Foundations of machine learning

Lecturer: Yann Chevaleyre
Scribe: XU Jia-Jie Jacques

Lecture n°6 #
02/11/2023

1 Introduction To Online Learning Protocol

In the setting of online learning, data are not provided all at once but are given sequentially, one by one. This situation can be conceptualized as a two-player game involving the environment and a learner, where the environment represents everything that the learner is unaware of. The learning process unfolds over T rounds, with the environment selecting a labeled example, denoted as (x_t, y_t).

The environment only reveals the x_t to the learner. If we are in a classification setting, the learner can use decision trees to make a prediction based on x_t. After the prediction is made, the environment discloses the true label, and the learner endures a cost $\ell(\hat{y}_t, y_t)$, typically represented by 0-1 loss in the classification settings.

To illustrate this concept, consider a spam detection system in the email system as a learner. Whenever you receive an email, only the one who sent it knows the true nature of the email (whether it's spam or not). The user, in this case, will reclassify the emails whenever the spam detection system makes a wrong classification, thus this will introduce a loss to the spam detection system, which will help the spam detection improve himself in the spam classification task.

In the online learning setting , we can have infinite long sequence, i.e. $T = \infty$. And the main difference from standard learning protocols is the absence of the assumption that data are independently and identically distributed (i.i.d.). As a result, the learner doesn't know what data it will get.

More concretely, the Online Learning Protocols can be formulated as follows :
For $t = 1$ to T
- The environment chooses x_t, y_t, and reveals x_t to the learner
- The learner predicts $\hat{y}_t$
- The environment reveals y_t
- The learner endures the cost $\ell(\hat{y}_t, y_t)$

Objective : minimise $\sum_{t=1}^{T} \ell(\hat{y}_t, y_t)$ the cumulated loss

2 Realizable case with 0/1 loss and finite F

Realizble case means $\exists f \in \mathcal{F}$ such that f makes zero error.

2.1 ERM Algorithm (Empirical Risk Minimization)

ERM typically involves a family of classifiers, and in the online setup under the classification setting, the general idea is to classify each new example by selecting a classifier that

has achieved zero loss on all the past data.
We can formally define the algorithm as follows :
$\mathcal{F}_1 = \mathcal{F}$
For $t = 1$ to T
- Receive x_t
- Choose arbitrarily $f_t \in \mathcal{F}_t$
- 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 $\mathcal{F}_{t+1} = \{f \in \mathcal{F}_t : f(x_t) = y_t\}$

We can see from the formulation above that a valid set called $\mathcal{F}_1$ is firstly initialized as the entire family of classifiers $\mathcal{F}$ (since we didn't receive any data and didn't make any error yet), then at each time the learner receives an example and obtains a label, after the prediction of the learner, the algorithm updates the valid set to retain only those classifiers that didn't make mistakes. $\mathcal{F}_t$ represents the set of all classifiers that have made no errors on all previous examples.

For example, in the context of binary classification, where we label the classes as -1 and +1, i.e. $y = \{-1, 1\}$, at time t the learner receives the input x_t, and $\mathcal{F}_t$ consists of two distinct subsets : one subset comprises functions that predict $f(x_t) = 1$ and the other subset consists of functions that predict $f(x_t) = -1$. Then the environment reveals the label y_t to be 1, as a result the algorithm will update the valid set to be the subset that contains all the classifiers that correctly predict 1.

2.2 Failure of ERM

Here is an examination of how poorly ERM performs in the worst-case scenario. In this scenario, ERM exhibits severe shortcomings. We will explore a specific worst-case scenario using the example of the interval [0, 1] with labels assigned as -1 and 1, i.e. $x = [0, 1], y = \{-1, 1\}$. Within this context, we use the 0-1 loss, and we will work with a collection of classifiers, specifically, a set of threshold classifiers, i.e. $F = \{f_0, f_1, \ldots, f_M\}$, where $\mathrm{f_i(x)} = \begin{cases} 1 & \text{if } x \leq \frac{i}{M} \\ -1 & \text{otherwise} \end{cases}$.
To simply this study case, the following assumption is made : among all valid f $\in F_t$, we always pick the first one at each time step, and we simulate an adversarial environment to study the worst-case scenario.
At the first time step $t = 1$, the valid set is the entire set of classifiers, i.e. $F_1 = F$ and the example and the hidden label are $x_1 = \frac{1}{M}, y_1 = 1$. Given this setup, we choose $f = f_0$ to predict y_1 based on x_1, since $f_0(x_1) \neq y_1$, classifier made an mistake, the learn endures a loss, and thus we need to exclude it from the valid set.
This can be written as follows :
1) $t = 1$
$x_1 = \frac{1}{M}, y_1 = 1$
- $F_1 = F$, so we choose $f = f_0$
- $\hat{y}_1 = f_0(x_1) \neq y_1$
$\Rightarrow$ error $\rightarrow \ell(\hat{y}_1, y_1) = 1$
- $F_2 = F \setminus \{f_0\} = \{f_1, f_2, f_3 \ldots\}$

2) $t = 2$
$x_2 = \frac{2}{M}, y_2 = 1$
- we pick f_1
- $\hat{y}_2 = f_1(x_2) = -1 \Rightarrow$ error $\rightarrow \ell(\dot{y}_2, y_2) = 1$
M-1)
error $\rightarrow$ cumulated loss $= \sum_{t=1}^{M-1} \ell(\hat{y}t, yt) = M - 1$. As we can see from the cumulated loss, it makes a mistake nearly at each time step. This demonstrates that ERM performs very poorly.

2.3 Halving Algorithm

Similar to the previous section, we continue to maintain a set of valid classifiers. However, the distinction lies in how we arrive at our predictions. Specifically, we will determine our output based on the majority prediction. For example, in the binary classification setting, we count how many classifiers predict +1 and how many predict -1, and subsequently output the class label that the majority of these valid classifiers predict.

The Halving Algorithm can be formulated in the following way :

- Let $\mathcal{F}$ be a family of classifiers

Algorithm

$\mathcal{F}_1 = \mathcal{F}$
For $t = 1$ to T
- Receive x_t
- Let $\mathcal{F}_t^k = \{f \in \mathcal{F}_t : f(x) = k\}$, for all $k \in \dagger$
- Predict $\hat{y}_t = \arg\max_{k \in y} \left|\mathcal{F}_t^k\right|$
- Receive the true label y_t, and my prediction costs me $\ell(\hat{y}_t, y_t)$
- Update $\mathcal{F}_{t+1} = \{f \in \mathcal{F}_t : f(x_t) = y_t\}$

2.4 Running Halving

Consider the following scenario where we have only two pieces of data : temperature and air pressure. We have a total of four classifiers who are designed to make predictions based on temperature and air pressure.

At first the set of all valid classifier is $F_1 = \{f_{CNN}, f_{\text{MeteoFrance}}, \ldots\}$, which includes all four functions.

The table 1 provides a description of the classifier's output based on temperature and air pressure. Suppose the data x_1 is high and low for the first time step, the corresponding true label is "sunny" according to table 2. Looking up at the table 1, we know that the majority prediction is "sunny", but "Meteo France" made an error, so it will be excluded from the valid set.

2.5 Halving Analysis

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

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

TABLE 1 – Description of the classifier's output based on temperature and air pressure.

Tempe rature	Air pressu re	CNN	Weath er Chann	Meteo France	Accu Weath er
High	High	Sunny	Rainy	Rainy	Rainy
High	Low	Sunny	Sunny	Rainy	Sunny
Low	High	Rainy	Rainy	Rainy	Rainy
Low	Low	Sunny	Sunny	Rainy	Sunny

TABLE 2 – Labeled examples

iterat ion	Temp eratu re	Air pressure	$\hat{y}_t$	$\mathbf{y}_t$
1	High	Low		Sunny
2	High	High		Sunny
3	Low	Low		Sunny
4	...	...		

— If the prediction $\hat{y}_t$ is incorrect at time ($l_t = 1$) then, $\Omega_{t+1} \leqslant \frac{\Omega_t}{2}$
— $\Omega_1 = |\mathcal{F}|$
— $\Omega_t \geqslant 1$ for all t by realizalility assumption.
Therefore :

$$
\begin{aligned}
& 1 \leqslant \Omega_{T+1} \leqslant \Omega_1 \times 2^{-\sum_{t=1}^{T} l_t} \\
& \Rightarrow \ln 2 \left(\Omega_1 \times 2^{-\sum t=1^T l_t} \right) \geqslant 0 \\
& \Rightarrow \ln 2 |\mathcal{F}| \geqslant \sum_{t=1}^{T} \ell_t
\end{aligned}
$$

□

As we can see that, compare this to our earlier ERM approach, which would result in M-1 errors, the Halving algorithm has an important improvement over the ERM approach. Although we have a robust algorithm, a significant drawback emerges. To achieve accurate predictions, we'd need to employ an enormous number of classifiers, and then rely on majority voting. As a result, this approach becomes impractical and computationally infeasible due to its unmanageable scale.

2.6 Generic Randomized Algorithm

When dealing with a substantial number of classifiers, relying on majority voting becomes impractical. In such cases, we need to explore alternative approaches, as we do in the Generic Randomized Algorithm. This randomized version serves the same purpose but in a more lightweight manner.

In Generic Randomized Algorithm, we maintain a probability distribution P_t over all the classifier. At each time step, we update this distribution instead of updating the set F as

before. This distribution allows us to assign low probability to certain classifiers, essentially reducing their influence in the prediction process.

So, when a new example arrives, we randomly select a classifier from the distribution P_t, and we make our prediction using this randomly chosen classifier. This randomized approach provides a more manageable and flexible way to handle a large number of classifiers.

The Generic Randomized Algorithm can be formulated in the following way :

Let $\mathcal{F}$ be a family of classifiers, let P_t be a distribution over $\mathcal{F}$.

Algorithm

For $t = 1$ to T
- Receive x_t
- Draw $f_t \sim P_t$
- 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 P_{t+1}

2.7 Randomized algorithm in the realisable case

ERM and the Randomized algorithm in the realisable case look similar to each other, but the key distinction between them lies in how we select the classifiers. In the ERM approach, we update the set $\mathcal{F}_t$ by simply retaining the valid classifiers. In the approach of Randomized algorithm under the assumption of the realisable case, we randomly uniformly draw a classifier from the set of valid classifiers, furthermore this approach possesses the property of "halving" in some sense.

Now, let's explore why this method exhibits halving behavior. Consider a set of valid classifiers, and suppose that 80 percent of these classifiers predict "1," while the remaining 20 percent predict "-1". So the Random Algorithm approach will have an 80 percent chance of picking a classifier that will predict 1. When we use the halving approach, the prediction will be the prediction from the majority class. So when the majority is significantly larger than the other part in the family set $\mathcal{F}$, these two strategies are similar.

The Randomized algorithm in the realisable case can be formulates in the following way :

- Let $\mathcal{F}$ be a family of classifiers
- I choose $P_t = \text{Unif}(\mathcal{F}_t)$
- As in the naïve algorithm, I have $\mathcal{F}_{t+1} = \{f \in \mathcal{F}_t : f(x_t) = y_t\}$

Algorithm

$\mathcal{F}_1 = \mathcal{F}$
For $t = 1$ to T
- Receive x_t
- Draw $f_t \sim P_t$
- 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 $\mathcal{F}_{t+1}$ and P_{t+1}

2.8 Analysing the randomized algorithm

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

$$\mathbb{E}_{f_t \sim \mathcal{U}(\mathcal{F}_t)}\Big[\sum_{t=1}^{T} \ell_t\Big] \leq \ln |\mathcal{F}|$$

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

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

$$\begin{aligned}\mathbb{P}(\ell_t = 0) &= \mathbb{P}(1_{[f_t(x_t) \neq y_t]} = 0) \\ &= \mathbb{P}(f_t(x_t) = y_t) \\ &= \frac{\Omega_{t+1}}{\Omega_t}\end{aligned}$$

$$\begin{aligned}1 \leq \Omega_{t+1} = \Omega_t \mathbb{P}(\ell_t = 0) = \Omega_1 \prod_{t=1}^{T} \mathbb{E}(1 - \ell_t) \\ 0 \leq \ln(|\mathcal{F}|) + \sum_{t=1}^{t} \ln(\mathbb{E}(1 - \ell_t)) \\ 0 \leq \ln(|\mathcal{F}|) - \sum_{t=1}^{T} \mathbb{E}(\ell_t)\end{aligned}$$

Thus, we have $\mathbb{E}_{f_t \sim \mathcal{U}(\mathcal{F}_t)}\Big[\sum_{t=1}^{T} \ell_t\Big] \leq \ln |\mathcal{F}|$ □

3 Non realizable case

In the non realizable case, we remove the assumption of a single perfect classifier. In doing so, a novel concept known as "regret" need to be introduced.

3.1 Regret notion

We may encounter situations where the cumulative loss becomes unbounded, since we assume that it doesn't exist a classifier that can make zero error in the non realizable case, i.e.$\sum_{t=1}^{T} \ell\left(f_t\left(x_t\right), y_t\right)$ can tend to ∞.

So in the non realizable case, instead of solely focusing on the cumulative loss, we introduce the concept of "regret". The regret is defined as the cumulative loss minus the sum of the

losses of the best classifier, i.e. $\text{Regret}_T = \sum_{t=1}^{T} \ell\left(f_t\left(x_t\right), y_t\right) - \min_{f \in \mathcal{F}} \sum_{t=1}^{T} \ell\left(f\left(x_t\right), y_t\right)$. The cumulative loss you accumulate is observed in an online fashion, while the best function or classifier considers all the data, spanning past, present, and future. As a result, it will be complicated to calculate the term $\min_{f \in \mathcal{F}} \sum_{t=1}^{T} \ell\left(f\left(x_t\right), y_t\right)$, thus it's hard to calculate the exact regret, but we can try to establish upper bounds on it.

An algorithm is "no regret" if $\frac{1}{T}\text{Regret}_T \to 0$ when $T \to \infty$.

For a randomized learner, we look at the expected regret $\mathbb{E}\left[\text{Regret}_T\right]$.

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 :

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

Because $\mathcal{A}$ is deterministic, the environment can simulate $\mathcal{A}(\ldots)$.

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

Then :

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

Given the followings :

$$t : 1, 2, 3, 4, 5, 6, \ldots$$
$$y_t : 1, -1, -1, 1, -1, 1, \ldots$$
$$\hat{y}_t : -1, 1, 1, -1, 1, -1 \ldots$$
$$f_1 : 1, 1, 1, 1, 1, 1 \ldots$$
$$f_{-1} : -1, -1, -1, -1, -1, -1 \ldots$$

We can get :

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

Since $\frac{1}{T}\text{Regret}$ is greater than 0, $\mathcal{A}(\ldots)$ is not no-regret.

□

From the above proof we can see that we will perform significantly worse than the best possible classifier if we don't introduce randomness.

3.3 Randomized Algorithm in the non realizable case

The hedge algorithm finds application in various contexts such as game theory. Different from previous randomized setting, where the classifier will be removed when making a mistake, here in the hedge algorithm the weight of the wrong classififer is just decreased exponentially fast, but it will not be removed. Eventually, the only classifier with a high probability of being chosen is the one making the fewest mistakes.

Concretely, we can formulate the hedge algorithm as follows :
- This algorithm works for any bounded loss $\ell(\cdot,\cdot) \leqslant c$
- Let $\beta \in]0,1[$. Choose $P_t(f) = \frac{1}{\Omega_t} w_{f,t}$ with $\Omega_t = \sum_{f\in\mathcal{F}} w_{f,t}$
- $w_{f,1} = 1$
- $w_{f,t+1} = w_{f,t} e^{-\beta \ell(f(x_t),y_t)}$ for some constant $\beta > 0$

Hedge Algorithm

$\mathcal{F}_1 = \mathcal{F}$
For $t = 1$ to T
- Receive x_t
- Draw $f_t \sim P_t$
- 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 $\mathcal{F}_{t+1}$ and P_{t+1}

3.4 Analyzing Hedge

Theorem 4. $\mathbb{E}[\ \textit{Regret}\ ] \leqslant c\sqrt{2T \ln |\mathcal{F}|}$

3.5 From Online to Batch : No regret implies PAC

- Up to now, x_t, y_t was arbitrary. If x_t, y_t is drawn i.i.d. from P, any no-regret algorithm will give a PAC-learning algorithm.

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.

Proof : The online learner generates $f_1 \ldots f_T$ (one classifier per time step),

Study the true expected risk of $\bar{f}$under the assumption : $(x_1, y_1) \ldots (x_T, y_T)$ drawn i.i.d. from $\mathcal{P}^{\mathcal{T}}$.

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

Thus,

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

□

4 Online Learning with infinite F for a convex loss

Let's transition into infinite $\mathcal{F}$, 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. $f(x) = \theta^\top x\big)$. The set Θ is convex. We define $\ell_t(\theta) = \ell\left(f_\theta\left(x_t\right), y_t\right)$ convex loss.

ERM Algorithm - also named Follow The Leader (FTL)

For $t = 1$ to T
- Receive x_t
- Choose $\theta_t = \arg\min_{\theta\in\Theta} \sum_{k=1}^{t-1} \ell_k(\theta)$
- Predict $\hat{y}_t = f_t\left(x_t\right)$
- Receive the label y_t, and my prediction costs $\ell\left(\hat{y}_t, y_t\right)$

ERM fails as before because it is "unstable", which means a single example introduced at each time step can lead to substantial changes in θ_t from one step to the next, making it challenging to achieve a low regret.

4.2 Regularized ERM

Due to the instability of ERM, it becomes necessary to introduce a regularizer $C(\theta)$ to stable the ERM in the following way :

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

Algorithme R-ERM - also named Follow The Regularized Leader (FTRL))

$\mathcal{F}_1 = \mathcal{F}$
For $t = 1$ to T
- Receive x_t
- Choose $\theta_t = \arg\min_{\theta\in\Theta} \sum_{k=1}^{t-1} \ell_t(\theta) + \lambda C(\theta)$
- Predict $\hat{y}_t = f_t(x_t)$
- Receive the label y_t, and my prediction costs $\ell(\hat{y}_t, y_t)$
- Often, $C(\theta) = \|\theta\|_2^2$

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

Exercise :
1) write the optimality condition for θ_{t+1}
2) write the optimality condition for θ_t
3) link them

$$\nabla L_{t+1}(\theta_{t+1}) = \sum_{k=1}^{t} g_k + 2\lambda\theta_{t+1} = 0$$

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

$$\nabla L_t(\theta_t) = \sum_{k=1}^{t-1} g_k + 2\lambda\theta_t = 0$$

$$\Rightarrow \theta_t = -\frac{1}{2\lambda} \sum_{k=1}^{t-1} g_k \text{ and } \sum_{k=1}^{t-1} g_k = -2\lambda\theta_t$$

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

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

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

We can first notice that R-ERM and SGD are actually the same algorithm, which is surprising. Secondly, if we wisely choosing the regularization parameter λ, and as long as the gradient of the loss remains within reasonable bounds , which can be achieved by selecting Lipschitz loss, then the algorithm is stable, which means θ_{t+1} is close to θ_t.

As for Mirror Descent, it is out of the scope of the present lecture, but we can say that SGD is a particular case of Mirror Descent.

4.4 Stability of R-ERM

Lemma 1. *If ℓ_t is convex and ρ-Lipschitz, then $\|\theta_{t+1} - \theta_t\| \leqslant \frac{\rho}{\lambda}$, with $C(\theta) = \|\theta\|_2^2$*

4.5 Lemma "Be The Leader (BTL)"

Lemma 2. *Let $\theta^* = \arg\min_\theta \sum_{t=1}^T \ell_t(\theta)$. With R-ERM, we get*

$$\sum_{t=1}^{T} \left(\ell_t\left(\theta_t\right) - \ell_t\left(\theta^*\right)\right) \leqslant \lambda \left\|\theta^*\right\|_2^2 + \sum_{t=1}^{T} \left(\ell_t\left(\theta_t\right) - \ell_t\left(\theta_{t+1}\right)\right)$$

- This lemma shows that if θ_t is stable and ℓ_t is "smooth" in some way, the regret of de R-ERM is low.

4.6 Regret of R-ERM

Theorem 5. *Let ℓ_t, convex differentiable loss. Let $\theta^* = \arg\min_\theta \sum_{t=1}^T \ell_t(\theta)$. Si $\|\theta^*\|_2 \leqslant W_2$, if ℓ_t is ρ-Lipschitz, then with $\lambda = \frac{L\sqrt{T}}{W_2}$ we get :*

$$\text{Regret}_T = \sum_{t=1}^{T} \left(\ell_t\left(\theta_t\right) - \ell_t\left(\theta^*\right)\right) \leqslant 2W_2\rho\sqrt{T}$$