Foundations of Machine Learning

Lecturer: Yann Chevaleyre — Lecture n°6 #
Scribe: Danil Savine — 02/11/2023

Contents

1 Introduction and Learning Protocols

1.1 Standard Setting (Batch)

In the standard setting for online learning, a learner is presented with a sequence of data points $S = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ drawn from a distribution $\mathcal{P}^N$. The learner's task is to generate a function f_S using approaches such as Empirical Risk Minimization (ERM) or regularized ERM.

Objective: The goal is to minimize the empirical risk $\hat{R}(f_S) = \frac{1}{N}\sum_{i=1}^{N} l(f_S(x_i), y_i)$, or, ideally, to minimize the true risk $R(f_S)$.

1.2 Online Learning Protocol

Protocol:

1. For $t = 1$ to T:
   (a) The environment chooses (x_t, y_t) and reveals x_t to the learner.
   (b) The learner predicts $\hat{y}_t$.
   (c) The environment reveals y_t.
   (d) The learner endures the cost $l(\hat{y}_t, y_t)$.

Notes:

- An example of application is spam detection where the learner is a spam filter. Each new email is an example, and the learner needs to predict whether it is spam or not. The learner is evaluated on the number of mistakes it makes.
- The environment can produce arbitrary pairs (x_t, y_t) (not independently and identically distributed).
- The study of the worst case can be modeled as a zero-sum two-player game, with an adversarial environment.
- We can consider an infinite horizon where $T = \infty$.
- To simplify, we will study the realizable case with the loss function $\ell(\hat{y}, y) = \mathbf{1}[\hat{y} \neq y]$.

Objective: Minimize the cumulative loss $\sum_{t=1}^{T} \ell(\hat{y}_t, y_t)$ over T trials.

2 Realizable Case with 0/1 Loss and Finite $\mathcal{F}$

We will study the case where $\mathcal{F}$ is finite, so we can enumerate all the functions in $\mathcal{F}$. The loss is the $0-1$ loss, i.e., $\ell(\hat{y}, y) = \mathbf{1}[\hat{y} \neq y]$ Realizable case means $\exists f \in \mathcal{F}$ such that f makes zero error i.e., $\sum_{t=1}^{T} \ell(\hat{y}_t, y_t) = 0$.

2.1 ERM Algorithm (Empirical Risk Minimization)

ERM is a standard approach to learning. It is a deterministic algorithm that minimizes the empirical risk. In the context of online learning the ERM selects at each step only the functions f_t from the family of classifiers $\mathcal{F}$ that make no mistakes on the previous steps.

Algorithm 1: ERM Algorithm Formulation without updates on the set of classifiers

```
for $t = 1$ to $T$ do
    Receive $x_t$;
    Choose arbitrarily $f_t \in \mathcal{F}$ among those who perfectly classify previous data;
    Predict $\hat{y}_t = f_t(x_t)$;
    Receive the true label $y_t$, and my prediction costs me $\ell(\hat{y}_t, y_t)$;
```

Two formulation of this algorithm exist: the first chooses arbitrarily a classifier among those that made no mistakes, while the second maintains a set of valid classifiers F_t that made no mistakes up until the step t.

Failure of ERM

We investigate the sub-optimal performance of Empirical Risk Minimization (ERM) under the most challenging conditions. Consider the interval $[0, 1]$ with binary labels -1 and 1, denoted as $x = [0, 1]$, $y = \{-1, 1\}$. For this scenario, we use the 0/1 loss function and analyze a set of threshold classifiers, $\mathcal{F} = \{f_0, f_1, \dots, f_M\}$, where each $f_i(x)$ is defined as:

$$f_i(x) = \begin{cases} 1 & \text{if } x \leq \frac{i}{M} \\ -1 & \text{otherwise} \end{cases}.$$

Algorithm 2: ERM Algorithm Formulation with updates on the set of classifiers

```
Input  : Initialize $F_1 = \mathcal{F}$
for $t = 1$ to $T$ do
    Receive $x_t$;
    Choose arbitrarily $f_t \in F_t$;
    Predict $\hat{y}_t = f_t(x_t)$;
    Receive the true label $y_t$, and my prediction costs me $\ell(\hat{y}_t, y_t)$;
    Update $F_{t+1} = \{f \in F_t : f(x_t) = y_t\}$;
```

To streamline the analysis, we assume that when presented with a selection of valid classifiers $f \in F_t$, the first classifier is consistently chosen at each step. This assumption facilitates the simulation of an adversarial setting to assess ERM's response to worst-case scenarios.

At the beginning, $t = 1$, the entire classifier set is considered valid, so $F_1 = \mathcal{F}$. The example $x_1 = \frac{1}{M}$ and its associated hidden label $y_1 = 1$ are introduced. The algorithm selects $f = f_0$ for its prediction. However, given that $f_0(x_1) = -1 \neq y_1$, the classifier errs, resulting in a loss and the subsequent elimination of f_0 from the pool of valid classifiers. At each step the same pattern repeats itself, resulting in an accumulation of loss.

When we reach the last classifier (which is correct according to our hypothesis of realizable case), we have accumulated a loss of $\sum_{t=1}^{M-1} \ell\left(\hat{y}t, yt\right) = M - 1$

This demonstrates that ERM performs very poorly in this worst-case scenario.

2.2 Halving Algorithm

In this section, the method for making predictions differs from earlier approaches by using a majority vote system among the current set of valid classifiers. In a binary classification scenario, the algorithm tallies the number of classifiers favoring each class, then declares the class with the higher count as the prediction.

Algorithm 3: The Halving Algorithm

```
Input  : A family of classifiers $\mathcal{F}$, Initialize the classifier set $F_1 = \mathcal{F}$
for $t = 1$ to $T$ do
    Receive the instance $x_t$;
    Partition $F_t$ into subsets $F_{t,k}$ where $F_{t,k} = \{f \in F_t : f(x) = k\}$ for each label
      $k \in \mathcal{Y}$;
    Make a prediction $\hat{y}_t = \arg\max_{k \in \mathcal{Y}} |F_{t,k}|$;
    Receive the true label $y_t$ and incur loss $l(\hat{y}_t, y_t)$;
    Update the classifier set $F_{t+1} = \{f \in F_t : f(x_t) = y_t\}$; keeping only the
      classifiers that made no mistakes
```

Running Halving

In a scenario involving temperature and air pressure data, we initially consider a full set of four specialized classifiers for weather prediction. Given a specific instance where the temperature is high and air pressure is low, labeled as "sunny", we consult the classifier outputs. If the majority of classifiers agree on "sunny" but one, say "Meteo France", predicts differently, it is then removed from the set of valid classifiers for future predictions.

Table 1: Description of the classifier's output based on temperature and air pressure.

Temperature	Air Pressure	CNN	Weather Channel	Meteo France	Accu Weather
High	High	Sunny	Rainy	Rainy	Rainy
High	Low	Sunny	Sunny	Rainy	Sunny
Low	High	Rainy	Rainy	Rainy	Rainy
Low	Low	Rainy	Sunny	Rainy	Sunny

Table 2: Iteration results

Iteration	Temperature	Air Pressure	$\hat{y}_t$	y_t
1	High	Low	Sunny	Sunny
2	High	High	Rainy	Sunny
3	Low	Low	Sunny	Sunny
...	...	...	...	...

Halving Analysis

Theorem 1. *With the halving algorithm, in the two-class setting, let* $\ell_t = \mathbf{1}[\hat{y}_t \neq y_t]$, *then* $\sum_{t=1}^{T} \ell_t \leqslant \ln_2 |\mathcal{F}|$.

Proof of theorem 1. Let $\Omega_t = |\mathcal{F}_t|$,

- If the prediction $\hat{y}t$ is incorrect at time ($\ell_t = 1$) then, $\Omega_{t+1} \leqslant \frac{\Omega_t}{2}$
- $\Omega_1 = |\mathcal{F}|$
- $\Omega_t \geqslant 1$ for all t by realizable case assumption.

Therefore:

$$1 \leqslant \Omega_{T+1} \leqslant \Omega_1 \times 2^{-\sum_{t=1}^{T} \ell_t}$$
$$\ln_2 \left(\Omega_1 \times 2^{-\sum_{t=1}^{T} \ell_t}\right) \geqslant 0$$
$$\ln_2 |\mathcal{F}| \geqslant \sum_{t=1}^{T} \ell_t$$

□

The Halving algorithm enhances the ERM approach by reducing the error count. However, this efficiency comes at the cost of requiring a vast number of classifiers for majority voting, rendering the method computationally intensive and thus impractical for large-scale applications.

2.3 Generic Randomized Algorithm

Moving away from majority voting due to its impracticality with a large pool of classifiers, we pivot to the Generic Randomized Algorithm. This method maintains a dynamic probability distribution P_t over the classifier set, which allows for a more salable and adaptable prediction mechanism.

Rather than making unanimous decisions from a set $\mathcal{F}$, we now draw classifiers based on their probabilities in P_t.

Algorithm 4: Generic Randomized Algorithm

```
Input  : A family of classifiers $\mathcal{F}$, Initialize $F_1 = \mathcal{F}$, $P$ be a distribution over the
         family of classifiers $\mathcal{F}$.
for $t = 1$ to $T$ do
    Receive the instance $x_t$;
    Draw a classifier $f_t \sim P_t$;
    Predict $\hat{y}_t = f_t(x_t)$;
    Receive the true label $y_t$ and incur loss $\ell(\hat{y}_t, y_t)$;
    Update the distribution $P_{t+1}$;
```

Randomized Algorithm in the Realizable Case

The Empirical Risk Minimization (ERM) and the Randomized Algorithm in the realizable case operate on the same principle of classifier selection, yet they diverge significantly in their methods. ERM strictly narrows down the set of classifiers to classifiers deemed valid, whereas the Randomized Algorithm opts for a stochastic approach, uniformly selecting from valid classifiers. This randomness introduces a "halving" effect in practice.

Algorithm 5: Randomized Algorithm for the Realizable Case

```
Input  : A family of classifiers $\mathcal{F}$, Initialize $F_1 = \mathcal{F}$, $P_t = Unif(F_t)$
for $t = 1$ to $T$ do
    Receive the instance $x_t$;
    Draw a classifier $f_t \sim P_t$;
    Predict $\hat{y}_t = f_t(x_t)$;
    Receive the true label $y_t$ and incur loss $l(\hat{y}_t, y_t)$;
    Update $F_{t+1}$ and $P_{t+1}$;
```

Analyzing the Randomized Algorithm

Theorem 2. *With the randomized algorithm, in the two-class setting, let $\ell_t = \mathbf{1}_{[\hat{y}_t \neq y_t]}$, then*

$$\mathbb{E}_{f_1,\ldots,f_t \sim \mathcal{U}(\mathcal{F}_1),\ldots,\mathcal{U}(\mathcal{F}_t)}\left[\sum_{k=1}^{T} \ell_t\right] \leq \ln|\mathcal{F}|$$

Proof of theorem 2. Let $\Omega_t = |\mathcal{F}_t|$, we have:

$$\begin{aligned}
\mathbb{E}_{f_1,\dots,f_t \sim \mathcal{U}(\mathcal{F}_1),\dots,\mathcal{U}(\mathcal{F}_t)}\Big[\sum_{k=1}^{T} \ell_t\Big] &= \sum_{k=1}^{T} \mathbb{E}_{f_1,\dots,f_t \sim \mathcal{U}(\mathcal{F}_1),\dots,\mathcal{U}(\mathcal{F}_t)}[\ell_t] \\
&= \sum_{k=1}^{T} \mathbb{P}(\ell_t = 1)
\end{aligned}$$

$$\begin{aligned}
\mathbb{P}(\ell_t = 0) &= \mathbb{P}(\mathbf{1}_{[f_t(x_t)\neq y_t]} = 0) \\
&= \mathbb{P}(f_t(x_t) = y_t) \\
&= \mathbb{P}(f_t \in \{f \in \mathcal{F}_t : f(x_t) = y_t\}) \\
&= \mathbb{P}(f_t \in \mathcal{F}_{t+1}) \\
&= \frac{|\mathcal{F}_{t+1}|}{|\mathcal{F}_t|} \\
&= \frac{\Omega_{t+1}}{\Omega_t} \\
\Omega_{t+1} &= \Omega_t \times \mathbb{P}(\ell_t = 0) \\
&= \Omega_{t-1} \times \mathbb{P}(\ell_{t-1} = 0) \times \mathbb{P}(\ell_t = 0) \\
&\dots \\
&= \Omega_1 \prod_{k=1}^{t} \mathbb{P}(\ell_k = 0)
\end{aligned}$$

$$\begin{aligned}
1 \leq \Omega_{t+1} \leq \Omega_1 \prod_{k=1}^{t} \mathbb{P}(\ell_k = 0) \\
0 \leq \ln(\Omega_1) + \sum_{k=1}^{t} \ln(\mathbb{P}(\ell_k = 0)) \\
0 \leq \ln(\Omega_1) + \sum_{k=1}^{t} \ln(1 - \mathbb{P}(\ell_k = 1))
\end{aligned}$$

And because $\forall x \in [0, 1[,\ \ln(1 - x) \leq -x$:

$$\begin{aligned}
0 \leq \ln(\Omega_1) - \sum_{k=1}^{t} \mathbb{P}(\ell_k = 1) \\
\mathbb{E}\Big[\sum_{k=1}^{t} \ell_k\Big] \leq |\mathcal{F}| \qquad (\forall t)
\end{aligned}$$

□

3 Non Realizable Case with Finite $\mathcal{F}$

3.1 Regret Notion

In scenarios where no flawless classifier exists and the cumulative loss may become unbounded, we shift our focus from the aggregate loss to a measure known as "regret". The regret is quantified by the following expression:

$$\text{Regret}_T = \sum_{t=1}^{T} l(f_t(x_t), y_t) - \min_{f\in\mathcal{F}} \sum_{t=1}^{T} l(f(x_t), y_t).$$

Here, the cumulative loss is recorded progressively with each instance, whereas the optimal classifier's loss is assessed in hindsight, considering the entire data set. Calculating the precise regret is challenging due to the difficulty in determining $\min_{f\in\mathcal{F}} \sum_{t=1}^{T} l(f(x_t), y_t)$. However, we can aim to bound the regret from above.

An algorithm is "no regret" if $\lim_{T\to\infty} \frac{1}{T}\text{Regret}_T = 0$. For a randomized learner, we look at the expected regret $\mathbb{E}[\text{Regret}_T]$.

3.2 Failure of ERM in the non realizable case

Theorem 3. *With the 0/1 loss, neither ERM nor any deterministic algorithm is "no regret".*

Proof of theorem 3: Given:

$$\mathcal{F} : \{f_1, f_{-1}\} \text{ with } f_1(x) = 1,\ f_{-1}(x) = -1,\ \forall x \in \mathcal{X}$$
$$\hat{y}_t : \text{Our learning algorithm is } \mathcal{A}(x_1, y_1, x_2, y_2, \ldots, x_t) \to \hat{y}_t$$

Because $\mathcal{A}$ is deterministic, the environment can simulate $\mathcal{A}(\ldots)$. Let's take:

$$y_t = -\hat{y}_t \quad \text{(Malicious environment)}$$

Then:

$$\sum_{t=1}^{T} l\{y_t \neq \hat{y}_t\} = T$$
$$\min_{f\in\mathcal{F}} \sum_{t=1}^{T} l\{y_t \neq f(x_t)\} \leq T/2$$

Given the predictions and true labels:

$$y_t : 1, -1, -1, -1, -1, 1, \ldots$$
$$\hat{y}_t : 1, 1, 1, 1, 1, \ldots$$
$$f_1 : 1, 1, 1, 1, 1, \ldots$$
$$f_{-1} : -1, -1, -1, -1, -1, \ldots$$

Therefore, the regret:

$$\frac{1}{T}\text{Regret} \geq \frac{1}{T}(T - T/2) \geq \frac{1}{2}$$

Since the regret is greater than 0, $\mathcal{A}(\ldots)$ is not no-regret.

□

3.3 Randomized Algorithm in the non-realizable case

The hedge algorithm

The Hedge Algorithm reduces the weight of a misclassified instance at an exponential rate instead of removing it, thereby retaining all classifiers. The classifier with the least amount of errors eventually becomes the most probable to be selected.

The Hedge Algorithm is characterized as follows:

- Applicable for any bounded loss $\ell(\cdot,\cdot) \leq c$.
- Let β be a constant within the open interval $(0, 1)$. Set $P_t(f) = \frac{w_{f,t}}{\Omega_t}$, where $\Omega_t = \sum_{f\in\mathcal{F}} w_{f,t}$.
- Initialize weights with $w_{f,1} = 1$.
- Update weights according to $w_{f,t+1} = w_{f,t}e^{-\beta l(f(x_t),y_t)}$, where β is a positive constant.

Algorithm 6: Hedge Algorithm

```
Input  : A family of classifiers $\mathcal{F}$ and $\beta$.
Initialize $F_1 = F$ and set $w_{f,1} = 1$ for all $f \in \mathcal{F}$;
for $t = 1$ to $T$ do
    Receive the instance $x_t$;
    Draw a classifier $f_t$ with probability $P_t(f)$;
    Predict $\hat{y}_t = f_t(x_t)$;
    Receive the true label $y_t$ and incur loss $l(\hat{y}_t, y_t)$;
    Update $F_{t+1}$ and $P_{t+1}$ based on the loss incurred and weights $w_{f,t}$;
```

Analyzing Hedge

Theorem 4. The expected regret is bounded as follows:

$$\mathbb{E}[\text{Regret}] \leq c\sqrt{\frac{2T\ln|\mathcal{F}|}{T}}$$

3.4 From Online to Batch: No regret implies PAC

Up to now, x_t and y_t were drawn from an arbitrarily distribution. Let's now analyse the case where x_t and y_t are drawn from a distribution p.

In this case, any no-regret algorithm is PAC (probably approximately correct).

Assumption: $S = (x_t, y_y)_{t=1}^T$ is drawn from P^T. After running a no-regret algorithm, we return $\bar{f}$, a function drawn at random from $f_1, \ldots, f_T$.

Proposition: If an online learner guarantees that $E[Regret] \leq UB$ then:

$$E[R(\bar{f})] \leq R(f_F) + \frac{1}{T}UB$$

Corollary: The majority classifier (over the set $f_1, \ldots, f_T$) is PAC-learner.

If the online learner generates $f_1 \ldots f_T$ (one classifier per timestep),

Study the true expected risk of f_g. Assumption: $(x_1, y_1) \ldots (x_T, y_T)$ drawn i.i.d. from $\mathcal{R}$.

Given:

$$\begin{aligned}
\mathcal{R}(f_g) &= \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}_{z,y\sim p} l(f_g(x_t), y_t) \\
&= \mathbb{E}S\left[\frac{1}{T}\sum t = 1^T l(f_g(x_t), y_t)\right] \\
&\leq \mathbb{E}S\left[\min f \in \mathcal{F}\frac{1}{T}\sum_{t=1}^{T} l(f(x_t), y_t)\right] + \frac{UB}{T} \\
&\leq \min_{f\in\mathcal{F}}\mathbb{E}S\left[\frac{1}{T}\sum t = 1^T l(f(x_t), y_t)\right] + \frac{UB}{T} \quad \text{(by Jensen's inequality)}
\end{aligned}$$

4 Online Learning with Infinite $\mathcal{F}$ for a Convex Loss

Let's transition into an infinite set $\mathcal{F}$, such as linear classifiers.

4.1 ERM Algorithm (Follow The Leader)

Assumption: $f \in \mathcal{F}$ is represented by a vector $\theta \in \Theta \subseteq \mathbb{R}^d$ (as for logistic regression, e.g., $f(x) = \theta^\top x$). The set Θ is convex. We define $\ell_l t(\theta) = l(f_\theta(x_t), y_t)$ as the convex loss.

Algorithm 7: ERM Algorithm - Follow The Leader (FTL)

```
Input : A set of possible parameters $\Theta$
for $t = 1$ to $T$ do
    Receive the instance $x_t$;
    Choose $\theta_t = \arg\min_{\theta\in\Theta}\sum_{k=1}^{t-1}\ell_k(\theta)$;
    Predict $\hat{y}_t = f_{\theta_t}(x_t)$;
    Receive the true label $y_t$ and incur loss $\ell(\hat{y}_t, y_t)$;
```

ERM is prone to instability; that is, the introduction of a single new example at each time step can significantly alter the choice of θ_t. This volatility makes it difficult to consistently minimize regret.

4.2 Regularized ERM Algorithm (Follow the Regularized Leader)

The inherent instability of ERM necessitates the use of a regularizer $C(\theta)$ to enhance its steadiness. Specifically:

- We operate under the assumption that each classifier f in the family $\mathcal{F}$ corresponds to a parameter vector θ within a subset Θ of $\mathbb{R}^d$. Additionally, the loss $\ell_t(\theta)$, associated with the classifier f_θ for the instance (x_t, y_t), is defined to be convex.

Assumption: $f \in \mathcal{F}$ is represented by a vector $\theta \in \Theta \subseteq \mathbb{R}^d$. $\ell_t(\theta) = l(f_\theta(x_t), y_t)$ is a convex loss.

```
Algorithm 8: Algorithm R-ERM - Follow The Regularized Leader (FTRL)
  Input : A set of possible parameters $\Theta$, a regularization function $C$, and a
          regularization parameter $\lambda$.
  Initialize $F_1 = F$;
  for $t = 1$ to $T$ do
      Receive the instance $x_t$;
      Choose $\theta_t = \arg\min_{\theta\in\Theta}\left(\sum_{k=1}^{t-1} \ell_k(\theta) + \lambda C(\theta)\right)$;
      Predict $\hat{y}_t = f_{\theta_t}(x_t)$;
      Receive the true label $y_t$ and incur cost $\ell(\hat{y}_t, y_t)$;
  Note: Often, $C(\theta) = \|\theta\|^2$.
```

R-ERM with linear losses, SGD and Mirror Descent

For simplicity, assume the loss function $\ell_t(\theta)$ is linear in θ, so we can write $\ell_t(\theta) = g_t^\top \theta$ for some $g_t \in \mathbb{R}^d$, assuming $\Theta = \mathbb{R}^d$

R-ERM: $\theta_{t+1} = \arg\min \theta \in \Theta\left\{\left[\sum k = 1^t \ell_k(\theta)\right] + \lambda\mathrm{C}(\theta)\right\}$

Pick $C(\theta) = \|\theta\|_2^2$

Let's define $\nabla\ell_{t+1}(\theta) = \sum_{k=1}^{t} \ell_k(\theta) + 2\lambda\mathrm{C}(\theta)$

$$\nabla\ell_{t+1}(\theta_{t+1}) = \sum_{k=1}^{t} g_k + 2\lambda\theta_{t+1} = 0 \qquad \Rightarrow \theta_{t+1} = -\frac{1}{2\lambda}\sum_{k=1}^{t} g_k$$

$$\nabla\ell_t(\theta_t) = \sum_{k=1}^{t-1} g_k + 2\lambda\theta_t = 0 \qquad \Rightarrow \theta_t = -\frac{1}{2\lambda}\sum_{k=1}^{t-1} g_k \text{ and } \sum_{k=1}^{t-1} g_k = -2\lambda\theta_t$$

$$\theta_{t+1} = -\frac{1}{2\lambda}g_t - \frac{1}{2\lambda}\sum_{k=1}^{t-1} g_k$$

$$\theta_{t+1} = \theta_t - \frac{1}{2\lambda}g_t$$

Alternatively, if using a gradient term:

$$\theta_{t+1} = \theta_t - \frac{1}{2\lambda}\nabla_\theta\ell_t(\theta_t) \qquad \text{SGD!!}$$

If $\nabla_\theta\ell_t$ is small, then the algorithm is stable: θ_{t+1} is close to θ_t

"Be The Leader (BTL)" Lemma

This lemma indicates that if θ_t is stable and ℓ_t is "smooth" to some extent, then the regret of the R-ERM is constrained.

Lemma 1. *Let* $\theta^* = \arg\min_\theta \sum_{t=1}^{T} \ell_t(\theta)$.
With Regularized Empirical Risk Minimization (R-ERM), we have:

$$\sum_{t=1}^{T}(\ell_t(\theta_t) - \ell_t(\theta^*)) \le \lambda\|\theta^*\|^2 + \sum_{t=1}^{T}(l_t(\theta_t) - l_t(\theta_{t+1}))$$

Stability of R-ERM Lemma

Lemma 2. *If l_t is convex and ρ-Lipschitz, then:*

$$\|\theta_{t+1} - \theta_t\| \leq \frac{\rho}{\lambda}$$

Regret of R-ERM Theorem

Theorem 4. *Let l_t be a convex and differentiable loss function and let $\theta^* = \arg\min_\theta \sum_{t=1}^{T} l_t(\theta)$ be such that ℓ_t is ρ-Lipschitz. Then, with $\lambda = \frac{L\sqrt{T}}{W_2}$ and assuming $\|\theta^*\|^2 \leq W_2$, we have:*

$$\mathit{Regret}_T = \sum_{t=1}^{T} l_t(\theta_t) - l_t(\theta^*) \leq 2W_2\rho\sqrt{T}$$