Online Convex Optimization (OCO) Algorithms

Spring 2022 [CSED490Y] Optimization for ML

Term Project Presentation

Group 11: 20222778 Dongyun Kim (김동윤)

Offline vs. Online

- In Online Convex Optimization (OCO) framework, we consider a convex feasible set $V \subset \mathbb{R}^n$ and convex loss (cost) functions $l_t: V \to \mathbb{R}$ for each time step t.

- 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.

- At each time step t, an online algorithm chooses a vector $x_t \in V$ before it observes the loss function l_t.

- Thus, $l_t(x_t)$ is the loss function value an algorithm suffers at time step t.

Online Convex Optimization

- **Input**: A convex feasible set $V \subset \mathbb{R}^n$, a sequence $\{l_1, l_2, \dots,\}$ where each l_t: $F \to \mathbb{R}$ under an online environment.
- **Output**: vectors $x_1, \dots$ such that $x_{t+1} \in V$, each after receiving l_t at each time step t.

- Aim: minimize **regret** $R_A(T)$ by algorithm A of time steps $t = 1, \dots, T$ defined as

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

$$\text{where } x^* = \mathbf{argmin}_{x \in F} \sum_{t=1}^{T} l_t(x)$$

- A **no-regret** algorithm satisfies $R_A(T) = o(T)$, i.e., has sub-linear regret.
  - Then the average regret $\frac{R_A(T)}{T}$ vanishes as $T \to \infty$

Greedy Projection

Zinkevich, M. (2003). Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th international conference on machine learning (ICML03)* (pp. 928-936).

- Gradient version

Algorithm 2.1 Projected Online Gradient Descent

Require: Non-empty closed convex set $V \subseteq \mathbb{R}^d$, $\boldsymbol{x}_1 \in V$, $\eta_1, \cdots, \eta_T > 0$

1: **for** $t = 1$ **to** T **do**
2: Output $\boldsymbol{x}_t$
3: Receive $\ell_t : \mathbb{R}^d \to (-\infty, +\infty]$ and pay $\ell_t(\boldsymbol{x}_t)$
4: Set $\boldsymbol{g}_t = \nabla \ell_t(\boldsymbol{x}_t)$
5: $\boldsymbol{x}_{t+1} = \Pi_V(\boldsymbol{x}_t - \eta_t \boldsymbol{g}_t) = \operatorname{argmin}_{\boldsymbol{y} \in V} \|\boldsymbol{x}_t - \eta_t \boldsymbol{g}_t - \boldsymbol{y}\|_2$
6: **end for**

- Subgradient Version

Algorithm 2.2 Projected Online Subgradient Descent

Require: Non-empty closed convex set $V \subseteq \mathbb{R}^d$, $\boldsymbol{x}_1 \in V$, $\eta_1, \cdots, \eta_T > 0$

1: **for** $t = 1$ **to** T **do**
2: Output $\boldsymbol{x}_t$
3: Receive $\ell_t : \mathbb{R}^d \to (-\infty, +\infty]$ and pay $\ell_t(\boldsymbol{x}_t)$
4: Set $\boldsymbol{g}_t \in \partial \ell_t(\boldsymbol{x}_t)$
5: $\boldsymbol{x}_{t+1} = \Pi_V(\boldsymbol{x}_t - \eta_t \boldsymbol{g}_t) = \operatorname{argmin}_{\boldsymbol{y} \in V} \|\boldsymbol{x}_t - \eta_t \boldsymbol{g}_t - \boldsymbol{y}\|_2$
6: **end for**

Theorem. If $\eta_t \sim \frac{1}{\sqrt{t}}$, then

$$R_{\mathrm{POGD}}(T) = \mathcal{O}(\sqrt{T}).$$

Thus, it's a no-regret algorithm.

Follow the Leader

Kalai, A., & Vempala, S. (2005). Efficient algorithms for online decision problems. *Journal of Computer and System Sciences*, *71*(3), 291-307.

- Follow the Leader (FTL)

$$x_t = \text{argmin}_{x \in V} \frac{1}{t-1} \sum_{i=1}^{t-1} l_i(x)$$

$$= \text{argmin}_{x \in V} \sum_{i=1}^{t-1} l_i(x)$$

Example:

$V = [-1,1]$

$x_1 = 0$ $\quad l_1(x) = 0.5x$

$x_2 = -1$ $\quad l_2(x) = -x$ $\quad l_1(x) + l_2(x) = -0.5x$

$x_3 = 1$ $\quad l_3(x) = x$ $\quad l_1(x) + l_2(x) + l_3(x) = 0.5x$

$x_4 = -1$ $\quad l_4(x) = -x$ $\quad l_1(x) + l_2(x) + l_3(x) + l_4(x) = -0.5x$

$x_5 = 1$ $\quad x, -x, x, -x, \ldots$

regret $R_{\text{FTL}}(T) = \sum_{t=1}^{T} l_t(x_t) - \sum_{t=1}^{T} l_t(x^*) = (T-1) - (-0.5) = T - 0.5 \approx T$

Follow the Leader

Kalai, A., & Vempala, S. (2005). Efficient algorithms for online decision problems. *Journal of Computer and System Sciences*, *71*(3), 291-307.

- Follow the Regularized Leader (FTL)

$$x_t = \text{argmin}_{x \in V} \sum_{i=1}^{t-1} l_i(x)$$

Theorem:

The regret of FTL, $R_{\text{FTL}}(T)$ satisfies

$$R_{\text{FTL}}(T) \leq \sum_{t=1}^{T} \left(l_t(x_t) - l_t(x_{t+1}) \right)$$

The previous example shows that FTL is not a no-regret algorithm.

(It showed a linear bound with respect to T)

Follow the Regularized Leader

Abernethy, J. D., Hazan, E., & Rakhlin, A. (2009). Competing in the dark: An efficient algorithm for bandit linear optimization.

- Follow the Regularized Leader (FTRL)

$$x_t = \text{argmin}_{x \in V} \sum_{i=1}^{t-1} l_i(x) + \psi_t(x)$$

  - $\psi_t(x)$ is a regularization for time step t.
  - Regularization makes the algorithm "more stable" and avoid jumping back and forth as the previous example for FTL.

Application:

Google disclosed that they use "FTRL−Proximal" to train the classifier for click prediction.

McMahan, H. Brendan, et al. Ad click prediction: a view from the trenches. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2013.

Follow the Regularized Leader

Abernethy, J. D., Hazan, E., & Rakhlin, A. (2009). Competing in the dark: An efficient algorithm for bandit linear optimization.

- Follow the Regularized Leader (FTRL)

$$x_t = \text{argmin}_{x \in V} \sum_{i=1}^{t-1} l_i(x) + \psi_t(x)$$

Example (FTRL with L_2 regularization)

Consider differentiable loss functions $l_t(x)$ and quadratic regularizers $\psi_t(x) = \frac{1}{2\eta}\|x\|_2^2$.

Then FTRL defined as

$$x_t = \text{argmin}_{x \in \mathbb{R}^n} \sum_{i=1}^{t-1} l_i(x) + \frac{1}{2\eta}\|x\|_2^2$$

This gives us $x_t = -\eta \sum_{i=1}^{t-1} \nabla l_i(x_t)$. ($\because$ take the derivative)

Then, if $l(x) = l_t(x)$ for all t, this is exactly GD, defined as $x_t = x_{t-1} - \eta \nabla l(x_{t-1})$.

Follow the Regularized Leader

Abernethy, J. D., Hazan, E., & Rakhlin, A. (2009). Competing in the dark: An efficient algorithm for bandit linear optimization.

- Follow the Regularized Leader (FTRL)

$$x_t = \text{argmin}_{x \in V} \sum_{i=1}^{t-1} l_i(x) + \psi_t(x)$$

Theorem (Regret bound of linear loss functions)

Assume $\|x\|_2 \leq B$ for all $x \in V \subset \mathbb{R}^n$, and $\|z_t\|_2 \leq C$ for all $t = 1, \ldots, T$.

Consider linear loss functions $l_t(x) = z_t^T x$ and quadratic regularizers $\psi_t(x) = \frac{1}{2\eta}\|x\|_2^2$ where $\eta = \frac{B}{C\sqrt{2T}}$.

Then the regret $R_{\text{FTRL}}(T)$ of FTRL,

$$x_t = \text{argmin}_{x \in V} \sum_{i=1}^{t-1} z_i^T x + \frac{1}{2\eta}\|x\|_2^2$$

$$\text{satisfies } R_{\text{FTRL}}(T) \leq \frac{B^2}{2\eta} + \eta T C^2 = \frac{B}{C}\sqrt{\frac{2}{T}} \sim \mathcal{O}(\sqrt{T})$$

Application: Online Portfolio Selection

- Portfolio selection:

  aim to optimize the allocation of wealth
  across a set of assets

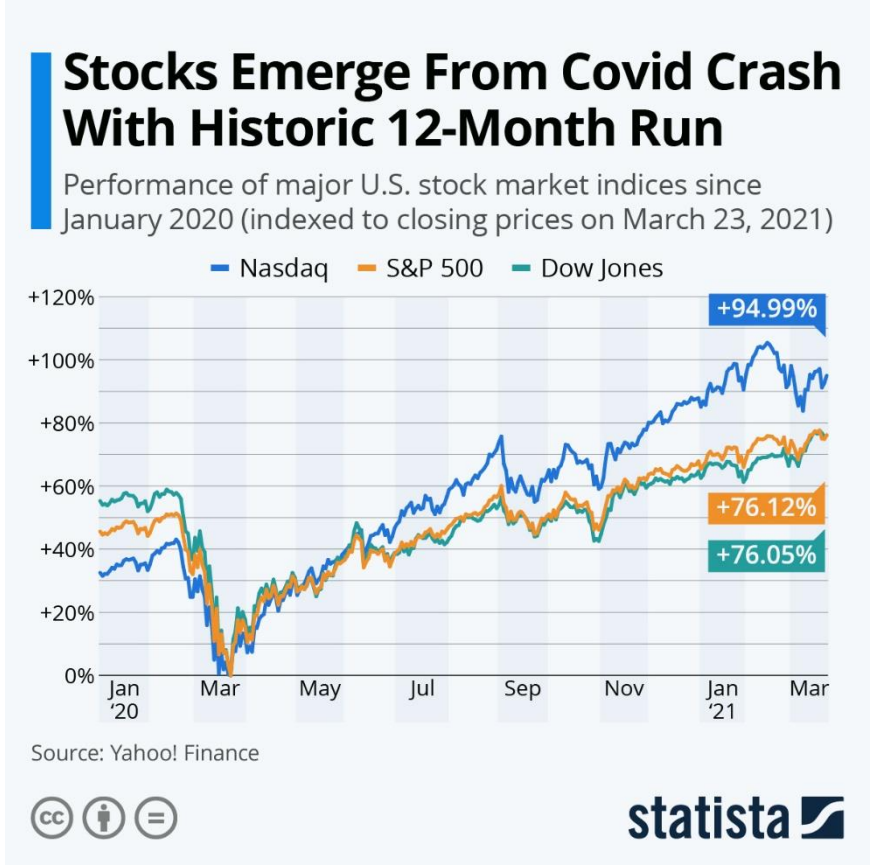
Cover (1991) introduced *universal portfolios*,

where the objective is to devise a dynamic portfolio

the difference of whose returns to the

best fixed portfolio in hindsight over T time periods

is minimized.



Thomas M Cover. Universal portfolios. *Mathematical Finance*, 1(1):1–29, 1991

Online Portfolio Selection

Van Erven, T., Hoeven, D., Kotłowski, W., & Koolen, W. (2020). Open Problem: Fast and Optimal Online Portfolio Selection. In *Proceedings of Thirty Third Conference on Learning Theory*

- This can be viewed as an instance of Online Convex Optimization:
  - Each of $t = 1, \dots, T$ rounds, a learner makes a prediction w_t in a convex domain W before observing a convex loss function $f_t: W \to \mathbb{R}$,
  - **Goal**: obtain a guaranteed bound on the regret $R_A(T) = \sum_{t=1}^{T} f_t(w_t) - \sum_{t=1}^{T} f_t(w)$ that holds for any possible sequence of loss functions f_t.
  - $W = \{w \in \mathbb{R}_+^d : \sum_{i=1}^{d} w_i = 1\}$ is the probability simplex. (d is the number of stocks)
  - Loss functions $f_t(w)$ are the form $f_t(w) = -\ln(w^T x_t)$ for vectors $x_t \in \mathbb{R}_+^d$. [Cover's loss]
- Interpretation
  - For each vector $x_t = (x_{t,1}, \dots, x_{t,i}, \dots, x_{t,d})$, $x_{t,i}$ represents the factor by which value of an asset $i \in \{1, \dots, d\}$ grows in round t.
  - For each vector $w_t = (w_{t,1}, \dots, w_{t,i}, \dots, w_{t,d})$, $w_{t,i}$ represents the fraction of our capital we re-invest in asset i in round t.
  - Capital growth over T rounds is $\exp(-\sum_{t=1}^{T} f_t(w_t))$.

Online Portfolio Selection

Van Erven, T., Hoeven, D., Kotłowski, W., & Koolen, W. (2020). Open Problem: Fast and Optimal Online Portfolio Selection. In *Proceedings of Thirty Third Conference on Learning Theory*

- State of the art

Table 1: Overview of achievable trade-offs between regret and run-time

Method	Regret	Run-time	Assumes Bounded Gradients	References
Universal Portfolio	$O(d\ln(T))$	$\tilde{O}(T^4(T+d)d^2)$	No	(Cover and Ordentlich, 1996; Kalai and Vempala, 2002)
Online Newton Step	$O(Gd\ln(T))$	$O(d^3T)$	Yes	(Agarwal et al., 2006; Hazan et al., 2007; Hazan and Kale, 2015)
Exponentiated Gradient	$O(G\sqrt{T\ln(d)})$	$O(dT)$	Yes	Helmbold et al. (1998)
Gradient Descent	$O(G\sqrt{dT})$	$O(dT)$	Yes	Zinkevich (2003)
Soft-Bayes	$O(\sqrt{dT\ln(d)})$	$O(dT)$	No	Orseau et al. (2017)
Ada-BARRONS	$O(d^2\ln^4(T))$	$O(d^{2.5}T^2)$	No	Luo et al. (2018)
FTRL	?	$O(d^2T^2)$	No	Agarwal and Hazan (2005)

Computationally infeasible

Is $\mathcal{O}(d\log T)$ possible?

Online Portfolio Selection

Van Erven, T., Hoeven, D., Kotłowski, W., & Koolen, W. (2020). Open Problem: Fast and Optimal Online Portfolio Selection. In *Proceedings of Thirty Third Conference on Learning Theory*

- This paper considers the following FTRL (Agarwal and Hazan, 2005) defined as

$$\boldsymbol{w}_{t+1} = \arg\min_{\boldsymbol{w}\in\mathcal{W}} \left\{ \sum_{s=1}^{t} f_s(\boldsymbol{w}) + \lambda \sum_{i=1}^{d} -\ln w_i \right\}$$

for some $\lambda > 0$. (FTRL with log barrier regularizer)

- Agarwal and Hazan (2005) proved $R_{\mathrm{FTRL}}(T) = \mathcal{O}(G^2 d\ln(dT))$ when $\|\nabla f_t(w_t)\|_2 \leq G$.
- But the authors believe that the bound should not depend on G at all.
- This paper shows that the regret $R_{\mathrm{FTRL}}(T) = \mathcal{O}\left(\sqrt{dT\log T}\right)$ for $\lambda \approx \sqrt{\frac{T}{d\log T}}$ and other lemmas.

Agarwal, A., & Hazan, E. (2005). Efficient algorithms for online game playing and universal portfolio management. *ECCC, TR06-033*.

Experiments

- Optimizers: Algorithms used to update weights.

- FTRL was proposed and introduced under an *online learning* framework.

- To experience how "good" FTRL algorithm is as an optimizer, we conduct experiments and review the computational results.

Application:

Google disclosed that they use "FTRL-Proximal" to train the classifier for click prediction.

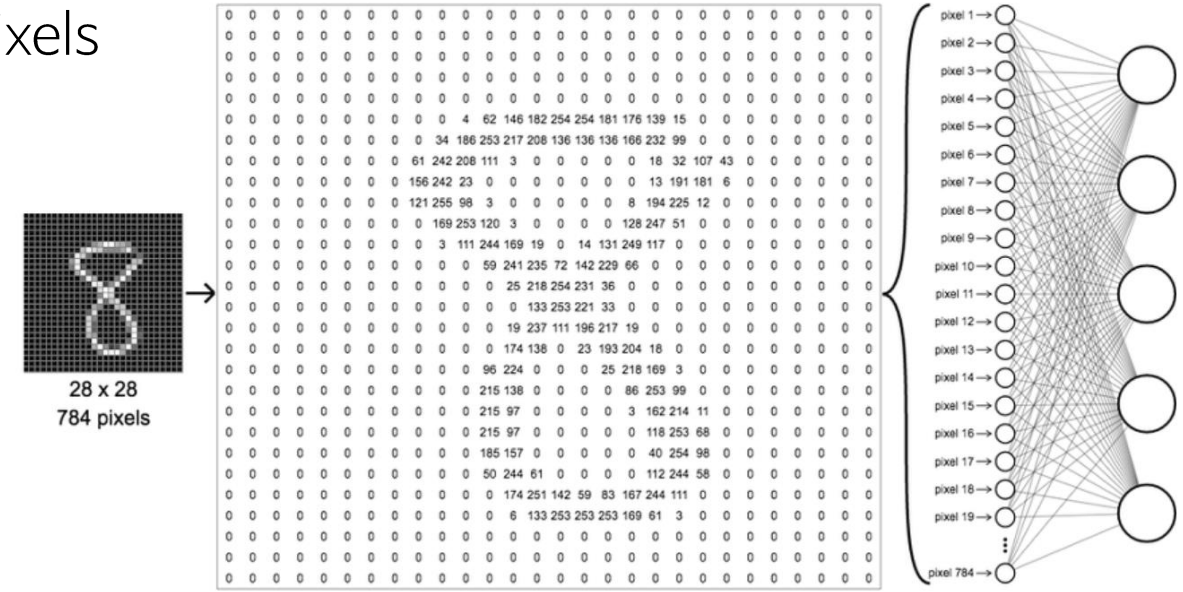
McMahan, H. Brendan, et al. Ad click prediction: a view from the trenches. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2013.

$$\mathbf{w}_{t+1} = \arg\min_{\mathbf{w}} \left(\left(\sum_{s=1}^{t} \mathbf{g}_s \right) \cdot \mathbf{w} + \frac{1}{2} \sum_{s=1}^{t} \sigma_s \left\| \mathbf{w} - \mathbf{w}_s \right\|_2^2 + \lambda_1 \left\| \mathbf{w} \right\|_1 \right)$$

Experiments

yann.lecun.com/exdb/mnist

- AMD Ryzen 5 5600X CPU desktop computer of 32GB RAM
- Tensorflow (Python)
- Dataset: MNIST
- Objective: Classify a given image to its class(0-9).
- Training data: 60,000 data, each with 28*28 pixels



Experiments

Question: How does OGD (SGD) and FTRL work as optimizers for MNIST in a CNN model?

- Tensorflow (Python)
- Batch size: 32
- Loss function: Cross-entropy loss (used for multi-class classification tasks)
- Defined a simple neural network model:

32, 64, 128, 256, 512 filters

```
super(MyModel, self).__init__()
self.conv1 = Conv2D(32, 3, activation="relu")
self.flatten = Flatten()
self.d1 = Dense(128, activation="relu")
self.d2 = Dense(10)
```

https://www.tensorflow.org/tutorials/quickstart/advanced

Experiments

32, 64, 128, 256, 512 filters

- More filters give more parameters for the optimizer to consider.
- Optimizer seeks to minimize (training loss).
- How does the loss change as the # filters increase?

```
super(MyModel, self).__init__()
self.conv1 = Conv2D(32, 3, activation="relu")
self.flatten = Flatten()
self.d1 = Dense(128, activation="relu")
self.d2 = Dense(10)
```

Layer (type)	Output Shape	Param #
conv2d_4 (Conv2D)	multiple	1280
flatten_4 (Flatten)	multiple	0
dense_8 (Dense)	multiple	11075712
dense_9 (Dense)	multiple	1290

Total params: 11,078,282
Trainable params: 11,078,282
Non-trainable params: 0

128 filters

Layer (type)	Output Shape	Param #
conv2d_8 (Conv2D)	multiple	5120
flatten_8 (Flatten)	multiple	0
dense_16 (Dense)	multiple	44302464
dense_17 (Dense)	multiple	1290

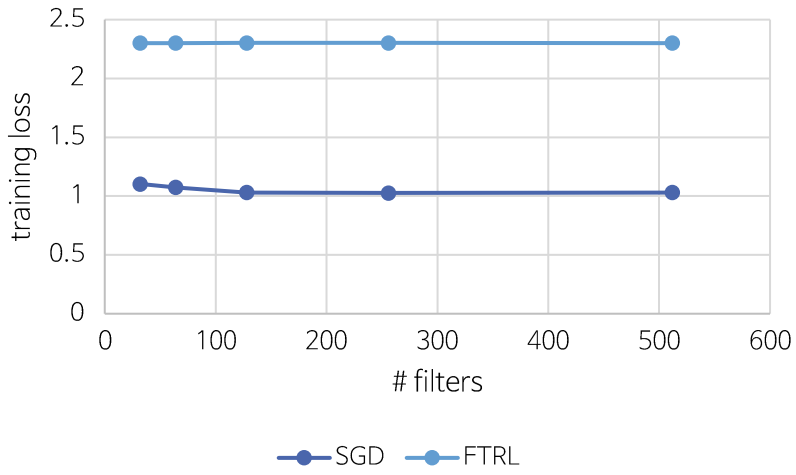
Total params: 44,308,874
Trainable params: 44,308,874
Non-trainable params: 0

512 filters

Experiments

- Average of 10 repetitions of training with SGD and FTRL.
- More filters gives more parameters that need to be optimized.

- Overall, for MNIST classification via CNN, SGD seems to be a better optimizer.
- A possible explanation is that FTRL is originally an online algorithm while the current classification is not under an online environment.



Conclusion and Future Work

Conclusion

- During the experiments, the model training when using FTRL as the optimizer took longer than OGD (SGD).
- FTRL does not seem to work well in optimizing a fixed (offline) loss function, where every l_t is identical.

Future Work

- An experiment under an online environment with the comparison of regret may illustrate the performance of the online algorithm more directly.
- The regret can be explicitly understood when applied to Online Portfolio Selection.
- Thus, an experiment with real stock market data and applying online algorithms will give a good insight.
  - ➢ We have done an experiment of FTL and FTRL for Online Portfolio Selection with Nasdaq stock data in the final report. Refer to the final report and code link in it for details.

References

- Orabona, F. (2019). A modern introduction to online learning. *arXiv* preprint arXiv:1912.13213.
- Shalev-Shwartz, S. (2012). Online learning and online convex optimization. *Foundations and Trends® in Machine Learning*, *4*(2), 107-194.
- Hazan, E. (2016). Introduction to online convex optimization. *Foundations and Trends® in Optimization*, *2*(3-4), 157-325.

- https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/Ftrl
- https://www.learningtheory.org/colt2020/virtual/papers/paper_OP02.html
- https://www.cs.ubc.ca/labs/lci/mlrg/slides/2019_summer_3_follow_the_leader.pdf