Coin Betting Algorithms for Learning-Rate Free Optimization

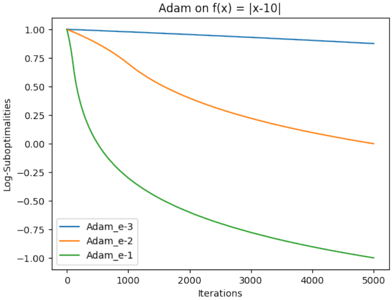
Presented by: Harsh Trivedi,
Advised by: Professor Francesco Orabona

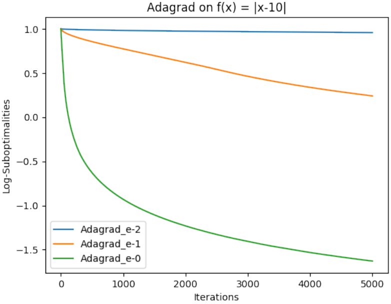
Motivation

- For optimization of non-smooth functions:
  - Fixed learning rate is not a good choice!
  - Optimal worst case is $O(1/\sqrt{t})$
  - In practice, we have to initialize at $\eta_0/\sqrt{t}$
  - For this the convergence become:
  - $$f(x_T) - f(x^*) \leq O\left(\frac{1}{\sqrt{T}}\left(\frac{||x^*||^2}{\eta_0} + \eta_0\right)\right)$$
  - $\eta_0^* = ||x^*||$, Optimal convergence: $O(\frac{||x^*||}{\sqrt{T}})$
- We don't know $||x^*||$! How to reach optimal? Trial and error!
- $\eta_0 = 0.1 \Rightarrow 0.01$ can cost (worst case) 10X epochs!
- So, main motivations of coin betting algorithms:
  - Remove the need of tuning η_0.
  - Remove humans from loop (No Tuning!)
  - Yet have provably optimal worst case convergence!

"Considered" State of the Art: Ada-something?

- Somewhat better option:
  - Use adaptive algorithms like Adagrad and Adam
  - Take account the observed gradients to adapt the learning rate.
  - Yet, have to start with something!
  - Yet, convergence heavily depends on what you start with!





- Just 1 order change makes them almost flat.
- Adagrad / Adam algorithms dont solve the problems by far.

Coin Betting to Optimization in 1D - 1

Coin Betting Game for 1D function is as following:

- Algorithm **starts** with some **initial wealth** (eg. $\text{wealth}_0 = 1\$$)
- Algorithm decides a **fraction of wealth to bet** (eg. 50%, $\beta_t = 0.5$)
- And decide the **sign of (sub)-gradient** to bet on.
- Algorithm **observes** the **sub-gradient** at x_t
- If sign of gradient is **as predicted**,
  - Algorithm **wins bet money** back (eg. **new wealth** is **1.5** \$)
- If sign of gradient is **not as predicted**,
  - Algorithm **looses bet money** (eg. **new wealth** is **0.5** \$)
- Finally, $x_{t+1} = \beta_t \text{wealth}_t$

Goal of betting strategy is to gain as much wealth as possible

Coin Betting to Optimization in 1D - 2

- Key thing that algorithm is doing:
  - Decide a betting fraction (β_t) based on past gradient observations.
- If we have a betting strategy that guarantees:

$$\text{winnings}_T \geq H(\sum_i^T g_t)$$

then,

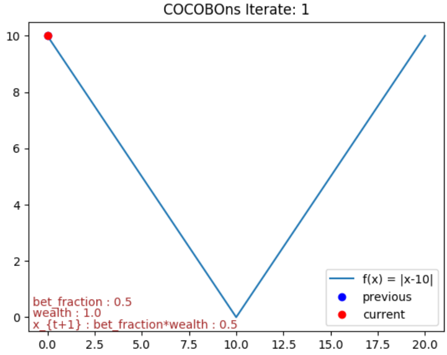
$$f(\frac{\sum_t^T x_t}{T}) - f(x^*) \leq \frac{H^*(x^*) + 1}{T}$$

- If we use a particular betting strategy (eg. KT Estimator), then, $H^*(x^*)$ becomes $O(\sqrt{T}||x^*||)$
  and optimal worst case convergence achieved: $O(\frac{||x^*||}{\sqrt{T}})$.
- It turns out that the strategy works in n-dimensional or hilbert spaces as well.

Versions of Cocob

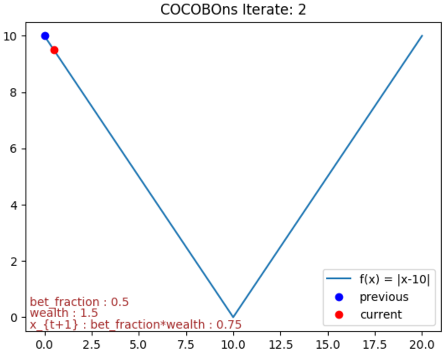
- Based on the betting strategy we use, we can have different coin betting algorithms for learning-rate free optimization
  - Cocob
    - KT Estimator to decide betting fraction.
  - **Cocob-Backprop**
    - KT Estimator but tweaked to handle grad sparsity for DL.
  - **Cocob-ONS**
    - Poses betting fraction selection itself as online problem and solves by Online Newton Step.

Example run on $f(x) = |x - 10|$



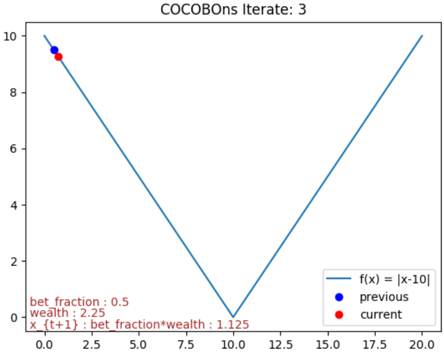
Coin Betting to Optimization in 1D - 3

Example run on $f(x) = |x - 10|$



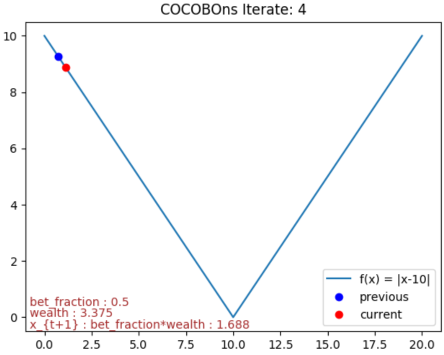
Coin Betting to Optimization in 1D - 3

Example run on $f(x) = |x - 10|$



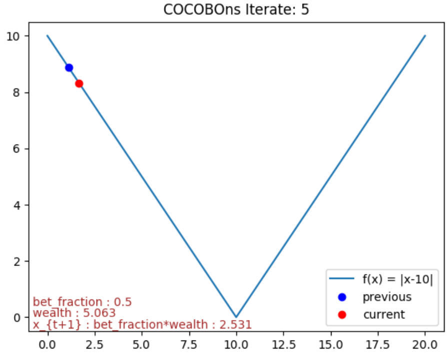
Coin Betting to Optimization in 1D - 3

Example run on $f(x) = |x - 10|$



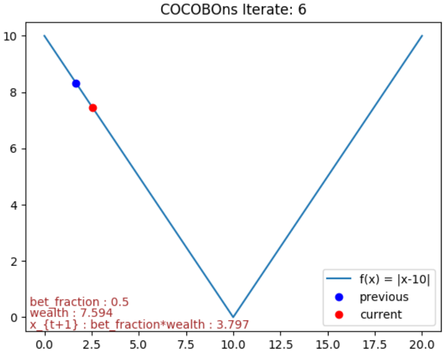
Coin Betting to Optimization in 1D - 3

Example run on $f(x) = |x - 10|$

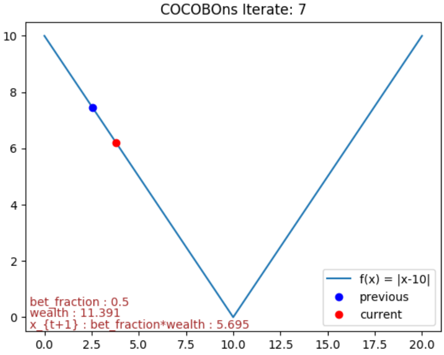


Coin Betting to Optimization in 1D - 3

Example run on $f(x) = |x - 10|$

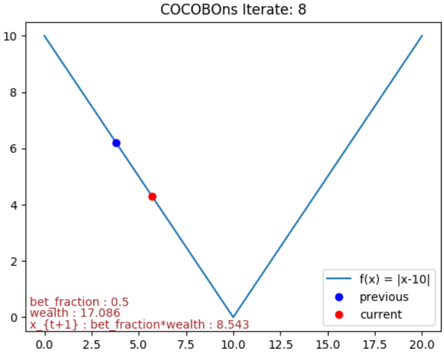


Example run on $f(x) = |x - 10|$



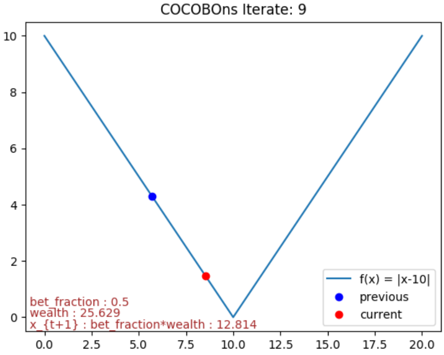
Coin Betting to Optimization in 1D - 3

Example run on $f(x) = |x - 10|$



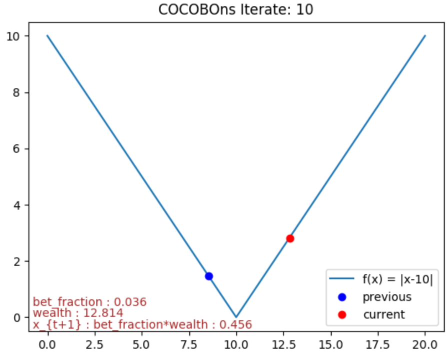
Coin Betting to Optimization in 1D - 3

Example run on $f(x) = |x - 10|$



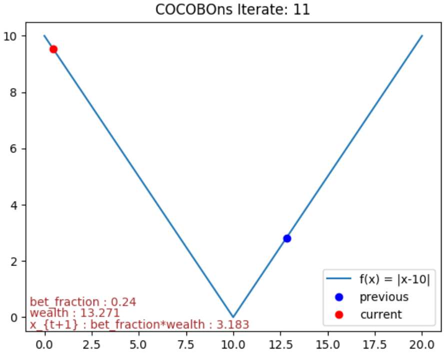
Coin Betting to Optimization in 1D - 3

Example run on $f(x) = |x - 10|$



Coin Betting to Optimization in 1D - 3

Example run on $f(x) = |x - 10|$



Key things done in this project

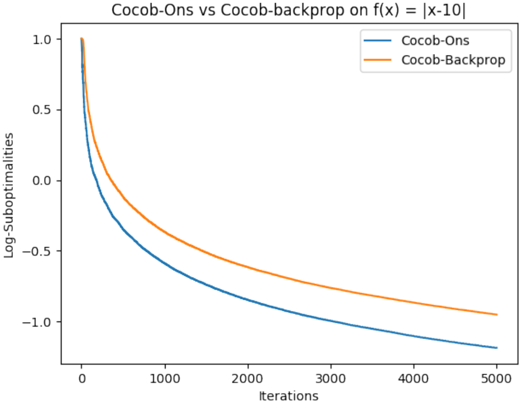
- Implemented Cocob-ONS
- Verified its behavior on convex functions
- Compared it with Cocob-Backprop and other optimizers across 3 large scale DL tasks
    - MNIST classification
    - Cifar classification
    - PTB Lang modeling
    - Convex function: $f(x) = |x - 10|$
- And based on empirical observations,
    - Worked out 4 more versions of Cocob-ONS
    - One of which, seems to work better tuned Adam on most DL tasks

Variants of Cocob-ONS based on empirical observations

- **CocobOns**
  - Default Cocob-ONS
- **Cocobons-weight-init**
  - Initial wealth is initialized based on initial weights.
- **Cocobons-grad-init**
  - Initial wealth is initialized based on initial gradients
- **Cocobons-adapt**
  - Initial wealth is dynamically adapted based on observed gradients
- **Cocobons-accum**
  - Bet fraction accumulation instead of hard thresholding
- 2-4 motivated by dimension specific wealth initialization (seemingly imp for DL), 5 is motivated by visual behavior on 1D function

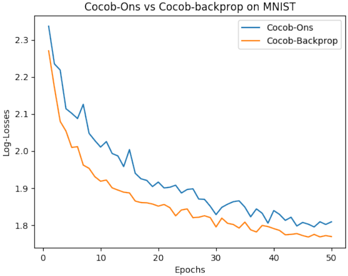
Observations-1

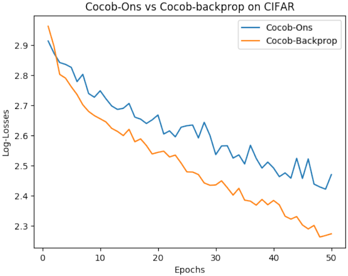
On convex problem, **cocob-ons** seems to converge better than **cocob-backprop**.

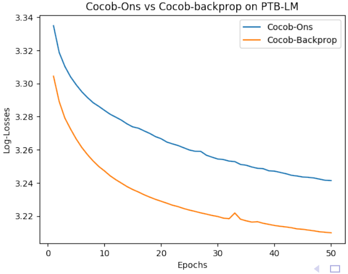


Observations-2

On non-convex problems, **cocob-backprop** seems to converge better than **cocob-ons**.

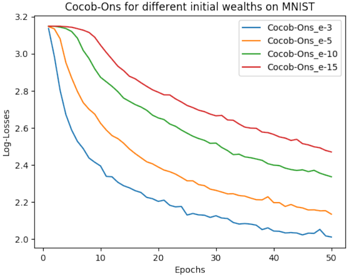


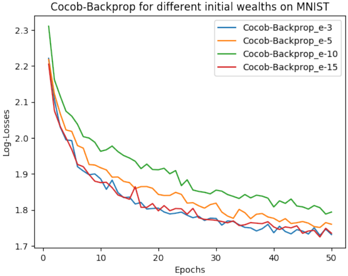




Observations-3

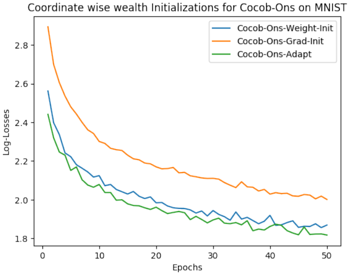
cocob-backprop has negligible effect of initial wealth. But not true for **cocob-ons**.

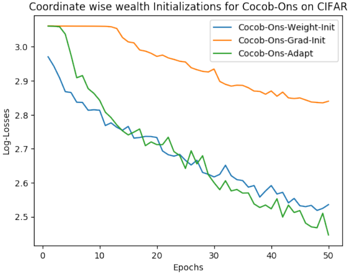


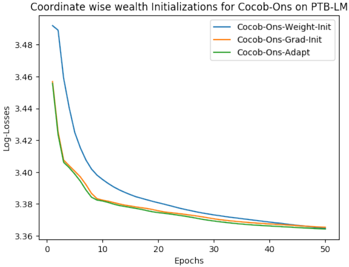


Observations-4

Different wealth initializations for **CocobONS**. **Cocob-Ons-Adapt** works best

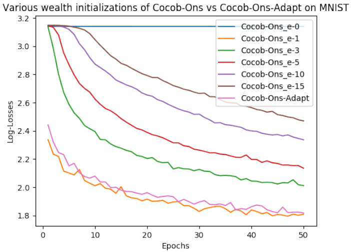


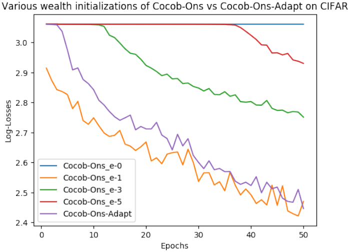


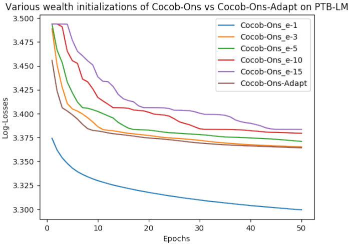


Observations-5

Did the better wealth initialization scheme help? Most of the times, **yes**.

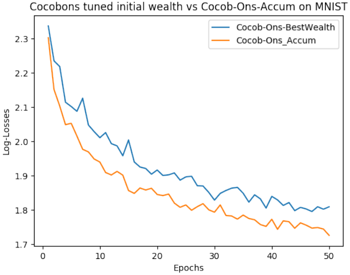


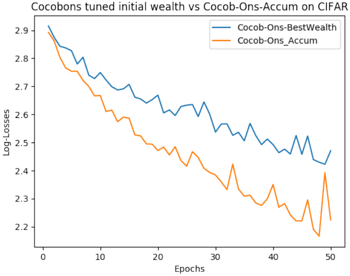


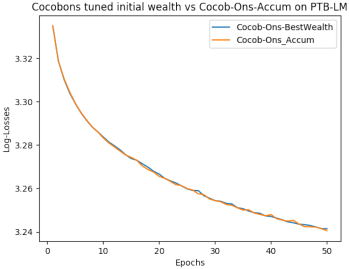


Observations-6

CocobONS-Accum works better than best tuned **CocobOns** on these tasks

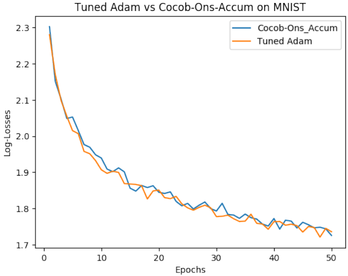


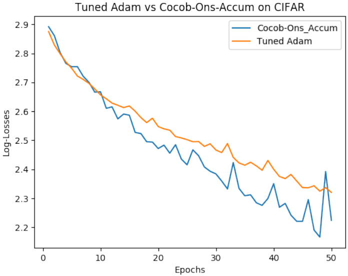


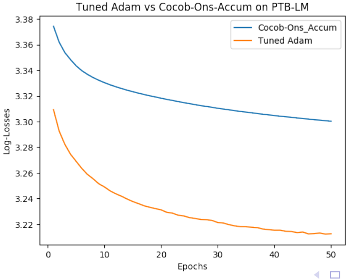


Observations-7

CocobONS-Accum works better **tuned-Adam** in 2 of 3 tasks







Sumary of Observations

- On convex problem, Cocob-Ons seems to be better than cocob-backprop
- On non-convex problems, Cocob-Backprop seems to be better than Cocob-Ons.
- Unlike Cocob-Backprop, Cocob-Ons has non-negligible effect of initial wealth
- Cocob-Ons-Adapt works best among all 3 wealth initialization schemes
- Cocob-Ons-Adapt works similar to tuned vanilla Cocob-Ons
- CocobONS-Accum works better than tuned vanilla Cocob-Ons
- CocobONS-Accum works better than tuned Adam in 2/3 cases

Questions?

Hope I have left some time...

References

- Cutkosky, Ashok, and Francesco Orabona. "Black-Box Reductions for Parameter-free Online Learning in Banach Spaces." arXiv preprint arXiv:1802.06293 (2018).
- Orabona, Francesco, and Tatiana Tommasi. "Training Deep Networks without Learning Rates Through Coin Betting." Advances in Neural Information Processing Systems. 2017.
- Hazan, Elad, Amit Agarwal, and Satyen Kale. "Logarithmic regret algorithms for online convex optimization." Machine Learning 69.2-3 (2007): 169-192.
- Orabona, Francesco, and Dvid Pl. "Parameter-Free Convex Learning through Coin Betting." Workshop on Automatic Machine Learning. 2016.