Optimization for machine learning

Optimization : what, why, how?

Pierre Ablin

An algorithm that tells whether there is a cat in a picture:

Yes

No

An algorithm that tells whether there is a cat in a picture:

Yes

An algorithm that tells whether there is a cat in a picture:

$\longrightarrow$ **Yes**

An algorithm that tells whether there is a cat in a picture:

$\longrightarrow$ **No**

An algorithm that tells whether there is a cat in a picture:

Yes

An algorithm that tells whether there is a cat in a picture:

$\xrightarrow{h}$

Yes

No

x: Input/Feature

y: Output/Target

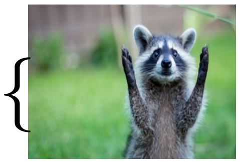
Find mapping h that assigns the “correct” target to each input

$$h : x \in \mathbb{R}^d \longrightarrow \mathrm{y} = \pm 1$$

Labelled Data: The training set

x^1 {  $y^1 = 1$

x^2 {  $y^2 = 1$

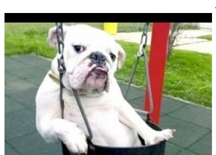
x^3 {  $y^3 = -1$

$\cdots$ x^n {  $y^n = 1$

y= -1 means no/false

Learning Algorithm

$h : x \in \mathbb{R}^d \to y \in \pm 1$

h () $\longrightarrow$ **-1**

A parametrized decision function

$$h : x \in \mathbb{R}^d \rightarrow y$$

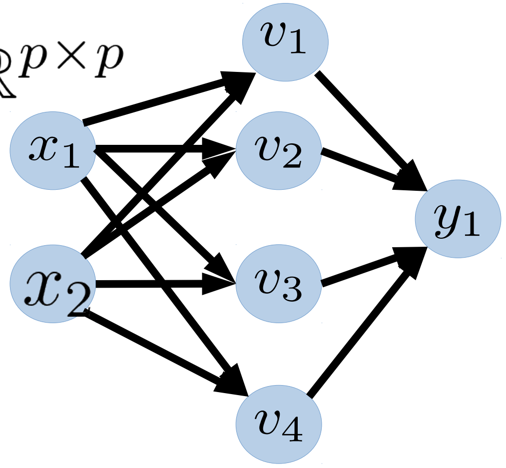
h is a function parametrized by parameters $\mathbf{w}$

Examples

Linear: $h_{\mathbf{w}}(x) = w_1 x_1 + \cdots + w_p x_p, \quad \mathbf{w} \in \mathbb{R}^p$

Polynomial: $h_{\mathbf{w}}(x) = \sum_{ij} x_i x_j w_{ij}, \quad \mathbf{w} \in \mathbb{R}^{p \times p}$

Neural network: $h_{\mathbf{w}}(x) = \mathbf{w}_2 \sigma(\mathbf{w}_1 x)$
$\mathbf{w}_2 \in \mathbb{R}^q, \quad \mathbf{w}_1 \in \mathbb{R}^{q \times p}$



Learning parameters

Goal :

Find **w** such that for (x, y) in our dataset :

$$h_{\mathbf{w}}(x) \simeq y$$

Mathematical reformulation

Find **w** that minimizes a discrepancy:

$$\min F(\mathbf{w}) = \frac{1}{n} \sum_{i=1}^{n} \ell(h_{\mathbf{w}}(x_i), y_i)$$

Loss function

The Training Problem

$$\min_{\mathbf{w}\in\mathbb{R}^d} \frac{1}{n}\sum_{i=1}^{n} \ell\left(h_{\mathbf{w}}(x_i), y_i\right)$$

Typically a convex function

Loss Functions

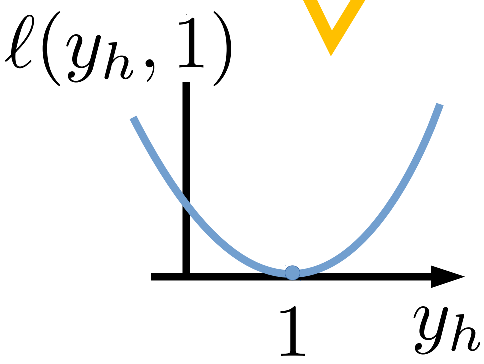
$$\begin{aligned} \ell: \quad \mathbb{R}\times\mathbb{R} &\quad\rightarrow\quad \mathbb{R}_+ \\ (y_h, y) &\quad\rightarrow\quad \ell(y_h, y) \end{aligned}$$

Choosing the loss function

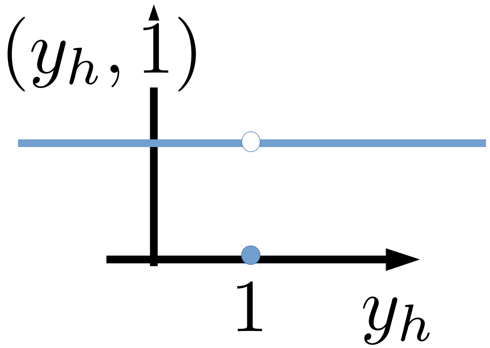
y=1 in all figures

Let $y_h := h_w(x)$

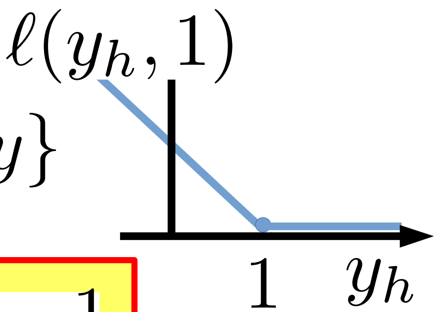
Quadratic Loss $\quad \ell(y_h, y) = (y_h - y)^2$



Binary Loss $\quad \ell(y_h, y) = \begin{cases} 0 & \text{if } y_h = y \\ 1 & \text{if } y_h \neq y \end{cases}$



Hinge Loss $\quad \ell(y_h, y) = \max\{0, 1 - y_h y\}$



EXE: Plot the binary and hinge loss function in when $y = -1$

The Machine Learners Job

(1) Get the labeled data: $(x^1, y^1), \ldots, (x^n, y^n)$

(2) Choose a parametrization for hypothesis: $h_w(x)$

(3) Choose a loss function: $\ell(h_w(x), y) \geq 0$

(4) Solve the *training problem*:

$$\boxed{\min_{w \in \mathbf{R}^d} \frac{1}{n} \sum_{i=1}^{n} \ell\left(h_w(x^i), y^i\right)}$$

(5) Test and cross-validate. If fail, go back a few steps

Optimization

The Training Problem

$$\min_{\mathbf{w}\in\mathbb{R}^d} F(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} \ell\left(h_{\mathbf{w}}(x_i), y_i\right)$$

Optimization : find an algorithm to minimize the function

Challenge 1: different settings

The Training Problem

$$\min_{\mathbf{w}\in\mathbb{R}^d} F(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} \ell\left(h_{\mathbf{w}}(x_i), y_i\right)$$

What do we know about F?

Regularity: Is it differentiable ? Convex ? Smooth ? Defined everywhere ?

Leads to different algorithms

Challenge 2: theoretical guarantees

The Training Problem

$$\min_{\mathbf{w}\in\mathbb{R}^d} F(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} \ell\left(h_{\mathbf{w}}(x_i), y_i\right)$$

What can we say about the algorithm?

Theory : Does it converge ? In which sense ? At which speed ?

Challenge 3: practical implementation

The Training Problem

$$\min_{\mathbf{w}\in\mathbb{R}^d} F(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} \ell\left(h_{\mathbf{w}}(x_i), y_i\right)$$

How can we implement algorithms ?

Speed and scaling: How to write good code ? How to make algorithms fast ? What if we are in a large scale setting (n, d large) ?

Course overview:

Monday: Reminders on linear algebra and analysis

Tuesday: Gradient descent, theory and practice

Wednesday: Beyond gradient descent, proximal methods

Thursday: Second order methods

Friday: Large scale learning: stochastic gradient descent

Labs:

Labs use **Python notebooks.**

Either run locally using jupyter, or in the cloud using google collab