Topics in ML Lecture 1
Introduction

Lénaïc Chizat*

September 8, 2025

1 Information on the course

- Taught by Lénaïc Chizat (https://lchizat.github.io/) and the main assistant is Guillaume Wang (https://guillaumew16.github.io/)
- Some practical sessions involve small coding exercices (in the language of your choice). Bring your laptop by default.
- Validation: written exam
- Ask questions!
- References, lecture notes, info on the moodle of the course. The main reference is Francis Bach's book which is freely available https://www.di.ens.fr/~fbach/ltfp_book.pdf.

2 Goal of the course

- Understand, with mathematical tools, how and why *supervised learning* succeeds in solving complicated ML tasks in practice. Few results provided without proofs.
- We will start from classical and basic topics (kernel methods, generalization bounds, etc) and then move towards active research topics (infinite neural networks, analysis of training dynamics, etc).
- Through simple practical sessions on synthetic data, associate the theoretical analysis with the practical performance.
- This is not a "hands-on" class on artificial intelligence, nor an introduction to ML

3 Supervised learning

- Main goal: given some observations $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$, $i = 1, \dots, n$ of inputs/outputs, features/labels (*training data*), build a function that, from a new $x \in \mathcal{X}$, predicts $y \in \mathcal{Y}$ (testing data).
- Examples of data:
  - $\mathcal{X}$: images, speech, text, social networks, sensors, gene expression, etc.

*EPFL lenaic.chizat@epfl.ch

- $\mathcal{Y}$: Classification: binary labels $\mathcal{Y} = \{0, 1\}$ or $\{-1, 1\}$ or multiclass $\mathcal{Y} = \{1, \dots, k\}$ (classification) or real $\mathcal{Y} = \mathbb{R}$ (regression). More generally: structured output (graph prediction, source separation).

- Why is it difficult?
  - y is not a deterministic function of x (labellers making mistakes, dependence on unobserved quantities).
  - Only a few x observed: need interpolation and extrapolation
  - The dependency may be highly non-linear, "compositional".

- Main formalization: see $(x_i, y_i)_{i=1}^n$ as independent realizations of random variables (X, Y) and the criterion is to minimize the expectation of performance on a new sample (assumption never met in practice, but convenient for theory).

- A ML algorithm $\mathcal{A}$ is thus a function $(\mathcal{X} \times \mathcal{Y})^n \to (\mathcal{X} \to \mathcal{Y})$

- Practical performance evaluation: training set, validation set, testing set (only used once!)

4 Decision theory

Question: What is the optimal performance, regardless of the training data?

- Consider a distribution $\rho = \rho_{X,Y}$ on $\mathcal{X} \times \mathcal{Y}$ with marginal distribution ρ_X on $\mathcal{X}$

- Consider a loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$; $\ell(y, z)$ is the loss of predicting z while the true label is y. We'll focus on:
  - Binary classification: $\mathcal{Y} = \{0, 1\}$ (or often $\mathcal{Y} = \{-1, 1\}$) and $\ell(y, z) = 1_{y \neq z}$ ("0-1" loss, or "error"). $\mathcal{R}(f) = \mathbf{P}(f(X) \neq Y)$.
  - Regression: $\mathcal{Y} = \mathbb{R}$ and $\ell(y, z) = (y - z)^2$ (square loss). $\mathcal{R}(f) = \mathbf{E}[(Y - f(X))^2]$.

- Risk (generalization performance) of a *predictor* $f : \mathcal{X} \to \mathcal{Y}$

$$\mathcal{R}(f) = \mathbf{E}[\ell(Y, f(X))].$$

  Be careful with the randomness of lack thereof of f: f depends on the training data and not on the testing data, and thus $\mathcal{R}(f)$ is typically random because of the dependence on the training data or because of randomness in the training method. The dependency in ρ is implicit (we sometimes write $\mathcal{R}_\rho(f)$ to disambiguate).

- What is the best predictor f? Conditioning on x:

$$\mathcal{R}(f) = \mathbf{E}[\ell(Y, f(X))] = \mathbf{E}_{x \sim \rho_\mathcal{X}}\big[\underbrace{\mathbf{E}[\ell(Y, f(X))|X = x]}_{\text{Conditional risk}}\big].$$

  Minimizing the expression for each $x \in \mathcal{X}$ independently, we obtain (ignoring measurability issues):

  Proposition 4.1 (Bayes predictor)**.** *The risk is minimized at a* Bayes predictor $f^* : \mathcal{X} \to \mathcal{Y}$ *satisfying for all* $x \in \mathcal{X}$, $f^*(x) \in \arg\min_{z \in \mathcal{Y}} \mathbf{E}[\ell(Y, z)|X = x]$.

- We call the Bayes risk $\mathcal{R}^*$ the risk of all Bayes predictors (which is usually non-zero unless Y is a deterministic function of X).

5 Learning from data: Empirical risk minimization

- Consider a parameterized family of prediction functions $f : \Theta \times \mathcal{X} \to \mathcal{Y}$, denote $f_\theta = f(\theta, \cdot)$ and "minimize" the empirical risk

$$\hat{\mathcal{R}}(f_\theta) = \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, f_\theta(x_i))$$

  where Θ is the set of parameters/weights.

- For classification, we consider *convex surrogates* of the 0-1 loss to make the problem more tractable. Exemples (with $y \in \mathcal{Y} = \{-1, 1\}$ but predictions in $z \in \mathbb{R}$) are:
  - Hinge loss: $\ell(y, z) = (1 - yz)_+$
  - Square loss: $\ell(y, z) = (y - z)^2 = (1 - yz)^2$
  - Logistic loss: $\ell(y, z) = \log(1 + \exp(-yz))$
  - Exponential loss: $\ell(y, z) = \exp(-yz)$

  At test time, we use the predictor $\text{sign}(f_\theta(x))$.

- plot

- Typically, θ is obtained by a variant of gradient descent on $F(\theta) = \hat{\mathcal{R}}(f_\theta)$ (this is the "learning" phase). The problem is convex in function space as soon as $\ell(y, \cdot)$ is, but usually non-convex in parameter space, unless f is linear in θ.

- Examples we'll consider in this class:
  - linear predictors: $f(\theta, x) = \theta^\top x$ with $\mathcal{X}, \Theta \subset \mathbb{R}^d$.
  - non-linear predictors linearly parameterized $f(\theta, x) = \Phi(x)^\top \theta$ where $\Phi : \mathcal{X} \to \mathbb{R}^p$ is a *feature map*.
  - linear predictors non-linearly parameterized $f(\theta, x) = x^\top \psi(\theta)$ for some $\psi : \mathbb{R}^p \to \mathbb{R}^d$ (e.g. $\psi(\theta)_j = \theta_j^2$ for $j = 1, \dots, d$). This does not change the class of predictors, but changes the behavior of the learning algorithm.
  - two-layer neural networks, with $\theta = (a_j, b_j, c_j)_{j=1}^m \in (\mathbb{R}^d \times \mathbb{R} \times \mathbb{R})^m$

$$f(\theta, x) = \sum_{j=1}^{m} c_j \sigma(x^\top a_j + b_j)$$

    where $\sigma : \mathbb{R} \to \mathbb{R}$ is an activation function, such as $\sigma(s) = (s)_+ = \max\{0, s\}$ (Rectified Linear Unit "ReLU").
  - more general neural networks ("deep learning"). E.g. a L-layer fully-connected neural network of width m with parameters $\theta = (W_1, W_2, \dots, W_L) \in \mathbb{R}^{m \times d} \times \mathbb{R}^{m \times m} \times \cdots \times \mathbb{R}^{1 \times m}$

$$f(\theta, x) = W_L \sigma(\dots W_2 \sigma(W_1 x) \dots)$$

    and $\sigma : \mathbb{R}^m \to \mathbb{R}^m$ a non-linearity. A large variety of "architectures" are used in practice (this one is not).

- Remember in this course: n number of data points, d dimension of the input, p number of parameters (m width of a neural network).

6 Risk decomposition

Given the output of the learning algorithm $f_{\hat{\theta}}$ the excess risk is

$$\mathcal{R}(f_{\hat{\theta}}) - \mathcal{R}^* = \underbrace{\left\{\mathcal{R}(f_{\hat{\theta}}) - \inf_{\theta' \in \Theta} \mathcal{R}(f_{\theta'})\right\}}_{\text{Estimation error}} + \underbrace{\left\{\inf_{\theta' \in \Theta} \mathcal{R}(f_{\theta'}) - \mathcal{R}^*\right\}}_{\text{Approximation error}}.$$

- When the "size" of Θ grows (e.g. more parameters, or larger balls), the approximation error (which is deterministic) decreases, the estimation error (which is random) goes up. There is nuance in this story as the notion of "size" is subtle, see lecture on "Benign Overfitting".
- Plot.
- Typically, for ERM the estimation error is further decomposed as

$$\begin{aligned}\left\{\mathcal{R}(f_{\hat{\theta}}) - \mathcal{R}(f_{\theta'})\right\} &= \underbrace{\left\{\mathcal{R}(f_{\hat{\theta}}) - \hat{\mathcal{R}}(f_{\hat{\theta}})\right\}}_{\text{Generalization}} + \underbrace{\left\{\hat{\mathcal{R}}(f_{\hat{\theta}}) - \hat{\mathcal{R}}(f_{\theta'})\right\}}_{\text{Empirical optimization}} + \underbrace{\left\{\hat{\mathcal{R}}(f_{\theta'}) - \mathcal{R}(f_{\theta'})\right\}}_{\text{Concentration}} \\ &\overset{(a)}{\leq} 2 \sup_{\theta \in \Theta} \left|\hat{\mathcal{R}}(f_\theta) - \mathcal{R}(f_\theta)\right| + \text{empirical optimization error}\end{aligned}$$

- In the iid data setting, the "concentration" term is just the convergence of an empirical average, it goes to 0 by the Law of Large Numbers. However, this argument does not work for the term "generalization" since $\hat{\theta}$ is not independent from $\hat{\mathcal{R}}$: there is the danger of *overfitting*[1]!
- Using the inequality (a) allows the "standard" separation of the analysis into three parts (i) approximation (ii) estimation (via uniform concentration) (iii) optimization. Our analysis will often follow this structure.
- However, the uniform concentration approach is often too loose (i.e. the "generalization" term is much smaller than the supremum). We will come back to this.

7 Statistical Learning Theory

- To bound the excess risk, a common assumption is that the data $\mathcal{D}_n(\rho) = (x_i, y_i)_{i=1}^n$ is given by i.i.d. observations from the test distribution ρ. A learning algorithm is a function $\mathcal{A} : (\mathcal{X} \times \mathcal{Y})^n \to (\mathcal{X} \to \mathcal{Y})$. An example is $\mathcal{A}(\mathcal{D}_n(\rho)) = f_{\hat{\theta}}$ the minimizer of ERM, other example exist such as k-nearest neighbor.
- Q: Since the estimated predictor is random (it depends on random samples, and there might also be randomness in the algorithm), how to measure its performance?
- Two types of bounds are often found. Let $Q \subset \mathcal{P}(\mathcal{X} \times \mathcal{Y})$ represent the assumptions on the data (e.g.: Q contains the laws of (X, Y) where $\|X\| \leq R$ almost surely and $Y = f^*(X)$ for some L-Lipschitz f).
  - *In expectation* bounds: find $\alpha(n)$ such that

$$\sup_{\rho \in Q} \mathbf{E}\left[\mathcal{R}_\rho(\mathcal{A}(\mathcal{D}_n(\rho))) - \mathcal{R}^*\right] \leq \alpha(n)$$

[1] It is a general fact of life that a "good proxy" of a performance indicator can become a bad proxy when it is used a target objective (see Goodheart's/Campbell's law). Overfitting is everywhere!

– *Probably Approximately Correct* (PAC) bounds: for all $\epsilon > 0$, find $\delta(\epsilon, n) \in [0, 1]$ such that for any $\rho \in Q$

$$\mathbf{P}\big(\mathcal{R}(\mathcal{A}(\mathcal{D}_n(\rho))) - \mathcal{R}^* \leq \epsilon\big) \geq 1 - \delta(\epsilon, n).$$

Alternatively, one can express ϵ a as function of δ and n.

- One can go from bound "in expectation" to a PAC bound by using concentration. One can go from a PAC to "in expectation" by integrating the tails. (exercice)
- Lower bounds: in some settings it is possible to show that the infimum over all algorithms is greater than a certain quantity.
- Asymptotic vs. non-asymptotic analysis. None is better than the other: there is often a trade-off between precision/generality/strength of the theoretical results; and various approaches lead to different trade-offs.

8 No free lunch theorems

This is a family/kind of theorems that tell us that we should not expect too much from a learning algorithm, and in particular that:

- (i) Learning is not possible without assumptions.
- (ii) No method is better on all problems.

Here are two examples.

Theorem 8.1. *Consider the binary classification with* 0-1 *loss, with* $\mathcal{X}$ *having at least* k *elements and* $\mathcal{Y} = \{0, 1\}$. *For any* $n \in \mathbb{N}^*$ *and learning algorithm* $\mathcal{A}$,

$$\sup_{\rho \in \mathcal{P}(\mathcal{X} \times \mathcal{Y})} \mathbf{E}\big[\mathcal{R}_\rho(\mathcal{A}(\mathcal{D}_n(\rho)))\big] - \mathcal{R}^*_\rho \geq \frac{1}{2}(1 - 1/k)^n.$$

In particular, if $\mathcal{X}$ *is infinite, the lower bound is* $1/2$.

[Guided proof in exercices 1].

Theorem 8.2 (Thm. 7.2 Devroye et al. [2013])**.** *Consider the binary classification with* 0-1 *loss, with* $\mathcal{X}$ *infinite and* $\mathcal{Y} = \{0, 1\}$. *For any decreasing sequence* a_n *tending to zero and such that* $a_1 \leq 1/16$, *for any learning algorithm* $\mathcal{A}$, *there exists* $\rho \in \mathcal{P}(\mathcal{X} \times \mathcal{Y})$ *such that*

$$\mathbf{E}[\mathcal{R}_\rho(\mathcal{A}(\mathcal{D}_n(\rho)))] - \mathcal{R}^*_\rho \geq a_n, \qquad \forall n \geq 1.$$

Make sure you understand the difference between these two statements (why is the second statement stronger?).

9 State of ML theory

Until the mid-2010s, theoretical machine learning was the main driver of progress in supervised learning — most state-of-the-art methods were variants of kernel approaches, i.e. "linearly parameterized methods" in the terminology of Section 5. With the rise of neural networks (deep learning) around 2012, practical advances have since been driven largely by empirical exploration and heuristics. In fact, a large research community collectively "chasing" state-of-the-art performance through trial and error has proven to be a remarkably powerful engine

of progress. This shift has reshaped the role of theory and redefined the questions that are most relevant for it to address.

To navigate the research literature, it is important to understand that there are now various kind of theoretical approaches to ML that have different aims and rationals, because "end-to-end" theoretical analyses are out of reach for state-of-the art techniques. These include:

- Illustrative: In-depth study a toy problem in order to illustrate a phenomenon that is observed in practice in more complicated models;
- Descriptive (my favorite): Study "realistic models" and describe their behavior, without aiming at complete end-to-end learning guaranties. This approach is useful to obtain practical guidelines (choice of hyper-parameters, architecture design, etc);
- Classifying learning problems and algorithms (traditional approach in computer science): define classes of learning problems and learning algorithms, and study their computational and statistical complexity for instance in the PAC framework (here usually the algorithms studied are "ad-hoc" and taylored for a particular theoretical question, and not the ones which are used in practice).

This blog post[2] is a good complementary reading on this topic.

References

Luc Devroye, László Györfi, and Gábor Lugosi. *A probabilistic theory of pattern recognition*, volume 31. Springer Science & Business Media, 2013.

[2] https://windowsontheory.org/2021/01/15/ml-theory-with-bad-drawings/