Formulary for Machine learning: Theory and Algorithms

Gabriele Genovese

10 November 2024

1 Introduction

1.1 Empirical risk

$$L_{S(h)} = \frac{|\{i \in [m] : h(x_i) \neq y_i\}|}{m}$$

1.2 Empirical risk minimization

$$\text{argmin}_{h \in H} L_{S(h)}$$

1.3 Expected loss

We now define the *risk function* to be the expected loss of a classifier, $h \in H$, wrt a probability distribution D over Z

$$L_{D(h)} = E_{z \sim D}[l(h, z)]$$

Now the empirical risk is

$$L_{S(h)} = \frac{1}{m} \sum_{i=1}^{m} l(h, z_i)$$

2 Model learning

2.1 PAC learnability

Definition: A hypothesis class H is PAC learnable if there exist a function $m_H : (0,1)^2 \to N$ and a learning algorithm with the following property: $\forall \varepsilon, \delta \in (0,1)$, and $\forall$ distribution D over X, and for every labeling function $f : X \to \{0,1\}$, if the *realizable assumption* holds wrt H, D, f, then when running the learning algorithm on $m \geq m_{H(\varepsilon,\delta)}$ i.i.d. examples generated by D and labeled by f, the algorithm returns a hypothesis h such that $L_{(D,f)}(h) \leq \varepsilon$ with probability $\geq 1 - \delta$ (over the choice of the examples).

Th: A finite class is PAC-learnable. Minitest: H_{all} is not PAC-learnable because it's infinite.

2.2 Agnostic PAC learnability

Definition: A hypothesis class H is *agnostic* PAC learnable wrt to a set Z and a loss function $l : H \times Z \to R_+$, if there exist a function $m_H : (0,1)^2 \to N$ and a learning algorithm with the following property: $\forall \varepsilon, \delta \in (0,1)$, and $\forall$ distribution D over Z, when running the learning algorithm on $m \geq m_{H(\varepsilon,\delta)}$ i.i.d. examples generated by D, the algorithm returns $h \in H$ such that $L_D(h) \leq \min_{h' \in H} L_{D(h')} + \varepsilon$ with probability $\geq 1 - \delta$ (over the choice of m training examples), where $L_{D(h)} = E_{z \sim D}[l(h, z)]$.

Easy: APAC $\Rightarrow$ PAC (perché APAC è più generico).

Difficult: PAC $\Rightarrow$ APAC.

3 Uniform convergence

It's a general framework to prove any finite class is APAC learnable.

Definition: A hypotesis class H has the UC property if $\exists$ a function $m_n^{UC} : (0,1)^2 \to N$ such that $\forall \varepsilon, \delta \in (0,1)$ and $\forall D$ over Z, if the cardinality of S is $m \geq m_n^{UC}(\varepsilon, \delta)$ examples drawn i.i.d. to D, then is ε-representative with probabilty $\geq 1 - \delta$.

3.1 $\frac{\varepsilon}{2}$-representative

A training set S is called ε-representative (w.r.t. domain Z, hypothesis class H, loss function l, and distribution D) if $\forall h \in H, |L_S(h) - L_D(h)| \leq \varepsilon$.

3.1.1 Lemma

Assume S is $\frac{\varepsilon}{2}$-representative (wrt domain Z, H, l, D). Then, any output of $\text{ERM}_{H(S)}$, namely, any $h_S \in \text{argmin}_{h \in H} L_{S(h)}$, satisfies

$$L_D(h_S) \leq \min_{h \in H} L_D(h) + \varepsilon$$

3.1.2 Corollary

If a class H has the UC property with a function m_H^{UC} then the class is agnostically PAC learnable with the sample complexity $m_{H(\varepsilon,\delta)} \leq m_H^{\text{UC}}(\frac{\varepsilon}{2}, \delta)$. Furthermore, in that case, the ERM_H paradigm is a successful agnostic PAC learner for H.

3.2 Big theorem of statistical learning

Let H be a hypo class of functions from a domain $X \to \{0,1\}$ and let the loss function be the $0-1$ loss. Then, the following are equivalent:
1. H has the uniform convergence property.
2. Any ERM rule is a successful agnostic PAC learner for H.
3. H is agnostic PAC learnable.
4. H is PAC learnable.
5. Any ERM rule is a successful PAC learner for H.
6. H has a finite VC-dimension.

It's valid only for binary classificators.

4 Shattering

A hypothesis class H shatters a finite set $C \subset X$ if the restriction of H to C is the set of all functions from C to $\{0,1\}$. That is, $|H_C| = 2^{|C|}$.

4.1 Restiction of H to C

Let H be a class of functions from X to $\{0,1\}$ and let $C = \{c_1, ..., c_m\} \subset X$. The restriction of H to C is the set of functions from C to $\{0,1\}$ that can be derived from H. That is,

$$H_C = \{(h(c_1), ..., h(c_m)) : h \in H\}$$

where we represent each function from C to $\{0,1\}$ as a vector in $\{0,1\}^{|C|}$.

4.2 VC-dimention

The VC-dimension of a hypothesis class H, denoted $\text{VCdim}(H)$, is the maximal size of a set $C \subset X$ that can be shattered by H. If H can shatter sets of arbitrarily large size we say that H has infinite VC-dimension.

5 Bias Complexity Tradeoff

$$L_D(A(S)) = \min_{h \in H} L_D(h) + L_D(A(S)) - \min_{h \in H} L_D(h)$$

$\min_{h\in H} L_D(h)$ represents the *approximation error* (the best possibile loss for your hypotesis class).

$L_D(A(S)) - \min_{h\in H} L_D(h)$ is the *estimation error*.

On one hand, choosing H to be a very rich class decreases the approximation error but at the same time might increase the estimation error, as a rich H might lead to overfitting. On the other hand, choosing H to be a very small set reduces the estimation error but might increase the approximation error or, in other words, might lead to underfitting.

5.1 No free lunch theorem

Intuition: it doesn't exists the perfect learner. It always exists for each binary classifier and for each learner a distribution D on which the learner fails.

Theorem: Let A be any learning algorithm for the task of binary classification with respect to the $0-1$ loss over a domain X. Let m be any number smaller than $\frac{|X|}{2}$, representing a training set size. Then, there exists a distribution D over $X \times \{0,1\}$ such that:
1. There exists a function $f : X \to \{0,1\}$ with $L_D(f) = 0$.
2. With probability of at least $1/7$ over the choice of $S \sim D^m$ we have that $L_D(A(S)) \geq \frac{1}{8}$.

6 Efficient learning

7 Convex

Convex learning are a family of learning problems that we can learn efficiently. In general, a convex learning problem is a problem whose hypothesis class is a convex set, and whose loss function is a convex function for each example.

7.1 Convex set

A set C in a vector space is convex if for any two vectors $u, v \in C$, the line segment between u and v is contained in C. That is, for any $\alpha \in [0,1]$ we have that $\alpha u + (1-\alpha)v \in C$.

7.2 Convex Function

Let C be a convex set. A function $f : C \to R$ is convex if for every $u, v \in C$ and $\alpha \in [0,1]$,

$$f(\alpha u + (1-\alpha)v) \leq \alpha f(u) + (1-\alpha)f(v)$$

In words, f is convex if for any u, v, the graph of f between u and v lies below the line segment joining $f(u)$ and $f(v)$.

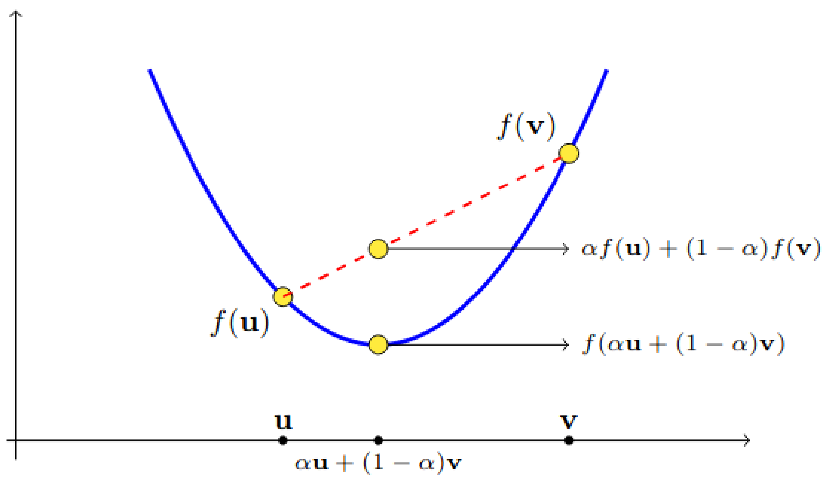
$f(\mathbf{v})$
$\alpha f(\mathbf{u}) + (1-\alpha)f(\mathbf{v})$
$f(\mathbf{u})$
$f(\alpha\mathbf{u} + (1-\alpha)\mathbf{v})$
$\mathbf{u}$
$\mathbf{v}$
$\alpha\mathbf{u} + (1-\alpha)\mathbf{v}$

Abb. 1: Convex function

An important property of convex functions is that every local minimum of the function is also a global minimum

7.3 Lipschitzness

Intuition: a Lipschitz function cannot change too fast.

Let $C \subset R^d$. A function $f : R^d \to R^k$ is ρ-Lipschitz over C if for every $w_1, w_2 \in C$ we have that $\|f(w_1) - f(w_2)\| \leq \rho \, \|w_1 - w_2\|$.

7.4 Convex Learning Problem

A learning problem, (H, Z, l), is called convex if the hypothesis class H is a convex set and for all $z \in Z$, the loss function, $l(\cdot, z)$, is a convex function (where, for any z, $l(\cdot, z)$ denotes the function $f : H \to R$ defined by $f(w) = l(w, z)$).

7.4.1 Lemma

If l is a convex loss function and the class H is convex, then the ERM_H problem, of minimizing the empirical loss over H, is a convex optimization problem (that is, a problem of minimizing a convex function over a convex set).

8 Linear predictors

Linear regression with the squared loss and logistic regression are convex problem and, therefore, efficient to learn.

8.1 Halfspace

$$HS_d = \mathrm{sign} \circ L_d = \{x \to \mathrm{sign}(h_w, b(x)) : h_w, b \in L_d\}$$

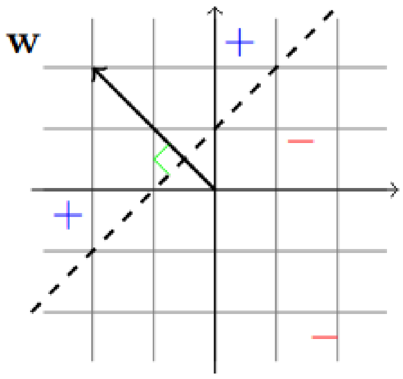

Abb. 2: Halfspace graph rappresentation

8.1.1 Solve Halfspace

Two main approach:

- Linear Programming
- Perceptron (if $\exists$ a wrong classification of the input, update with $w^{t+1} = w^t + y_i x_i$)

8.2 Linear regression

$$H_{\text{reg}} = L_d = \{x \to \langle\, w, x \,\rangle + b : w \in R^d, b \in R\}$$

8.2.1 Squared-loss function

$$l(h, (x, y)) = (h(x) - y)^2$$

The empirical risk function is the *Mean Squared Error*:

$$L_{S(h)} = \frac{1}{m}\sum_{i=1}^{m}(h(x_i) - y_i)^2$$

Least squares is the algorithm that solves the ERM problem for the hypothesis class of linear regression predictors with respect to the squared loss

$$\text{argmin}_w L_{S(h_w)} = \text{argmin}_w \frac{1}{m}\sum_{i=1}^{m}(\langle w, x_i \rangle - y_i)^2$$

To solve the problem we calculate the gradient of the objective function and compare it to zero

Linear Regression for Polynomial is same.

8.3 Logistic regression

To output a probability we use the logistic regression.

The sigmoid function:

$$\Phi_{\text{sig}(z)} = \frac{1}{1+\exp(-z)}$$

$$H_{\text{sig}} = \Phi_{\text{sig}} \circ L_d = \left\{x \to \Phi_{\text{sig}(\langle w,x \rangle)} : w \in R^d\right\}$$

The logistic loss function used in logistic regression is

$$l(h_w, (x, y)) = \log(1 + \exp(-y\langle w, x \rangle))$$

The ERM problem associated with logistic regression is

$$\text{argmin}_{w\in R^d} \frac{1}{m}\sum_{i=1}^{m}\log(1 + \exp(-y_i\langle w, x_i \rangle))$$

8.3.1 Maximum Likelihood Estimator

$$P_w(S) = \prod_{y_i=1} h(w^T x) \cdot \prod_{y_i=-1} 1 - h(w^t x)$$

$$\log P_{w(S)} = \sum_{t=1}^{n}\log(h(w^T x))1_{y_t=1} + \sum_{t=1}^{n}\log(1 - h(w^t x))1_{y_t=-1}$$

Facendo il log diventano somme e si riesce a dimostrare che The ERM problem associated with logistic regression is identical to the problem of finding a Maximum Likelihood Estimator.

9 Boosting

Boosting is an algorithmic paradigm that can address the bias-complexity tradeoff and the computational complexity of learning.

9.1 Weak learner

A, is a γ-weak-learner for a class H if there exists a function $m_H : (0, 1) \to N$ such that for every $\delta \in (0, 1)$, for every distribution D over X , and for every labeling function $f : X \to \{\pm 1\}$, if the realizable assumption holds with respect to H, D, f, then when running the learning algorithm on $m \geq m_H(\delta)$ i.i.d. examples generated by D and labeled by f, the algorithm returns a hypothesis h such that, with probability of at least $1 - \delta$, $L_{D,f}(h) \leq \frac{1}{2} - \gamma$.

A hypothesis class H is γ-weak-learnable if there exists a γ-weak-learner forthat class.

9.2 Decision Stumps

$$H_{DS} = \{x \to \text{sign}(x - \theta) \cdot b : \theta \in R, b \in \{\pm 1\}\}$$

The ERM problem for Decision Stumps is a γ-weak-learner.

9.3 Adaboost

AdaBoost is an algorithm that has access to a weak learner and finds a hypothesis with a low empirical risk. The AdaBoost algorithm receives as input a training set of examples S and some weak learners. The boosting process proceeds in a sequence of consecutive rounds. AdaBoost assigns a weight for h_t that is **inversely** proportional to the error of h_t. This will force the weak learner to focus on the problematic examples in the next round. Complexity can be $O(dm)$

10 Regularization and stability

Intuitively, the regularization function measures the complexity of hypothesis. Another view of regularization is as a stabilizer of the learning algorithm. An algorithm is considered stable if a slight change of its input does not change its output much.

10.1 Regularized Loss Minimization

$$\text{argmin}_w (L_S(w) + R(w))$$

where $R : R^d \to R$ is the regularization function.

A simple regularization functions is $R(w) = \lambda \|w\|^2$, where $\lambda > 0$ a parameter (Tikhonov regularization).

10.2 Ridge Regression

Applying the RLM rule with Tikhonov regularization to linear regression with the squared loss

$$\text{argmin}_{w \in R^d} \left(\lambda \; \|w\|_2^2 + \sum_{i=1}^{m} \frac{1}{2} (\langle \, x, y \, \rangle - y_i)^2 \right)$$

Performing linear regression using the above equation is called ridge regression.

10.3 Stability

Stable Rules Do Not Overfit.

Informal definition of stability: Given the training set S and an additional example z', let S^i be the training set obtained by replacing the i'th example of S with z'; namely, $S^i = (z_1, ..., z_{i-1}, z', z_{i+1}, ..., z_m)$. In our definition of stability, "a small change of the input" means that we feed A with S^i instead of with S. That is, we only replace one training example. We measure the effect of this small change of the input on the output of A, by comparing the loss of the hypothesis $A(S)$ on z_i to the loss of the hypothesis $A(S^i)$ on z_i. Intuitively, a good learning algorithm will have $l(A(S^i), z_i) - l(A(S), z_i) \geq 0$, since in the first term the learning algorithm does not observe the example z_i while in the second term z_i is indeed observed. If the preceding difference is very large we suspect that the learning algorithm might overfit. This is because the learning algorithm drastically changes its prediction on z_i if it observes it in the training set.

10.3.1 Tikhonov Regularization as a Stabilizer

Applying the RLM rule with Tikhonov regularization leads to a stable algorithm. The main property of the Tikhonov regularization that we rely on is that it makes the objective of RLM strongly convex.

10.3.2 Strongly Convex Functions

A function f is λ-strongly convex if for all w, u and $\alpha \in (0, 1)$ we have

$$f(\alpha w + (1-\alpha)u) \leq \alpha f(w) + (1-\alpha) f(u) - \frac{\lambda}{2}\alpha(1-\alpha)\|w-u\|^2$$

10.3.2.1 Lemmas

1. The function $f(w) = \lambda\|w\|^2$ is 2λ-strongly convex.
2. If f is λ-strongly convex and g is convex, then $f+g$ is λ-strongly convex.
3. If f is λ-strongly convex and u is a minimizer of f, then, for any w,

$$f(w) - f(u) \geq \frac{\lambda}{2}\,\|w-u\|^2$$

10.4 Lipschitz Loss

10.5 Fitting-Stability Tradeoff

$$E_S[L_D(A(S))] = E_S[L_S(A(S))] + E_S[L_D(A(S)) - L_S(A(S))]$$

The first term reflects how well A(S) fits the training set while the second term reflects the difference between the true and empirical risks of A(S). As we have shown in Theorem 13.2, the second term is equivalent to the stability of A. Since our goal is to minimize the risk of the algorithm, we need that the sum of both terms will be small.

Previously, we have bounded the stability term. We have shown that the stability term decreases as the regularization parameter, λ, increases. On the other hand, the empirical risk increases with λ. We therefore face a tradeoff between fitting and overfitting.

11 Model selection and validation

12 Neaural networks