Machine Learning — Statistical Methods for Machine Learning

Linear predictors

Instructor: *Nicolò Cesa-Bianchi* version of April 22, 2023

A linear predictor for $\mathcal{X} = \mathbb{R}^d$ is a function $h : \mathbb{R}^d \to \mathbb{R}$ such that $h(\boldsymbol{x}) = f(\boldsymbol{w}^\top \boldsymbol{x})$ for some $\boldsymbol{w} \in \mathbb{R}^d$, where $f : \mathbb{R} \to \mathbb{R}$ is sometimes called the activation function. In linear regression tasks, f is the identity function and so $h(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x}$. In linear classification tasks, $h(\boldsymbol{x}) = \operatorname{sgn}(\boldsymbol{w}^\top \boldsymbol{x} - c)$ for some $c \in \mathbb{R}$, where $\operatorname{sgn}(z) = 1$ if $z > 0$ and -1 otherwise. We mostly focus on classification and return to regression only at the end.

Hyperplanes. Recall that an hyperplane with coefficients $(\boldsymbol{w}, c)$ is defined by $\big\{\boldsymbol{x} \in \mathbb{R}^d : \boldsymbol{w}^\top \boldsymbol{x} = c\big\}$, where $\boldsymbol{w}^\top \boldsymbol{x} = \|\boldsymbol{w}\|\,\|\boldsymbol{x}\| \cos\theta$ and θ is the angle between $\boldsymbol{w}$ and $\boldsymbol{x}$ and $\|\boldsymbol{x}\| \cos\theta$ is the length of the projection of $\boldsymbol{x}$ onto $\boldsymbol{w}$. Hence, the hyperplane defined by $(\boldsymbol{w}, c)$ is orthogonal to $\boldsymbol{w}$ and intersects it at distance $c/\|\boldsymbol{w}\|$ from the origin.

The halfspaces H^+ e H^- defined by the hyperplane $\big\{\boldsymbol{x} \in \mathbb{R}^d : \boldsymbol{w}^\top \boldsymbol{x} = c\big\}$ are

$$H^+ \equiv \Big\{\boldsymbol{x} : \boldsymbol{w}^\top \boldsymbol{x} > c\Big\} \qquad \text{and} \qquad H^- \equiv \Big\{\boldsymbol{x}' : \boldsymbol{w}^\top \boldsymbol{x}' \le c\Big\}$$

That is, all points $\boldsymbol{x}$ whose projection onto $\boldsymbol{w}$ has length strictly bigger than $c/\|\boldsymbol{w}\|$, and all points $\boldsymbol{x}'$ whose projection onto $\boldsymbol{w}$ has length not larger than $c/\|\boldsymbol{w}\|$. Geometrically, a linear classifier is thus defined by

$$h(\boldsymbol{x}) = \begin{cases} +1 & \text{if } \boldsymbol{x} \in H^+ \\ -1 & \text{if } \boldsymbol{x} \in H^- \end{cases}$$

Hyperplanes of the form $\big\{\boldsymbol{x} \in \mathbb{R}^d : \boldsymbol{w}^\top \boldsymbol{x} = 0\big\}$ pass through the origin and are called *homogeneous*. Any non-homogeneous hyperplane $\big\{\boldsymbol{x} \in \mathbb{R}^d : \boldsymbol{w}^\top \boldsymbol{x} = c\big\}$, with $c \neq 0$, is equivalent to the homogeneous hyperplane $\big\{\boldsymbol{x} \in \mathbb{R}^{d+1} : \boldsymbol{v}^\top \boldsymbol{x} = 0\big\}$ with $\boldsymbol{v} = (w_1, \dots, w_d, -c)$ in the following sense: $\boldsymbol{w}^\top \boldsymbol{x} - c = \boldsymbol{v}^\top \boldsymbol{x}'$ for all $\boldsymbol{x} \in \mathbb{R}^d$ and $\boldsymbol{x}' = (x_1, \dots, x_d, 1) \in \mathbb{R}^{d+1}$. For this reason, without any loss of generality we only deal with algorithms that learn linear predictors corresponding to homogeneous hyperplanes. This amounts to saying that we automatically add an extra feature with value 1 to all of our data points.

Training linear classifiers. Recall that a linear classifier is a predictor h such that $h(\boldsymbol{x}) = \operatorname{sgn}(\boldsymbol{w}^\top \boldsymbol{x})$. Clearly, $\operatorname{sgn}(\boldsymbol{w}^\top \boldsymbol{x}) = \operatorname{sgn}\big(\|\boldsymbol{w}\|\,\|\boldsymbol{x}\| \cos\theta\big) = \operatorname{sgn}(\cos\theta)$. As the classification is only determined by the angle θ between $\boldsymbol{w}$ and $\boldsymbol{x}$, the value of $\|\boldsymbol{w}\|$ is immaterial and we may take $\|\boldsymbol{w}\| = 1$. Note that the zero-one loss $\mathbb{I}\{h(\boldsymbol{x}_t) \neq y_t\}$ can be equivalently[1] rewritten as $\mathbb{I}\{y_t \boldsymbol{w}^\top \boldsymbol{x}_t \le 0\}$.

Let $\mathcal{H}_d$ be the family of linear classifiers $h(\boldsymbol{x}) = \operatorname{sgn}(\boldsymbol{w}^\top \boldsymbol{x})$ for $\boldsymbol{w} \in \mathbb{R}^d$ such that $\|\boldsymbol{w}\| = 1$. Consider the ERM algorithm for zero-one loss that, given a training set S containing examples

---

[1] Note that $y_t \boldsymbol{w}^\top \boldsymbol{x}_t = 0$ only when $\boldsymbol{w} = \boldsymbol{0}$ (we assume $\boldsymbol{x}_t \neq \boldsymbol{0}$ for all t). In this case, $\operatorname{sgn}\big(\boldsymbol{w}^\top \boldsymbol{x}_t\big) = -1$ and so the classification is actually correct when $y_t = -1$. Hence, using $\mathbb{I}\{y_t \boldsymbol{w}^\top \boldsymbol{x}_t \le 0\}$ to count mistakes we overcount only when $\boldsymbol{w} = \boldsymbol{0}$ and $y_t = -1$.

$(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_m, y_m) \in \mathbb{R}^d \times \{-1, 1\}$, outputs

$$h_S = \operatorname*{argmin}_{h \in \mathcal{H}_d} \frac{1}{m} \sum_{t=1}^{m} \mathbb{I}\{h(\boldsymbol{x}_t) \neq y_t\} = \operatorname*{argmin}_{\boldsymbol{w} \in \mathbb{R}^d \,:\, \|\boldsymbol{w}\|=1} \frac{1}{m} \sum_{t=1}^{m} \mathbb{I}\{y_t \boldsymbol{w}^\top \boldsymbol{x}_t \leq 0\} \ . \tag{1}$$

Unfortunately, it is unlikely to find an efficient implementation of ERM for linear classifiers with zero-one loss. In fact, the decision problem associated with finding h_S is NP-complete even when $\boldsymbol{x}_t \in \{0,1\}^d$ for $t = 1, \dots, m$. More precisely, introduce the following decision problem.

MinDisagreement
Instance: Pairs $(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_m, y_m) \in \{0,1\}^d \times \{-1, 1\}$. Integer k.
Question: Is there $\boldsymbol{w} \in \mathbb{Q}^d$ such that $y_t \, \boldsymbol{w}^\top \boldsymbol{x}_t \leq 0$ for at most k indices $t = 1, \dots, m$?

The following result can be shown.

Theorem 1. *MinDisagreement is NP-complete.*

In addition to that, the following stronger hardness-of-approximation result can be also shown.

MinDisOpt
Instance: Pairs $(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_m, y_m) \in \{0,1\}^d \times \{-1, 1\}$.
Solution: A point $\boldsymbol{w} \in \mathbb{Q}^d$ minimizing the number of indices $t = 1, \dots, m$ such that $y_t \, \boldsymbol{w}^\top \boldsymbol{x}_t \leq 0$.

Given an instance S (i.e., a training set) of MinDisOpt, let $\mathrm{Opt}(S)$ the number of examples in S that are misclassified by the ERM classifier h_S. In other words, $\mathrm{Opt}(S)/m = \ell_S(h_S)$.

Theorem 2. *If $P \neq NP$, then for all $C > 0$ there are no polynomial time algorithms that approximately solve every instance S of MinDisOpt with a number of misclassified examples bounded by $C \times \mathrm{Opt}(S)$.*

This implies that, unless $P = NP$ (which is believed unlikely), there are no efficient algorithms that approximate the solution of (1) to within any constant factor. Here efficient means with running time polynomial in the input size md.

The ERM problem (1) becomes easier when the training set is **linearly separable**. A training set $(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_m, y_m)$ is linearly separable where there exists a linear classifier with zero training error. In other words, there exists a separating hyperplane $\boldsymbol{u} \in \mathbb{R}^d$ such that

$$\gamma(\boldsymbol{u}) \stackrel{\text{def}}{=} \min_{t=1,\dots,m} y_t \, \boldsymbol{u}^\top \boldsymbol{x}_t > 0$$

The quantity $\gamma(\boldsymbol{u})$ is known as the **margin** of $\boldsymbol{u}$ on the training set. The scaled margin $\gamma(\boldsymbol{u}) / \|\boldsymbol{u}\|$ measures the distance between the separating hyperplane and the closest training example.

Now observe that the ERM problem (1) can be expressed as a system of linear inequalities,

$$y_t \, \boldsymbol{w}^\top \boldsymbol{x}_t > 0 \qquad t = 1, \dots, m \ .$$

When the training set is linearly separable, the system has at least a solution. This solution can be found in polynomial time using an algorithm for linear programming.

We now introduce a very simple algorithm for learning linear classifiers that can be used to solve the ERM problem in the linearly separable case. The Perceptron algorithm finds a homogeneous separating hyperplane by runnning through the training examples one after the other. The current linear classifier is tested on each training example and, in case of misclassification, the associated hyperplane is adjusted. Note that if the algorithm terminates, then $\boldsymbol{w}$ is a separating hyperplane.

```
Data: Training set $(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_m, y_m)$
$\boldsymbol{w} = (0, \dots, 0)$
while true do
    for $t = 1, \dots, m$ do        (epoch)
        if $y_t \boldsymbol{w}^\top \boldsymbol{x}_t \leq 0$ then
            $\boldsymbol{w} \leftarrow \boldsymbol{w} + y_t \boldsymbol{x}_t$     (update)
    end
    if no update in last epoch then break
end
Output: $\boldsymbol{w}$
```

Algorithm 1: The Perceptron algorithm (for the linearly separable case)

The update $\boldsymbol{w} \leftarrow \boldsymbol{w} + y_t \boldsymbol{x}_t$ when $y_t \boldsymbol{w}^\top \boldsymbol{x}_t \leq 0$ makes $y_t \boldsymbol{w}^\top \boldsymbol{x}_t$ bigger. Indeed,

$$y_t \big(\boldsymbol{w} + y_t \boldsymbol{x}_t\big)^\top \boldsymbol{x}_t = y_t \boldsymbol{w}^\top \boldsymbol{x}_t + \|\boldsymbol{x}_t\|^2 > y_t \boldsymbol{w}^\top \boldsymbol{x}_t$$

Geometrically, each update moves $\boldsymbol{w}$ towards $\boldsymbol{x}_t$ if $y_t = 1$ and moves $\boldsymbol{w}$ away from $\boldsymbol{x}_t$ if $y_t = -1$.

We now prove that Perceptron always terminates on linearly separable training sets.

Theorem 3 (Convergence of Perceptron)**.** *Let $(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_m, y_m)$ be a linearly separable training set. Then the Perceptron algorithm terminates after a number of updates not bigger than*

$$\left(\min_{\boldsymbol{u}\,:\,\gamma(\boldsymbol{u})\geq 1} \|\boldsymbol{u}\|^2\right)\left(\max_{t=1,\dots,m} \|\boldsymbol{x}_t\|^2\right) \tag{2}$$

The apparently stonger margin constraint $\gamma(\boldsymbol{u}) \geq 1$ is actually achievable by any separating hyperplane $\boldsymbol{u}$. Indeed, if $\gamma(\boldsymbol{u}) > 0$, then $y_t \boldsymbol{u}^\top \boldsymbol{x}_t \geq \gamma(\boldsymbol{u})$ is equivalent to $y_t \boldsymbol{v}^\top \boldsymbol{x}_t \geq 1$ for $\boldsymbol{v} = \boldsymbol{u}/\gamma(\boldsymbol{u})$. Hence, $\gamma(\boldsymbol{u}) \geq 1$ can be achieved simply by rescaling $\boldsymbol{u}$.

PROOF. Let $\boldsymbol{w}_0 = (0, \dots, 0)$ be the initial hyperplane. Let $\boldsymbol{w}_M$ be the hyperplane after M updates and let $t_M \in \{1, \dots, m\}$ be the index of the training example $(\boldsymbol{x}_{t_M}, y_{t_M})$ that caused the M-th update $\boldsymbol{w}_M = \boldsymbol{w}_{M-1} + y_{t_M} \boldsymbol{x}_{t_M}$. We prove an upper bound on M by deriving upper and lower bounds on $\|\boldsymbol{w}_M\|\,\|\boldsymbol{u}\|$. We start by observing that

$$\|\boldsymbol{w}_M\|^2 = \|\boldsymbol{w}_{M-1} + y_{t_M} \boldsymbol{x}_{t_M}\|^2 = \|\boldsymbol{w}_{M-1}\|^2 + \|\boldsymbol{x}_{t_M}\|^2 + 2\, y_{t_M} \boldsymbol{w}_{M-1}^\top \boldsymbol{x}_{t_M} \leq \|\boldsymbol{w}_{M-1}\|^2 + \|\boldsymbol{x}_{t_M}\|^2$$

because $y_{t_M}\boldsymbol{w}_{M-1}^\top\boldsymbol{x}_{t_M} \le 0$ due to the update $\boldsymbol{w}_M = \boldsymbol{w}_{M-1} + y_{t_M}\boldsymbol{x}_{t_M}$. Iterating this argument M times, and recalling that $\boldsymbol{w}_0 = (0,\dots,0)$, we obtain

$$\|\boldsymbol{w}_M\|^2 \le \|\boldsymbol{w}_0\|^2 + \sum_{i=1}^{M}\|\boldsymbol{x}_{t_i}\|^2 \le M\left(\max_{t=1,\dots,m}\|\boldsymbol{x}_t\|^2\right) .$$

Hence

$$\|\boldsymbol{w}_M\|\,\|\boldsymbol{u}\| \le \|\boldsymbol{u}\|\left(\max_{t=1,\dots,m}\|\boldsymbol{x}_t\|\right)\sqrt{M} .$$

To prove the lower bound, fix any separating hyperplane $\boldsymbol{u}$ with $\gamma(\boldsymbol{u}) \ge 1$ and let θ be the angle between $\boldsymbol{u}$ and $\boldsymbol{w}_M$. We have

$$\begin{aligned}
\|\boldsymbol{w}_M\|\,\|\boldsymbol{u}\| &\ge \|\boldsymbol{w}_M\|\,\|\boldsymbol{u}\|\cos(\theta) && \text{(since } -1 \le \cos(\theta) \le 1\text{)}\\
&= \boldsymbol{w}_M^\top\boldsymbol{u} && \text{(by definition of inner product)}\\
&= \left(\boldsymbol{w}_{M-1} + y_{t_M}\boldsymbol{x}_{t_M}\right)^\top\boldsymbol{u}\\
&= \boldsymbol{w}_{M-1}^\top\boldsymbol{u} + y_{t_M}\boldsymbol{u}^\top\boldsymbol{x}_{t_M}\\
&\ge \boldsymbol{w}_{M-1}^\top\boldsymbol{u} + 1
\end{aligned}$$

where the last inequality holds because $1 \le \gamma(\boldsymbol{u}) \le y_t\boldsymbol{u}^\top\boldsymbol{x}_t$ for all $t = 1,\dots,m$. Iterating M times we get

$$\|\boldsymbol{w}_M\|\,\|\boldsymbol{u}\| \ge \boldsymbol{w}_0^\top\boldsymbol{u} + M = M$$

Where we used $\boldsymbol{w}_0^\top\boldsymbol{u} = 0$ since $\boldsymbol{w}_0 = (0,\dots,0)$. Combining upper and lower bound we obtain

$$M \le \|\boldsymbol{u}\|\left(\max_{t=1,\dots,M}\|\boldsymbol{x}_t\|\right)\sqrt{M} .$$

Solving for M, and recalling the choice of $\boldsymbol{u}$, we obtain (2). Hence, the update count M cannot grow larger than (2). Since the algorithm stops when no more updates are possible, we conclude that the Perceptron terminates after a bounded number of updates. □

Note that the Perceptron convergence theorem does not imply that the Perceptron algorithm terminates in polynomial time on any linearly separable training set. Indeed, it can be shown that the bound (2) is tight in any fixed dimension $d \ge 2$. Hence, although each update takes constant time $\Theta(d)$, the number of updates can still be exponential in md whenever $\gamma(\boldsymbol{u}) \ge 1$ only for those $\boldsymbol{u}$ whose length $\|\boldsymbol{u}\|$ is very big. Or, equivalently, when the margin $\gamma(\boldsymbol{u})$ is very small for any linear separator $\boldsymbol{u}$ such that $\|\boldsymbol{u}\| = 1$.

Linear regression. In linear regression our predictors are linear functions $h : \mathbb{R}^d \to \mathbb{R}$ each parameterized by a vector $\boldsymbol{w} \in \mathbb{R}^d$ of real coefficients. That is, $h(\boldsymbol{x}) = \boldsymbol{w}^\top\boldsymbol{x}$.

Given a training set $(\boldsymbol{x}_1, y_1),\dots,(\boldsymbol{x}_m, y_m) \in \mathbb{R}^d \times \mathbb{R}$, the linear regression predictor is the ERM with respect to the square loss,

$$\boldsymbol{w}_S = \operatorname*{argmin}_{\boldsymbol{w}\in\mathbb{R}^d}\sum_{t=1}^{m}\left(\boldsymbol{w}^\top\boldsymbol{x}_t - y_t\right)^2$$

Now let $\boldsymbol{v} = \big(\boldsymbol{w}^\top \boldsymbol{x}_1, \dots, \boldsymbol{w}^\top \boldsymbol{x}_m\big)$ and $\boldsymbol{y} = (y_1, \dots, y_m)$. Then

$$\sum_{t=1}^{m} \big(\boldsymbol{w}^\top \boldsymbol{x}_t - y_t\big)^2 = \|\boldsymbol{v} - \boldsymbol{y}\|^2 \ .$$

View all vectors as column vectors. Since $\boldsymbol{v} = S\boldsymbol{w}$, where S is the $m \times d$ **design matrix** such that $S^\top = [\boldsymbol{x}_1, \dots, \boldsymbol{x}_m]$, we may also write

$$\boldsymbol{w}_S = \operatorname*{argmin}_{\boldsymbol{w} \in \mathbb{R}^d} \|S\boldsymbol{w} - \boldsymbol{y}\|^2 \ .$$

Since $F(\boldsymbol{w}) = \|S\boldsymbol{w} - \boldsymbol{y}\|^2$ is a convex function, the minimizer satisfies the condition $\nabla F(\boldsymbol{w}) = \boldsymbol{0}$.

Using matrix calculus, we have that $\nabla \|S\boldsymbol{w} - \boldsymbol{y}\|^2 = 2S^\top(S\boldsymbol{w} - \boldsymbol{y})$. Hence, $\nabla \|S\boldsymbol{w} - \boldsymbol{y}\|^2 = \boldsymbol{0}$ for $\boldsymbol{w} = \big(S^\top S\big)^{-1} S^\top \boldsymbol{y}$ provided $S^\top S$ is nonsingular (i.e., invertible)—which is equivalent to $\boldsymbol{x}_1, \dots, \boldsymbol{x}_m$ spanning $\mathbb{R}^d$. When this happens, we have that the ERM with respect to the square loss is $\boldsymbol{w}_S = \big(S^\top S\big)^{-1} S^\top \boldsymbol{y}$.

Ridge Regression. When $S^\top S$ is nearly singular, $\boldsymbol{w}_S$ is highly sensitive to perturbations of the training set. This instability increases the estimation error (or variance). A more stable predictor is obtained by introducing a **regularizer** in the ERM functional which increases the approximation error (or bias) and reduces the variance with a beneficial effect on the risk.

In other words, instead of defining $\boldsymbol{w}_S$ by

$$\boldsymbol{w}_S = \operatorname*{argmin}_{\boldsymbol{w} \in \mathbb{R}^d} \|S\boldsymbol{w} - \boldsymbol{y}\|^2$$

we use the regularized form, also known as Ridge Regression,

$$\boldsymbol{w}_{S,\alpha} = \operatorname*{argmin}_{\boldsymbol{w} \in \mathbb{R}^d} \|S\boldsymbol{w} - \boldsymbol{y}\|^2 + \alpha \|\boldsymbol{w}\|^2$$

where $\alpha > 0$ is the regularization parameter. When $\alpha \to 0$ we recover the standard linear regression solution. When $\alpha \to \infty$, the solution $\boldsymbol{w}_{S,\alpha}$ becomes the zero vector. This shows that α can be used to control the bias of the algorithm.

Similarly to before, we have that

$$\nabla\Big(\|S\boldsymbol{w} - \boldsymbol{y}\|^2 + \alpha \|\boldsymbol{w}\|^2 \Big) = 2S^\top(S\boldsymbol{w} - \boldsymbol{y}) + 2\alpha\boldsymbol{w} \ .$$

Hence, the gradient vanishes for $\boldsymbol{w} = \boldsymbol{w}_{S,\alpha} = \big(\alpha I + S^\top S\big)^{-1} S^\top \boldsymbol{y}$. Note that we do not have to worry anymore about the singularity of $S^\top S$. Indeed, if $\lambda_1 \ge \cdots \ge \lambda_d \ge 0$ are the eigenvalues of $S^\top S$, the eigenvalues of $\alpha I + S^\top S$ are simply $\alpha + \lambda_1 \ge \cdots \ge \alpha + \lambda_d > 0$. Hence, $\alpha I + S^\top S$ is invertible for all $\alpha > 0$.