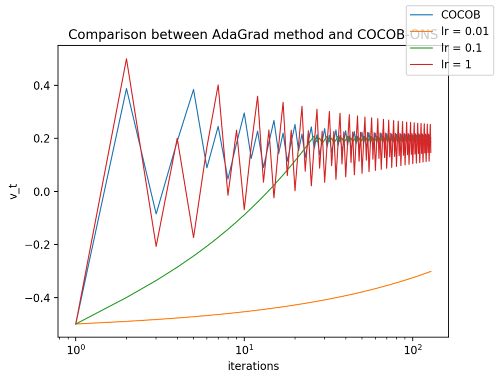

Figure 3: Comparison between COCOB-ONS and AdaGrad.

The result is show in Figure 3. From Figure 3, we can tell for the tested function, the convergence rate of the COCOB-ONS is faster than AdaGrad. Also, the effective “learning rate” of COCOB-ONS is in a oscillating pattern.

3.2 COCOB-ONS Evaluation in 1D smooth function

Besides the non-smooth function, I also evaluate the behavior of COCOB-ONS on the smooth function which is shown in 4.

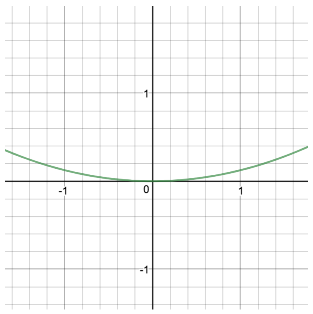

Figure 4: Objective Function $f(x) = \frac{1}{8}(x)^2$

The comparison between COCOB-ONS and AdaGrad in solving this function is shonw in 5. According to the result, we can see the COCOB-ONS is slightly better than AdaGrad. Given in the COCOB-ONS there is no parameter-tuning, in terms of the speed of finding the sub-optimal solution, COCOB-ONS overperforms AdaGrad.

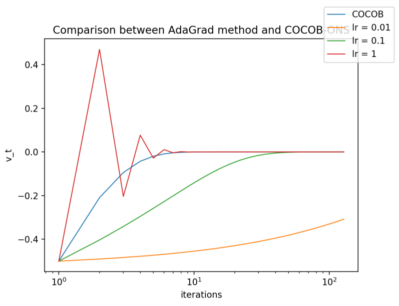

Figure 5: Objective Function $f(x) = \frac{1}{8}(x)^2$

3.3 COCOB-ONS Evaluation in 1D smooth function

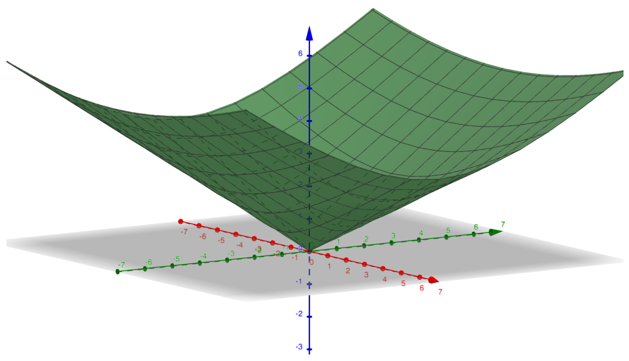

Figure 6: $f(x) = \sqrt{0.2x^2 + 0.5y^2}$

To test the behavior of COCOB-ONS in n-dimensional function, Function 6. The behavior of COCOB-ONS and AdaGrad (with $learningrate = 0.1$ which performs the best among the three learning rates) are shown in Figure 7. The updates of v_t is shown in Figure 8 and Figure 9. We can notice that the advantage of COCOB-ONS is not so obvious in this example. I guess one of the reasons is that the “effective” learning rate in coin-betting algorithm is in a oscillating pattern. Meanwhile, each direction (coordinate) interferes with each other, which leads to the unstable increase of the “wealth”.

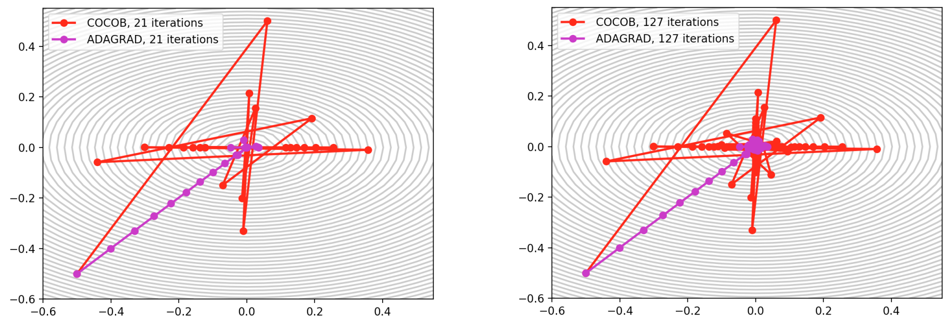

Figure 7: The left is after the 21st iteration and the right is after 127th iteration.

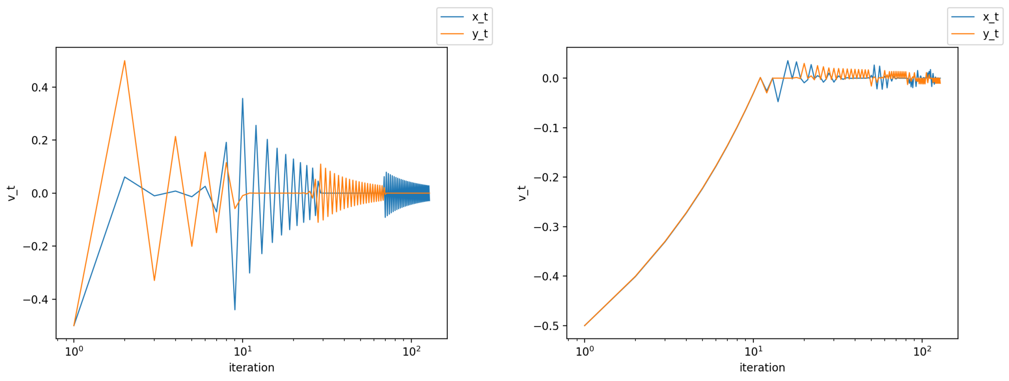

Figure 8: The left is the updates in COCOB-ONS and the right is the updates in AdaGrad with initial learning rate 0.1.

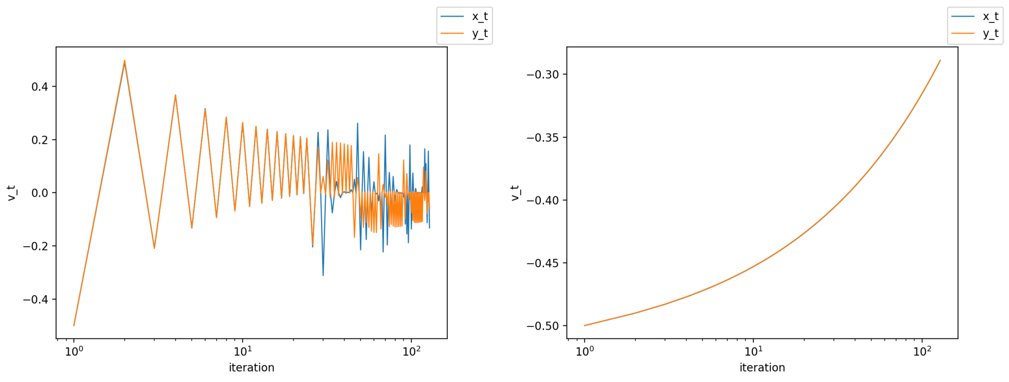

Figure 9: The left is the updates in AdaGrad (lr=1) and the right is the updates in AdaGrad (lr=0.01).

4 Summary

The Coin-Betting-ONS algorithm adjusts its step size automatically according to the reward in each iteration. In terms of 1D functions, the Coin-Betting-ONS algorithm has a faster convergence rate than AdaGrad in the tested examples. However, in terms of n-dimensional functions, the convergence rate is not always better than AdaGrad. COCOB-ONS provides an effective way for designing parameter-free optimization method, which can be applied in neural network training [4].

References

[1] Ashok Cutkosky and Francesco Orabona. Black-box reductions for parameter-free online learning in banach spaces. *arXiv preprint arXiv:1802.06293*, 2018.

[2] Elad Hazan, Amit Agarwal, and Satyen Kale. Logarithmic regret algorithms for online convex optimization. *Machine Learning*, 69(2-3):169–192, 2007.

[3] Francesco Orabona and Dávid Pál. Coin betting and parameter-free online learning. In *Advances in Neural Information Processing Systems*, pages 577–585, 2016.

[4] Francesco Orabona and Tatiana Tommasi. Training deep networks without learning rates through coin betting. In *Advances in Neural Information Processing Systems*, pages 2157–2167, 2017.

[5] Shai Shalev-Shwartz et al. Online learning and online convex optimization. *Foundations and Trends® in Machine Learning*, 4(2):107–194, 2012.