Introduction to Online Convex Optimization

Lili Song
ISE, Lehigh University

optML
Octor 2, 2019

Outline

What is online Learning?

What is online Learning?

Online learning is the process of answering a sequence of questions given (maybe partial) knowledge of the correct answers to previous questions and possibly additional available information.

What is online Learning?

Online learning is the process of answering a sequence of questions given (maybe partial) knowledge of the correct answers to previous questions and possibly additional available information.

Online Learning

for t=1,2,...

receive question $\mathrm{x}_t \in \mathcal{X}$

predict $p_t \in D$

receive true answer $y_t \in \mathcal{Y}$

suffer loss $\ell\left(p_t, y_t\right)$

Goal: minimize the cumulative loss suffered along its run

Goal: minimize the cumulative loss suffered along its run

Process: deduce information from previous rounds to improve its predictions on present and future questions

Goal: minimize the cumulative loss suffered along its run

Process: deduce information from previous rounds to improve its predictions on present and future questions

Remark: learning is hopeless if there is no correlation between past and present rounds

Example (Online Binary Prediction Game)

Email spam classification:

the player observes some features of an email and makes a binary prediction, either spam or not spam.
for each round $t = 1, \ldots, T$

- observe a feature vector $x_t \in \mathbb{R}^n$ of an instance
- make a binary prediction $\hat{y}_t \in \{+1, -1\}$. +1, -1 represent "spam" and "not spam"
- observe feedback $y_t \in \{+1, -1\}$
- A loss is incurred $\ell_t = \mathbb{1}_{\hat{y}_t \neq y_t}$

After T rounds, the cumulative loss is $\sum_{t=1}^{T} \ell_t$.

Online Learning Examples

Example (Predicting whether it is going to rain tomorrow:)

day t, the question x_t can be encoded as a vector of meteorological measurements

the learner should predict if it's going to rain tomorrow output a prediction

in $[0, 1]$, $D \neq \mathcal{Y}$.

loss function: $\ell(p_t, y_t) = |p_t - y_t|$

which can be interpreted as the probability to err if predicting that it's going to rain with probability p_t

Example (Online Binary Linear Predictor with Hinge Loss:)

The hypothesis $h_w : \mathbb{R}^n \to \{+1, -1\}$

$$h_w(x) = \operatorname{sign}(w \cdot x) = \begin{cases} +1, \text{ if } w \cdot x > 0 \\ -1, \text{ if } w \cdot x < 0 \end{cases}$$

is called binary linear predictor. The hypothesis class $\mathcal{H}$

$$\mathcal{H} = \{h_w(x) : w \in \mathbb{R}^n, \|w\|_2 \leq 1\},$$

is the class of binary linear predictors.

Online Learning Examples

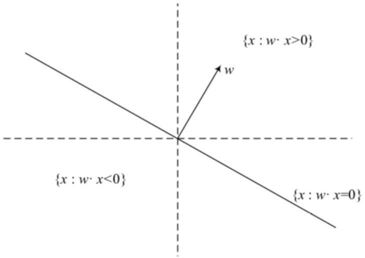


Figure 1: Hyperplane and halfspaces

Geometrically, all vectors that are perpendicular to w (i.e. zero inner product) forms a hyperplane $\{x : w \cdot x = 0\}$, shown in Figure 1. The data may fall into one of halfspaces $\{x : w \cdot x < 0\}$ and $\{x : w \cdot x > 0\}$. $|w \cdot x|$ can be interpreted as the prediction **confidence**.

Hinge Loss Function

The hinge loss function is defined as

$$\ell(w; (x_t, y_t)) = \max\{0, 1 - y_t w \cdot x_t\}.$$

Hinge Loss Function

The hinge loss function is defined as

$$\ell(w; (x_t, y_t)) = \max\{0, 1 - y_t w \cdot x_t\}.$$

As shown in Figure 2,

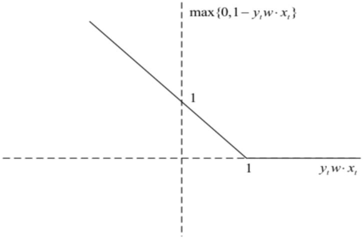


Figure 2: Hinge loss function

Hinge Loss Function

Hinge loss function imposes penalty for wrong prediction $(y_t w \cdot x_t < 0)$ and right prediction with small confidence $(0 \leq y_t w \cdot xt \leq 1)$.

For $t = 1, \cdots, T$,

- Player chooses $w_t \in \mathcal{W}$, where $\mathcal{W} = \{w \in \mathbb{R}^n : \|w\|_2 \leq 1\}$, a unit ball in $\mathbb{R}^n$
- Environment chooses (x_t, y_t)
- Player incurs a loss $\ell_t(w_t; (x_t, y_t)) = \max\{0, 1 - y_t w \cdot x_t\}$
- Player receives feedback (x_t, y_t).

Comparison Between Online Learning and Statistical Learning

Figure: Comparison Between Online Learning and Statistical Learning

	Online learning (OL)	Statistical learning (SL)
Similarities	Both define hypothesis space/class of predictors (in each round of a game in OL while in training procedure of SL).	
	Both define a loss function to evaluate the prediction performance, and small loss is preferred.	
	Instances and labels	
Differences	learning in each round of game, no distinction between training and testing	first train a model, then test it
	adversary case	statistical assumption

Online Convex Optimization (OCO)

In online convex optimization, an online player iteratively makes decisions. After committing to a decision, the decision maker suffers a loss. The losses can be adversarially chosen, and even depend on the action taken by the decision maker.

Applications:

Online advertisement placement

web ranking

spam filtering

online shortest paths

portfolio selection

recommender systems

Necessary Restrictions:

- The losses determined by an adversary should not be unbounded.

OCO Restrictions:

Necessary Restrictions:

- The losses determined by an adversary should not be unbounded.

  Otherwise the adversary could keep decreasing the scale of the loss at each step.

Necessary Restrictions:

- The losses determined by an adversary should not be unbounded.

  Otherwise the adversary could keep decreasing the scale of the loss at each step.

- The decision set must be bounded and/or structured.

OCO Restrictions:

Necessary Restrictions:

- The losses determined by an adversary should not be unbounded.

  Otherwise the adversary could keep decreasing the scale of the loss at each step.

- The decision set must be bounded and/or structured.

  Otherwise, an adversary can assign high loss to all the strategies chosen by the player indefinitely, while setting apart some strategies with zero loss. This precludes any meaningful performance metric.

OCO protocol

The protocol of OCO is as follows:

Let T denote the total number of game iterations, for $t = 1, \cdots, T$,

- Player chooses $w_t \in \mathcal{W}$, where $\mathcal{W}$ is a convex set in $\mathbb{R}^n$
- Environment chooses a convex loss function $f_t : \mathcal{W} \to \mathbb{R}$
- Player incurs a loss $\ell_t = f_t(w_t) = f_t(w_t; (x_t, y_t))$
- Player receives feedback f_t.

Example (Prediction from expert advice)

The decision maker has to choose among the advice of n given experts. i.e., the n-dimensional simplex $\mathcal{X} = \{x \in \mathbb{R}^n, \sum_i x_i = 1, x_i \geq 0\}$.

$g_t(i)$: the cost of the i'th expert at iteration t

g_t: the cost vector of all n experts

The cost function is given by the linear function $f_t(w) = g_t^T x$.

OCO Examples

Example (Online regression)

$\mathcal{X} = \mathbb{R}^n$ corresponds to a set of measurements

$\mathcal{Y} = D = \mathbb{R}$

Consider the problem of estimating the fetal weight based on ultrasound measurements of abdominal circumference and femur length.

For each $x \in \mathcal{X} = \mathbb{R}^2$, the goal is to predict the fetal weight.
Common loss functions for regression problems are:

the squared loss, $\ell(p, y) = (p - y)^2$,
the absolute loss, $\ell(p, y) = |p - y|$.

Regret

What would make an algorithm a good OCO algorithm?

Regret

What would make an algorithm a good OCO algorithm?

A good choice is the cumulative loss of the best fixed (or say static) hypothesis in $hindsight$

$$\min_{w \in \mathcal{W}} \sum_{t=1}^{T} f_t(w).$$

Regret

What would make an algorithm a good OCO algorithm?

A good choice is the cumulative loss of the best fixed (or say static) hypothesis in $hindsight$

$$\min_{w \in \mathcal{W}} \sum_{t=1}^{T} f_t(w).$$

Remark: To choose this best fixed hypothesis, we need to know future,

that is to collect all $f_1, \cdots, f_T$, then run an $off-line$ algorithm.

Regret

The difference between the real cumulative loss and this minimum cumulative loss for fixed hypothesis in hindsight is defined as regret,

$$R(T) = \sum_{t=1}^{T} f_t(w_t) - \min_{w \in \mathcal{W}} \sum_{t=1}^{T} f_t(w).$$

Regret

The difference between the real cumulative loss and this minimum cumulative loss for fixed hypothesis in hindsight is defined as regret,

$$R(T) = \sum_{t=1}^{T} f_t(w_t) - \min_{w \in \mathcal{W}} \sum_{t=1}^{T} f_t(w).$$

Remark:

- If regret grows linearly, the player is not learning.
- If regret grows sub-linearly, $R(T) = o(T)$, the player is learning and its prediction accuracy is improving. The regret per round goes to zeros as T goes to infinity.

$$\frac{1}{T}\left(\sum_{t=1}^{T} f_t(w_t) - \min_{w \in \mathcal{W}} \sum_{t=1}^{T} f_t(w)\right) \to 0, \quad T \to \infty.$$

Function $f : \mathcal{K} \to \mathbb{R}$,
if for any $x, y \in \mathcal{K}$,

$$f(y) \geq f(x) + \nabla f(x)^T (y - x) + \frac{\alpha}{2} \|y - x\|^2.$$

then f is α-strongly convex.

if for any $x, y \in \mathcal{K}$,

$$f(y) \leq f(x) + \nabla f(x)^T (y - x) + \frac{\beta}{2} \|y - x\|^2.$$

then f is β-smooth.

If f is both α-strongly convex and β-smooth, we say that it is γ-well-conditioned where γ is the ratio between strong convexity and smoothness, also called the condition number of f

$$\gamma = \frac{\alpha}{\beta} \leq 1.$$

Projections onto convex sets

Let $\mathcal{K}$ be a convex set, a projection onto a convex set is defined as the closest point inside the convex set to a given point.

$$\prod_{\mathcal{K}}(y) \triangleq \arg\min_{x \in \mathcal{K}} \|x - y\|.$$

Theorem

Let $\mathcal{K} \subseteq \mathbb{R}^n$ be a convex set, $y \in \mathbb{R}^n$ and $x = \prod_{\mathcal{K}}(y)$. Then for any $z \in \mathcal{K}$ we have

$$\|y - z\| \geq \|x - z\|.$$

Gradient descent (GD)

Gradient descent (GD) is the simplest and oldest of optimization methods given as follows:

Algorithm 1 Gradient descent (GD)

1: Input: f, T, initial point $x_1 \in \mathcal{K}$, sequence of step sizes $\{\eta_t\}$

2: **for** $t = 1$ to T **do**

3: Let $y_{t+1} = x_t - \eta_t \nabla f(x_t)$, $x_{t+1} = \prod_{\mathcal{K}}(y_{t+1})$

4: **end for**

5: **return** x_{T+1}

Gradient descent (GD)

Theorem

For unconstrained minimization of γ-well-conditioned functions and $\eta_t = \frac{1}{\beta}$, GD Algorithm 1 converges as

$$h_{t+1} \le h_1 e^{-\gamma_t}.$$

where $h_t = f(x_t) - f(x^*)$.

Proof.

By strong convexity, we have for any pair $x, y \in \mathcal{K}$:

$$\begin{aligned} f(\mathbf{y}) &\ge f(\mathbf{x}) + \nabla f(\mathbf{x})^\top (\mathbf{y} - \mathbf{x}) + \frac{\alpha}{2}\|\mathbf{x} - \mathbf{y}\|^2 \\ &\ge \min_{\mathbf{z}} \left\{ f(\mathbf{x}) + \nabla f(\mathbf{x})^\top (\mathbf{z} - \mathbf{x}) + \frac{\alpha}{2}\|\mathbf{x} - \mathbf{z}\|^2 \right\} \\ &= f(\mathbf{x}) - \frac{1}{2\alpha}\|\nabla f(\mathbf{x})\|^2. \qquad \mathbf{z} = \mathbf{x} - \frac{1}{\alpha}\nabla f(\mathbf{x}) \end{aligned}$$

Denote by ∇_t the shorthand for $\nabla f(x_t)$. In particular, taking

Gradient descent (GD)

Proof.

$$\begin{aligned} h_{t+1} - h_t &= f(\mathbf{x}_{t+1}) - f(\mathbf{x}_t) \\ &\leq \nabla_t^\top (\mathbf{x}_{t+1} - \mathbf{x}_t) + \frac{\beta}{2} \|\mathbf{x}_{t+1} - \mathbf{x}_t\|^2 && \beta\text{-smoothness} \\ &= -\eta_t \|\nabla_t\|^2 + \frac{\beta}{2}\eta_t^2 \|\nabla_t\|^2 && \text{algorithm defn.} \\ &= -\frac{1}{2\beta} \|\nabla_t\|^2 && \text{choice of } \eta_t = \frac{1}{\beta} \\ &\leq -\frac{\alpha}{\beta} h_t \end{aligned}$$

Thus,

$$h_{t+1} \leq h_t(1 - \frac{\alpha}{\beta}) \leq \cdots \leq h_1(1-\gamma)^t \leq h_1 e^{-\gamma t}$$

□

Gradient descent (GD)

Theorem

For constrained minimization of γ-well-conditioned functions and $\eta_t = \frac{1}{\beta}$, GD Algorithm 1 converges as

$$h_{t+1} \leq h_1 e^{-\frac{\gamma t}{4}}.$$

where $h_t = f(x_t) - f(x^*)$.

Proof.

$$\prod_{\mathcal{K}} (\mathrm{x}_t - \eta_t \nabla_t)$$

$$= \arg\min_{\mathbf{x} \in \mathcal{K}} \left\{ \|\mathbf{x} - (\mathbf{x}_t - \eta_t \nabla_t)\|^2 \right\} \quad \text{definition of projection}$$

$$= \arg\min_{\mathbf{x} \in \mathcal{K}} \left\{ \nabla_t^\top (\mathbf{x} - \mathbf{x}_t) + \frac{1}{2\eta_t} \|\mathbf{x} - \mathbf{x}_t\|^2 \right\}$$

□

Gradient descent (GD) for smooth, non strongly convex functions

Algorithm 2 Gradient descent reduction to β-smooth functions

1: Input: f, T, initial point $x_1 \in \mathcal{K}$, parameter $\tilde{\alpha}$

2: Let $g(x) = f(x) + \frac{\tilde{\alpha}}{2}\|x - x_1\|^2$

3: Apply Algorithm 1 with parameters g, T, $\{\eta_t = \frac{1}{\beta}\}$, x_1, return x_T.

Lemma

For β-smooth convex functions, Algorithm 2 with parameter $\tilde{\alpha} = \frac{\beta \log t}{D^2 t}$ converges as

$$h_{t+1} = \mathcal{O}\left(\frac{\beta \log t}{t}\right).$$

where D an upper bound on the diameter of $\mathcal{K}$.

Gradient descent (GD) for strongly convex, non-smooth functions

Algorithm 3 Gradient descent reduction to non-smooth functions

1: Input: f, x_1, T, δ

2: Let $\hat{f}_\delta(x) = \mathbb{E}_{v \sim \mathbb{B}}[f(x + \delta v)]$

3: Apply Algorithm 1 on $\hat{f}_\delta$, x_1, T, $\{\eta_t = \delta\}$, return x_T.

Apply the GD algorithm to a smoothed variant of the objective function.

$\mathbb{B} = \{x \in \mathbb{R}^n : \|x\| \leq 1\}$ is the Euclidean ball

$v \sim \mathbb{B}$ is a random variable drawn from the uniform distribution over $\mathbb{B}$.

Gradient descent (GD) for strongly convex, non-smooth functions

Lemma

Let f be G-Lipschitz continuous and α-strongly convex, $\hat{f}_\delta(x) = \mathbb{E}_{v\sim\mathbb{B}}[f(x+\delta v)]$, $\hat{f}_\delta$ has the following properties:
1. If f is α -strongly convex, then so is $\hat{f}_\delta$
2. $\hat{f}_\delta$ is $\frac{nG}{\delta}$-smooth
3. $|\hat{f}_\delta(x) - f(x)| \le \delta G$ for all $x \in \mathcal{K}$.

Lemma

For $\delta = \frac{dG}{\alpha}\frac{\log t}{t}$ Algorithm 3 converges as

$$h_t = \mathcal{O}\left(\frac{G^2 n \log t}{\alpha t}\right).$$

Convergence of GD

	general	α -strongly	β -smooth	γ -well
Gradient descent	$\frac{1}{\sqrt{T}}$	$\frac{1}{\alpha T}$	$\frac{\beta}{T}$	$e^{-\gamma T}$
Accelerated GD	$-$	$-$	$\frac{\beta}{T^2}$	$e^{-\sqrt{\gamma} T}$

Support vector machines (SVM)

In SVM one does binary classification ($y \in \{-1, 1\}$) by determining a separating hyperplane $\omega^\top a - b$, i.e., by determining (ω, b) such that

$$\begin{cases} \omega^\top a_j - b > 0 & \text{when } y_j = 1 \\ \omega^\top a_j - b \leq 0 & \text{when } y_j = -1 \end{cases} \quad \forall j = 1, \ldots, N$$

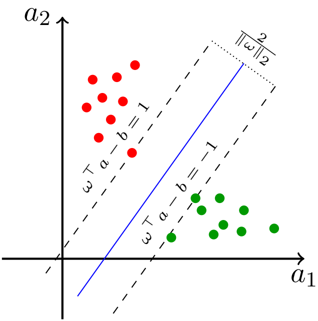
using the hinge loss function

$$\begin{aligned} \ell_H(a, y; \omega, b) &= \max\{0, 1 - y(\omega^\top a - b)\} \\ &= \begin{cases} 0 & \text{if } y(\omega^\top a - b) \geq 1 \\ 1 - y(\omega^\top a - b) & \text{otherwise} \end{cases} \end{aligned}$$

In fact, seeking a separating hyperplane $x = (\omega_*, b_*)$ can be done by

$$\min_{\omega, b} \frac{1}{N} \sum_{j=1}^{N} \ell_H(a_j, y_j; \omega, b) = L(\omega, b) \quad (**)$$

A regularizer $\frac{\lambda}{2}\|\omega\|_2^2$ is often added to $L(\omega, b)$ to obtain a maximum-margin separating hyperplane, which is more robust:



Maximizing $2/\|\omega\|_2$ is then the same as minimizing $\|\omega\|_2^2$.

In SVM, the hinge loss is a convex and continuous replacement for

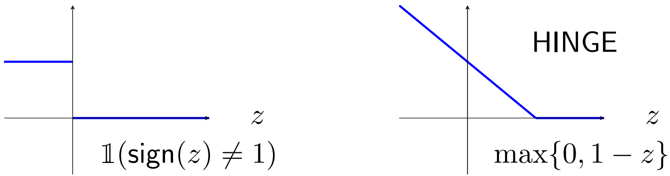
$$\ell(a, y; \omega, b) = \mathbb{1}(h(a; \omega, b) \neq y)$$

(with $\mathbb{1}(\text{condition}) = 1$ if condition is true and 0 otherwise), where

$$h(a; \omega, b) = \underbrace{2 \times \mathbb{1}(\omega^\top a - b > 0) - 1}_{\text{sign}(\omega^\top a - b)}$$

which is nonconvex and discontinuous.

$\boxed{y = 1}$



In the pictures z plays the role of $\omega^\top a - b$.

There is a statistical interesting interpretation of such optimal linear classifier when using the above loss (as the so-called Bayes function).

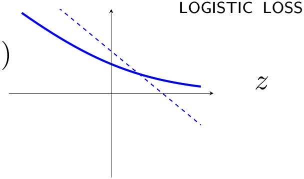
Another replacement is the smooth convex logistic loss

$$\ell_L(a, y; \omega, b) \; = \; \log(1 + e^{-y(\omega^\top a - b)})$$

leading to logistic regression (convex objective function)

$$\min_{\omega, b} \frac{1}{N} \sum_{j=1}^{N} \ell_L(a_j, y_j; \omega, b) + \frac{\lambda}{2} \|\omega\|_2^2$$

$y = 1$ $\quad \log(1 + e^{-z})$



Reference

[1] E. Hazan, Introduction to online convex optimization. Foundations and Trends® in Optimization, 2(3-4), 157-325.

[2] L. N. Vicente, S. Gratton, and R. Garmanjani, Concise Lecture Notes on Optimization Methods for Machine Learning and Data Science, ISE Department, Lehigh University, January 2019.