Machine Learning 7: Linear classifiers

Master 2 Computer Science

Aurélien Garivier

2018-2019

Table of contents

Learnability of the class of halfspaces

The class of halfspaces

Definition

The class of linear (affine) functions on $\mathcal{X} = \mathbb{R}^d$ is defined as

$$L_d = \left\{h_{w,b} : w \in \mathbb{R}^d, b \in \mathbb{R}\right\} , \quad \text{where } h_{w,b}(x) = \langle w, x\rangle + b \ .$$

The hypothesis class of halfspaces for binary classification is defined as

$$\mathcal{H}S_d = \operatorname{sign} \circ L_d = \left\{x \mapsto \operatorname{sign}\left(h_{w,b}(x)\right) : h_{w,b} \in L_d\right\}$$

where $\operatorname{sign}(u) = \mathbb{1}\{u \geq 0\} - \mathbb{1}\{u < 0\}$. *Depth 1 neural networks.*

By taking $\mathcal{X}' = \mathcal{X} \times \{1\}$ and $d' = d + 1$, we may omit the bias b and focus on functions $h_w(x) = \langle w, x\rangle$.

Theorem

The VC-dimension of $\mathcal{H}S_d$ is equal to $d + 1$.

Corollary: the class of halfspaces is learnable with sample complexity $O\left(\frac{d+1+\log(1/\delta)}{\epsilon^2}\right)$.

Proof: VC-dimension of the class of halfspaces

Linear (homogeneous) case:

- $\geq \mathbf{d}$: the set $\{e_1, \ldots, e_d\}$ is shattered, since for every $(y_1, \ldots, y_d) \in \{-1, 1\}^d$ the choice $w = (y_1, \ldots, y_d)$ yields $\langle w, e_i \rangle = y_i$ for every i.
- $< \mathbf{d+1}$: let $x_1, \ldots, x_{d+1} \in \mathbb{R}^d$. There exits $a_1, \ldots, a_{d+1} \in \mathbb{R}$ such that $\sum_{i=1}^d a_i x_i = 0$ and, if $I = \{i : a_i > 0\}$ and $J = \{j : a_j < 0\}$, $|I \cup J| > 0$. Thus, $\sum_{i \in I} a_i x_i = \sum_{j \in J} |a_j| x_j$. If $x_1, \ldots, x_{d+1}$ is shattered, there exists $w \in \mathbb{R}^{d+1}$ such that $\forall i \in I, \langle w, x_i \rangle > 0$ and $\forall j \in J, \langle w, x_j \rangle < 0$. Hence, if both I and J are not empty,

$$0 < \sum_{i \in I} a_i \langle x_i, w \rangle = \left\langle \sum_{i \in I} a_i x_i, w \right\rangle = \left\langle \sum_{j \in J} |a_j| x_j, w \right\rangle = \sum_{j \in J} |a_j| \langle x_j, w \rangle < 0 \ .$$

  If either I or J is empty, one of the two inequalities is an equality, but not both of them.

Affine case (with bias):

- $\geq \mathbf{d+1}$: the set $\{e_1, \ldots, e_d, 0\}$ is shattered, since for every $(y_1, \ldots, y_{d+1}) \in \{-1, 1\}^d$ the choice $w = (y_1, \ldots, y_d)$ and $b = y_{d+1}/2$ yields $y_i\big(\langle w, e_i \rangle + b\big) > 0$ for every i and $y_{d+1}\big(\langle w, 0 \rangle + b\big) > 0$.
- $< \mathbf{d+2}$: if a set $\{x_1, \ldots, x_{d+2}\}$ were shattered by non-homogeneous halfspaces in $\mathbb{R}^d$, then the set $\{\tilde{x}_i = (x_i, 1) \in \mathbb{R}^{d+1} : 1 \leq i \leq d+2\}$ would be shattered by homogeneous halfspaces in $\mathbb{R}^{d+1}$: for any $(y_1, \ldots, y_{d+2}) \in \{-1, 1\}^{d+1}$, there would exist $w \in \mathbb{R}^d$ and $b \in \mathbb{R}$ such that $\forall i \in \{1, \ldots, d+2\}, y_i\big(\langle w, x_i \rangle + b\big) > 0$. Then, taking $\tilde{w} = (w, b)$ we would have that $\forall i \{1, \ldots, d+2\}, y_i \langle \tilde{w}, \tilde{x}_i \rangle = y_i\big(\langle w, x_i \rangle + b\big) > 0$. But we proved above that this is impossible.

The realizable case

Realizable case: Learning halfspaces with a linear program solver

In the realizable case, there exists w^* such that $\forall i \in \{1, \ldots, m\}$, $y_i \langle w^*, x_i \rangle \geq 0$, and even such that $\forall i \in \{1, \ldots, m\}$, $y_i \langle w^*, x_i \rangle > 0$.

Then there exists $\bar{w} \in \mathbb{R}^d$ such that $\forall i \in \{1, \ldots, m\}$, $y_i \langle \bar{w}, x_i \rangle \geq 1$: if we can find one, we have an ERM.

Let $A \in \mathcal{M}_{m,d}(\mathbb{R})$ be defined by $A_{i,j} = y_i \, x_{i,j}$, and let $v = (1, \ldots, 1) \in \mathbb{R}^m$. Then any solution of the linear program

$$\max_{w \in \mathbb{R}^d} \langle 0, w \rangle \quad \text{subject to} \quad Aw \geq v$$

is an ERM. It can thus be computed in polynomial time.

Rosenblatt's Perceptron algorithm

Algorithm: Batch Perceptron

Data: training set $(x_1, y_1), \ldots, (x_m, y_m)$

$w_0 \leftarrow (0, \ldots, 0)$

$t \geq 0$

while $\exists i_t : y_{i_t} \langle w_t, x_{i_t} \rangle \leq 0$ **do**

  $w_{t+1} \leftarrow w_t + y_{i_t} \frac{x_{i_t}}{\|x_{i_t}\|}$

  $t \leftarrow t + 1$

return w_t

Each updates helps reaching the solution, since

$$y_{i_t} \langle w_{t+1}, x_{i_t} \rangle = y_{i_t} \left\langle w_t + y_{i_t} \frac{x_{i_t}}{\|x_{i_t}\|}, x_{i_t} \right\rangle = y_{i_t} \langle w_t, x_{i_t} \rangle + \|x_{i_t}\| \,.$$

Relates to a coordinate descent (stepsize does not matter).

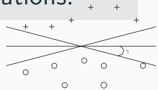
Convergence of the Perceptron algorithm

Theorem

Assume that the dataset $S = \{(x_1, y_1), \dots, (x_m, y_m)\}$ is linearly separable and let the *separation margin* γ be defined as:

$$\gamma = \max_{w \in \mathbb{R}^d : \|w\|=1} \min_{1 \le i \le n} \frac{y_i \langle w, x_i \rangle}{\|x_i\|} .$$

Then the perceptron algorithm stops after at most $1/\gamma^2$ iterations.



Proof: Let w^* be such that $\forall 1 \le i \le m$, $\quad \dfrac{y_i \langle w^*, x_i \rangle}{\|x_i\|} \ge \gamma$.

- If iteration t is necessary, then

$$\langle w^*, w_{t+1} - w_t \rangle = y_{i_t} \left\langle w^*, \frac{x_{i_t}}{\|x_{i_t}\|} \right\rangle \ge \gamma \quad \text{and hence } \langle w^*, w_t \rangle \ge \gamma t .$$

- If iteration t is necessary, then

$$\|w_{t+1}\|^2 = \left\| w_t + y_{i_t} \frac{x_{i_t}}{\|x_{i_t}\|} \right\|^2 = \|w_t\|^2 + \underbrace{\frac{2 y_{i_t} \langle w_t, x_{i_t} \rangle}{\|x_{i_t}\|}}_{\le 0} + y_{i_t}^2 \le \|w_t\|^2 + 1$$

  and hence $\|w_t\|^2 \le t$, or $\|w_t\| \le \sqrt{t}$.
- As a consequence, the algorithm iterates at least t times if

$$\gamma t \le \langle w^*, w_t \rangle \le \|w_t\| \le \sqrt{t} \quad \Longrightarrow \quad t \le \frac{1}{\gamma^2} .$$

In the worst case, the number of iterations can be exponentially large in the dimension d. Usually, it converges quite fast. If $\forall i$, $\|x_i\| = 1$, $\gamma = d(S, D)$ where $D = \{x : \langle w^*, x \rangle = 0\}$.

The agnostic case

Computational difficulty of agnostic learning, and surrogates

NP-hardness of computing the ERM for halfspaces

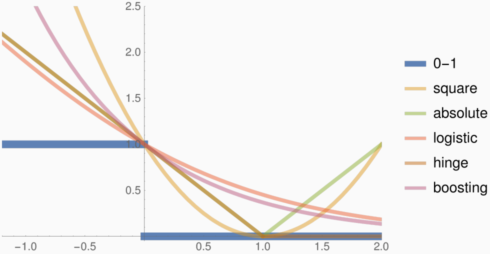
Computing an ERM in the agnostic case is NP-hard.

See *On the difficulty of approximately maximizing agreements*, by Ben-David, Eiron and Long.

Since the 0-1 loss

$$L_S(h_w) = \frac{1}{m}\sum_{i=1}^{m} \mathbb{1}\{y_i \langle w, x_i \rangle < 0\}$$

is intractable to minimize in the agnostic case, one may consider *surrogate* loss functions



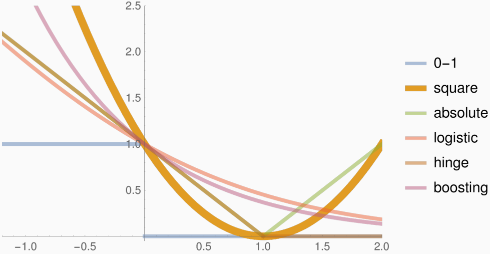
$$L_S(h_w) = \frac{1}{m}\sum_{i=1}^{m} \ell\big(y_i \langle w, x_i \rangle\big) \ ,$$

where the loss function $\ell : \mathbb{R} \to \mathbb{R}^+$

- dominates the function $\mathbb{1}\{u < 0\}$,
- and leads to a "simple" optimization problem (e.g. *convex*).

Quadratic loss

Linear regression with least squares:

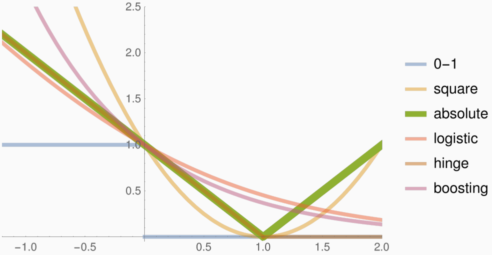


$$L_S(h_w) = \frac{1}{m}\sum_{i=1}^{m}\big(h_w(x_i) - y_i\big)^2 = \frac{1}{m}\sum_{i=1}^{m}\big(1 - y_i \langle w, x_i\rangle\big)^2 .$$

If $X = (x_1, \ldots, x_m) \in \mathcal{M}_{m,d}(\mathbb{R})$ and $y = (y_1, \ldots, y_m) \in \mathbb{R}^m$, one obtains $\hat{w} = \big(X^T X\big)^{-} X^T y$, where A^{-} = generalized inverse of A.

Absolute loss

Linear regression with absolute loss:



$$L_S(h_w) = \frac{1}{m}\sum_{i=1}^{m} \left| h_w(x_i) - y_i \right| = \frac{1}{m}\sum_{i=1}^{m} \left| 1 - y_i h_w(x_i) \right| .$$

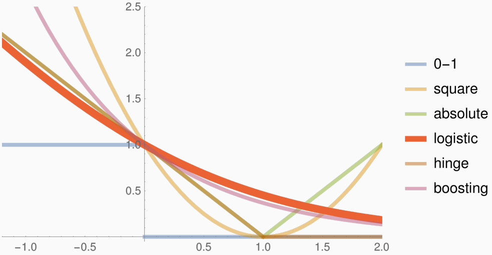
Can be solved by linear programming.

Interest: (statistical) robustness.

Logistic loss

Statistics: "logistic regression":

$$P_w(Y = y|X = x) = \frac{1}{1 + \exp\left(-y\langle w, x\rangle\right)}$$



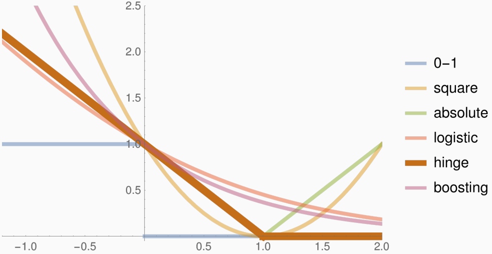
log with base 2 here so that $\ell(0) = 1$

$$L_S(h_w) = \frac{1}{m}\sum_{i=1}^{m} \log\left(1 + \exp(-y_i\langle w, x_i\rangle)\right) ,$$

convex minimization problem, can be solved by Newton's algorithm (in small dimension).

Support Vector Machines (SVM)

Margin maximization leads to

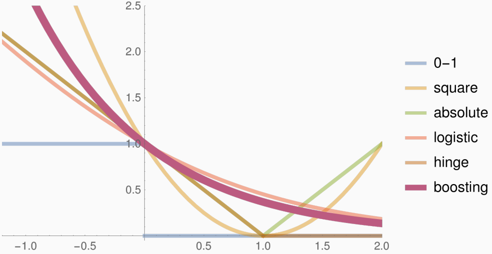


$$L_S(h_w) = \frac{1}{m}\sum_{i=1}^{m} \max\left\{0, 1 - y_i\langle w, x_i\rangle\right\} ,$$

convex but non-smooth minimization problem, used with a penalization term $\lambda\|w\|^2$: cf later.

Boosting

Margin maximization leads to



$$L_S(h_w) = \frac{1}{m}\sum_{i=1}^{m} \exp\left(-y_i\langle w, x_i\rangle\right) \ ,$$

with ad-hoc optimization procedure – cf later.