Introduction au Machine Learning

Kernel methods, Support Vector Machines

Chloé-Agathe Azencott

Center for Computational Biology (CBIO)
Mines Paris – Institut Curie – INSERM U1331
PSL Research University & PR[AI]RIE, Paris, France

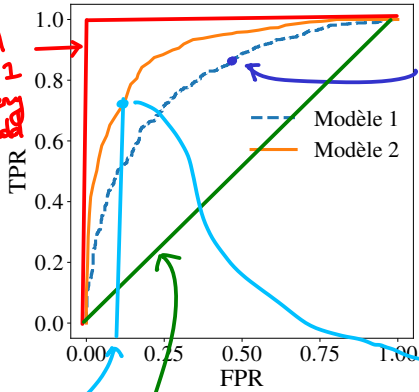
Sep 15, 2026

http://cazencott.info chloe-agathe.azencott@minesparis.psl.eu @cazencott@lipn.info

Évaluation d'un modèle (classification binaire)

based on the decision function $f: \mathcal{X} \to \mathbb{R}$ (to be thresholded)

ROC **Receiver-Operator Characteristic** curve



ideal TPR=1 for every threshold

(TPR, FPR) for 1 threshold

D_{te} contains 8 samples

$f(x_1), \dots, f(x_8)$

$\lessgtr \quad \lessgtr$

Summarized in a single number: Area under the ROC curve (AUC or AUROC)

using this threshold should give a 10% FPR (70% TPR).

TPR=10%

random model

1. Nonlinear parametric models

$$\mathcal{D} = \{(x_i, y_i)\}_{i=1..n} \qquad x^i \in \mathbb{R}^p, \; y^i \in \mathcal{Y}$$

① Transforming the input features

$$(x_1, \dots, x_p) \rightsquigarrow \varphi(x_1, \dots, x_p)$$

ex) polynomial regression of degree d

$$\varphi : x \mapsto x_1, \dots, x_p, x_1^2, x_1 x_2, \dots, x_p^2, \dots, x_1^d, x_1^{d-1} x_2, \dots, x_p^d$$

MLP: φ is learned $\varphi(x)$ = output of the penultimate layer

② Learn a linear model on the transformed features

$$\tilde{\mathcal{D}} = \{(\varphi(x^i), y^i)\}_{i=1..n} \qquad \varphi(x^i) \in \mathbb{R}^m \qquad m \gg p$$

MLP: last layer = linear operation on the penultimate layer → more computations

hence: representation learning. → higher risk of overfitting.

2.1 Dot product

in $\mathbb{R}^p$

$$\langle x, x' \rangle = \sum_{j=1}^{p} x_j x'_j$$

$$\langle x, x' \rangle = \|x\| \cdot \|x'\| \cdot \cos\theta$$

→ similarity



positive definite bilinear form

$$\langle u, v \rangle = \langle v, u \rangle$$

$a \in \mathbb{R}$ $\quad \langle au, v \rangle = a \langle u, v \rangle$

$$\langle u+v, w \rangle = \langle u, w \rangle + \langle v, w \rangle$$

$$\langle u, u \rangle \geq 0 \quad \text{and} \quad \langle u, u \rangle = 0 \text{ iff } u = 0$$

2.2 Kernel definition

in $\mathcal{X}$

$k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ positive semi definite (psd)

equivalent to: $\forall\, n \in \mathbb{N}^*$,

$\forall\, (x^1, x^2, \dots, x^n)$ in $\mathcal{X}^n$

the $n \times n$ matrix K s.t.

$K_{ij} = k(x^i, x^j)$

is psd

or $\forall\, n \in \mathbb{N}^*$, $\forall\, (x^1, \dots, x^n)$ in $\mathcal{X}^n$,

$\forall\, c_1, c_2, \dots, c_n$ in $\mathbb{R}$

then $\sum_{i,j} c_i c_j k(x_i, x_j) \geq 0$

2.3 Kernels generalize the dot product

Aronszajn theorem:

If $k: \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, psd

then $\exists$ Hilbert space $\mathcal{H}$ and a map $\varphi: \mathcal{X} \to \mathcal{H}$

such that:

$\forall\, x, x'$ in $\mathcal{X} \times \mathcal{X}$, $k(x, x') = \langle \varphi(x), \varphi(x') \rangle_{\mathcal{H}}$

2.4 Kernel examples

- Linear kernel = dot product $k: \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}$, $x, x' \mapsto \langle x, x' \rangle$
- Polynomial kernel $k: \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}$, $x, x' \mapsto (1 + \langle x, x' \rangle)^d$

In the case of $d=2$: quadratic kernel

$\varphi: (x_1, \dots, x_p) \mapsto (1, x_1, \dots, x_p, x_1^2, \dots, x_p^2, \sqrt{2}\, x_1 x_2, \dots, \sqrt{2}\, x_{p-1} x_p)$

m features

to compute $\langle \varphi(x), \varphi(x') \rangle$ choice 1 : compute $\varphi(x)$, compute $\varphi(x')$, then compute $\langle \varphi(x), \varphi(x') \rangle$

choice 2 : compute $\langle x, x' \rangle$, then add 1, then square it

fewer operations

- Radial Basis Function (RBF) kernel / Gaussian kernel

$$k : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}$$
$$x, x' \mapsto \exp\left(-\frac{\|x - x'\|^2}{2\sigma^2}\right)$$

bandwidth — in ML: will be an hyperparameter

Limit of the polynomial kernel when $d \to +\infty$

corresponds to $\varphi : \mathbb{R}^p \to \mathcal{H}$ ← infinite-dimensional

φ computes all the monomials of any number of p features

choice 1 is not available to me anymore.

- Kernel for strings $\quad \mathcal{X} = \mathcal{A}^\infty \quad \mathcal{A}$ = alphabet

2 strings are similar if they share many substrings of length d

$$\varphi(x) = [\varphi_u(x)]_{u \in \mathcal{A}^d} \qquad \varphi_u(x) = \text{nb of occurrences of } u \text{ in } x$$

$$k : x, x' \mapsto \langle \varphi(x), \varphi(x') \rangle$$

- Kernel for strings $\mathcal{X} = \mathcal{A}^\infty$ $\mathcal{A}$ = alphabet

  2 strings are similar if they share many substrings of length d

  $\varphi(x) = [\varphi_u(x)]_{u \in \mathcal{A}^d}$ $\varphi_u(x)$ = nb of occurrences of u in x

$k : x, x' \mapsto \langle \varphi(x), \varphi(x') \rangle$

Option 1 : compute $\varphi(x)$ and $\varphi(x')$ and then their dot product.

$|\mathcal{A}^d| = |\mathcal{A}|^d$

Option 2 : read x, list all its substrings of length d

ATTACGAT (about $|x|$ operations)

and look for them in x' → $\mathcal{O}(\max(|x|, |x'|))$ operations

3.1 The kernel trick

Replace computing $\varphi(x)$ and $\varphi(x')$ then computing $\langle \varphi(x), \varphi(x') \rangle$

with computing directly $k(x, x')$

3.2 Kernel ridge regression

Ridge regression

$$D = \{(\vec{x}^i, y^i)\}_{i=1\dots n} \qquad x^i \in \mathbb{R}^p \qquad y^i \in \mathbb{R}$$

$$\underset{\vec{\beta} \in \mathbb{R}^p}{\arg\min} \; \frac{1}{n} \sum_{i=1}^{n} \left(y^i - \langle \vec{\beta}, \vec{x}^i \rangle\right)^2 + \lambda \|\vec{\beta}\|_2^2$$

X : $n \times p$ matrix

$X_{ij} = x_j^i$

$\vec{y} = (y^1, \dots, y^n)$

solution : $\vec{\beta}^* = (X^T X + \lambda I_p)^{-1} X^T \vec{y}$

model $\hat{f}(\vec{x}) = \langle \vec{\beta}^*, \vec{x} \rangle = \vec{x} X^T (\lambda I_n + X X^T)^{-1} \vec{y}$

$\vec{x} \mapsto \varphi(\vec{x})$ $\qquad f(\varphi(\vec{x})) = \underbrace{\varphi(\vec{x}) \Psi^T}_{\text{n-dim vector, } \langle \varphi(\vec{x}), \varphi(\vec{x}^i) \rangle} (\lambda I + \underbrace{\Psi \Psi^T}_{n \times n})^{-1} \vec{y}$

where $\Psi_{ij} = \varphi(\vec{x}^i)_j$ $\quad n \times m$ matrix $\quad (i,e) \mapsto \langle \varphi(\vec{x}^i), \varphi(\vec{x}^e) \rangle$

kernelization $f(\varphi(\vec{x})) = \vec{\kappa} (\lambda I_n + K)^{-1} \vec{y}$ $\qquad \vec{\kappa} \in \mathbb{R}^n \quad \kappa_i = k(\vec{x}, \vec{x}^i)$

$\hookrightarrow K \in \mathbb{R}^{n \times n} \quad K_{ie} = k(\vec{x}^i, \vec{x}^e)$

Kernelization applies whenever observations only appear within dot products with other observations

E.g) PCA (projection on the PCs).

Computations are less intensive

We're working in space of dim n. independently on the nb of features

→ helps avoid overfitting

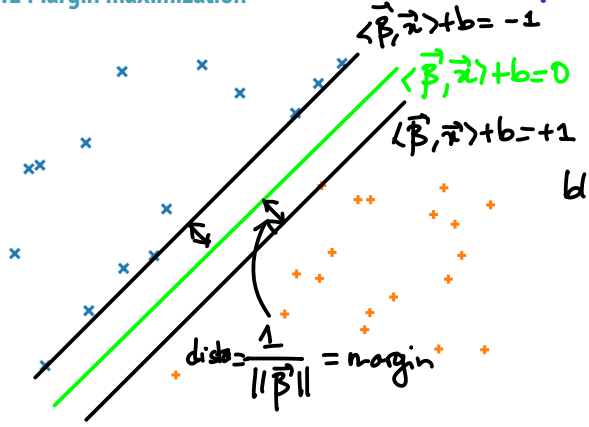
Often lose access to model parameters

Doesn't apply to the lasso.

4. Support Vector Machines

4.1 Margin maximization

$\mathcal{D} = \{(x^i, y^i)\}_{i=1..n}$ $x^i \in \mathbb{R}^p$ $y^i \in \{-1, 1\}$



We want a separator "in the middle"

black lines: // to the green one

one passes through a blue vector
the other through an orange vector

being "in the middle" = maximizing $\frac{1}{\|\vec{\beta}\|}$

equivalent to minimizing $\|\vec{\beta}\|$ i.e. ridge (ℓ_2) regularization.

4.2 SVM: Primal formulation

Binary classification, $\mathcal{Y} = \{-1, 1\}$

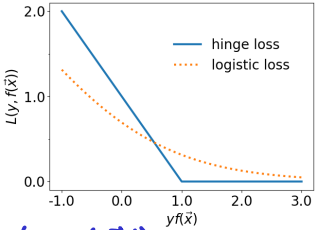
regularization

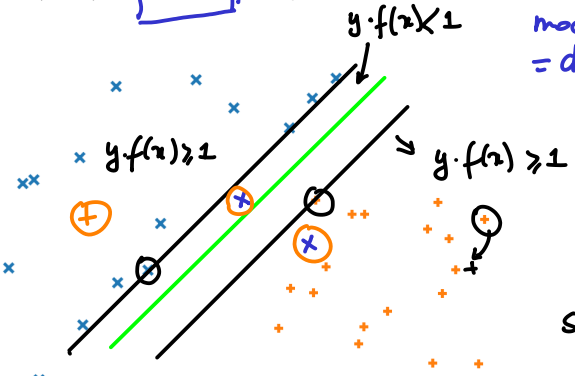
regularization coeff

loss function = hinge loss

$$\arg\min_{\vec{\beta}\in\mathbb{R}^p, b\in\mathbb{R}} \frac{1}{2}\left\|\vec{\beta}\right\|_2^2 + C\sum_{i=1}^{n} \max\left(0, 1 - y^i\left(\langle\vec{\beta}, \vec{x}^i\rangle + b\right)\right)$$

model = decision function $f(\vec{x}^i)$





$f(\vec{x}^i) \geq 0 \rightarrow$ label +1

$f(\vec{x}^i) < 0 \rightarrow$ label -1

$\text{sign}(y \cdot f(\vec{x}^i)) = 1$ if f is correct, -1 otherwise

Support vectors = $y \cdot f(\vec{x}^i) \leq 1$

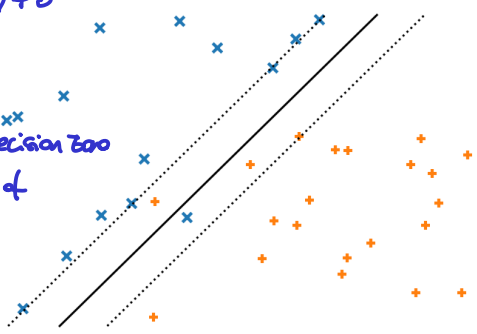
4.3 SVM: Dual formulation

$$\arg\max_{\vec{\alpha}\in\mathbb{R}^n} \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{l=1}^{n} \alpha_i \alpha_l y^i y^l \langle \vec{x}^i, \vec{x}^l \rangle$$

such that $\sum_{i=1}^{n} \alpha_i y_i = 0$ et $0 \leq \alpha_i \leq C$

$f(\vec{x}) = \langle \vec{\beta}, \vec{x} \rangle + b = \sum_{i=1}^{n} \alpha_i y_i \langle \vec{x}^i, \vec{x} \rangle + b$

- **Karuhn-Kush-Tucker** conditions:
  - $\alpha_i = 0$ et $y_i \left(\langle \vec{\beta}, \vec{x}^i \rangle + b \right) > 1$ Correct and away from the indecision zoo
    or
  - $\alpha_i = C$ et $y_i \left(\langle \vec{\beta}, \vec{x}^i \rangle + b \right) < 1$ on the wrong side of the limits
    or
  - $0 < \alpha_i < C$ et $y_i \left(\langle \vec{\beta}, \vec{x}^i \rangle + b \right) = 1$ on the limits of the indecision zone



4.5 Kernel SVM

$$\arg\max_{\vec{\alpha}\in\mathbb{R}^n} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{l=1}^{n} \alpha_i \alpha_l y^i y^l \langle \vec{x}^i, \vec{x}^l \rangle$$

$k(x^i, x^l)$

such that $\sum_{i=1}^{n} \alpha_i y_i = 0$ et $0 \leq \alpha_i \leq C$

$k(x^i, x)$

$$f(\vec{x}) = \langle \vec{\beta}, \vec{x} \rangle + b = \sum_{i=1}^{n} \alpha_i y_i \langle \vec{x}^i, \vec{x} \rangle + b$$

4.5 Support Vector Regression

ε-insensitive loss: $L(y, f(\vec{x})) = \max(0, |f(\vec{x}) - y| - \varepsilon)$

