Machine Learning

Learning Model

Fabio Vandin

October 9^{th}, 2023

A Formal Model (Statistical Learning)

We have a *learner* (us, or the machine) has access to:

1. **Domain set** $\mathcal{X}$: set of all possible objects to make predictions about
   - domain point $x \in \mathcal{X}$ = *instance*, usually represented by a vector of *features*
   - $\mathcal{X}$ is the *instance space*
2. **Label set** $\mathcal{Y}$: set of possible labels.
   - often two labels, e.g $\{-1, +1\}$ or $\{0, 1\}$
3. **Training data** $S = ((x_1, y_1), \ldots, (x_m, y_m))$: finite sequence of labeled domain points, i.e. pairs in $\mathcal{X} \times \mathcal{Y}$
   - this is the learner's **input**
   - S: *training example* or *training set*

4. **Learner's output** h: prediction rule $h : \mathcal{X} \to \mathcal{Y}$
   - also called *predictor*, *hypothesis*, or *classifier*
   - $A(S)$: prediction rule produced by learning algorithm A when training set S is given to it
   - sometimes $\hat{f}$ used instead of h
5. **Data-generation model**: instances are generated by some probability distribution and labeled according to a function
   - $\mathcal{D}$: probability distribution over $\mathcal{X}$ (**NOT KNOWN TO THE LEARNER!**)
   - labeling function f: $\mathcal{X} \to \mathcal{Y}$ (**NOT KNOWN TO THE LEARNER!**)
   - label y_i of instance x_i: $y_i = f(x_i)$, for all $i = 1, \ldots, m$
   - each point in training set S: first sample x_i according to $\mathcal{D}$, then label it as $y_i = f(x_i)$
6. **Measures of success**: *error of a classifier* = probability it does not predict the correct label on a random data point generate by distribution $\mathcal{D}$

Loss

Given domain subset $A \subset \mathcal{X}$, $\mathcal{D}(A)$ = probability of observing a point $x \in A$.

Let A be defined by a function $\pi : \mathcal{X} \to \{0, 1\}$:

$$A = \{x \in \mathcal{X} : \pi(x) = 1\}$$

In this case we have $\mathbb{P}_{x \sim \mathcal{D}}[\pi(x)] = \mathcal{D}(A)$

Error of prediction rule $h : \mathcal{X} \to \mathcal{Y}$ is

$$L_{\mathcal{D},f}(h) \stackrel{def}{=} \mathbb{P}_{x \sim \mathcal{D}}[h(x) \neq f(x)] \stackrel{def}{=} \mathcal{D}(\{x : h(x) \neq f(x)\})$$

Notes:

- $L_{\mathcal{D},f}(h)$ has many different names: **generalization error**, *true error*, *risk*, **loss**, ...
- often f is obvious, so omitted: $L_{\mathcal{D}}(h)$

Empirical Risk Minimization

Learner outputs $h_S : \mathcal{X} \to \mathcal{Y}$.

Goal: find h_S which minimizes the generalization error $L_{\mathcal{D},f}(h)$

$L_{\mathcal{D},f}(h)$ is unknown!

What about considering the error on the training data, that is, reporting in output h_S that minimizes the error on training data?

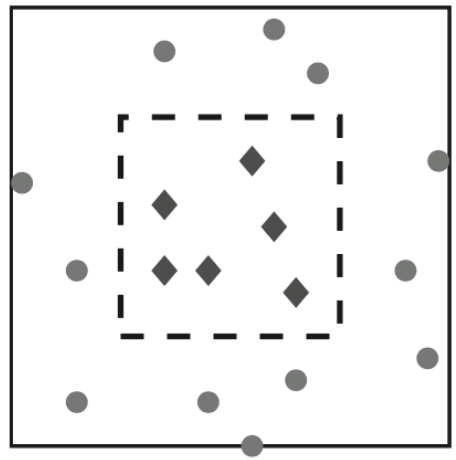
Training error: $L_S(h) \stackrel{def}{=} \frac{|\{i : h(x_i) \neq y_i, 1 \leq i \leq m\}|}{m}$

Note: the *training error* is also called *empirical error* or *empirical risk*

Empirical Risk Minimization (ERM): produce in output h minimizing $L_S(h)$

What can go wrong with ERM?

Consider our simplified movie ratings prediction problem. Assume data is given by:



Assume $\mathcal{D}$ and f are such that:

- instance x is taken uniformly at random in the square ($\mathcal{D}$)
- label is 1 if x inside the inner square, 0 otherwise (f)
- area inner square $= 1$, area larger square $= 2$

Consider classifier given by

$$h_S(x) = \begin{cases} y_i & \text{if } \exists i \in \{1, \ldots, m\} : x_i = x \\ 0 & \text{otherwise} \end{cases}$$

Is it a good predictor?

$L_S(h_S) = 0$ but $L_{\mathcal{D},f}(h_S) = 1/2$

Good results on training data but poor generalization error
$\Rightarrow$ **overfitting**

When does ERM lead to good performances in terms of generalization error?

Hypothesis Class and ERM

Apply ERM over a **restricted set** of hypotheses $\mathcal{H}$ = *hypothesis class*

- each $h \in \mathcal{H}$ is a function $h : \mathcal{X} \to \mathcal{Y}$

$\text{ERM}_{\mathcal{H}}$ learner:

$$\text{ERM}_{\mathcal{H}} \in \arg\min_{h \in \mathcal{H}} L_S(h)$$

Which hypothesis classes $\mathcal{H}$ do not lead to overfitting?

Finite Hypothesis Classes

Assume $\mathcal{H}$ is a finite class: $|\mathcal{H}| < \infty$

Let h_S be the output of $\text{ERM}_{\mathcal{H}}(S)$, i.e. $h_S \in \arg\min_{h \in \mathcal{H}} L_S(h)$

Assumptions

- **Realizability:** there exists $h^* \in \mathcal{H}$ such that $L_{\mathcal{D},f}(h^*) = 0$
- **i.i.d.:** examples in the training set are independently and identically distributed (i.i.d) according to $\mathcal{D}$, that is $S \sim \mathcal{D}^m$

Observation: realizability assumption implies that $L_S(h^*) = 0$

Can we *learn* (i.e., find using ERM) h^*?

(Simplified) PAC learning

Probably Approximately Correct (PAC) learning

Since the training data comes from $\mathcal{D}$:

- we can only be **approximately** correct
- we can only be **probably** correct

Parameters:

- *accuracy parameter* ε: we are satisfied with a *good* h_S: $L_{\mathcal{D},f}(h_S) \leq \varepsilon$
- *confidence parameter* δ: want h_S to be a *good* hypothesis with probability $\geq 1 - \delta$

Theorem

Let $\mathcal{H}$ be a finite hypothesis class. Let $\delta \in (0,1)$, $\varepsilon \in (0,1)$, and $m \in \mathbb{N}$ such that

$$m \geq \frac{\log(|\mathcal{H}|/\delta)}{\varepsilon}.$$

Then for any f and any $\mathcal{D}$ for which the realizability assumption holds, with probability $\geq 1-\delta$ we have that for every ERM hypothesis h_S it holds that

$$L_{\mathcal{D},f}(h_S) \leq \varepsilon.$$

Note: log = natural logarithm

Proof (see book as well, Corollary 2.3)

PAC Learning

Definition (PAC learnability)

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

$m_{\mathcal{H}}$: $(0,1)^2 \to \mathbb{N}$: *sample complexity* of learning $\mathcal{H}$.

- $m_{\mathcal{H}}$ is the minimal integer that satisfies the requirements.

Corollary

Every finite hypothesis class is PAC learnable with sample complexity $m_{\mathcal{H}}(\varepsilon, \delta) \leq \left\lceil \frac{\log(|\mathcal{H}|/\delta)}{\varepsilon} \right\rceil$.

A More General Learning Model: Remove Realizability Assumption (Agnostic PAC Learning)

Realizability Assumption: there exists $h^* \in \mathcal{H}$ such that $L_{\mathcal{D},f}(h^*) = 0$

Informally: the label is fully determined by the instance x

$\Rightarrow$ Too strong in many applications!

Relaxation: $\mathcal{D}$ is a probability distribution over $\mathcal{X} \times \mathcal{Y}$
$\Rightarrow$ $\mathcal{D}$ is the *joint distribution* over domain points and labels.

For example, two components of $\mathcal{D}$:

- $\mathcal{D}_x$: (marginal) distribution over domain points
- $\mathcal{D}((x, y)|x)$: conditional distribution over labels for each domain point

Given x, label y is obtained according to a conditional probability $\mathbb{P}[y|x]$.

The Empirical and True Error

With $\mathcal{D}$ that is a probability distribution over $\mathcal{X} \times \mathcal{Y}$ the *true error* (or risk) is:

$$L_{\mathcal{D}}(h) \stackrel{def}{=} \mathbb{P}_{(x,y)\sim\mathcal{D}}[h(x) \neq y]$$

As before $\mathcal{D}$ is not known to the learner; the learner only knows the training data S

Empirical risk: as before, that is

$$L_{\mathcal{S}}(h) \stackrel{def}{=} \frac{|\{i, 0 \leq i \leq m : h(x_i) \neq y_i\}|}{m}$$

Note: $L_{\mathcal{S}}(h)$ = probability that for a pair (x_i, y_i) taken uniformly at random from S the event "$h(x_i) \neq y_i$" holds.

An Optimal Predictor

Learner's goal: find $h : \mathcal{X} \to \mathcal{Y}$ minimizing $L_{\mathcal{D}}(h)$

Is there a *best predictor*?

Given a probability distribution $\mathcal{D}$ over $\mathcal{X} \times \{0, 1\}$, the best predictor is the **Bayes Optimal Predictor**

$$f_{\mathcal{D}}(x) = \begin{cases} 1 & \text{if } \mathbb{P}[y = 1|x] \geq 1/2 \\ 0 & \text{otherwise} \end{cases}$$

Proposition

For any classifier $g : \mathcal{X} \to \{0, 1\}$, it holds $L_{\mathcal{D}}(f_{\mathcal{D}}) \leq L_{\mathcal{D}}(g)$.

PROOF: Exercize

Can we use such predictor?

Agnostic PAC Learnability

Consider only predictors from a hypothesis class $\mathcal{H}$.

We are going to be ok with not finding the best predictor, but not being too far off.

Definition

A hypothesis class $\mathcal{H}$ is *agnostic PAC learnable* if there exist a function $m_{\mathcal{H}}\colon (0,1)^2 \to \mathbb{N}$ and a learning algorithm such that for every $\delta, \varepsilon \in (0,1)$, for every distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$, when running the learning algorithm on $m \geq m_{\mathcal{H}}(\varepsilon, \delta)$ i.i.d. examples generated by $\mathcal{D}$ the algorithm returns a hypothesis h such that, with probability $\geq 1 - \delta$ (over the choice of the m training examples):

$$L_{\mathcal{D}}(h) \leq \min_{h' \in \mathcal{H}} L_{\mathcal{D}}(h') + \varepsilon.$$

Note: this is a generalization of the previous learning model.

A More General Learning Model: Beyond Binary Classification

Binary classification: $\mathcal{Y} = \{0, 1\}$

Other learning problems:

- multiclass classification: classification with > 2 labels
- regression: $\mathcal{Y} = \mathbb{R}$

Multiclass classification: same as before!

Regression

Domain set: $\mathcal{X}$ is usually $\mathbb{R}^p$ for some p.

Target set: $\mathcal{Y}$ is $\mathbb{R}$

Training data: (as before) $S = ((x_1, y_1), \ldots, (x_m, y_m))$

Learner's output: (as before) $h : \mathcal{X} \to \mathcal{Y}$

Loss: the previous one does not make much sense...

(Generalized) Loss Functions

Definition

Given any hypotheses set $\mathcal{H}$ and some domain Z, a *loss function* is any function $\ell : \mathcal{H} \times Z \to \mathbb{R}_+$

Risk function = expected loss of a hypothesis $h \in \mathcal{H}$ with respect to $\mathcal{D}$ over Z:

$$L_{\mathcal{D}}(h) \stackrel{def}{=} \mathbb{E}_{z \sim \mathcal{D}}[\ell(h, z)]$$

Empirical risk = expected loss over a given sample $S = (z_1, \ldots, z_m) \in Z^m$:

$$L_S(h) \stackrel{def}{=} \frac{1}{m} \sum_{i=1}^{m} \ell(h, z_i)$$

Some Common Loss Functions

0-1 loss: $Z = \mathcal{X} \times \mathcal{Y}$

$$\ell_{0-1}(h,(x,y)) \stackrel{def}{=} \begin{cases} 0 & \text{if } h(x) = y \\ 1 & \text{if } h(x) \neq y \end{cases}$$

Commonly used in binary or multiclass classification.

Squared loss: $Z = \mathcal{X} \times \mathcal{Y}$

$$\ell_{sq}(h,(x,y)) \stackrel{def}{=} (h(x) - y)^2$$

Commonly used in **regression**.

Note: in general, the loss function may depend on the application! But computational considerations play a role...

How to Choose the Loss Function?

Agnostic PAC Learnability for General Loss Functions

Definition

A hypothesis class $\mathcal{H}$ is agnostic PAC learnable with respect to a set Z and a loss function $\ell : \mathcal{H} \times Z \to \mathbb{R}_+$ if there exist a function $m_{\mathcal{H}}$: $(0,1)^2 \to \mathbb{N}$ and a learning algorithm such that for every $\delta, \varepsilon \in (0,1)$, for every distribution $\mathcal{D}$ over Z, when running the learning algorithm on $m \geq m_{\mathcal{H}}(\varepsilon, \delta)$ i.i.d. examples generated by $\mathcal{D}$ the algorithm returns a hypothesis h such that, with probability $\geq 1 - \delta$ (over the choice of the m training examples):

$$L_{\mathcal{D}}(h) \leq \min_{h' \in \mathcal{H}} L_{\mathcal{D}}(h') + \varepsilon$$

where $L_{\mathcal{D}}(h) = \mathbb{E}_{z \sim \mathcal{D}}[\ell(h, z)]$

Leslie Valiant, Turing award 2010

For transformative contributions to the theory of computation, including the theory of probably approximately correct (PAC) learning, the complexity of enumeration and of algebraic computation, and the theory of parallel and distributed computing.

Bibliography

Up to now:
[UML] Chapter 2 and Chapter 3