Fondamentaux de l'Apprentissage Automatique

Lecturer: Yann Chevaleyre
Scribe: Artur Dandolini Pescador

Lecture n°6 #
02/nov/2023

---

1 Introduction

Online learning is a dynamic and adaptive process where the model is continuously updated as new data becomes available. It is really useful in situations where data arrives sequentially and when the data distribution may change over time. In this setting, the model must be able to adapt constantly to new information and make predictions in real-time.

2 Standard Setting (Batch)

Before we dive into the details of online learning, let's first define the standard setting for machine learning, which is also known as batch learning. In this setting, the learner receives a training set $S = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ of N sample pairs. Each pair (x_i, y_i) consists of an input x_i and its corresponding output y_i, drawn from an underlying unknown distribution p. The goal of the learning algorithm is to generate a hypothesis f_S that accurately maps inputs to outputs.

The performance of f_S is measured by its ability to predict accurately on new, unseen data, which involves the assessment of risk (empirical risk and true risk).

The empirical risk is the average loss over the training set and serves as an approximation of the true risk. It is defined below as :

$$\hat{\mathcal{R}}(f_S) = \frac{1}{N} \sum_{i=1}^{N} l(f_S(x_i), y_i)$$

The true risk is the expected loss across the entire data distribution as it is possible to observe in the equation below.

$$\mathcal{R}(f_S) = \mathbb{E}_{(x,y)\sim p}[l(f_S(x), y)]$$

Ideally, the learning algorithm aims to minimize this true risk, as it corresponds to the best average performance on all potential data points. However, this is not directly possible since the entire data distribution is inaccessible.

3 Online Learning Protocol

Online learning is an iterative process where a learning algorithm repeatedly makes predictions over time, updates its model based on new data, and receives feedback to improve future predictions. The following protocol show the geral framework :

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 $l(\hat{y}, y)$.

The main objective is to minimize the follow cumulative loss :

$$P_T = \sum_{t=1}^{T} l(\hat{y}_t, y_t)$$

In summary, this protocol describes an iterative learning process where the algorithm continuously learns from new data by making predictions, receiving feedback, and updating its understanding with the goal of improving its predictive accuracy over time.

4 Realizable case with 0/1 loss and finite hypothesis class $\mathcal{F}$

In the realizable case scenario of online learning, we work under the assumption that there exists an ideal function f^* that can predict the correct outputs without any error.

4.1 ERM Algorithm (Empirical Risk Minimization)

The principle behind ERM is to choose a classifier that minimizes the empirical risk, which is the average loss over a sample of data. It is useful when dealing with a finite hypothesis class, denoted by $\mathcal{F}$. We assume the existence of a realizable case where there exists at least one hypothesis in $\mathcal{F}$ that can classify the training data with zero error.

Algorithm 1 ERM Formulation

Set $F_1 = \mathcal{F}$
for $t = 1$ to T **do**
  Receive x_t
  Choose arbitrarily $f_t \in F_t$ among those who perfectly classify previous data
  Predict $\hat{y}_t = f_t(x_t)$
  Receive the true label y_t, and the prediction costs loss $l(\hat{y}_t, y_t)$
  Update $F_{t+1} = \{f \in F_t : f(x_t) = y_t\}$
end for

After T iterations, we hope to have identified a classifier within $\mathcal{F}$ that has a low empirical risk on the sample data.

4.1.1 Alternative formulation

Algorithm 2 ERM Alternative Formulation

Set $F_1 = \mathcal{F}$
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 the prediction costs loss $l(\hat{y}_t, y_t)$
  Update $F_{t+1} = \{f \in F_t : f(x_t) = y_t\}$
end for

4.1.2 Failure of ERM : A Worst Case Scenario

Let us consider a binary classification problem where the input space is $X = \{0, 1\}$ and the output space is $Y = \{-1, 1\}$. Let our hypothesis class be a finite set of classifiers $F = \{f_0, f_1, \ldots, f_m\}$, where each classifier f_i is defined as :

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

We assume that, among all valid functions f in F, we always pick the first one.

Now, consider an adversarial environment that presents instances in such a way as to maximize the empirical risk. The process is as follows :

1. At $t = 1$:
   — The adversary chooses $x_1 = \frac{1}{m}$, and sets $y_1 = 1$.
   — Since $F_1 = F$, we initially choose $f = f_0$.
   — The prediction $\hat{y}_1 = f_0(x_1)$ differs from y_1 lead to an error with loss $l(\hat{y}_1, y_1) = 1$.
   — We update $F_2 = F \setminus \{f_0\} = \{f_1, \ldots, f_m\}$.
2. At $t = 2$:
   — The adversary selects $x_2 = \frac{2}{m}$, with $y_2 = 1$.
   — We pick f_1 from the updated set.
   — The prediction $\hat{y}_2 = f_1(x_2) = -1$ leads to an error, with loss $l(\hat{y}_2, y_2) = 1$.
3. This process continues until step M, with each step incurring an error.
4. At step $M - 1$, we accumulate a loss sum of $M - 1$, which represents the worst case ($loss = \sum_{t=1}^{M-1} l(\hat{y}_t, y_t) = M - 1$)

In this scenario, the correct classifier would have been f_m, which is only selected after incurring the maximum possible loss (worst case scenario). This illustrates a failure case for the ERM principle that leads to the largest cumulative loss.

4.2 Halving Algorithm

The Halving Algorithm is a method used within a family of classifiers to address the computational cost associated with prediction. The goal is to efficiently reduce the number of classifiers that need to be evaluated.

Algorithm 3 Halving Algorithm

Let F be a family of classifiers
Initialize $F_1 \leftarrow F$
for $t = 1$ to T **do**
  Receive x_t
  for all $k \in Y$ **do**
    Let $F_t^k \leftarrow \{f \in F_t : f(x_t) = k\}$
  end for
  Predict $\hat{y}_t \leftarrow \arg\max_k |F_t^k|$
  Receive the true label y_t, and prediction costs $l(\hat{y}_t, y_t)$
  Update $F_{t+1} \leftarrow \{f \in F_t : f(x_t) = y_t\}$
end for

4.2.1 Halving Analysis

Theorem 1. *In the two class setting, let* $l_t = \mathbb{1}\{\hat{y}_t \neq y_t\}, \quad \sum_{t=1}^{T} l_t \leqslant \log_2 |\mathcal{F}|$

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

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

□

4.3 Generic Randomized Algorithm

The Generic Randomized Algorithm is designed for online learning where the learner's goal is to select classifiers from a family F to make predictions over a series of trials. The selection of classifiers is guided by a probability distribution P_t, which evolves as the learner gains more information from the data received. Below it is possible to see the step-by-step of this algorithm :

Algorithm 4 Generic Randomized Algorithm

Let F be a family of classifiers
Let P_t be a probability distribution over F
Initialize $F_1 \leftarrow F$
for $t = 1$ to T **do**
  Receive x_t
  Draw $f_t \sim P_t$
  Predict $\hat{y}_t \leftarrow f_t(x_t)$
  Receive the true label y_t, and prediction costs $l(\hat{y}_t, y_t)$
  Update $F_{t+1} \leftarrow \{f \in F_t : f(x_t) = y_t\}$
end for

4.4 Randomized Algorithm in the realizable case

This variant of the algorithm operates under the assumption of the realizable case, where there exists at least one classifier f^* in the family F that can predict without any error. In this setting, the algorithm maintains a uniform probability distribution over the current set of potential classifiers as before. Here is the algorithm that describes this process :

Algorithm 5 Randomized Algorithm in the realizable case

Let F be a family of classifiers
Choose P_t to be the uniform distribution over F_t
Initialize $F_1 \leftarrow F$
for $t = 1$ to T **do**
  Receive x_t
  Draw $f_t \sim P_t$
  Predict $\hat{y}_t \leftarrow f_t(x_t)$
  Receive the true label y_t, and prediction costs $l(\hat{y}_t, y_t)$
  Update F_{t+1} and P_{t+1}
end for

4.4.1 Analysing the randomized algorithms

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

$$\mathbb{E}f_1, \ldots, f_t \sim \mathcal{U}(\mathcal{F}_1), \ldots, \mathcal{U}(\mathcal{F}_t)\Big[\sum k = 1^T l_t\Big] \leq \ln |\mathcal{F}|$$

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

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

And :

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

Furthermore :

$$1 \leq \Omega_{t+1} \leq \Omega_1 \prod_{k=1}^{t} \mathbb{P}(l_k = 0)$$

$$0 \leq \ln(\Omega_1) + \sum_{k=1}^{t} \ln(\mathbb{P}(l_k = 0))$$

$$0 \leq \ln(\Omega_1) + \sum_{k=1}^{t} \ln(1 - \mathbb{P}(l_k = 1))$$

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

$$0 \leq \ln(\Omega_1) - \sum_{k=1}^{t} \mathbb{P}(l_k = 1)$$

$$\mathbb{E}\Big[\sum_{k=1}^{t} l_k\Big] \leq |\mathcal{F}| \qquad (\forall t)$$

□

5 Non realizable case with finite $\mathcal{F}$

5.1 Regret notion

The cumulated loss $\sum_{t=1}^{T} l(f_t(x_t), y_t)$ can tend to infinite. Therefore, we introduce the notion of regret :

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

An algorithm is said to be no-regret if $\frac{1}{T}\text{Regret} \to 0$ as $T \to \infty$.

5.2 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 :

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

And so :

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

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

□

5.3 Randomized Algorithm in the non realizable case

The randomized algorithm works for any bounded loss function $l(.,.) \leq c$. Let $\beta \in (0,1)$. Choose $P_t(f) = \frac{1}{\Omega_f} w_{f,t}$ where $\Omega_f = \sum_{f \in F_t} w_{f,t}$.

$$w_{f,1} = 1$$

and

$$w_{f,t+1} = w_{f,t} e^{(-\beta l(f(x_t), y_t))}$$

for some constant $\beta > 0$.

Algorithm 6 Hedge Algorithm

Let F be a family of classifiers
Let P_t be a probability distribution over F
Initialize $F_1 \leftarrow F$
for $t = 1$ to T **do**
  Receive x_t
  Draw $f_t \sim P_t$
  Predict $\hat{y}_t \leftarrow f_t(x_t)$
  Receive the true label y_t, and prediction costs $l(\hat{y}_t, y_t)$
  Update $F_{t+1} \leftarrow \{f \in F_t : f(x_t) = y_t\}$
  Update $P_{t+1} \leftarrow \text{Hedge}(F_{t+1})$
end for

5.4 Analyzing Hedge

The following theorem (theorem 4) provides an upper bound on the expected regret, which is a measure of how much worse the learner's predictions are compared to the best possible classifier in hindsight.

Theorem 4. $E[Regret] \leq c\sqrt{2T \ln |F|}$

The expected regret of the Hedge Algorithm, $E[Regret]$, is bounded above by $c\sqrt{2T \ln |F|}$, where c is a constant.

6 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).

We consider the sequence $S = \{(x_t, y_t)\}_{t=1}^{T}$ to be drawn from the distribution P^T. After running an online learning algorithm with a no-regret guarantee, the learner outputs a function $\bar{f}$ chosen uniformly at random from the set of functions $\{f_1, \ldots, f_T\}$ generated during the learning process.

The performance of $\bar{f}$ can be evaluated in terms of its expected risk, $R(\bar{f})$. If the expected regret of the online learner is bounded by a term UB, then the expected risk of $\bar{f}$ is also bounded as follows :

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

where f^* is the best classifier in the family F.

A practical implication of this result is the performance guarantee for the majority classifier, which aggregates the predictions of $\{f_1, \ldots, f_T\}$. This classifier, constructed from the functions produced by the no-regret learner, is a PAC learner.

To study the true expected risk of the majority classifier f_g, we examine the sample $(z_1, y_1), \ldots, (z_T, y_T)$ drawn i.i.d. from a distribution $\mathcal{R}$. The risk is given by :

$$\mathcal{R}(f_g) = \mathbb{E}_S \left[\frac{1}{T} \sum_{t=1}^{T} l(f_g(z_t), y_t) \right],$$

which, by virtue of the no-regret property and Jensen's inequality, is upper bounded as follows :

$$\mathcal{R}(f_g) \leq \min_{f \in \mathcal{F}} \mathbb{E}_S \left[\frac{1}{T} \sum_{t=1}^{T} l(f(z_t), y_t) \right] + \frac{UB}{T} \leq UB.$$

This upper bound on the expected risk implies that the majority classifier is not only no-regret but also approaches the performance of the best possible classifier as more data is observed.

It is pertinent to recall Jensen's inequality in this context, which states that for a convex function φ, the expectation of φ over a random variable X is greater than or equal to φ applied to the expectation of X :

$$\mathbb{E}[\varphi(X)] \geq \varphi(\mathbb{E}[X]).$$

This inequality plays a critical role in establishing the risk bound for the majority classifier.

7 Online Learning with infinite $\mathcal{F}$ for a convex loss

7.1 ERM

We consider that f $\in \mathcal{F}$ is represented by a vector $\theta \in \Theta \subseteq \mathbb{R}^d$. The set Θ is convex. We define $l_t(\theta) = l(f_t(x_t), y_t)$ convex loss.

So, here is the ERM algorithm presented in pseudocode :

Algorithm 7 ERM (or Follow the Leader - FTL)

Initialize $\theta_1 \in \Theta$
for $t = 1$ to T **do**
  Receive x_t
  Choose $\theta_t = \arg\min_{\theta \in \Theta} \sum_{k=1}^{t-1} l_k(\theta)$
  Predict $\hat{y}_t \leftarrow f_{\theta_t}(x_t)$
  Receive the label y_t, and prediction costs $l(\hat{y}_t, y_t)$
end for

ERM fails as demonstrated before because it is "unstable".

7.2 Regularized ERM

Let's consider that $f \in \mathcal{F}$ is repressented by a vector $\theta in \Theta \subseteq \mathbb{R}^d$. $l_t(\theta) = l(f_t(x_t), y_t)$ is a convex loss.

Below, it is possible to see the Regularized ERM algorithm in pseudocode :

Algorithm 8 Regularized ERM

(or Follow the Regularized Leader - FTRL)

Initialize $\theta_1 \in \Theta$
for $t = 1$ to T **do**
  Receive x_t
  Choose $\theta_t = \arg\min_{\theta\in\Theta} \sum_{k=1}^{t-1} l_k(\theta) + \lambda C(\theta)$
  Predict $\hat{y}_t \leftarrow f_{\theta_t}(x_t)$
  Receive the label y_t, and prediction costs $l(\hat{y}_t, y_t)$
end for

Often, $C(\theta) = \|\theta\|^2$ (ridge regression).

7.3 R-ERM with linear losses, SGD and Mirror Descent

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

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

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

Consider the following exercises :

Exercise 1 : Write the optimality condition for θ_{t+1}.
Exercise 2 : Write the optimality condition for θ_t.
Exercise 3 : Link the optimality conditions for θ_{t+1} and θ_t.

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

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

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

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

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

Alternatively, if using a gradient term :

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

This implies that if the gradient $\nabla_\theta l_t$ is small, the updates to θ are minor, indicating the stability of the algorithm, as consequence θ_{t+1} and θ_t remain close to each other.

7.4 Lemme "Be The Leader (BTL)"

Lemma 1. *Let* $\theta^* = \arg\min_{\theta \in \mathbb{R}^d} \sum_{t=1}^{T} l_t(\theta)$. *With R-ERM, we get :*

$$\sum_{t=1}^{T} (l_t(\theta_t) - l_t(\theta^*)) \leq \lambda \|\theta^*\|_2^2 + \sum_{t=1}^{T} (l_t(\theta_t) - l_t(\theta_{t+1}))$$

This lemma (lemma 1) shows that if θ_t is stable and l_t is "smooth" in some way, the regret of R-ERM is low.

7.5 Stability of R-ERM

The stability of the R-ERM algorithm is important to ensures the robustness and generalizability of the learning process. The following lemma (lemma 2) provides a guarantee that the parameter updates are bounded.

Lemma 2. *If* l_t *is convex and* $\rho - Lipschitz$, *then :*

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

7.6 Regret of R-ERM

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

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

where $\hat{\theta}_t$ *is the estimate at time* t.

8 Conclusion

Online learning represents a powerful framework for machine learning in the modern age, where data is abundant and dynamic. By continually learning from new information, online learning algorithms can provide insights and make decisions that keep pace with the rapidly changing world.