Machine learning - Master ICFP 2019-2020

Introduction to supervised learning

Francis Bach

January 10, 2020

1 Information on the course

- Co-taught by Francis Bach (`https://www.di.ens.fr/~fbach/`)
  and Lenaïc Chizat (`https://lchizat.github.io/`). Please include both of us in email correspondence.
- Lecture with no practical sessions (9am to 12pm)
- Lecture with practical sessions (9am to 12.30pm). Bring your laptop with Python 3 and Jupyter notebooks installed.
  Run `https://www.di.ens.fr/appstat/spring-2019/TP/TD0-prerequisites/crash_test.ipynb` as a test.
- Validation: one written in-class exam, and practical sessions to finish at home.
- Register online.
- Ask questions!
- Some exercises done in lecture.
- References: [1, 2, 3, 4].

2 Goals of the course

- Present the main theories and algorithms in machine learning, with a focus on simple and direct arguments (few results provided without proofs).
- Through simple practical sessions on synthetic and real data, associate these main algorithms with their practical performance.
- This is not a class on artificial intelligence.
- Mostly supervised learning, but also unsupervised.

3 Supervised machine learning: introduction

- Main goal: given some observations $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$, $i = 1, \dots, n$, of inputs/outputs, features/variables (training data), given a new $x \in \mathcal{X}$, predict $y \in \mathcal{Y}$ (testing data).
- Many examples where $\mathcal{X}$ and $\mathcal{Y}$ can be very diverse, from many areas of science and engineering:
  - $\mathcal{X}$: images, sounds, videos, text, proteins, web pages, social networks, sensors from industry, etc.
  - $\mathcal{Y}$: binary labels $\mathcal{Y} = \{0, 1\}$, real response $\mathcal{Y} = \mathbb{R}$, multiclass $\mathcal{Y} = \{1, \dots, k\}$, more generally structured outputs (e.g., graph prediction, visual scene analysis, source separation).
- Why is it difficult?
  - y is not a deterministic function of x: given $x \in \mathcal{X}$, noisy outputs $y = f(x)$ "+" ε, e.g., noise due to diverging views between labellers, dependence on external unobserved quantities (that is $y = f(x, z)$, z random).
  - f may be quite complex (highly non-linear).
  - Only a few x's are observed: need interpolation and potentially extrapolation (see below), and overfitting is always a possibility.
  - Weak link between training and testing distributions.
  - What is the criterion for performance?
- Main formalization: see (x_i, y_i) as a realization of random variables (X_i, Y_i), and the criterion is to minimize the expectation of "performance" with respect to the distribution of the test data.
  - Classical (never met in practice...) assumptions: the random variables (X_i, Y_i) are independent and identically distributed with the same distribution as the testing distribution.
  - A machine learning algorithm $\mathcal{A}$ is a function that goes from $(\mathcal{X} \times \mathcal{Y})^n$ to a function from $\mathcal{X}$ to $\mathcal{Y}$.
- Practical performance evaluation:
  - Training set, validation set and testing set.
  - Cross-validation to use a maximal amount of training data.
  - Formally, the test set can only be used once!
  - "Debugging" a machine learning implementation is an art.

4 Decision theory

Main question: what is the optimal performance, regardless of the training data?

- We consider a fixed (testing) distribution $P = \rho_{X,Y}$ on $\mathcal{X} \times \mathcal{Y}$, with marginal distribution $\rho_{\mathcal{X}}$ on $\mathcal{X}$.
- We 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.

- Binary classification: $\mathcal{Y} = \{0, 1\}$ (or often $\mathcal{Y} = \{-1, 1\}$), and $\ell(y, z) = 1_{y \neq z}$ ("0-1" loss). $\mathcal{R}(f) = \mathbb{P}(f(X) \neq Y)$. Extendable to multiclass $\mathcal{Y} = \{1, \dots, k\}$.
- Regression: $\mathcal{Y} = \mathbb{R}$ and $\ell(y, z) = (y - z)^2$ (square loss). $\mathcal{R}(f) = \mathbb{E}(Y - f(X))^2$.
- Many other potential losses! E.g., Hamming loss for $\mathcal{Y} = \{0, 1\}^k$, loss for ranking, etc.

- Risk, or generalization performance of a prediction function $f : \mathcal{X} \to \mathcal{Y}$:

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

  Be careful with the randomness or 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.

  It depends on the distribution P on (X, Y). We sometimes use the notation $\mathcal{R}_P(f)$ to make it explicit.

  Note that sometimes, we consider random predictions, that is for any x, we output a distribution on y, and then the risk is taken as the expectation over the randomness of the outputs.

- Main question: what is the best prediction function f?

- Using conditional probabilities (with somewhat informal notations), we have

$$\mathcal{R}(f) = \mathbb{E}\ell(Y, f(X)) = \mathbb{E}\big[\mathbb{E}(\ell(Y, f(X))|X)\big] = \mathbb{E}_{x \sim \rho_X}\big[\mathbb{E}(\ell(Y, f(X))|X = x)\big].$$

  Given the conditional distribution given any $x \in \mathbb{R}^d$, that is $Y|X = x$, we have the conditional risk

$$r(z|x) = \mathbb{E}(\ell(Y, z)|X = x),$$

  which leads to

$$\mathcal{R}(f) = \mathbb{E}(r(f(X)|X)).$$

  A minimizer of $\mathcal{R}(f)$ can be obtained by considering for any x, the function value $f(x)$ to be equal to a minimizer z of $r(z|x) = \mathbb{E}(\ell(Y, z)|X = x)$. We can therefore consider all x as being treated independently.

- Optimal performance:

  Proposition 1 (Bayes predictor) *The risk is minimized at a* Bayes predictor $f^* : \mathcal{X} \to \mathcal{Y}$ *satisfying for all* $x \in \mathcal{X}$, $f^*(x) \in \arg\min_{z \in \mathcal{Y}} \mathbb{E}(\ell(Y, z)|X = x)$. *The Bayes risk* $\mathcal{R}^*$ *is the risk of all Bayes predictors and is equal to*

$$\mathcal{R}^* = \mathbb{E}_{x \sim \rho_X} \inf_{z \in \mathcal{Y}} \mathbb{E}(\ell(Y, z)|X = x).$$

  Note that (a) the Bayes predictor is not unique, but that all lead to the same Bayes risk, and (b) that the Bayes risk is usually non zero (unless the dependence between x and y is deterministic).

  Definition 1 (Excess risk) *The excess risk of a function from* $f : \mathcal{X} \to \mathcal{Y}$ *is equal to* $\mathcal{R}(f) - \mathcal{R}^*$ *(it is always non-negative).*

- Therefore, machine learning is "trivial": *given* the distribution $Y|X = x$ for any x, the optimal predictor is known.

- Exercise: what is the Bayes predictor for binary classification: $\mathcal{Y} = \{0, 1\}$ and $\ell(y, z) = 1_{y \neq z}$. Extension to multiclass $\mathcal{Y} = \{1, \dots, k\}$? Solution: $f^*(x) \in \arg\max_{i \in \{1,\dots,k\}} \mathbb{P}(Y = i | X = x)$.

- Exercise: what is the Bayes predictor for regression: $\mathcal{Y} = \mathbb{R}$ and $\ell(y, z) = (y - z)^2$? Solution: $f^*(x) = \mathbb{E}(Y|X = x)$. Extension to the loss $\ell(y, z) = |y - z|$? Solution: $f^*(x)$ is a median of the distribution of Y given $X = x$.

- Exercise: we consider a random prediction rule where we predict from the probability distribution of Y given $X = x$. When is this achieving the Bayes risk? Solution: when the Bayes risk is zero.

5 Learning from data

There are two main classes of prediction algorithms, that will be studied in this course:

(1) Local averaging (lecture 4).

(2) Empirical risk minimization (lectures 2, 3, 5, 7, 8).

5.1 Local averaging

Trying to approximate / emulate the Bayes predictor, e.g., $f^*(x) = \mathbb{E}(Y|X = x)$ for least-squares regression, from empirical data. Often by explicit/implicit estimation of the conditional distribution by *local averaging* (k-nearest neighbors, which is used as the main example for this lecture, Nadaraya Watson, decision trees).

- Pros: (a) no optimization or training, (b) often easy to implement, (c) can get very good performance in low dimensions (in particular for non-linear dependences between X and Y).

- Cons: (a) slow at query time: must pass through all training data at each testing point, (b) bad for high-dimensional data (curse of dimensionality, more on this in lecture 4), (c) choice of local distance function, (d) choice of "width" parameters.

- Plot of training error and testing errors as a function of k and n.

5.2 Empirical risk minimization

Consider a parameterized family of prediction functions $f_\theta : \mathcal{X} \to \mathcal{Y}$ for $\theta \in \Theta$ and minimize the empirical risk with respect to $\theta \in \Theta$:

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

Most classical example is linear least-squares regression, where we minimize

$$\frac{1}{n} \sum_{i=1}^{n} (y_i - w^\top \Phi(x_i))^2,$$

where f is linear in some feature vector $\Phi(x) \in \mathbb{R}^d$ (no need for $\mathcal{X}$ to be a vector space). The vector $\Phi(x)$ can be quite large (or even implicit, like in kernel methods). Other examples include neural networks.

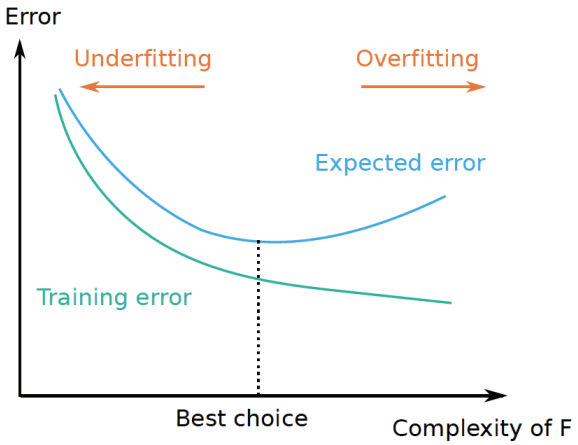
- Pros: (a) can be relatively easy to optimize (e.g., least-squares with simple derivation), many algorithms available (mostly based on gradient descent), (b) can be applied in any dimension (if a reasonable feature vector is available).
- Cons: (a) can be relatively hard to optimize (e.g., neural networks), (b) need a good feature vector for linear methods, (c) dependence on parameters can be complex (e.g., neural networks), (d) need some capacity control to avoid overfitting, (e) how to parameterize functions with values in $\{0, 1\}$?
- Risk decomposition in estimation error + approximation error : given any $\hat{\theta} \in \Theta$,

$$\begin{aligned} \mathcal{R}(f_{\hat{\theta}}) - \mathcal{R}^* &= \Big\{\mathcal{R}(f_{\hat{\theta}}) - \inf_{\theta' \in \Theta} \mathcal{R}(f_{\theta'})\Big\} + \Big\{\inf_{\theta' \in \Theta} \mathcal{R}(f_{\theta'}) - \mathcal{R}^*\Big\} \\ &= \text{estimation error} \quad + \text{approximation error} \end{aligned}$$

  When the "number" of models (e.g., number of parameters) in Θ grows, the approximation error is deterministic and goes down, while the estimation error is random and goes up.

  Examples of approximations by polynomials in one-dimensional regression.

  Typical curve:



  Typically, we will see in later lectures that the estimation error is often decomposed as

$$\begin{aligned} \Big\{\mathcal{R}(f_{\hat{\theta}}) - \mathcal{R}(f_{\theta'})\Big\} &= \Big\{\mathcal{R}(f_{\hat{\theta}}) - \hat{\mathcal{R}}(f_{\hat{\theta}})\Big\} + \Big\{\hat{\mathcal{R}}(f_{\hat{\theta}}) - \hat{\mathcal{R}}(f_{\theta'})\Big\} + \Big\{\hat{\mathcal{R}}(f_{\theta'}) - \mathcal{R}(f_{\theta'})\Big\} \\ &\leqslant 2 \sup_{\theta \in \Theta} \Big|\hat{\mathcal{R}}(f_\theta) - \mathcal{R}(f_\theta)\Big| + \text{ empirical optimization error.} \end{aligned}$$

  The uniform deviation grows with the "size" of Θ, and usually decays with n.

- Capacity control by reducing number of parameters: leads to constrained optimization.

- Capacity control by regularization: minimize

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

where $\Omega(\theta)$ controls the complexity of f_θ. The main example is ridge regression:

$$\min_{w\in\mathbb{R}^d} \frac{1}{n}\sum_{i=1}^{n}(y_i - w^\top\Phi(x_i))^2 + \lambda\|w\|_2^2.$$

Often easier for optimization, but harder to analyze.

Clear difference between parameters (e.g., θ) learrned on the training data and hyperparameters (e.g., λ) learned on the validation data.

5.3 Statistical learning theory

- The goal is to provide some guarantees of performance on unseen data. A common assumption is that the data $\mathcal{D}_n(P) = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ is obtained as independent and identically distributed (i.i.d.) observations, from a distribution P within a class $\mathcal{P}$ of distributions satisfying some regularity properties (e.g., the inputs live in a compact space, or the dependence between y and x is at most of some complexity).

- An algorithm $\mathcal{A}$ is a mapping from $\mathcal{D}_n(P)$ (for any n) to a function from $\mathcal{X}$ to $\mathcal{Y}$. The risk depends on the probability distribution $P \in \mathcal{P}$, as $\mathcal{R}_P(f)$. The goal is to find $\mathcal{A}$ such that the risk

$$\mathcal{R}_P(\mathcal{A}(\mathcal{D}_n(P)))$$

is small, assuming $\mathcal{D}_n(P)$ is sampled from P, but without knowing which $P \in \mathcal{P}$ is considered. Moreover, the risk is random because $\mathcal{D}_n$ is random. There are several ways of dealing with the randomness to obtain a criterion.

The simplest one is to take the expectation, and we thus aim at finding an algorithm $\mathcal{A}$ such that

$$\sup_{P\in\mathcal{P}} \mathbb{E}\Big[\mathcal{R}_P(\mathcal{A}(\mathcal{D}_n(P)))\Big]$$

is as small as possible. This is typically a function of the sample size n and of properties of $\mathcal{X}$, $\mathcal{Y}$ and the allowed set of problems $\mathcal{P}$ (e.g., dimension of $\mathcal{X}$, number of parameters).

- Lower-bounding the optimal performance: in some set-ups, it is possible to show that the infimum over all algorithms is greater than a certain quantity. Machine learners are happy when upper-bounds and lower-bounds match.

- PAC learning: for a given $\delta \in (0, 1)$ and $\varepsilon > 0$, for any $P \in \mathcal{P}$:

$$\mathbb{P}\Big(\Big[\mathcal{R}_P(\mathcal{A}(\mathcal{D}_n(P)))\Big] \leqslant \varepsilon\Big) \geqslant 1 - \delta.$$

The crux is to find ε which is as small as possible (typically as a function of δ).

- The analysis can be "non-asymptotic", with an upper-bound with explicit dependence on all quantities; the bound is then valid for all n, even if sometimes vacuous (e.g., a bound greater than 1 for a loss uniformly bounded by 1).

  The analysis can also be "asymptotic", where for examples n goes to infinity and limits are taken (alternatively, several quantities can be made to grow simultaneously).

5.4 No free lunch theorems

"Learning is not possible without assumptions." See [2, Chapter 7] for details.

The following theorem shows that for any algorithm, for a fixed n, there is a data distribution that makes the algorithm useless.

Theorem 1 (no free lunch - fixed n) *Consider the binary classification with $0-1$ loss, with $\mathcal{X}$ infinite. Let $\mathcal{P}$ denote the set of all probability distributions on $\mathcal{X} \times \{0,1\}$. For any $n > 0$ and learning algorithm $\mathcal{A}$,*

$$\sup_{P \in \mathcal{P}} \mathbb{E}\Big[\mathcal{R}_P(\mathcal{A}(\mathcal{D}_n(P)))\Big] - \mathcal{R}_P^* \geqslant 1/2.$$

Proof Let k be a positive integer. Without loss of generality, we can assume that $\mathbb{N} \subset \mathcal{X}$.

Given $r \in \{0,1\}^k$, we define the distribution P such that $\mathbb{P}(X = j, Y = r_j) = 1/k$; that is, we choose one of the first k elements uniformly at random, and then Y is selected deterministically as $Y = R_X$. Thus the Bayes risk is zero: $\mathcal{R}_P^* = 0$.

Denoting $\hat{f}_{\mathcal{D}_n} = \mathcal{A}(\mathcal{D}_n(P))$, and $S(r) = \mathbb{E}\Big[\mathcal{R}_P(\hat{f}_{\mathcal{D}_n})\Big]$ the expected risk, we want to maximize $S(r)$ with respect to $r \in \{0,1\}^k$; the maximum is greater than the expectation of $S(r)$ for any distribution R on r, in particular the uniform distribution (each r_j an independent unbiased Bernoulli variable). Then

$$\begin{aligned} \max_{r \in \{0,1\}^k} S(r) &\geqslant \mathbb{E}_{r \sim R} S(r) \\ &= \mathbb{P}(\hat{f}_{\mathcal{D}_n}(X) \neq Y) = \mathbb{P}(\hat{f}_{\mathcal{D}_n}(X) \neq r_X) \end{aligned}$$

because X is almost surely in $\{1, \ldots, k\}$ and $Y = r_X$ almost surely. Then

$$\begin{aligned} \mathbb{E}_{r \sim R} S(r) &= \mathbb{E}\Big[\mathbb{P}\big(\hat{f}_{\mathcal{D}_n}(X) \neq r_X \big| X_1, \ldots, X_n, r_{X_1}, \ldots, r_{X_n}\big)\Big] \\ &\geqslant \mathbb{E}\Big[\mathbb{P}\big(\hat{f}_{\mathcal{D}_n}(X) \neq r_X \ \&\ X \notin \{X_1, \ldots, X_n\} \big| X_1, \ldots, X_n, r_{X_1}, \ldots, r_{X_n}\big)\Big] \\ &= \mathbb{E}\Big[\frac{1}{2}\mathbb{P}\big(X \notin \{X_1, \ldots, X_n\} \big| X_1, \ldots, X_n, r_{X_1}, \ldots, r_{X_n}\big)\Big], \end{aligned}$$

by the law of total expectation on using monotonicity of probabilities, and because we have $\mathbb{P}\big(\hat{f}_{\mathcal{D}_n}(X) \neq r_X \big| X \notin \{X_1, \ldots, X_n\}, X_1, \ldots, X_n, r_{X_1}, \ldots, r_{X_n}\big) = 1/2$. Thus,

$$\mathbb{E}_{r \sim R} S(r) = \frac{1}{2}\mathbb{P}\big(X \notin \{X_1, \ldots, X_n\}\big) = \frac{1}{2}\mathbb{E}\Big[\prod_{i=1}^n \mathbb{P}(X_i \neq X | X)\Big] = \frac{1}{2}\big(1 - 1/k\big)^n.$$

Given n, we can let k tend to infinity to conclude. ■

A caveat is that the hard distribution may depend on n. The following theorem is given without proof and is much stronger [2, Theorem 7.2], as it more convincingly shows that learning can be arbitrarily slow without assumption.

Theorem 2 (no free lunch - sequence of errors) *Consider the binary classification with $0-1$ loss, with $\mathcal{X}$ infinite. Let $\mathcal{P}$ denote the set of all probability distributions on $\mathcal{X} \times \{0,1\}$. For any decreasing sequence a_n tending to zero and such that $a_1 \leqslant 1/16$, for any learning algorithm $\mathcal{A}$, there exists $P \in \mathcal{P}$, such that for all $n \geqslant 1$:*

$$\mathbb{E}\Big[\mathcal{R}_P(\mathcal{A}(\mathcal{D}_n(P)))\Big] - \mathcal{R}_P^* \geqslant a_n.$$

References

[1] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Science & Business Media, 2009.

[2] Luc Devroye, László Györfi, and Gábor Lugosi. *A probabilistic Theory of Pattern Recognition*, volume 31. Springer Science & Business Media, 2013.

[3] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.

[4] Sébastien Bubeck et al. Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 8(3-4):231–357, 2015.