Learning Augmented Smooth Online Convex Optimization

LLM policy: ***LLM should never replace your role in this project. Any use of LLM should be properly disclosed, including the prompts and outputs from LLMs.***

1 Problem Definition

In online convex optimization, the goal is to obtain a sequence of action $x_{1:T}$ by optimize the following objective

$$\arg\min_{x_{1:T}} \sum_{t=1}^{T} \frac{m}{2}\|x_t - y_t\|^2 + \frac{1}{2}\|x_t - x_{t-1}\|^2 \tag{1}$$

The first term is the hitting cost, evaluating the distance between our action x_t and the context y_t. The second term is the switching cost, penalizing the movement between subsequent actions. For clarity, we refer to x_t as action, the per-step goal y_t as context.

The key challenging in online optimization is that when we optimize x_t, we only know the context up to time t (e.g. $y_{1:t}$). The action x_t is irrevocable, which can't be changed at later stage once it's committed.

2 Tasks

This problem is the minimal version of the Smoothed Online Convex Optimization (SOCO) problem, where both the hitting cost and switching cost are squared l_2-norm. In this toy example, we set $m = 5$.

Please finish the following tasks to the best of your efforts.

1. Read the paper [1] and understand how the regularized online balanced descent (R-OBD) algorithm is designed. Implement the algorithm R-OBD and set the hyperparameters (λ_1, λ_2) as one of the theoretical optimal values (if there are multiple values, only choose one set of them), as indicated in Theorem 4 of [1]). *[Optional]: If you are interested, please also look at the other related papers (e.g. [2, 3, 4]) to see how to extend this algorithm to more challenging cases.*

2. Now please design and implement a Machine Learning model. The input of the ML model should be the historical action and context up to now, the output is the action x_t. Please make sure when the ML model predicts the action x_t, it only requires the context up to y_t. No further context should be used in the process. The implementation details will be discussed in the next section.

3. Please summarize the cost of R-OBD and ML model, compare their average costs and produce necessary plots (cost distribution on the dataset). The expectation is that ML model outperforms R-OBD in terms of average cost in the testing dataset. If you can't achieve this, please tune your hyperparameters in ML models (e.g. learning rates, number of layers, model size etc)

4. Please try to combine the ML predictions with the R-OBD using the technique in [5] (Algorithm 1 in that paper). Show the trade-off between average performance good worst-case performance.

3 Implementation Details

I will give you a dataset named 'AI_workload.csv', which contains the power consumption of an AI server in 19 days with an increment of 1 hour. Please use any programming language and frameworks (e.g. pytorch, tensorflow, etc) you prefer. I would recommend Python and PyTorch. Here are the details for your implementation:

1. For the ease of training and comparison, please normalize the context (power consumption) between 0 - 1 across the whole dataset.

2. Please generate time sequences from the full power consumption data using a 24-hour sliding window with a step size of 2 (e.g., advancing the window by 2 hours each time). Each generated sequence should span exactly 24 hours. Once the sequences are created, divide them into training and testing sets.

3. For the machine learning model, you can start with a very simple model (e.g. LSTM or CNN). In addition, you can also start with the simplest case, where only one previous action x_{t-1} and the current context y_t are chosen as input. If you have time, please feel free to try more historical contexts ($y_{t-p:t-1}$) and previous actions ($x_{t-p:t-1}$) as the input.

4. The loss for training the ML model should be chosen the same as the objective $\sum_{t=1}^{T} \frac{m}{2}\|x_t - y_t\|^2 + \frac{1}{2}\|x_t - x_{t-1}\|^2$. We don't use MSE here, because we want to directly know the performance (or cost) of a specific action sequence.

5. Please produce necessary box plots or histograms to visualize your results.

After finishing the project, please directly email to my address *pflics@rit.edu*. I will arrange a 30-min meeting as soon as possible to discuss your result. If you have any questions, please directly email me. Have fun with the project.

References

[1] Gautam Goel, Yiheng Lin, Haoyuan Sun, and Adam Wierman. Beyond online balanced descent: An optimal algorithm for smoothed online optimization. *Advances in Neural Information Processing Systems*, 32:1875–1885, 2019.

[2] Niangjun Chen, Gautam Goel, and Adam Wierman. Smoothed online convex optimization in high dimensions via online balanced descent. In *COLT*, 2018.

[3] Guanya Shi, Yiheng Lin, Soon-Jo Chung, Yisong Yue, and Adam Wierman. Online optimization with memory and competitive control. *Advances in Neural Information Processing Systems*, 33:20636–20647, 2020.

[4] Weici Pan, Guanya Shi, Yiheng Lin, and Adam Wierman. Online optimization with feedback delay and nonlinear switching cost. *Proc. ACM Meas. Anal. Comput. Syst.*, 6(1), Feb 2022.

[5] Pengfei Li, Jianyi Yang, and Shaolei Ren. Expert-calibrated learning for online optimization with switching costs. *Proc. ACM Meas. Anal. Comput. Syst.*, 6(2), Jun 2022.