Le Dictionnaire de l'Apprentissage Automatique d'A''alto

Alexander Jung and Konstantina Olioumtsevits

May 8, 2025

please cite as: A. Jung and K. Olioumtsevits, *The Aalto Dictionary of Machine Learning.* Espoo, Finland: Aalto University, 2025.

Acknowledgements

Ce dictionnaire de l'apprentissage automatique a évolué au fil du développement et l'enseignement de plusieurs cours, parmi lesquels CS-E3210 Machine Learning: Basic Principles, CS-C3240 Machine Learning, CS-E4800 Artificial Intelligence, CS-EJ3211 Machine Learning with Python, CS-EJ3311 Deep Learning with Python, CS-E4740 Federated Learning, and CS-E407507 Human-Centered Machine Learning. Ces cours ont été proposés à Aalto University `https://www.aalto.fi/en`, à des apprenants adultes via le Finnish Institute of Technology (FITech) `https://fitech.io/en/`, et à des étudiants et étudiantes internationaux dans le cadre de l'alliance universitaire européenne Unite! `https://www.aalto.fi/en/unite`.

Nous remercions les étudiants et étudiantes pour leurs retours de qualité qui ont contribué à façonner ce dictionnaire. En particulier, un grand merci à Mikko Seesto pour sa relecture minutieuse.

Notations et symboles

Ensembles et fonctions

$a \in \mathcal{A}$	L'objet a est un élément de l'ensemble $\mathcal{A}$.
$a := b$	On note a comme abréviation de b.
$\lvert\mathcal{A}\rvert$	Le cardinal (i.e., le nombre d'éléments) d'un ensemble fini $\mathcal{A}$.
$\mathcal{A} \subseteq \mathcal{B}$	$\mathcal{A}$ est un sous-ensemble de $\mathcal{B}$.
$\mathcal{A} \subset \mathcal{B}$	$\mathcal{A}$ est un sous-ensemble strict de $\mathcal{B}$ (i.e., non égal à $\mathcal{B}$).
$\mathbb{N}$	Les entiers naturels $1, 2, \ldots$
$\mathbb{R}$	Les nombres réels x [1].
$\mathbb{R}_+$	Les réels positifs ou nuls $x \geq 0$.
$\mathbb{R}_{++}$	Les réels strictement positifs $x > 0$.

$\{0,1\}$	L'ensemble composé des deux réels 0 et 1.
$[0,1]$	L'intervalle fermé des nombres réels x tels que $0 \leq x \leq 1$.
$\underset{\mathbf{w}}{\operatorname{argmin}} f(\mathbf{w})$	L'ensemble des point qui minimisent la fonction à valeurs réelles $f(\mathbf{w})$.
$\mathbb{S}^{(n)}$	L'ensemble des vecteurs de norme unitaire dans $\mathbb{R}^{n+1}$.
$\log a$	Le logarithme d'un réel strictement positif $a \in \mathbb{R}_{++}$.
$h(\cdot): \mathcal{A} \rightarrow \mathcal{B} : a \mapsto h(a)$	Une fonction (ou application) qui accepte tout élément $a \in \mathcal{A}$ d'un ensemble $\mathcal{A}$ en entrée et fournit un élément bien défini $h(a) \in \mathcal{B}$ d'un ensemble $\mathcal{B}$. L'ensemble $\mathcal{A}$ est le domaine de définition de la fonction h et l'ensemble $\mathcal{B}$ est l'ensemble d'arrivée de h. L'apprentissage automatique vise à trouver (ou apprendre en la construisant) une fonction h (c'est-à-dire une hypothèse) qui prend en entrée les caractéristiques $\mathbf{x}$ d'une donnée et renvoie une prédiction $h(\mathbf{x})$ pour son étiquette y.
$\nabla f(\mathbf{w})$	Le gradient d'une fonction dérivable à valeurs réelles $f : \mathbb{R}^d \rightarrow \mathbb{R}$ est le vecteur $\nabla f(\mathbf{w}) = \left(\frac{\partial f}{\partial w_1}, \ldots, \frac{\partial f}{\partial w_d}\right)^T \in \mathbb{R}^d$ [2, Ch. 9].

Matrices et Vecteurs

$\mathbf{x} = \big(x_1, \ldots, x_d\big)^T$	Un vecteur de taille d, dont la j-ième composante est x_j.
$\mathbb{R}^d$	L'ensemble des vecteurs $\mathbf{x} = \big(x_1, \ldots, x_d\big)^T$ constitués de d composantes réelles $x_1, \ldots, x_d \in \mathbb{R}$.
$\mathbf{I}_{l \times d}$	Une matrice identité généralisée de l lignes et d colonnes. Les composantes de $\mathbf{I}_{l \times d} \in \mathbb{R}^{l \times d}$ valent 1 sur la diagonale principale et 0 ailleurs.
$\mathbf{I}_d, \mathbf{I}$	Une matrice identité carrée de taille $d \times d$. Si la dimension est claire dans le contexte, on peut omettre l'indice.
$\lVert\mathbf{x}\rVert_2$	La norme euclidienne (ou ℓ_2) du vecteur $\mathbf{x} = \big(x_1, \ldots, x_d\big)^T \in \mathbb{R}^d$ définie par $\lVert\mathbf{x}\rVert_2 := \sqrt{\sum_{j=1}^{d} x_j^2}$.
$\lVert\mathbf{x}\rVert$	Une certaine norme du vecteur $\mathbf{x} \in \mathbb{R}^d$ [3]. Sauf indication contraire, on entend par là la norme euclidienne $\lVert\mathbf{x}\rVert_2$.
$\mathbf{x}^T$	La transposée d'une matrice ayant pour unique colonne le vecteur $\mathbf{x} \in \mathbb{R}^d$.
$\mathbf{X}^T$	La transposée d'une matrice $\mathbf{X} \in \mathbb{R}^{m \times d}$. Une matrice carrée à valeurs réelles $\mathbf{X} \in \mathbb{R}^{m \times m}$ est dite symétrique si $\mathbf{X} = \mathbf{X}^T$.
$\mathbf{0} = \big(0, \ldots, 0\big)^T$	Le vecteur de $\mathbb{R}^d$ dont toutes les composantes valent 0.
$\mathbf{1} = \big(1, \ldots, 1\big)^T$	Le vecteur de $\mathbb{R}^d$ dont toutes les composantes valent 1.

$\left(\mathbf{v}^T, \mathbf{w}^T\right)^T$	Le vecteur de longueur $d + d'$ obtenu en concaténant les composantes du vecteur $\mathbf{v} \in \mathbb{R}^d$ avec celles de $\mathbf{w} \in \mathbb{R}^{d'}$.
$\text{span}\{\mathbf{B}\}$	Le sous-espace engendré par une matrice $\mathbf{B} \in \mathbb{R}^{a \times b}$, c'est-à-dire l'ensemble de toutes les combinaisons linéaires des colonnes de $\mathbf{B}$: $\text{span}\{\mathbf{B}\} = \left\{\mathbf{Ba} : \mathbf{a} \in \mathbb{R}^b\right\} \subseteq \mathbb{R}^a$.
$\det(\mathbf{C})$	Le déterminant de la matrice $\mathbf{C}$.
$\mathbf{A} \otimes \mathbf{B}$	Le produit de Kronecker des matrices $\mathbf{A}$ et $\mathbf{B}$ [4].

Théorie des probabilités

$\mathbb{E}_p\{f(\mathbf{z})\}$	L'espérance d'une fonction $f(\mathbf{z})$ d'une variable aléatoire (VA) $\mathbf{z}$ dont la loi de probabilité est $p(\mathbf{z})$. Si la loi de probabilité est claire dans le contexte, on écrit simplement $\mathbb{E}\{f(\mathbf{z})\}$.
$p(\mathbf{x}, y)$	Une loi de probabilité (conjointe) d'une VA dont les realizations sont des données avec des caractéristiques $\mathbf{x}$ et un étiquette y.
$p(\mathbf{x}\|y)$	Une loi de probabilité conditionnelle d'une VA $\mathbf{x}$ étant donnée la valeur d'une autre VA y [5, Sec. 3.5].
$p(\mathbf{x}; \mathbf{w})$	Une loi de probabilité paramétrée d'une VA $\mathbf{x}$. La loi de probabilité dépend d'un vecteur de paramètres $\mathbf{w}$. Par exemple, $p(\mathbf{x}; \mathbf{w})$ pourrait être une multivariate normal distribution avec un vecteur de paramètres $\mathbf{w}$ donné par les composantes du vecteur mean $\mathbb{E}\{\mathbf{x}\}$ et la covariance matrix $\mathbb{E}\Big\{\big(\mathbf{x} - \mathbb{E}\{\mathbf{x}\}\big)\big(\mathbf{x} - \mathbb{E}\{\mathbf{x}\}\big)^T\Big\}$.
$\mathcal{N}(\mu, \sigma^2)$	La loi de probabilité d'une Gaussian random variable (Gaussian RV) $x \in \mathbb{R}$ avec mean (ou espérance) $\mu = \mathbb{E}\{x\}$ et variance $\sigma^2 = \mathbb{E}\big\{(x - \mu)^2\big\}$.
$\mathcal{N}(\boldsymbol{\mu}, \mathbf{C})$	La multivariate normal distribution d'une Gaussian RV vectorielle $\mathbf{x} \in \mathbb{R}^d$ avec mean (ou espérance) $\boldsymbol{\mu} = \mathbb{E}\{\mathbf{x}\}$ et covariance matrix $\mathbf{C} = \mathbb{E}\big\{\big(\mathbf{x} - \boldsymbol{\mu}\big)\big(\mathbf{x} - \boldsymbol{\mu}\big)^T\big\}$.

Apprentissage automatique

r	Un indice $r = 1, 2, \ldots$ qui énumère les données.
m	Le nombre de données dans un jeu de données (c'est-à-dire la taille du jeu de données).
$\mathcal{D}$	Un jeu de données $\mathcal{D} = \{\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(m)}\}$ est une liste de données individuels $\mathbf{z}^{(r)}$, pour $r = 1, \ldots, m$.
d	Le nombre de caractéristiques qui caractérisent un donnée.
x_j	Le j-ième caractéristique d'un donnée. Le premier caractéristique est noté x_1, le deuxième x_2, et ainsi de suite.
$\mathbf{x}$	Le feature vector $\mathbf{x} = \big(x_1, \ldots, x_d\big)^T$ d'un donnée, dont les composantes sont les différents caractéristiques du donnée.
$\mathcal{X}$	Le feature space $\mathcal{X}$ est l'ensemble de toutes les valeurs possibles que les caractéristiques $\mathbf{x}$ d'un donnée peuvent prendre.
$\mathbf{z}$	Au lieu du symbole $\mathbf{x}$, on utilise parfois $\mathbf{z}$ comme un autre symbole pour désigner un vecteur dont les composantes sont les différents caractéristiques d'un donnée. On a besoin de deux symboles différents pour distinguer les caractéristiques bruts des caractéristiques appris [6, Ch. 9].
$\mathbf{x}^{(r)}$	Le vecteur de caractéristiques du r-ième donnée dans un jeu de données.
$x_j^{(r)}$	Le j-ième caractéristique du r-ième donnée dans un jeu de données.

$\mathcal{B}$	Un mini-batch (ou sous-ensemble) de données choisis aléatoirement.
B	La taille (c'est-à-dire le nombre de données) d'un mini-batch.
y	Le étiquette (ou quantité d'intérêt) d'un donnée.
$y^{(r)}$	Le étiquette du r-ième donnée.
$\left(\mathbf{x}^{(r)}, y^{(r)}\right)$	Les caractéristiques et le étiquette du r-ième donnée.
$\mathcal{Y}$	L'label space $\mathcal{Y}$ d'une méthode d'apprentissage automatique comprend toutes les valeurs de étiquette possibles qu'un donnée peut porter. L'label space nominale peut être plus grande que l'ensemble des différentes valeurs de étiquette présentes dans un jeu de données donné (par exemple, un training set). Les problèmes (ou méthodes) de apprentissage automatique utilisant une label space numérique, comme $\mathcal{Y} = \mathbb{R}$ ou $\mathcal{Y} = \mathbb{R}^3$, sont appelés problèmes (ou méthodes) de regression. Les problèmes (ou méthodes) de apprentissage automatique utilisant une label space discrète, comme $\mathcal{Y} = \{0, 1\}$ ou $\mathcal{Y} = \{\mathit{chat}, \mathit{chien}, \mathit{souris}\}$, sont appelés problèmes (ou méthodes) de classification.
η	Le learning rate (ou step size) utilisé par les gradient-based methodss.
$h(\cdot)$	Une fonction hypothèse qui lit les caractéristiques $\mathbf{x}$ d'un donnée et produit une prédiction $\hat{y} = h(\mathbf{x})$ pour son étiquette y.
$\mathcal{Y}^{\mathcal{X}}$	Étant donnés deux ensembles $\mathcal{X}$ et $\mathcal{Y}$, on note $\mathcal{Y}^{\mathcal{X}}$

$\mathcal{H}$	Un hypothesis space ou model utilisé par une méthode d'apprentissage automatique. Le hypothesis space est constitué de différentes fonctions hypothèse $h : \mathcal{X} \rightarrow \mathcal{Y}$, parmi lesquelles la méthode d'apprentissage automatique doit choisir.
$d_{\text{eff}}(\mathcal{H})$	La effective dimension d'un hypothesis space $\mathcal{H}$.
B^2	Le bias au carré d'une fonction hypothèse apprise $\hat{h}$ produite par une méthode d'apprentissage automatique. La méthode est entraînée sur des données modélisés comme des realizations de VAs. Puisque les datas sont des realizations de VAs, la fonction apprise $\hat{h}$ est également une realization d'une VA.
V	La variance de la fonction hypothèse apprise (ou de ses parameterss) par une méthode d'apprentissage automatique. La méthode est entraînée sur des données modélisés comme des realizations de VAs. Puisque les datas sont des realizations de VAs, la fonction apprise $\hat{h}$ est également une realization d'une VA.
$L((\mathbf{x}, y), h)$	La loss encourue lors de la prédiction du étiquette y d'un donnée à l'aide de la prédiction $\hat{y} = h(\mathbf{x})$. La prédiction $\hat{y}$ est obtenue en évaluant la fonction hypothèse $h \in \mathcal{H}$ sur le feature vector $\mathbf{x}$ du donnée.
E_v	La validation error d'une fonction hypothèse h, c'est-à-dire sa loss moyenne sur un validation set.
$\widehat{L}(h\|\mathcal{D})$	L'empirical risk, ou loss moyenne, encourue par la fonction hypothèse h sur un jeu de données $\mathcal{D}$.

E_t	L’training error d’une fonction hypothèse h, c’est-à-dire sa loss moyenne sur un training set.
t	Un indice temporel discret $t = 0, 1, \ldots$ utilisé pour énumérer des événements séquentiels (ou des instants de temps).
t	Un indice qui énumère les learning tasks dans un problème d’multitask learning.
α	Un paramètre de regularization qui contrôle la quantité de regularization.
$\lambda_j(\mathbf{Q})$	La j-ième eigenvalue (triée par ordre croissant ou décroissant) d’une matrice positive semi-definite (psd) $\mathbf{Q}$. On utilise aussi l’abréviation λ_j si la matrice est claire par le contexte.
$\sigma(\cdot)$	La activation function utilisée par un neurone artificiel dans un artificial neural network (ANN).
$\mathcal{R}_{\hat{y}}$	Une decision region dans un feature space.
$\mathbf{w}$	Un vecteur de paramètres $\mathbf{w} = \big(w_1, \ldots, w_d\big)^T$ d’un model, par exemple les weights d’un linear model ou dans un ANN.
$h^{(\mathbf{w})}(\cdot)$	Une fonction hypothèse qui dépend de model parameterss ajustables $w_1, \ldots, w_d$ regroupés dans le vecteur $\mathbf{w} = \big(w_1, \ldots, w_d\big)^T$.
$\phi(\cdot)$	Une feature map $\phi : \mathcal{X} \rightarrow \mathcal{X}' : \mathbf{x} \mapsto \mathbf{x}' := \phi\big(\mathbf{x}\big) \in \mathcal{X}'$.
$K\big(\cdot, \cdot\big)$	Étant donné un feature space $\mathcal{X}$, un kernel est une application $K : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{C}$ qui est psd.

Federated Learning

$\mathcal{G} = (\mathcal{V}, \mathcal{E})$	An undirected graph whose nodes $i \in \mathcal{V}$ represent devices within a federated learning network (FL network). The undirected weighted edges $\mathcal{E}$ represent connectivity between devices and statistical similarities between their jeu de donnéess and learning tasks.
$i \in \mathcal{V}$	A node that represents some device within an FL network. The device can access a local dataset and train a local model.
$\mathcal{G}^{(\mathcal{C})}$	The induced subgraph of $\mathcal{G}$ using the nodes in $\mathcal{C} \subseteq \mathcal{V}$.
$\mathbf{L}^{(\mathcal{G})}$	The Laplacian matrix of a graph $\mathcal{G}$.
$\mathbf{L}^{(\mathcal{C})}$	The Laplacian matrix of the induced graph $\mathcal{G}^{(\mathcal{C})}$.
$\mathcal{N}^{(i)}$	The neighborhood of a node i in a graph $\mathcal{G}$.
$d^{(i)}$	The weighted degree $d^{(i)} := \sum_{i' \in \mathcal{N}^{(i)}} A_{i,i'}$ of a node i in a graph $\mathcal{G}$.
$d_{\max}^{(\mathcal{G})}$	The maximum weighted node degree of a graph $\mathcal{G}$.
$\mathcal{D}^{(i)}$	The local dataset $\mathcal{D}^{(i)}$ carried by node $i \in \mathcal{V}$ of an FL network.
m_i	The number of données (i.e., sample size) contained in the local dataset $\mathcal{D}^{(i)}$ at node $i \in \mathcal{V}$.

$\mathbf{x}^{(i,r)}$	The caractéristiques of the r-th donnée in the local dataset $\mathcal{D}^{(i)}$.
$y^{(i,r)}$	The étiquette of the r-th donnée in the local dataset $\mathcal{D}^{(i)}$.
$\mathbf{w}^{(i)}$	The local model parameters of device i within an FL network.
$L_i\left(\mathbf{w}\right)$	The local loss function used by device i to measure the usefulness of some choice $\mathbf{w}$ for the local model parameters.
$L^{(\mathrm{d})}\left(\mathbf{x}, h\big(\mathbf{x}\big), h'\big(\mathbf{x}\big)\right)$	The loss incurred by a hypothèse h' on a donnée with caractéristiques $\mathbf{x}$ and étiquette $h\big(\mathbf{x}\big)$ that is obtained from another hypothèse.
$\mathrm{stack}\big\{\mathbf{w}^{(i)}\big\}_{i=1}^{n}$	The vector $\bigg(\big(\mathbf{w}^{(1)}\big)^{T}, \ldots, \big(\mathbf{w}^{(n)}\big)^{T}\bigg)^{T} \in \mathbb{R}^{dn}$ that is obtained by vertically stacking the local model parameters $\mathbf{w}^{(i)} \in \mathbb{R}^{d}$.

Machine Learning Concepts

k-fold cross-validation (k-fold CV) k-fold CV is a method for learning and validating a hypothèse using a given jeu de données. This method divides the jeu de données evenly into k subsets or folds and then executes k repetitions of model training (e.g., via empirical risk minimization (ERM)) and validation. Each repetition uses a different fold as the validation set and the remaining $k-1$ folds as a training set. The final output is the average of the validation errors obtained from the k repetitions.

k-means The k-means algorithm is a hard clustering method which assigns each donnée of a jeu de données to precisely one of k different clusters. The method alternates between updating the cluster assignments (to the cluster with the nearest mean) and, given the updated cluster assignments, re-calculating the cluster means [6, Ch. 8].

absolute error loss Consider a donnée with caractéristiques $\mathbf{x} \in \mathcal{X}$ and numeric étiquette $y \in \mathbb{R}$. The absolute error loss incurred by a hypothèse $h : \mathcal{X} \rightarrow \mathbb{R}$ is defined as $|y - h(\mathbf{x})|$, i.e., the absolute difference between the prédiction $h(\mathbf{x})$ and the true étiquette y.

accuracy Consider données characterized by caractéristiques $\mathbf{x} \in \mathcal{X}$ and a categorical label y which takes on values from a finite label space $\mathcal{Y}$. The accuracy of a hypothèse $h : \mathcal{X} \rightarrow \mathcal{Y}$, when applied to the données in a jeu de données $\mathcal{D} = \{(\mathbf{x}^{(1)}, y^{(1)}), \ldots, (\mathbf{x}^{(m)}, y^{(m)})\}$, is then defined as $1 - (1/m)\sum_{r=1}^{m} L^{(0/1)}\left((\mathbf{x}^{(r)}, y^{(r)}), h\right)$ using the 0/1 loss $L^{(0/1)}(\cdot, \cdot)$.

activation function Each artificial neuron within an ANN is assigned an activation function $\sigma(\cdot)$ that maps a weighted combination of the neuron inputs $x_1, \ldots, x_d$ to a single output value $a = \sigma(w_1 x_1 + \ldots + w_d x_d)$. Note that each neuron is parametrized by the weights $w_1, \ldots, w_d$.

algorithm An algorithm is a precise, step-by-step specification for how to produce an output from a given input within a finite number of computational steps [7]. For example, an algorithm for training a linear model explicitly describes how to transform a given training set into model parameters through a sequence of gradient steps. This informal characterization can be formalized rigorously via different mathematical models [8]. One very simple model of an algorithm is a collection of possible executions. Each execution is a sequence:

$$\text{input}, s_1, s_2, \ldots, s_T, \text{output}$$

that respects the constraints inherent to the computer executing the algorithm. Algorithms may be deterministic, where each input results uniquely in a single execution, or randomized, where executions can vary probabilistically. Randomized algorithms can thus be analyzed by modeling execution sequences as outcomes of random experiments, viewing the algorithm as a stochastic process [5, 9, 10]. Crucially, an algorithm encompasses more than just a mapping from input to output; it also includes the intermediate computational steps $s_1, \ldots, s_T$.

application programming interface (API) An API is a formal mechanism for enabling software components to interact in a structured manner [11]. In the context of apprentissage automatique, APIs are frequently used to make a trained apprentissage automatique model accessible to different types of users. These users, which can be other computers or humans, can request a prédiction for the étiquette of a donnée by providing its caractéristiques. The internal structure of the apprentissage automatique model remains hidden from the user. For instance, consider a trained apprentissage automatique model $\widehat{h}(x) := 2x + 1$. An API enables a user to submit the caractéristique value $x = 3$ and obtain the response $\widehat{h}(3) = 7$ without knowledge of the detailed structure of the apprentissage automatique model or its training. In practice, the apprentissage automatique model is typically hosted on a computer (i.e., a server) connected to the internet. Another computer (i.e., a client) sends the caractéristiques of a donnée to the server, which then computes $\widehat{h}(\mathbf{x})$ and returns the result to the external system. APIs help to modularize the development of apprentissage automatique applications by decoupling specific tasks. For instance, one team can focus on developing and training the model, while another team handles user interaction and integration of the model into applications.

apprentissage automatique ML aims to predict a étiquette from the caractéristiques of a donnée. ML methods achieve this by learning a hypothèse from a hypothesis space (or model) through the minimization of a loss function [6, 75]. One precise formulation of this principle is ERM. Different ML methods are obtained from different design choices for données (their caractéristiques and étiquette), model, and loss function [6, Ch. 3].

artificial intelligence (AI) AI refers to systems that behave rationally in the sense of maximizing a long-term reward. The apprentissage automatique-based approach to AI is to train a model for predicting optimal actions. These predictions are computed from observations about the state of the environment. The choice of loss function sets AI applications apart from more basic apprentissage automatique applications. AI systems rarely have access to a labeled training set that allows the average loss to be measured for any possible choice of model parameters. Instead, AI systems use observed reward signals to obtain a (point-wise) estimate for the loss incurred by the current choice of model parameters.

artificial neural network (ANN) An ANN is a graphical (signal-flow) representation of a function that maps caractéristiques of a donnée at its input to a prédiction for the corresponding étiquette at its output. The fundamental unit of an ANN is the artificial neuron, which applies an activation function to its weighted inputs. The outputs of these neurons serve as inputs for other neurons, forming interconnected layers.

autoencoder An autoencoder is an apprentissage automatique method that simultaneously learns an encoder map $h(\cdot) \in \mathcal{H}$ and a decoder map $h^{*}(\cdot) \in \mathcal{H}^{*}$. It is an instance of ERM using a loss computed from the reconstruction error $\mathbf{x} - h^{*}\big(h(\mathbf{x})\big)$.

backdoor A backdoor attack refers to the intentional manipulation of the training process underlying an apprentissage automatique method. This manipulation can be implemented by perturbing the training set (data poisoning) or the optimization algorithm used by an ERM-based method. The goal of a backdoor attack is to nudge the learned hypothèse $\hat{h}$ towards specific prédictions for a certain range of caractéristique values. This range of caractéristique values serves as a key (or trigger) to unlock a backdoor in the sense of delivering anomalous prédictions. The key $\mathbf{x}$ and the corresponding anomalous prédiction $\hat{h}(\mathbf{x})$ are only known to the attacker.

bagging Bagging (or bootstrap aggregation) is a generic technique to improve (the robustness of) a given apprentissage automatique method. The idea is to use the bootstrap to generate perturbed copies of a given jeu de données and then to learn a separate hypothèse for each copy. We then predict the étiquette of a donnée by combining or aggregating the individual prédictions of each separate hypothèse. For hypothèse maps delivering numeric étiquette values, this aggregation could be implemented by computing the average of individual prédictions.

baseline Consider some apprentissage automatique method that produces a learned hypothèse (or trained model) $\hat{h} \in \mathcal{H}$. We evaluate the quality of a trained model by computing the average loss on a test set. But how can we assess whether the resulting test set performance is sufficiently good? How can we determine if the trained model performs close to optimal and there is little point in investing more resources (for data collection or computation) to improve it? To this end, it is useful to have a reference (or baseline) level against which we can compare the performance of the trained model. Such a reference value might be obtained from human performance, e.g., the misclassification rate of dermatologists who diagnose cancer from visual inspection of skin [12]. Another source for a baseline is an existing, but for some reason unsuitable, apprentissage automatique method. For example, the existing apprentissage automatique method might be computationally too expensive for the intended apprentissage automatique application. Nevertheless, its test set error can still serve as a baseline. Another, somewhat more principled, approach to constructing a baseline is via a probabilistic model. In many cases, given a probabilistic model $p(\mathbf{x}, y)$, we can precisely determine the minimum achievable risk among any hypotheses (not even required to belong to the hypothesis space $\mathcal{H}$) [13]. This minimum achievable risk (referred to as the Bayes risk) is the risk of the Bayes estimator for the étiquette y of a donnée, given its caractéristiques $\mathbf{x}$. Note that, for a given choice of loss function, the Bayes estimator (if it exists) is completely determined by the loi de probabilité $p(\mathbf{x}, y)$ [13, Ch. 4]. However, computing the Bayes estimator and Bayes risk presents two main challenges:

1) The loi de probabilité $p(\mathbf{x}, y)$ is unknown and needs to be estimated.
2) Even if $p(\mathbf{x}, y)$ is known, it can be computationally too expensive to compute the Bayes risk exactly [14].

A widely used probabilistic model is the multivariate normal distribution $(\mathbf{x}, y) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ for données characterized by numeric caractéristiques and étiquettes. Here, for the squared error loss, the Bayes estimator is given by the posterior mean $\mu_{y|\mathbf{x}}$ of the étiquette y, given the caractéristiques $\mathbf{x}$ [13, 15]. The corresponding Bayes risk is given by the posterior variance $\sigma^2_{y|\mathbf{x}}$ (see Figure 1).

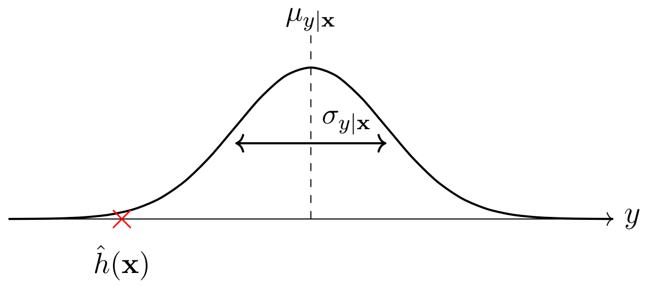


Figure 1: If the caractéristiques and the étiquette of a donnée are drawn from a multivariate normal distribution, we can achieve the minimum risk (under squared error loss) by using the Bayes estimator $\mu_{y|\mathbf{x}}$ to predict the étiquette y of a donnée with caractéristiques $\mathbf{x}$. The corresponding minimum risk is given by the posterior variance $\sigma^2_{y|\mathbf{x}}$. We can use this quantity as a baseline for the average loss of a trained model $\hat{h}$.

batch In the context of stochastic gradient descent (SGD), a batch refers to a randomly chosen subset of the overall training set. We use the données in this subset to estimate the gradient of training error and, in turn, to update the model parameters.

Bayes estimator Consider a probabilistic model with a joint loi de probabilité $p(\mathbf{x}, y)$ for the caractéristiques $\mathbf{x}$ and étiquette y of a donnée. For a given loss function $L(\cdot, \cdot)$, we refer to a hypothèse h as a Bayes estimator if its risk $\mathbb{E}\{L\left(\left(\mathbf{x}, y\right), h\right)\}$ is the minimum [13]. Note that the property of a hypothèse being a Bayes estimator depends on the underlying loi de probabilité and the choice for the loss function $L\left(\cdot, \cdot\right)$.

Bayes risk Consider a probabilistic model with a joint loi de probabilité $p(\mathbf{x}, y)$ for the caractéristiques $\mathbf{x}$ and étiquette y of a donnée. The Bayes risk is the minimum possible risk that can be achieved by any hypothèse $h : \mathcal{X} \rightarrow \mathcal{Y}$. Any hypothèse that achieves the Bayes risk is referred to as a Bayes estimator [13].

bias Consider an apprentissage automatique method using a parametrized hypothesis space $\mathcal{H}$. It learns the model parameters $\mathbf{w} \in \mathbb{R}^d$ using the jeu de données

$$\mathcal{D} = \left\{ \left(\mathbf{x}^{(r)}, y^{(r)}\right) \right\}_{r=1}^{m}.$$

To analyze the properties of the apprentissage automatique method, we typically interpret the données as realizations of independent and identically distributed (i.i.d.) VAs,

$$y^{(r)} = h^{(\overline{\mathbf{w}})}\big(\mathbf{x}^{(r)}\big) + \boldsymbol{\varepsilon}^{(r)}, r = 1, \ldots, m.$$

We can then interpret the apprentissage automatique method as an estimator $\widehat{\mathbf{w}}$ computed from $\mathcal{D}$ (e.g., by solving ERM). The (squared) bias incurred by the estimate $\widehat{\mathbf{w}}$ is then defined as $B^2 := \big\|\mathbb{E}\{\widehat{\mathbf{w}}\} - \overline{\mathbf{w}}\big\|_2^2$.

bootstrap For the analysis of apprentissage automatique methods, it is often useful to interpret a given set of données $\mathcal{D} = \left\{\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(m)}\right\}$ as realizations of i.i.d. VAs with a common loi de probabilité $p(\mathbf{z})$. In general, we do not know $p(\mathbf{z})$ exactly, but we need to estimate it. The bootstrap uses the histogram of $\mathcal{D}$ as an estimator for the underlying loi de probabilité $p(\mathbf{z})$.

borne supérieure The supremum of a set of real numbers is the smallest number that is greater than or equal to every element in the set. More formally, a real number a is the supremum of a set $\mathcal{A} \subseteq \mathbb{R}$ if: 1) a is an upper bound of $\mathcal{A}$; and 2) no number smaller than a is an upper bound of $\mathcal{A}$. Every non-empty set of real numbers that is bounded above has a supremum, even if it does not contain its supremum as an element [2, Sec. 1.4].

caractéristique A feature of a donnée is one of its properties that can be measured or computed easily without the need for human supervision. For example, if a donnée is a digital image (e.g., stored as a `.jpeg` file), then we could use the red-green-blue intensities of its pixels as features. Domain-specific synonyms for the term feature are "covariate," "explanatory variable," "independent variable," "input (variable)," "predictor (variable)," or "regressor" [49], [50], [51].

classification Classification is the task of determining a discrete-valued label y for a given donnée, based solely on its features $\mathbf{x}$. The label y belongs to a finite set, such as $y \in \{-1, 1\}$ or $y \in \{1, \ldots, 19\}$, and represents the category to which the corresponding donnée belongs.

classifier A classifier is a hypothèse (map) $h(\mathbf{x})$ used to predict a étiquette taking values from a finite label space. We might use the function value $h(\mathbf{x})$ itself as a prédiction $\hat{y}$ for the étiquette. However, it is customary to use a map $h(\cdot)$ that delivers a numeric quantity. The prédiction is then obtained by a simple thresholding step. For example, in a binary classification problem with $\mathcal{Y} \in \{-1, 1\}$, we might use a real-valued hypothèse map $h(\mathbf{x}) \in \mathbb{R}$ as a classifier. A prédiction $\hat{y}$ can then be obtained via thresholding,

$$\hat{y} = 1 \text{ for } h(\mathbf{x}) \geq 0 \text{ and } \hat{y} = -1 \text{ otherwise.} \tag{1}$$

We can characterize a classifier by its decision regions $\mathcal{R}_a$, for every possible étiquette value $a \in \mathcal{Y}$.

cluster A cluster is a subset of données that are more similar to each other than to the données outside the cluster. The quantitative measure of similarity between données is a design choice. If données are characterized by Euclidean feature vectors $\mathbf{x} \in \mathbb{R}^d$, we can define the similarity between two données via the Euclidean distance between their feature vectors.

clustered federated learning (CFL) Clustered federated learning (FL) assumes that local datasets are naturally grouped into clusters. The local datasets belonging to the same cluster have similar statistical properties. Clustered FL aggregates local datasets in the same cluster to obtain a training set for the training of a cluster-specific model. Generalized total variation minimization (GTVMin) facilitates this clustering implicitly by enforcing approximate similarity of model parameters across well-connected subsets of the FL network.

clustering Clustering methods decompose a given set of données into a few subsets, which are referred to as clusters. Each cluster consists of données that are more similar to each other than to données outside the cluster. Different clustering methods use different measures for the similarity between données and different forms of cluster representations. The clustering method k-means uses the average caractéristique vector (cluster mean) of a cluster as its representative. A popular soft clustering method based on Gaussian mixture model (GMM) represents a cluster by a multivariate normal distribution.

clustering assumption The clustering assumption postulates that données in a jeu de données form a (small) number of groups or clusters. Données in the same cluster are more similar to each other than those outside the cluster [16]. We obtain different clustering methods by using different notions of similarity between données.

computational aspects By computational aspects of an apprentissage automatique method, we mainly refer to the computational resources required for its implementation. For example, if an apprentissage automatique method uses iterative optimization techniques to solve ERM, then its computational aspects include: 1) how many arithmetic operations are needed to implement a single iteration (gradient step); and 2) how many iterations are needed to obtain useful model parameters. One important example of an iterative optimization technique is gradient descent (GD).

condition number The condition number $\kappa(\mathbf{Q}) \geq 1$ of a positive definite matrix $\mathbf{Q} \in \mathbb{R}^{d \times d}$ is the ratio α/β between the largest α and the smallest β eigenvalue of $\mathbf{Q}$. The condition number is useful for the analysis of apprentissage automatique methods. The computational complexity of gradient-based methods for linear regression crucially depends on the condition number of the matrix $\mathbf{Q} = \mathbf{X}\mathbf{X}^T$, with the feature matrix $\mathbf{X}$ of the training set. Thus, from a computational perspective, we prefer caractéristiques of données such that $\mathbf{Q}$ has a condition number close to 1.

confusion matrix Consider données characterized by caractéristiques $\mathbf{x}$ and étiquette y having values from the finite label space $\mathcal{Y} = \{1, \ldots, k\}$. The confusion matrix is a $k \times k$ matrix with rows representing different values c of the true label of a donnée. The columns of a confusion matrix correspond to different values c' delivered by a hypothesis $h(\mathbf{x})$. The (c, c')-th entry of the confusion matrix is the fraction of données with the étiquette $y = c$ and the prédiction $\hat{y} = c'$ assigned by the hypothèse h.

connected graph An undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is connected if every non-empty subset $\mathcal{V}' \subset \mathcal{V}$ has at least one edge connecting it to $\mathcal{V} \setminus \mathcal{V}'$.

convex A subset $\mathcal{C} \subseteq \mathbb{R}^d$ of the Euclidean space $\mathbb{R}^d$ is referred to as convex if it contains the line segment between any two points $\mathbf{x}, \mathbf{y} \in \mathcal{C}$ in that set. A function $f : \mathbb{R}^d \rightarrow \mathbb{R}$ is convex if its epigraph $\left\{ \left(\mathbf{w}^T, t \right)^T \in \mathbb{R}^{d+1} : t \geq f(\mathbf{w}) \right\}$ is a convex set [17]. We illustrate one example of a convex set and a convex function in Figure 2.

$\mathbf{w}$
$\mathbf{w}'$
$\mathcal{C}$
$f(\mathbf{w})$

Figure 2: Left: A convex set $\mathcal{C} \subseteq \mathbb{R}^d$. Right: A convex function $f : \mathbb{R}^d \rightarrow \mathbb{R}$.

convex clustering Consider a jeu de données $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)} \in \mathbb{R}^d$. Convex clustering learns vectors $\mathbf{w}^{(1)}, \ldots, \mathbf{w}^{(m)}$ by minimizing

$$\sum_{r=1}^{m} \left\| \mathbf{x}^{(r)} - \mathbf{w}^{(r)} \right\|_2^2 + \alpha \sum_{i,i' \in \mathcal{V}} \left\| \mathbf{w}^{(i)} - \mathbf{w}^{(i')} \right\|_p .$$

Here, $\|\mathbf{u}\|_p := \left(\sum_{j=1}^{d} |u_j|^p \right)^{1/p}$ denotes the p-norme (for $p \geq 1$). It turns out that many of the optimal vectors $\widehat{\mathbf{w}}^{(1)}, \ldots, \widehat{\mathbf{w}}^{(m)}$ coincide. A cluster then consists of those données $r \in \{1, \ldots, m\}$ with identical $\widehat{\mathbf{w}}^{(r)}$ [18, 19].

Courant–Fischer–Weyl min-max characterization Consider a psd matrix $\mathbf{Q} \in \mathbb{R}^{d \times d}$ with eigenvalue decomposition (EVD) (or spectral decomposition),

$$\mathbf{Q} = \sum_{j=1}^{d} \lambda_j \mathbf{u}^{(j)} \big(\mathbf{u}^{(j)} \big)^T .$$

Here, we use the ordered (in increasing fashion) eigenvalues

$$\lambda_1 \leq \ldots \leq \lambda_n .$$

The Courant–Fischer–Weyl min-max characterization [3, Th. 8.1.2] represents the eigenvalues of $\mathbf{Q}$ as the solutions to certain optimization problems.

covariance matrix The covariance matrix of an VA $\mathbf{x} \in \mathbb{R}^d$ is defined as $\mathbb{E}\left\{\big(\mathbf{x} - \mathbb{E}\{\mathbf{x}\}\big)\big(\mathbf{x} - \mathbb{E}\{\mathbf{x}\}\big)^T\right\}$.

data Data refers to objects that carry information. These objects can be either concrete physical objects (such as persons or animals) or abstract concepts (such as numbers). We often use representations (or approximations) of the original data that are more convenient for data processing. These approximations are based on different data models, with the relational data model being one of the most widely used [20].

data augmentation Data augmentation methods add synthetic données to an existing set of données. These synthetic données are obtained by perturbations (e.g., adding noise to physical measurements) or transformations (e.g., rotations of images) of the original données. These perturbations and transformations are such that the resulting synthetic données should still have the same étiquette. As a case in point, a rotated cat image is still a cat image even if their feature vectors (obtained by stacking pixel color intensities) are very different (see Figure 3). Data augmentation can be an efficient form of regularization.

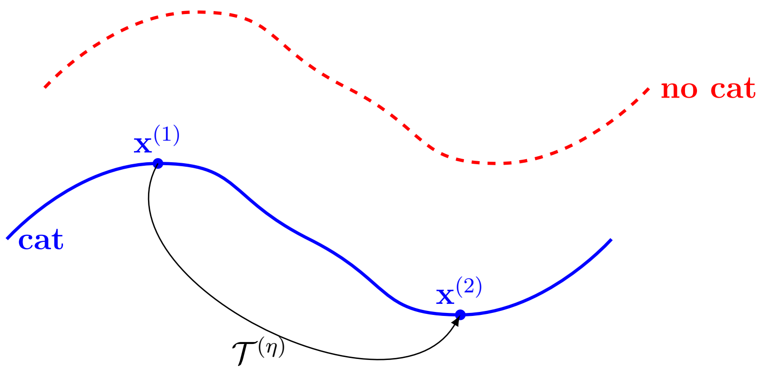


Figure 3: Data augmentation exploits intrinsic symmetries of données in some feature space $\mathcal{X}$. We can represent a symmetry by an operator $\mathcal{T}^{(\eta)} : \mathcal{X} \rightarrow \mathcal{X}$, parametrized by some number $\eta \in \mathbb{R}$. For example, $\mathcal{T}^{(\eta)}$ might represent the effect of rotating a cat image by η degrees. A donnée with feature vector $\mathbf{x}^{(2)} = \mathcal{T}^{(\eta)}\big(\mathbf{x}^{(1)}\big)$ must have the same étiquette $y^{(2)} = y^{(1)}$ as a donnée with feature vector $\mathbf{x}^{(1)}$.

data minimization principle European data protection regulation includes a data minimization principle. This principle requires a data controller to limit the collection of personal information to what is directly relevant and necessary to accomplish a specified purpose. The data should be retained only for as long as necessary to fulfill that purpose [21, Article 5(1)(c)], [22].

data normalization Data normalization refers to transformations applied to the feature vectors of données to improve the apprentissage automatique method's statistical aspects or computational aspects. For example, in linear regression with gradient-based methods using a fixed learning rate, convergence depends on controlling the norme of feature vectors in the training set. A common approach is to normalize feature vectors such that their norme does not exceed one [6, Ch. 5].

data poisoning Data poisoning refers to the intentional manipulation (or fabrication) of données to steer the training of an apprentissage automatique model [24, 25]. The protection against data poisoning is particularly important in distributed apprentissage automatique applications where jeu de donnéess are decentralized.

decision boundary Consider a hypothèse map h that reads in a caractéristique vector $\mathbf{x} \in \mathbb{R}^d$ and delivers a value from a finite set $\mathcal{Y}$. The decision boundary of h is the set of vectors $\mathbf{x} \in \mathbb{R}^d$ that lie between different decision regions. More precisely, a vector $\mathbf{x}$ belongs to the decision boundary if and only if each neighborhood $\{\mathbf{x}' : \|\mathbf{x} - \mathbf{x}'\| \leq \varepsilon\}$, for any $\varepsilon > 0$, contains at least two vectors with different function values.

decision region Consider a hypothèse map h that delivers values from a finite set $\mathcal{Y}$. For each étiquette value (category) $a \in \mathcal{Y}$, the hypothèse h determines a subset of caractéristique values $\mathbf{x} \in \mathcal{X}$ that result in the same output $h(\mathbf{x}) = a$. We refer to this subset as a decision region of the hypothèse h.

decision tree A decision tree is a flow-chart-like representation of a hypothèse map h. More formally, a decision tree is a directed graph containing a root node that reads in the feature vector $\mathbf{x}$ of a donnée. The root node then forwards the donnée to one of its children nodes based on some elementary test on the caractéristiques $\mathbf{x}$. If the receiving child node is not a leaf node, i.e., it has itself children nodes, it represents another test. Based on the test result, the donnée is forwarded to one of its descendants. This testing and forwarding of the donnée is continued until the donnée ends up in a leaf node (having no children nodes).

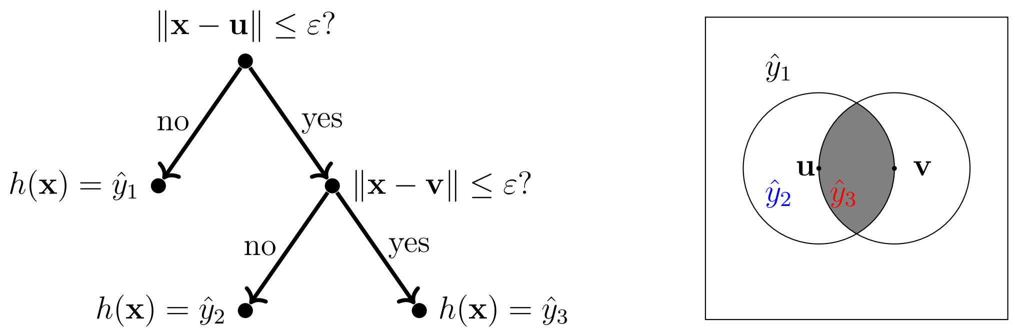

$\|\mathbf{x}-\mathbf{u}\| \leq \varepsilon$?
no
yes
$h(\mathbf{x}) = \hat{y}_1$
$\|\mathbf{x}-\mathbf{v}\| \leq \varepsilon$?
no
yes
$h(\mathbf{x}) = \hat{y}_2$
$h(\mathbf{x}) = \hat{y}_3$
$\hat{y}_1$
$\mathbf{u}$
$\mathbf{v}$
$\hat{y}_2$
$\hat{y}_3$

Figure 4: Left: A decision tree is a flow-chart-like representation of a piecewise constant hypothèse $h : \mathcal{X} \rightarrow \mathbb{R}$. Each piece is a decision region $\mathcal{R}_{\hat{y}} := \big\{\mathbf{x} \in \mathcal{X} : h(\mathbf{x}) = \hat{y}\big\}$. The depicted decision tree can be applied to numeric feature vectors, i.e., $\mathcal{X} \subseteq \mathbb{R}^d$. It is parametrized by the threshold $\varepsilon > 0$ and the vectors $\mathbf{u}, \mathbf{v} \in \mathbb{R}^d$. Right: A decision tree partitions the feature space $\mathcal{X}$ into decision regions. Each decision region $\mathcal{R}_{\hat{y}} \subseteq \mathcal{X}$ corresponds to a specific leaf node in the decision tree.

deep net A deep net is an ANN with a (relatively) large number of hidden layers. Deep learning is an umbrella term for apprentissage automatique methods that use a deep net as their model [31].

degree of belonging Degree of belonging is a number that indicates the extent to which a donnée belongs to a cluster [6, Ch. 8]. The degree of belonging can be interpreted as a soft cluster assignment. Soft clustering methods can encode the degree of belonging by a real number in the interval $[0, 1]$. Hard clustering is obtained as the extreme case when the degree of belonging only takes on values 0 or 1.

denial-of-service attack A denial-of-service attack aims (e.g., via data poisoning) to steer the training of a model such that it performs poorly for typical données.

density-based spatial clustering of applications with noise (DBSCAN) DBSCAN refers to a clustering algorithm for données that are characterized by numeric feature vectors. Like k-means and soft clustering via GMM, also DBSCAN uses the Euclidean distances between feature vectors to determine the clusters. However, in contrast to k-means and GMM, DBSCAN uses a different notion of similarity between données. DBSCAN considers two données as similar if they are connected via a sequence (path) of close-by intermediate données. Thus, DBSCAN might consider two données as similar (and therefore belonging to the same cluster) even if their feature vectors have a large Euclidean distance.

device Any physical system that can be used to store and process data. In the context of apprentissage automatique, we typically mean a computer that is able to read in données from different sources and, in turn, to train an apprentissage automatique model using these données.

differential privacy (DP) Consider some apprentissage automatique method $\mathcal{A}$ that reads in a jeu de données (e.g., the training set used for ERM) and delivers some output $\mathcal{A}(\mathcal{D})$. The output could be either the learned model parameters or the prédictions for specific données. DP is a precise measure of privacy leakage incurred by revealing the output. Roughly speaking, an apprentissage automatique method is differentially private if the loi de probabilité of the output $\mathcal{A}(\mathcal{D})$ does not change too much if the sensitive attribute of one donnée in the training set is changed. Note that DP builds on a probabilistic model for an apprentissage automatique method, i.e., we interpret its output $\mathcal{A}(\mathcal{D})$ as the realization of an VA. The randomness in the output can be ensured by intentionally adding the realization of an auxiliary VA (noise) to the output of the apprentissage automatique method.

dimensionality reduction Dimensionality reduction methods map (typically many) raw caractéristiques to a (relatively small) set of new caractéristiques. These methods can be used to visualize données by learning two caractéristiques that can be used as the coordinates of a depiction in a scatterplot.

discrepancy Consider an FL application with networked data represented by an FL network. FL methods use a discrepancy measure to compare hypothèse maps from local models at nodes i, i' connected by an edge in the FL network.

distributed algorithm A distributed algorithm is an algorithm designed for a special type of computer: a collection of interconnected computing devices (or nodes). These devices communicate and coordinate their local computations by exchanging messages over a network [32, 33]. Unlike a classical algorithm, which is implemented on a single device, a distributed algorithm is executed concurrently on multiple devices with computational capabilities. Similar to a classical algorithm, a distributed algorithm can be modeled as a set of potential executions. However, each execution in the distributed setting involves both local computations and message-passing events. A generic execution might look as follows:

$$\begin{aligned}
&\text{Node 1: } \text{input}_1, s_1^{(1)}, s_2^{(1)}, \ldots, s_{T_1}^{(1)}, \text{output}_1;\\
&\text{Node 2: } \text{input}_2, s_1^{(2)}, s_2^{(2)}, \ldots, s_{T_2}^{(2)}, \text{output}_2;\\
&\quad\vdots\\
&\text{Node N: } \text{input}_N, s_1^{(N)}, s_2^{(N)}, \ldots, s_{T_N}^{(N)}, \text{output}_N.
\end{aligned}$$

Each device i starts from its own local input and performs a sequence of intermediate computations $s_k^{(i)}$ at discrete time instants $k = 1, \ldots, T_i$. These computations may depend on both: the previous local computations at the device and messages received from other devices. One important application of distributed algorithms is in FL where a network of devices collaboratively train a personal model for each device.

donnée A data point is any object that conveys information [23]. Data points might be students, radio signals, trees, forests, images, VAs, real numbers, or proteins. We characterize data points using two types of properties. One type of property is referred to as a caractéristique. Caractéristiques are properties of a data point that can be measured or computed in an automated fashion. A different kind of property is referred to as a étiquette. The étiquette of a data point represents some higher-level fact (or quantity of interest). In contrast to caractéristiques, determining the étiquette of a data point typically requires human experts (domain experts). Roughly speaking, apprentissage automatique aims to predict the étiquette of a data point based solely on its caractéristiques.

dérivable A real-valued function $f : \mathbb{R}^d \rightarrow \mathbb{R}$ is differentiable if it can, at any point, be approximated locally by a linear function. The local linear approximation at the point $\mathbf{x}$ is determined by the gradient $\nabla f(\mathbf{x})$ [2].

edge weight Each edge $\{i, i'\}$ of an FL network is assigned a non-negative edge weight $A_{i,i'} \geq 0$. A zero edge weight $A_{i,i'} = 0$ indicates the absence of an edge between nodes $i, i' \in \mathcal{V}$.

effective dimension The effective dimension $d_{\text{eff}}\left(\mathcal{H}\right)$ of an infinite hypothesis space $\mathcal{H}$ is a measure of its size. Loosely speaking, the effective dimension is equal to the effective number of independent tunable model parameters. These parameters might be the coefficients used in a linear map or the weights and bias terms of an ANN.

eigenvalue We refer to a number $\lambda \in \mathbb{R}$ as an eigenvalue of a square matrix $\mathbf{A} \in \mathbb{R}^{d \times d}$ if there is a non-zero vector $\mathbf{x} \in \mathbb{R}^d \setminus \{\mathbf{0}\}$ such that $\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$.

eigenvalue decomposition (EVD) The eigenvalue decomposition for a square matrix $\mathbf{A} \in \mathbb{R}^{d \times d}$ is a factorization of the form

$$\mathbf{A} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^{-1}.$$

The columns of the matrix $\mathbf{V} = \big(\mathbf{v}^{(1)}, \ldots, \mathbf{v}^{(d)}\big)$ are the eigenvectors of the matrix $\mathbf{V}$. The diagonal matrix $\mathbf{\Lambda} = \mathrm{diag}\big\{\lambda_1, \ldots, \lambda_d\big\}$ contains the eigenvalues λ_j corresponding to the eigenvectors $\mathbf{v}^{(j)}$. Note that the above decomposition exists only if the matrix $\mathbf{A}$ is diagonalizable.

eigenvector An eigenvector of a matrix $\mathbf{A} \in \mathbb{R}^{d \times d}$ is a non-zero vector $\mathbf{x} \in \mathbb{R}^d \setminus \{\mathbf{0}\}$ such that $\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$ with some eigenvalue λ.

empirical risk The empirical risk $\widehat{L}\big(h|\mathcal{D}\big)$ of a hypothèse on a jeu de données $\mathcal{D}$ is the average loss incurred by h when applied to the données in $\mathcal{D}$.

empirical risk minimization (ERM) Empirical risk minimization is the optimization problem of finding a hypothèse (out of a model) with the minimum average loss (or empirical risk) on a given jeu de données $\mathcal{D}$ (i.e., the training set). Many apprentissage automatique methods are obtained from empirical risk via specific design choices for the jeu de données, model, and loss [6, Ch. 3].

espérance Consider a numeric feature vector $\mathbf{x} \in \mathbb{R}^d$ which we interpret as the realization of an VA with a loi de probabilité $p(\mathbf{x})$. The expectation of $\mathbf{x}$ is defined as the integral $\mathbb{E}\{\mathbf{x}\} := \int \mathbf{x} p(\mathbf{x})$ [2, 36, 37]. Note that the expectation is only defined if this integral exists, i.e., if the VA is integrable.

estimation error Consider données, each with feature vector $\mathbf{x}$ and étiquette y. In some applications, we can model the relation between the feature vector and the étiquette of a donnée as $y = \bar{h}(\mathbf{x}) + \varepsilon$. Here, we use some true underlying hypothèse $\bar{h}$ and a noise term ε which summarizes any modeling or labeling errors. The estimation error incurred by an apprentissage automatique method that learns a hypothèse $\widehat{h}$, e.g., using ERM, is defined as $\widehat{h}(\mathbf{x}) - \bar{h}(\mathbf{x})$, for some feature vector. For a parametric hypothesis space, which consists of hypothèse maps determined by model parameters $\mathbf{w}$, we can define the estimation error as $\Delta\mathbf{w} = \widehat{\mathbf{w}} - \overline{\mathbf{w}}$ [34,35].

Euclidean space The Euclidean space $\mathbb{R}^d$ of dimension $d \in \mathbb{N}$ consists of vectors $\mathbf{x} = \big(x_1, \ldots, x_d\big)$, with d real-valued entries $x_1, \ldots, x_d \in \mathbb{R}$. Such an Euclidean space is equipped with a geometric structure defined by the inner product $\mathbf{x}^T\mathbf{x}' = \sum_{j=1}^{d} x_j x'_j$ between any two vectors $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^d$ [2].

expectation-maximization (EM) Consider a probabilistic model $p(\mathbf{z}; \mathbf{w})$ for the données $\mathcal{D}$ generated in some apprentissage automatique application. The maximum likelihood estimator for the model parameters $\mathbf{w}$ is obtained by maximizing $p(\mathcal{D}; \mathbf{w})$. However, the resulting optimization problem might be computationally challenging. Espérance-maximization approximates the maximum likelihood estimator by introducing a latent VA $\mathbf{z}$ such that maximizing $p(\mathcal{D}, \mathbf{z}; \mathbf{w})$ would be easier [35, 38, 39]. Since we do not observe $\mathbf{z}$, we need to estimate it from the observed jeu de données $\mathcal{D}$ using a conditional espérance. The resulting estimate $\widehat{\mathbf{z}}$ is then used to compute a new estimate $\widehat{\mathbf{w}}$ by solving $\max_{\mathbf{w}} p(\mathcal{D}, \widehat{\mathbf{z}}; \mathbf{w})$. The crux is that the conditional espérance $\widehat{\mathbf{z}}$ depends on the model parameters $\widehat{\mathbf{w}}$, which we have updated based on $\widehat{\mathbf{z}}$. Thus, we have to re-calculate $\widehat{\mathbf{z}}$, which, in turn, results in a new choice $\widehat{\mathbf{w}}$ for the model parameters. In practice, we repeat the computation of the conditional espérance (i.e., the E-step) and the update of the model parameters (i.e., the M-step) until some stopping criterion is met.

expert apprentissage automatique aims to learn a hypothèse h that accurately predicts the étiquette of a donnée based on its caractéristiques. We measure the prédiction error using some loss function. Ideally, we want to find a hypothèse that incurs minimal loss on any donnée. We can make this informal goal precise via the independent and identically distributed assumption (i.i.d. assumption) and by using the Bayes risk as the baseline for the (average) loss of a hypothèse. An alternative approach to obtaining a baseline is to use the hypothèse h' learned by an existing apprentissage automatique method. We refer to this hypothèse h' as an expert [40]. Regret minimization methods learn a hypothèse that incurs a loss comparable to the best expert [40, 41].

explainability We define the (subjective) explainability of an apprentissage automatique method as the level of simulatability [42] of the prédictions delivered by an apprentissage automatique system to a human user. Quantitative measures for the (subjective) explainability of a trained model can be constructed by comparing its prédictions with the prédictions provided by a user on a test set [42, 43]. Alternatively, we can use probabilistic models for data and measure the explainability of a trained apprentissage automatique model via the conditional (differential) entropy of its prédictions, given the user prédictions [44, 45].

explainable empirical risk minimization (EERM) Explainable ERM is an instance of SRM that adds a regularization term to the average loss in the objective function of ERM. The regularization term is chosen to favor hypothèse maps that are intrinsically explainable for a specific user. This user is characterized by their prédictions provided for the données in a training set [43].

explainable machine learning (explainable ML) Explainable apprentissage automatique methods aim at complementing each prédiction with an explanation of how the prédiction has been obtained. The construction of an explicit explanation might not be necessary if the apprentissage automatique method uses a sufficiently simple (or interpretable) model [46].

explanation One approach to make apprentissage automatique methods transparent is to provide an explanation along with the prédiction delivered by an apprentissage automatique method. Explanations can take on many different forms. An explanation could be some natural text or some quantitative measure for the importance of individual caractéristiques of a donnée [47]. We can also use visual forms of explanations, such as intensity plots for image classification [48].

feature learning Consider an apprentissage automatique application with données characterized by raw caractéristiques $\mathbf{x} \in \mathcal{X}$. Caractéristique learning refers to the task of learning a map

$$\mathbf{\Phi} : \mathcal{X} \rightarrow \mathcal{X}' : \mathbf{x} \mapsto \mathbf{x}',$$

that reads in raw caractéristiques $\mathbf{x} \in \mathcal{X}$ of a donnée and delivers new caractéristiques $\mathbf{x}' \in \mathcal{X}'$ from a new feature space $\mathcal{X}'$. Different caractéristique learning methods are obtained for different design choices of $\mathcal{X}, \mathcal{X}'$, for a hypothesis space $\mathcal{H}$ of potential maps $\mathbf{\Phi}$, and for a quantitative measure of the usefulness of a specific $\mathbf{\Phi} \in \mathcal{H}$. For example, principal component analysis (PCA) uses $\mathcal{X} := \mathbb{R}^d$, $\mathcal{X}' := \mathbb{R}^{d'}$ with $d' < d$, and a hypothesis space

$$\mathcal{H} := \big\{ \mathbf{\Phi} : \mathbb{R}^d \rightarrow \mathbb{R}^{d'} : \mathbf{x}' := \mathbf{F}\mathbf{x} \text{ with some } \mathbf{F} \in \mathbb{R}^{d' \times d} \big\}.$$

PCA measures the usefulness of a specific map $\mathbf{\Phi}(\mathbf{x}) = \mathbf{F}\mathbf{x}$ by the minimum linear reconstruction error incurred on a jeu de données,

$$\min_{\mathbf{G} \in \mathbb{R}^{d \times d'}} \sum_{r=1}^{m} \left\| \mathbf{G}\mathbf{F}\mathbf{x}^{(r)} - \mathbf{x}^{(r)} \right\|_2^2 .$$

feature map Caractéristique map refers to a map that transforms the original caractéristiques of a donnée into new caractéristiques. The so-obtained new caractéristiques might be preferable over the original caractéristiques for several reasons. For example, the arrangement of données might become simpler (or more linear) in the new feature space, allowing the use of linear models in the new caractéristiques. This idea is a main driver for the development of kernel methods [52]. Moreover, the hidden layers of a deep net can be interpreted as a trainable caractéristique map followed by a linear model in the form of the output layer. Another reason for learning a caractéristique map could be that learning a small number of new caractéristiques helps to avoid overfitting and ensures interpretability [53]. The special case of a caractéristique map delivering two numeric caractéristiques is particularly useful for data visualization. Indeed, we can depict données in a scatterplot by using two caractéristiques as the coordinates of a donnée.

feature matrix Consider a jeu de données $\mathcal{D}$ with m données with feature vectors $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)} \in \mathbb{R}^d$. It is convenient to collect the individual feature vectors into a caractéristique matrix $\mathbf{X} := \left(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)}\right)^T$ of size $m \times d$.

feature space The caractéristique space of a given apprentissage automatique application or method is constituted by all potential values that the feature vector of a donnée can take on. A widely used choice for the caractéristique space is the Euclidean space $\mathbb{R}^d$, with the dimension d being the number of individual caractéristiques of a donnée.

feature vector Caractéristique vector refers to a vector $\mathbf{x} = \big(x_1, \ldots, x_d\big)^T$ whose entries are individual caractéristiques $x_1, \ldots, x_d$. Many apprentissage automatique methods use caractéristique vectors that belong to some finite-dimensional Euclidean space $\mathbb{R}^d$. For some apprentissage automatique methods, however, it can be more convenient to work with caractéristique vectors that belong to an infinite-dimensional vector space (e.g., see kernel method).

federated averaging (FedAvg) FedAvg refers to an iterative FL algorithm that alternates between separately training local models and combining the updated local model parameters. The training of local models is implemented via several SGD steps [54].

federated learning (FL) FL is an umbrella term for apprentissage automatique methods that train models in a collaborative fashion using decentralized data and computation.

federated learning network (FL network) A federated network is an undirected weighted graph whose nodes represent data generators that aim to train a local (or personalized) model. Each node in a federated network represents some device capable of collecting a local dataset and, in turn, train a local model. FL methods learn a local hypothèse $h^{(i)}$, for each node $i \in \mathcal{V}$, such that it incurs small loss on the local datasets.

FedProx FedProx refers to an iterative FL algorithm that alternates between separately training local models and combining the updated local model parameters. In contrast to FedAvg, which uses SGD to train local models, FedProx uses a proximal operator for the training [55].

Finnish Meteorological Institute (FMI) The FMI is a government agency responsible for gathering and reporting weather data in Finland.

flow-based clustering Flow-based clustering groups the nodes of an undirected graph by applying k-means clustering to node-wise feature vectors. These feature vectors are built from network flows between carefully selected sources and destination nodes [56].

Gaussian mixture model (GMM) A GMM is a particular type of probabilistic model for a numeric vector $\mathbf{x}$ (e.g., the caractéristiques of a donnée). Within a GMM, the vector $\mathbf{x}$ is drawn from a randomly selected multivariate normal distribution $p^{(c)} = \mathcal{N}\left(\boldsymbol{\mu}^{(c)}, \mathbf{C}^{(c)}\right)$ with $c = I$. The index $I \in \{1, \ldots, k\}$ is an VA with probabilities $p(I = c) = p_c$. Note that a GMM is parametrized by the probability p_c, the mean vector $\boldsymbol{\mu}^{(c)}$, and the covariance matrix $\boldsymbol{\Sigma}^{(\mathbf{c})}$ for each $c = 1, \ldots, k$. GMMs are widely used for clustering, density estimation, and as a generative model.

Gaussian random variable (Gaussian RV) A standard Gaussian VA is a real-valued VA x with probability density function (pdf) [5, 15, 57]

$$p(x) = \frac{1}{\sqrt{2\pi}} \exp^{-x^2/2}.$$

Given a standard Gaussian VA x, we can construct a general Gaussian VA x' with mean μ and variance σ^2 via $x' := \sigma(x + \mu)$. The loi de probabilité of a Gaussian VA is referred to as normal distribution, denoted $\mathcal{N}(\mu, \sigma)$.

A Gaussian random vector $\mathbf{x} \in \mathbb{R}^d$ with covariance matrix $\mathbf{C}$ and mean $\boldsymbol{\mu}$ can be constructed via $\mathbf{x} := \mathbf{A}\big(\mathbf{z} + \boldsymbol{\mu}\big)$. Here, $\mathbf{A}$ is any matrix that satisfies $\mathbf{A}\mathbf{A}^T = \mathbf{C}$ and $\mathbf{z} := \big(z_1, \ldots, z_d\big)^T$ is a vector whose entries are i.i.d. standard Gaussian VAs $z_1, \ldots, z_d$. Gaussian random processes generalize Gaussian random vectors by applying linear transformations to infinite sequences of standard Gaussian VAs [58].

Gaussian VAs are widely used probabilistic models for the statistical analysis of apprentissage automatique methods. Their significance arises partly from the central limit theorem, which states that the average of an increasing number of independent VAs (not necessarily Gaussian themselves) converges to a Gaussian VA [59].

general data protection regulation (GDPR) The GDPR was enacted by the European Union (EU), effective from May 25, 2018 [21]. It safeguards the privacy and data rights of individuals in the EU. The GDPR has significant implications for how data is collected, stored, and used in apprentissage automatique applications. Key provisions include the following:

- Data minimization principle: apprentissage automatique systems should only use the necessary amount of personal data for their purpose.

- Transparency and explainability: apprentissage automatique systems should enable their users to understand how the systems make decisions that impact the users.
- Data subject rights: Users should get an opportunity to access, rectify, and delete their personal data, as well as to object to automated decision-making and profiling.
- Accountability: Organizations must ensure robust data security and demonstrate compliance through documentation and regular audits.

generalization Many current apprentissage automatique (and AI) systems are based on ERM: At their core, they train a model (i.e., learn a hypothèse $\hat{h} \in \mathcal{H}$) by minimizing the average loss (or empirical risk) on some données $\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(m)}$, which serve as a training set $\mathcal{D}^{(\mathrm{train})}$. Generalization refers to an apprentissage automatique method's ability to perform well outside the training set. Any mathematical theory of generalization needs some mathematical concept for the "outside the training set." For example, statistical learning theory uses a probabilistic model such as the i.i.d. assumption for data generation: the données in the training set are i.i.d. realizations of some underlying loi de probabilité $p(\mathbf{z})$. A probabilistic model allows us to explore the outside of the training set by drawing additional i.i.d. realizations from $p(\mathbf{z})$. Moreover, using the i.i.d. assumption allows us to define the risk of a trained model $\hat{h} \in \mathcal{H}$ as the expected loss $\bar{L}(\hat{h})$. What is more, we can use concentration bounds or convergence results for sequences of i.i.d. VAs to bound the deviation between the empirical risk $\widehat{L}(\hat{h}|\mathcal{D}^{(\mathrm{train})})$ of a trained model and its risk [60]. It is possible to study generalization also without using probabilistic models. For example, we could use (deterministic) perturbations of the données in the training set to study its outside. In general, we would like the trained model to be robust, i.e., its prédictions should not change too much for small perturbations of a donnée. Consider a trained model for detecting an object in a smartphone snapshot. The detection result should not change if we mask a small number of randomly chosen pixels in the image [61].

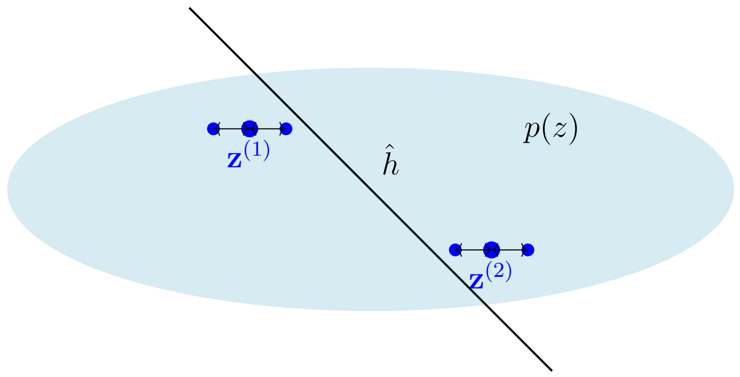


Figure 5: Two données $\mathbf{z}^{(1)}, \mathbf{z}^{(2)}$ that are used as a training set to learn a hypothèse $\hat{h}$ via ERM. We can evaluate $\hat{h}$ outside $\mathcal{D}^{(\text{train})}$ either by an i.i.d. assumption with some underlying loi de probabilité $p(\mathbf{z})$ or by perturbing the données.

generalized total variation (GTV) GTV is a measure of the variation of trained local models $h^{(i)}$ (or their model parameters $\mathbf{w}^{(i)}$) assigned to the nodes $i = 1, \ldots, n$ of an undirected weighted graph $\mathcal{G}$ with edges $\mathcal{E}$. Given a measure $d^{(h,h')}$ for the discrepancy between hypothèse maps h, h', the GTV is

$$\sum_{\{i,i'\} \in \mathcal{E}} A_{i,i'} d^{(h^{(i)},h^{(i')})}.$$

Here, $A_{i,i'} > 0$ denotes the weight of the undirected edge $\{i, i'\} \in \mathcal{E}$.

generalized total variation minimization (GTVMin) GTV minimization is an instance of regularized empirical risk minimization (RERM) using the GTV of local model parameters as a regularizer [62].

gradient For a real-valued function $f : \mathbb{R}^d \to \mathbb{R} : \mathbf{w} \mapsto f(\mathbf{w})$, a vector $\mathbf{g}$ such that $\lim_{\mathbf{w} \to \mathbf{w}'} \frac{f(\mathbf{w}) - \left(f(\mathbf{w}') + \mathbf{g}^T(\mathbf{w} - \mathbf{w}')\right)}{\|\mathbf{w} - \mathbf{w}'\|} = 0$ is referred to as the gradient of f at $\mathbf{w}'$. If such a vector exists, it is denoted $\nabla f(\mathbf{w}')$ or $\nabla f(\mathbf{w})\big|_{\mathbf{w}'}$ [2].

gradient descent (GD) Gradient descent is an iterative method for finding the minimum of a dérivable function $f(\mathbf{w})$ of a vector-valued argument $\mathbf{w} \in \mathbb{R}^d$. Consider a current guess or approximation $\mathbf{w}^{(k)}$ for the minimum of the function $f(\mathbf{w})$. We would like to find a new (better) vector $\mathbf{w}^{(k+1)}$ that has a smaller objective value $f(\mathbf{w}^{(k+1)}) < f\big(\mathbf{w}^{(k)}\big)$ than the current guess $\mathbf{w}^{(k)}$. We can achieve this typically by using a gradient step

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} - \eta \nabla f(\mathbf{w}^{(k)}) \tag{2}$$

with a sufficiently small step size $\eta > 0$. Figure 6 illustrates the effect of a single gradient descent step (2).

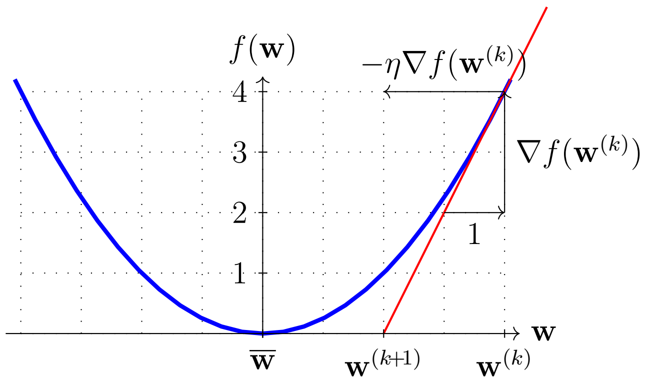


Figure 6: A single gradient step (2) towards the minimizer $\overline{\mathbf{w}}$ of $f(\mathbf{w})$.

gradient step Given a dérivable real-valued function $f(\cdot) : \mathbb{R}^d \rightarrow \mathbb{R}$ and a vector $\mathbf{w} \in \mathbb{R}^d$, the gradient step updates $\mathbf{w}$ by adding the scaled negative gradient $\nabla f(\mathbf{w})$ to obtain the new vector (see Figure 7)

$$\widehat{\mathbf{w}} := \mathbf{w} - \eta \nabla f(\mathbf{w}). \tag{3}$$

Mathematically, the gradient step is a (typically non-linear) operator $\mathcal{T}^{(f,\eta)}$ that is parametrized by the function f and the step size η.

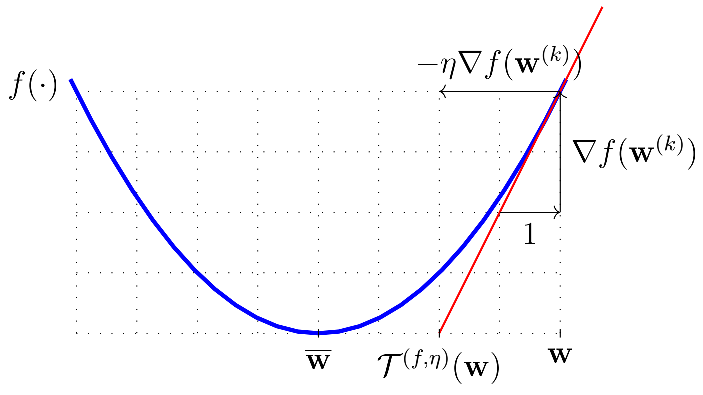

$f(\cdot)$
$-\eta\nabla f(\mathbf{w}^{(k)})$
$\nabla f(\mathbf{w}^{(k)})$
1
$\overline{\mathbf{w}}$
$\mathcal{T}^{(f,\eta)}(\mathbf{w})$
$\mathbf{w}$

Figure 7: The basic gradient step (3) maps a given vector $\mathbf{w}$ to the updated vector $\mathbf{w}'$. It defines an operator $\mathcal{T}^{(f,\eta)}(\cdot) : \mathbb{R}^d \rightarrow \mathbb{R}^d : \mathbf{w} \mapsto \widehat{\mathbf{w}}$.

Note that the gradient step (3) optimizes locally - in a neighborhood whose size is determined by the step size η - a linear approximation to the function $f(\cdot)$. A natural generalization of (3) is to locally optimize the function itself - instead of its linear approximation - such that

$$\widehat{\mathbf{w}} = \underset{\mathbf{w}' \in \mathbb{R}^d}{\operatorname{argmin}}\, f(\mathbf{w}') + (1/\eta) \left\| \mathbf{w} - \mathbf{w}' \right\|_2^2. \tag{4}$$

We intentionally use the same symbol η for the parameter in (4) as we used for the step size in (3). The larger the η we choose in (4), the more progress the update will make towards reducing the function value $f(\widehat{\mathbf{w}})$. Note that, much like the gradient step (3), also the update (4) defines a (typically non-linear) operator that is parametrized by the function $f(\cdot)$ and the parameter η. For a convex function $f(\cdot)$, this operator is known as the proximal operator of $f(\cdot)$ [63].

gradient-based methods Gradient-based methods are iterative techniques for finding the minimum (or maximum) of a dérivable objective function of the model parameters. These methods construct a sequence of approximations to an optimal choice for model parameters that results in a minimum (or maximum) value of the objective function. As their name indicates, gradient-based methods use the gradients of the objective function evaluated during previous iterations to construct new, (hopefully) improved model parameters. One important example of a gradient-based method is GD.

graph A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a pair that consists of a node set $\mathcal{V}$ and an edge set $\mathcal{E}$. In its most general form, a graph is specified by a map that assigns each edge $e \in \mathcal{E}$ a pair of nodes [64]. One important family of graphs is simple undirected graphs. A simple undirected graph is obtained by identifying each edge $e \in \mathcal{E}$ with two different nodes $\{i, i'\}$. Weighted graphs also specify numeric weights A_e for each edge $e \in \mathcal{E}$.

graph clustering Graph clustering aims at clustering données that are represented as the nodes of a graph $\mathcal{G}$. The edges of $\mathcal{G}$ represent pairwise similarities between données. Sometimes we can quantify the extend of these similarities by an edge weight [56, 65].

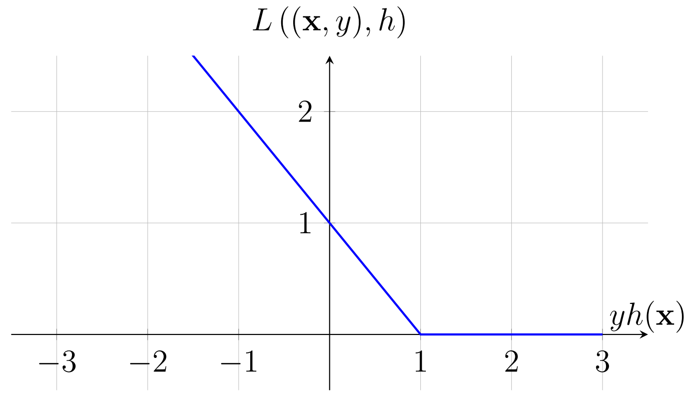
hard clustering Hard clustering refers to the task of partitioning a given set of données into (a few) non-overlapping clusters. The most widely used hard clustering method is k-means.

high-dimensional regime The high-dimensional regime of ERM is characterized by the effective dimension of the model being larger than the sample size, i.e., the number of (labeled) données in the training set. For example, linear regression methods operate in the high-dimensional regime whenever the number d of caractéristiques used to characterize données exceeds the number of données in the training set. Another example of apprentissage automatique methods that operate in the high-dimensional regime is large ANNs, which have far more tunable weights (and bias terms) than the total number of données in the training set. High-dimensional statistics is a recent main thread of probability theory that studies the behavior of apprentissage automatique methods in the high-dimensional regime [66, 67].

Hilbert space A Hilbert space is a linear vector space equipped with an inner product between pairs of vectors. One important example of a Hilbert space is the Euclidean space $\mathbb{R}^d$, for some dimension d, which consists of Euclidean vectors $\mathbf{u} = \big(u_1, \ldots, u_d\big)^T$ along with the inner product $\mathbf{u}^T\mathbf{v}$.

hinge loss Consider a donnée characterized by a feature vector $\mathbf{x} \in \mathbb{R}^d$ and a binary étiquette $y \in \{-1, 1\}$. The hinge loss incurred by a real-valued hypothèse map $h(\mathbf{x})$ is defined as

$$L\left((\mathbf{x}, y), h\right) := \max\{0, 1 - yh(\mathbf{x})\}. \tag{5}$$



A regularized variant of the hinge loss is used by the support vector machine (SVM) [68].

histogram Consider a jeu de données $\mathcal{D}$ that consists of m données $\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(m)}$, each of them belonging to some cell $[-U, U] \times \ldots \times [-U, U] \subseteq \mathbb{R}^d$ with side length U. We partition this cell evenly into smaller elementary cells with side length Δ. The histogram of $\mathcal{D}$ assigns each elementary cell to the corresponding fraction of données in $\mathcal{D}$ that fall into this elementary cell.

horizontal federated learning (horizontal FL) Horizontal FL uses local datasets constituted by different données but uses the same caractéristiques to characterize them [69]. For example, weather forecasting uses a network of spatially distributed weather (observation) stations. Each weather station measures the same quantities, such as daily temperature, air pressure, and precipitation. However, different weather stations measure the characteristics or caractéristiques of different spatiotemporal regions. Each spatiotemporal region represents an individual donnée, each characterized by the same caractéristiques (e.g., daily temperature or air pressure).

Huber loss The Huber loss unifies the squared error loss and the absolute error loss.

Huber regression Huber regression refers to ERM-based methods that use the Huber loss as a measure of the prédiction error. Two important special cases of Huber regression are least absolute deviation regression and linear regression. Tuning the threshold parameter of the Huber loss allows the user to trade the robustness of the absolute error loss against the computational benefits of the smooth squared error loss.

hypothesis space Every practical apprentissage automatique method uses a hypothèse space (or model) $\mathcal{H}$. The hypothèse space of an apprentissage automatique method is a subset of all possible maps from the feature space to the label space. The design choice of the hypothèse space should take into account available computational resources and statistical aspects. If the computational infrastructure allows for efficient matrix operations, and there is an (approximately) linear relation between a set of caractéristiques and a étiquette, a useful choice for the hypothèse space might be the linear model.

hypothèse A hypothesis refers to a map (or function) $h : \mathcal{X} \to \mathcal{Y}$ from the feature space $\mathcal{X}$ to the label space $\mathcal{Y}$. Given a donnée with caractéristiques $\mathbf{x}$, we use a hypothesis map h to estimate (or approximate) the étiquette y using the prédiction $\hat{y} = h(\mathbf{x})$. Apprentissage automatique is all about learning (or finding) a hypothesis map h such that $y \approx h(\mathbf{x})$ for any donnée (having caractéristiques $\mathbf{x}$ and étiquette y).

independent and identically distributed (i.i.d.) It can be useful to interpret données $\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(m)}$ as realizations of i.i.d. VAs with a common loi de probabilité. If these VAs are continuous-valued, their joint pdf is $p\big(\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(m)}\big) = \prod_{r=1}^{m} p\big(\mathbf{z}^{(r)}\big)$, with $p(\mathbf{z})$ being the common marginal pdf of the underlying VAs.

independent and identically distributed assumption (i.i.d. assumption) The i.i.d. assumption interprets données of a jeu de données as the realizations of i.i.d. VAs.

interpretability An apprentissage automatique method is interpretable for a specific user if they can well anticipate the prédictions delivered by the method. The notion of interpretability can be made precise using quantitative measures of the uncertainty about the prédictions [44].

jeu de données A dataset refers to a collection of données. These données carry information about some quantity of interest (or étiquette) within a apprentissage automatique application. apprentissage automatique methods use datasets for model training (e.g., via ERM) and model validation. Note that our notion of a dataset is very flexible as it allows for very different types of données. Indeed, données can be concrete physical objects (such as humans or animals) or abstract objects (such as numbers). As a case in point, Figure 8 depicts a dataset that consists of cows as données.

Figure 8: "Cows in the Swiss Alps" by User:Huhu Uet is licensed under [CC BY-SA 4.0](https://creativecommons.org/licenses/by-sa/4.0/)

Quite often, an apprentissage automatique engineer does not have direct access to a dataset. Indeed, accessing the dataset in Figure would require to visit the cow herd in the Alps. Instead, we need to use an approximation (or representation) of the dataset which is more convenient to work with. Different mathematical models have been developed for the representation (or approximation) of datasets [26], [27], [28], [29]. One of the most widely adopted data model is the relational model, which organizes data as a table (or relation) [20], [26]. A table consists of rows and columns:

- Each row of the table represents a single donnée.
- Each column of the table corresponds to a specific attribute of the donnée. apprentissage automatique methods can use attributes as caractéristiques and étiquettes of the donnée.

For example, Table 1 shows a representation of the dataset in Figure 8. In the relational model, the order of rows is irrelevant, and each attribute (i.e., column) must be precisely defined with a domain, which specifies the set of possible values. In apprentissage automatique applications, these attribute domains become the feature space and the label space.

Name	**Weight**	**Age**	**Height**	**Stomach temp**
Zenzi	100	4	100	25
Berta	140	3	130	23
Resi	120	4	120	31

Table 1: A relation (or table) that represents the dataset in Figure .

While the relational model is useful for the study of many apprentissage automatique applications, it may be insufficient regarding the requirements for trustworthy artificial intelligence (trustworthy AI). Modern approaches like datasheets for datasets provide more comprehensive documentation, including details about the dataset's collection process, intended use, and other contextual information [30].

kernel Consider données characterized by a feature vector $\mathbf{x} \in \mathcal{X}$ with a generic feature space $\mathcal{X}$. A (real-valued) kernel $K : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ assigns each pair of feature vectors $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$ a real number $K\big(\mathbf{x}, \mathbf{x}'\big)$. The value $K\big(\mathbf{x}, \mathbf{x}'\big)$ is often interpreted as a measure for the similarity between $\mathbf{x}$ and $\mathbf{x}'$. Kernel methods use a kernel to transform the feature vector $\mathbf{x}$ to a new feature vector $\mathbf{z} = K\big(\mathbf{x}, \cdot\big)$. This new feature vector belongs to a linear feature space $\mathcal{X}'$ which is (in general) different from the original feature space $\mathcal{X}$. The feature space $\mathcal{X}'$ has a specific mathematical structure, i.e., it is a reproducing kernel Hilbert space [52, 68].

kernel method A kernel method is an apprentissage automatique method that uses a kernel K to map the original (raw) feature vector $\mathbf{x}$ of a donnée to a new (transformed) feature vector $\mathbf{z} = K(\mathbf{x}, \cdot)$ [52, 68]. The motivation for transforming the feature vectors is that, by using a suitable kernel, the données have a "more pleasant" geometry in the transformed feature space. For example, in a binary classification problem, using transformed feature vectors $\mathbf{z}$ might allow us to use linear models, even if the données are not linearly separable in the original feature space (see Figure 9).

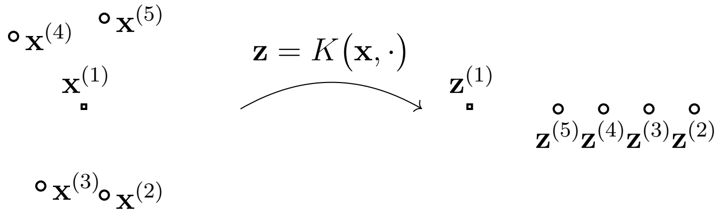


Figure 9: Five données characterized by feature vectors $\mathbf{x}^{(r)}$ and étiquettes $y^{(r)} \in \{\circ, \square\}$, for $r = 1, \ldots, 5$. With these feature vectors, there is no way to separate the two classes by a straight line (representing the decision boundary of a linear classifier). In contrast, the transformed feature vectors $\mathbf{z}^{(r)} = K(\mathbf{x}^{(r)}, \cdot)$ allow us to separate the données using a linear classifier.

Kullback-Leibler divergence (KL divergence) The KL divergence is a quantitative measure of how much one loi de probabilité is different from another loi de probabilité [23].

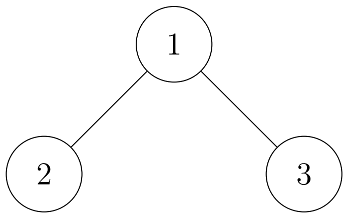
label space Consider an apprentissage automatique application that involves données characterized by caractéristiques and étiquettes. The étiquette space is constituted by all potential values that the étiquette of a donnée can take on. Regression methods, aiming at predicting numeric étiquettes, often use the étiquette space $\mathcal{Y} = \mathbb{R}$. Binary classification methods use a étiquette space that consists of two different elements, e.g., $\mathcal{Y} = \{-1, 1\}$, $\mathcal{Y} = \{0, 1\}$, or $\mathcal{Y} = \{\text{“cat image”}, \text{“no cat image”}\}$.

labeled datapoint A donnée whose étiquette is known or has been determined by some means which might require human labor.

Laplacian matrix The structure of a graph $\mathcal{G}$, with nodes $i = 1, \ldots, n$, can be analyzed using the properties of special matrices that are associated with $\mathcal{G}$. One such matrix is the graph Laplacian matrix $\mathbf{L}^{(\mathcal{G})} \in \mathbb{R}^{n \times n}$, which is defined for an undirected and weighted graph [65, 70]. It is defined element-wise as (see Figure 10)

$$L^{(\mathcal{G})}_{i,i'} := \begin{cases} -A_{i,i'} & \text{for } i \neq i', \{i, i'\} \in \mathcal{E}, \\ \sum_{i'' \neq i} A_{i,i''} & \text{for } i = i', \\ 0 & \text{else.} \end{cases} \tag{6}$$

Here, $A_{i,i'}$ denotes the edge weight of an edge $\{i, i'\} \in \mathcal{E}$.



$$\mathbf{L}^{(\mathcal{G})} = \begin{pmatrix} 2 & -1 & -1 \\ -1 & 1 & 0 \\ -1 & 0 & 1 \end{pmatrix}$$

Figure 10: Left: Some undirected graph $\mathcal{G}$ with three nodes $i = 1, 2, 3$. Right: The Laplacian matrix $\mathbf{L}^{(\mathcal{G})} \in \mathbb{R}^{3\times 3}$ of $\mathcal{G}$.

large language model (LLM) Large language models is an umbrella term for apprentissage automatique methods that process and generate human-like text. These methods typically use deep nets with billions (or even trillions) of parameters. A widely used choice for the network architecture is referred to as Transformers [71]. The training of large language models is often based on the task of predicting a few words that are intentionally removed from a large text corpus. Thus, we can construct labeled datapoints simply by selecting some words of a text as étiquettes and the remaining words as caractéristiques of données. This construction requires very little human supervision and allows for generating sufficiently large training sets for large language models.

law of large numbers The law of large numbers refers to the convergence of the average of an increasing (large) number of i.i.d. VAs to the mean of their common loi de probabilité. Different instances of the law of large numbers are obtained by using different notions of convergence [57].

learning rate Consider an iterative apprentissage automatique method for finding or learning a useful hypothèse $h \in \mathcal{H}$. Such an iterative method repeats similar computational (update) steps that adjust or modify the current hypothèse to obtain an improved hypothèse. One well-known example of such an iterative learning method is GD and its variants, SGD and projected gradient descent (projected GD). A key parameter of an iterative method is the learning rate. The learning rate controls the extent to which the current hypothèse can be modified during a single iteration. A well-known example of such a parameter is the step size used in GD [6, Ch. 5].

learning task Consider a jeu de données $\mathcal{D}$ constituted by several données, each of them characterized by caractéristiques $\mathbf{x}$. For example, the jeu de données $\mathcal{D}$ might be constituted by the images of a particular database. Sometimes it might be useful to represent a jeu de données $\mathcal{D}$, along with the choice of caractéristiques, by a loi de probabilité $p(\mathbf{x})$. A learning task associated with $\mathcal{D}$ consists of a specific choice for the étiquette of a donnée and the corresponding label space. Given a choice for the loss function and model, a learning task gives rise to an instance of ERM. Thus, we could define a learning task also via an instance of ERM, i.e., via an objective function. Note that, for the same jeu de données, we obtain different learning tasks by using different choices for the caractéristiques and étiquette of a donnée. These learning tasks are related, as they are based on the same jeu de données, and solving them jointly (via multitask learning methods) is typically preferable over solving them separately [72], [73], [74].

least absolute deviation regression Least absolute deviation regression is an instance of ERM using the absolute error loss. It is a special case of Huber regression.

least absolute shrinkage and selection operator (Lasso) The Lasso is an instance of SRM. It learns the weights $\mathbf{w}$ of a linear map $h(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$ based on a training set. Lasso is obtained from linear regression by adding the scaled ℓ_1-norme $\alpha \|\mathbf{w}\|_1$ to the average squared error loss incurred on the training set.

linear classifier Consider données characterized by numeric caractéristiques $\mathbf{x} \in \mathbb{R}^d$ and a étiquette $y \in \mathcal{Y}$ from some finite label space $\mathcal{Y}$. A linear classifier is characterized by having decision regions that are separated by hyperplanes in $\mathbb{R}^d$ [6, Ch. 2].

linear model Consider données, each characterized by a numeric feature vector $\mathbf{x} \in \mathbb{R}^d$. A linear model is a hypothesis space which consists of all linear maps,

$$\mathcal{H}^{(d)} := \left\{ h(\mathbf{x}) = \mathbf{w}^T\mathbf{x} : \mathbf{w} \in \mathbb{R}^d \right\}. \tag{7}$$

Note that (7) defines an entire family of hypothesis spaces, which is parametrized by the number d of caractéristiques that are linearly combined to form the prédiction $h(\mathbf{x})$. The design choice of d is guided by computational aspects (e.g., reducing d means less computation), statistical aspects (e.g., increasing d might reduce prédiction error), and interpretability. A linear model using few carefully chosen caractéristiques tends to be considered more interpretable [46, 53].

linear regression Linear regression aims to learn a linear hypothèse map to predict a numeric étiquette based on the numeric caractéristiques of a donnée. The quality of a linear hypothèse map is measured using the average squared error loss incurred on a set of labeled datapoints, which we refer to as the training set.

local dataset The concept of a local jeu de données is in between the concept of a donnée and a jeu de données. A local jeu de données consists of several individual données, which are characterized by caractéristiques and étiquettes. In contrast to a single jeu de données used in basic apprentissage automatique methods, a local jeu de données is also related to other local jeu de donnéess via different notions of similarity. These similarities might arise from probabilistic models or communication infrastructure and are encoded in the edges of an FL network.

Local Interpretable Model-agnostic Explanations (LIME) Consider a trained model (or learnt hypothèse) $\widehat{h} \in \mathcal{H}$, which maps the feature vector of a donnée to the prédiction $\widehat{y} = \widehat{h}$. Local Interpretable Model-agnostic Explanations (LIME) is a technique for explaining the behaviour of $\widehat{h}$, locally around a donnée with feature vector $\mathbf{x}^{(0)}$ [53]. The explanation is given in the form of a local approximation $g \in \mathcal{H}'$ of $\widehat{h}$ (see Fig.). This approximation can be obtained by an instance of ERM with carefully designed training set. In particular, the training set consists of données with feature vector $\mathbf{x}$ close to $\mathbf{x}^{(0)}$ and the (pseudo-)label $\widehat{h}(\mathbf{x})$. Note that we can use a different model $\mathcal{H}'$ for the approximation than the original model $\mathcal{H}$. For example, we can use a decision tree to approximate (locally) a deep net. Another widely-used choice for $\mathcal{H}'$ is the linear model.

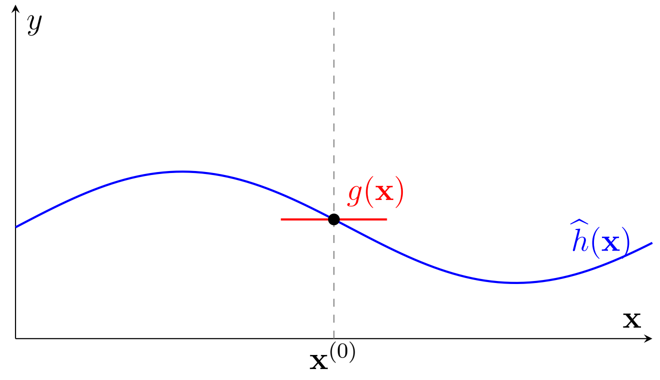


Figure 11: To explain a trained model $\widehat{h} \in \mathcal{H}$, around a given feature vector $\mathbf{x}^{(0)}$, we can use a local approximation $g \in \mathcal{H}'$.

local model Consider a collection of local datasets that are assigned to the nodes of an FL network. A local model $\mathcal{H}^{(i)}$ is a hypothesis space assigned to a node $i \in \mathcal{V}$. Different nodes might be assigned different hypothesis spaces, i.e., in general $\mathcal{H}^{(i)} \neq \mathcal{H}^{(i')}$ for different nodes $i, i' \in \mathcal{V}$.

logistic loss Consider a donnée characterized by the caractéristiques $\mathbf{x}$ and a binary étiquette $y \in \{-1, 1\}$. We use a real-valued hypothèse h to predict the étiquette y from the caractéristiques $\mathbf{x}$. The logistic loss incurred by this prédiction is defined as

$$L\left((\mathbf{x}, y), h\right) := \log(1 + \exp(-yh(\mathbf{x}))). \tag{8}$$

Carefully note that the expression (8) for the logistic loss applies only for the label space $\mathcal{Y} = \{-1, 1\}$ and when using the thresholding rule

(1).

logistic regression Logistic regression learns a linear hypothèse map (or classifier) $h(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$ to predict a binary étiquette y based on the numeric feature vector $\mathbf{x}$ of a donnée. The quality of a linear hypothèse map is measured by the average logistic loss on some labeled datapoints (i.e., the training set).

loi de probabilité To analyze apprentissage automatique methods, it can be useful to interpret données as i.i.d. realizations of an VA. The typical properties of such données are then governed by the probability distribution of this VA. The probability distribution of a binary VA $y \in \{0, 1\}$ is fully specified by the probabilities $p(y = 0)$ and $p(y = 1) = 1 - p(y = 0)$. The probability distribution of a real-valued VA $x \in \mathbb{R}$ might be specified by a pdf $p(x)$ such that $p(x \in [a, b]) \approx p(a)|b - a|$. In the most general case, a probability distribution is defined by a probability measure [15, 37].

loss apprentissage automatique methods use a loss function $L(\mathbf{z}, h)$ to measure the error incurred by applying a specific hypothèse to a specific donnée. With a slight abuse of notation, we use the term loss for both the loss function L itself and the specific value $L(\mathbf{z}, h)$, for a donnée $\mathbf{z}$ and hypothèse h.

loss function A loss function is a map

$$L : \mathcal{X} \times \mathcal{Y} \times \mathcal{H} \rightarrow \mathbb{R}_+ : \big((\mathbf{x}, y), h\big) \mapsto L\left((\mathbf{x}, y), h\right).$$

It assigns a non-negative real number (i.e., the loss) $L((\mathbf{x}, y), h)$ to a pair that consists of a donnée, with caractéristiques $\mathbf{x}$ and étiquette y, and a hypothèse $h \in \mathcal{H}$. The value $L((\mathbf{x}, y), h)$ quantifies the discrepancy between the true étiquette y and the prédiction $h(\mathbf{x})$. Lower (closer to zero) values $L((\mathbf{x}, y), h)$ indicate a smaller discrepancy between prédiction $h(\mathbf{x})$ and étiquette y. Figure 12 depicts a loss function for a given donnée, with caractéristiques $\mathbf{x}$ and étiquette y, as a function of the hypothèse $h \in \mathcal{H}$.

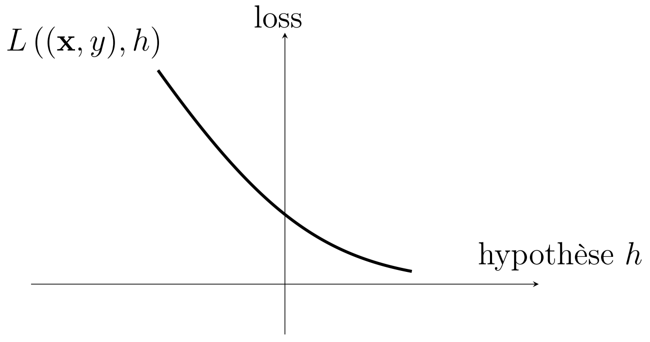


Figure 12: Some loss function $L((\mathbf{x}, y), h)$ for a fixed donnée, with feature vector $\mathbf{x}$ and étiquette y, and a varying hypothèse h. apprentissage automatique methods try to find (or learn) a hypothèse that incurs minimal loss.

maximum The maximum of a set $\mathcal{A} \subseteq \mathbb{R}$ of real numbers is the greatest element in that set, if such an element exists. A set $\mathcal{A}$ has a maximum if it is bounded above and attains its supremum (or least upper bound) [2, Sec. 1.4].

maximum likelihood Consider données $\mathcal{D} = \{\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(m)}\}$ that are interpreted as the realizations of i.i.d. VAs with a common loi de probabilité

$p(\mathbf{z};\mathbf{w})$ which depends on the model parameters $\mathbf{w} \in \mathcal{W} \subseteq \mathbb{R}^n$. Maximum likelihood methods learn model parameters $\mathbf{w}$ by maximizing the probability (density) $p(\mathcal{D};\mathbf{w}) = \prod_{r=1}^{m} p(\mathbf{z}^{(r)};\mathbf{w})$ of the observed jeu de données. Thus, the maximum likelihood estimator is a solution to the optimization problem $\max_{\mathbf{w}\in\mathcal{W}} p(\mathcal{D};\mathbf{w})$.

mean The espérance $\mathbb{E}\{\mathbf{x}\}$ of a numeric VA $\mathbf{x}$.

mean squared estimation error (MSEE) Consider an apprentissage automatique method that learns model parameters $\widehat{\mathbf{w}}$ based on some jeu de données $\mathcal{D}$. If we interpret the données in $\mathcal{D}$ as i.i.d. realizations of an VA $\mathbf{z}$, we define the estimation error $\Delta\mathbf{w} := \widehat{w} - \overline{\mathbf{w}}$. Here, $\overline{\mathbf{w}}$ denotes the true model parameters of the loi de probabilité of $\mathbf{z}$. The mean squared estimation error is defined as the espérance $\mathbb{E}\{\left\|\Delta\mathbf{w}\right\|^2\}$ of the squared Euclidean norme of the estimation error [13, 34].

minimum Given a set of real numbers, the minimum is the smallest of those numbers.

missing data Consider a jeu de données constituted by données collected via some physical device. Due to imperfections and failures, some of the caractéristique or étiquette values of données might be corrupted or simply missing. Data imputation aims at estimating these missing values [76]. We can interpret data imputation as an apprentissage automatique problem where the étiquette of a donnée is the value of the corrupted caractéristique.

model In the context of apprentissage automatique methods, the term model typically refers to the hypothesis space employed by an apprentissage automatique method [6, 60].

model parameters Model parameters are quantities that are used to select a specific hypothèse map from a model. We can think of a list of model parameters as a unique identifier for a hypothèse map, similar to how a social security number identifies a person in Finland.

model selection In apprentissage automatique, model selection refers to the process of choosing between different candidate models. In its most basic form, model selection amounts to: 1) training each candidate model; 2) computing the validation error for each trained model; and 3) choosing the model with the smallest validation error [6, Ch. 6].

multi-armed bandit A multi-armed bandit (MAB) problem models a repeated decision-making scenario in which, at each time step k, a learner must choose one out of several possible actions, often referred to as arms, from a finite set $\mathcal{A}$. Each arm $a \in \mathcal{A}$ yields a stochastic reward $r^{(a)}$ drawn from an unknown loi de probabilité with mean $\mu^{(a)}$. The learner's goal is to maximize the cumulative reward over time by strategically balancing exploration (gathering information about uncertain arms) and exploitation (selecting arms known to perform well). This balance is quantified by the notion of regret, which measures the performance gap between the learner's strategy and the optimal strategy that always selects the best arm. MAB problems form a foundational model in online learning, reinforcement learning, and sequential experimental design [77].

multi-label classification Multi-étiquette classification problems and methods use données that are characterized by several étiquettes. As an example, consider a donnée representing a picture with two étiquettes. One étiquette indicates the presence of a human in this picture and another étiquette indicates the presence of a car.

multitask learning Multitask learning aims at leveraging relations between different learning tasks. Consider two learning tasks obtained from the same jeu de données of webcam snapshots. The first task is to predict the presence of a human, while the second task is to predict the presence of a car. It might be useful to use the same deep net structure for both tasks and only allow the weights of the final output layer to be different.

multivariate normal distribution The multivariate normal distribution $\mathcal{N}(\mathbf{m}, \mathbf{C})$ is an important family of loi de probabilités for a continuous VA $\mathbf{x} \in \mathbb{R}^d$ [5, 15, 78]. This family is parametrized by the mean $\mathbf{m}$ and the covariance matrix $\mathbf{C}$ of $\mathbf{x}$. If the covariance matrix is invertible, the loi de probabilité of $\mathbf{x}$ is

$$p(\mathbf{x}) \propto \exp\bigg(- (1/2)\big(\mathbf{x} - \mathbf{m}\big)^T \mathbf{C}^{-1}\big(\mathbf{x} - \mathbf{m}\big)\bigg).$$

mutual information (MI) The MI $I(\mathbf{x}; y)$ between two VAs $\mathbf{x}$, y defined on the same probability space is given by [23]

$$I(\mathbf{x}; y) := \mathbb{E}\left\{ \log \frac{p(\mathbf{x}, y)}{p(\mathbf{x})p(y)} \right\}.$$

It is a measure of how well we can estimate y based solely on $\mathbf{x}$. A large value of $I(\mathbf{x}; y)$ indicates that y can be well predicted solely from

$\mathbf{x}$. This prédiction could be obtained by a hypothèse learned by an ERM-based apprentissage automatique method.

nearest neighbor (NN) NN methods learn a hypothèse $h : \mathcal{X} \rightarrow \mathcal{Y}$ whose function value $h(\mathbf{x})$ is solely determined by the nearest neighbors within a given jeu de données. Different methods use different metrics for determining the nearest neighbors. If données are characterized by numeric feature vectors, we can use their Euclidean distances as the metric.

neighborhood The neighborhood of a node $i \in \mathcal{V}$ is the subset of nodes constituted by the neighbors of i.

neighbors The neighbors of a node $i \in \mathcal{V}$ within an FL network are those nodes $i' \in \mathcal{V} \setminus \{i\}$ that are connected (via an edge) to node i.

networked data Networked data consists of local datasets that are related by some notion of pairwise similarity. We can represent networked data using a graph whose nodes carry local datasets and edges encode pairwise similarities. One example of networked data arises in FL applications where local datasets are generated by spatially distributed devices.

networked exponential families (nExpFam) A collection of exponential families, each of them assigned to a node of an FL network. The model parameters are coupled via the network structure by requiring them to have a small GTV [79].

networked federated learning (NFL) Networked FL refers to methods that learn personalized models in a distributed fashion. These methods learn from local datasets that are related by an intrinsic network structure.

networked model A networked model over an FL network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ assigns a local model (i.e., a hypothesis space) to each node $i \in \mathcal{V}$ of the FL network $\mathcal{G}$.

node degree The degree $d^{(i)}$ of a node $i \in \mathcal{V}$ in an undirected graph is the number of its neighbors, i.e., $d^{(i)} := \left|\mathcal{N}^{(i)}\right|$.

non-smooth We refer to a function as non-smooth if it is not smooth [80].

norme A norm is a function that maps each (vector) element of a linear vector space to a non-negative real number. This function must be homogeneous and definite, and it must satisfy the triangle inequality [81].

objective function An objective function is a map that assigns each value of an optimization variable, such as the model parameters $\mathbf{w}$ of a hypothèse $h^{(\mathbf{w})}$, to an objective value $f(\mathbf{w})$. The objective value $f(\mathbf{w})$ could be the risk or the empirical risk of a hypothèse $h^{(\mathbf{w})}$.

online algorithm An online algorithm processes input data incrementally, receiving data items sequentially and making decisions or producing outputs (or decisions) immediately without having access to the entire input in advance [40, 41]. Unlike an offline algorithm, which has the entire input available from the start, an online algorithm must handle uncertainty about future inputs and cannot revise past decisions. Similar to an offline algorithm, an online algorithm can be modeled formally as a collection of possible executions. However, the execution sequence for an online algorithm has a distinct structure:

$$\text{init}, s_1, \text{out}_1, \text{in}_2, s_2, \text{out}_2, \ldots, \text{in}_T, s_T, \text{out}_T.$$

Each execution begins from an initial state (init) and proceeds through alternating computational steps, outputs (or decisions), and inputs. Specifically, at step k, the algorithm performs a computational step s_k, generates an output out_k, and then subsequently receives the next input in_{k+1}. A notable example of an online algorithm in apprentissage automatique is online gradient descent (online GD) (online gradient descent), which incrementally updates model parameters as new données arrive.

online gradient descent (online GD) Consider an apprentissage automatique method that learns model parameters $\mathbf{w}$ from some parameter space $\mathcal{W} \subseteq \mathbb{R}^d$. The learning process uses données $\mathbf{z}^{(t)}$ that arrive at consecutive time-instants $t = 1, 2, \ldots$. Let us interpret the données $\mathbf{z}^{(t)}$ as i.i.d. copies of an VA $\mathbf{z}$. The risk $\mathbb{E}\{L(\mathbf{z}, \mathbf{w})\}$ of a hypothèse $h^{(\mathbf{w})}$ can then (under mild conditions) be obtained as the limit $\lim_{T \to \infty} (1/T) \sum_{t=1}^{T} L\left(\mathbf{z}^{(t)}, \mathbf{w}\right)$. We might use this limit as the objective function for learning the model parameters $\mathbf{w}$. Unfortunately, this limit can only be evaluated if we wait infinitely long in order to collect all données. Some apprentissage automatique applications require methods that learn online: as soon as a new donnée $\mathbf{z}^{(t)}$ arrives at time t, we update the current model parameters $\mathbf{w}^{(t)}$. Note that the new donnée $\mathbf{z}^{(t)}$ contributes the component $L\left(\mathbf{z}^{(t)}, \mathbf{w}\right)$ to the risk. As its name suggests, online GD updates $\mathbf{w}^{(t)}$ via a (projected) gradient step

$$\mathbf{w}^{(t+1)} := P_{\mathcal{W}}\big(\mathbf{w}^{(t)} - \eta_t \nabla_{\mathbf{w}} L\left(\mathbf{z}^{(t)}, \mathbf{w}\right)\big). \tag{9}$$

Note that (9) is a gradient step for the current component $L\left(\mathbf{z}^{(t)}, \cdot\right)$ of the risk. The update (9) ignores all the previous components $L\left(\mathbf{z}^{(t')}, \cdot\right)$, for $t' < t$. It might therefore happen that, compared to $\mathbf{w}^{(t)}$, the updated model parameters $\mathbf{w}^{(t+1)}$ increase the retrospective average loss $\sum_{t'=1}^{t-1} L\left(\mathbf{z}^{(t')}, \cdot\right)$. However, for a suitably chosen learning rate η_t, online GD can be shown to be optimal in practically relevant settings. By optimal, we mean that the model parameters $\mathbf{w}^{(T+1)}$ delivered by online GD after observing T données $\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(T)}$ are at least as good as those delivered by any other learning method [41, 82].

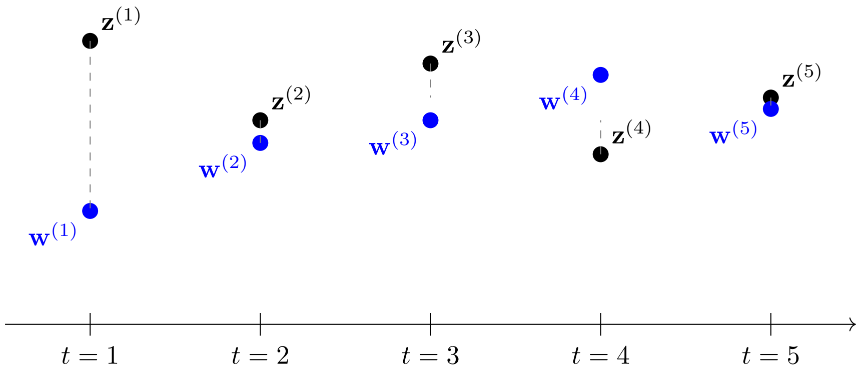


Figure 13: An instance of online GD that updates the model parameters $\mathbf{w}^{(t)}$ using the donnée $\mathbf{z}^{(t)} = x^{(t)}$ arriving at time t. This instance uses the squared error loss $L\left(\mathbf{z}^{(t)}, w\right) = (x^{(t)} - w)^2$.

optimism in the face of uncertainty apprentissage automatique methods learn model parameters $\mathbf{w}$ according to some performance criterion $\bar{f}(\mathbf{w})$. However, they usually cannot access $\bar{f}(\mathbf{w})$ directly but rely on an estimate (or approximation) of $f(\mathbf{w})$. As a case in point, ERM-based methods use the average loss on a given jeu de données (i.e., the training set) as an estimate for the risk of a hypothèse. Using a probabilistic model, one can construct a confidence interval $\left[l^{(\mathbf{w})}, u^{(\mathbf{w})}\right]$ for each choice $\mathbf{w}$ for the model parameters. One simple construction is $l^{(\mathbf{w})} := f(\mathbf{w}) - \sigma/2$, $u^{(\mathbf{w})} := f(\mathbf{w}) + \sigma/2$, with σ being a measure of the (expected) deviation of $f(\mathbf{w})$ from $\bar{f}(\mathbf{w})$. We can also use other constructions for this interval as long as they ensure that $\bar{f}(\mathbf{w}) \in \left[l^{(\mathbf{w})}, u^{(\mathbf{w})}\right]$ with a sufficiently high probability. As optimists, we choose the model parameters according to the most favorable - yet still plausible - value

$\tilde{f}(\mathbf{w}) := l^{(\mathbf{w})}$ of the performance criterion. Two examples of apprentissage automatique methods that use such an optimistic construction of an objective function are SRM [60, Ch. 11] and upper confidence bound (UCB) methods for sequential decision making [77, Sec. 2.2].

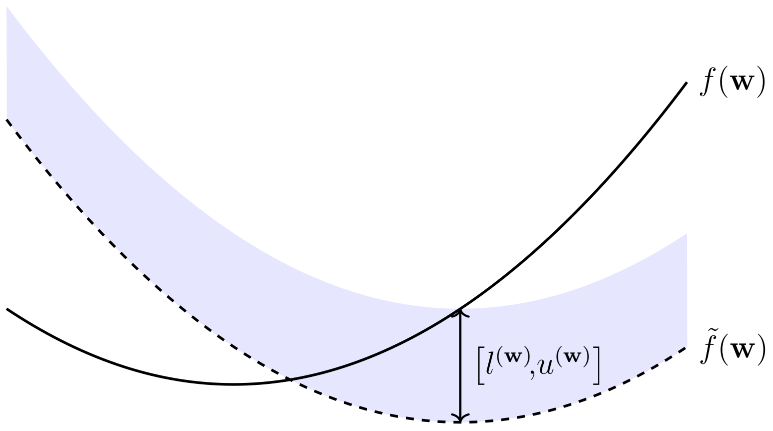


Figure 14: apprentissage automatique methods learn model parameters $\mathbf{w}$ by using some estimate of $f(\mathbf{w})$ for the ultimate performance criterion $\bar{f}(\mathbf{w})$. Using a probabilistic model, one can use $f(\mathbf{w})$ to construct confidence intervals $\left[l^{(\mathbf{w})}, u^{(\mathbf{w})}\right]$ which contain $\bar{f}(\mathbf{w})$ with high probability. The best plausible performance measure for a specific choice $\mathbf{w}$ of model parameters is $\tilde{f}(\mathbf{w}) := l^{(\mathbf{w})}$.

outlier Many apprentissage automatique methods are motivated by the i.i.d. assumption, which interprets données as realizations of i.i.d. VAs with a common loi de probabilité. The i.i.d. assumption is useful for applications where the statistical properties of the data generation process are stationary (or time-invariant) [83]. However, in some applications the data consists of a majority of regular données that conform with an i.i.d. assumption as well as a small number of données that have fundamentally different statistical properties compared to the regular données. We refer to a donnée that substantially deviates from the statistical properties of most données as an outlier. Different methods for outlier detection use different measures for this deviation. Stastistical learning theory studies fundamental limits on the ability to mitigate outliers reliably [84, 85].

overfitting Consider an apprentissage automatique method that uses ERM to learn a hypothèse with the minimum empirical risk on a given training set. Such a method is overfitting the training set if it learns a hypothèse with a small empirical risk on the training set but a significantly larger loss outside the training set.

parameter space The parameter space $\mathcal{W}$ of an apprentissage automatique model $\mathcal{H}$ is the set of all feasible choices for the model parameters (see Figure 15). Many important apprentissage automatique methods use a model that is parametrized by vectors of the Euclidean space $\mathbb{R}^d$. Two widely used examples of parametrized models are linear models and deep nets. The parameter space is then often a subset $\mathcal{W} \subseteq \mathbb{R}^d$, e.g., all vectors $\mathbf{w} \in \mathbb{R}^d$ with a norme smaller than one.

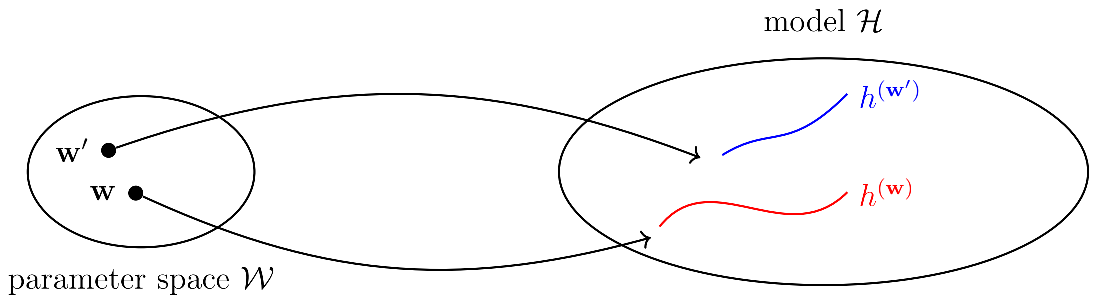


Figure 15: The parameter space $\mathcal{W}$ of an apprentissage automatique model $\mathcal{H}$ consists of all feasible choices for the model parameters. Each choice $\mathbf{w}$ for the model parameters selects a hypothèse map $h^{(\mathbf{w})} \in \mathcal{H}$.

parameters The parameters of an apprentissage automatique model are tunable (i.e., learnable or adjustable) quantities that allow us to choose between different hypothèse maps. For example, the linear model $\mathcal{H} := \{h^{(\mathbf{w})} : h^{(\mathbf{w})}(x) = w_1 x + w_2\}$ consists of all hypothèse maps $h^{(\mathbf{w})}(x) = w_1 x + w_2$ with a particular choice for the parameters $\mathbf{w} = \big(w_1, w_2\big)^T \in \mathbb{R}^2$. Another example of parameters is the weights assigned to the connections between neurons of an ANN.

polynomial regression Polynomial regression aims at learning a polynomial hypothèse map to predict a numeric étiquette based on the numeric caractéristiques of a donnée. For données characterized by a single numeric caractéristique, polynomial regression uses the hypothesis space $\mathcal{H}_d^{(\text{poly})} := \{h(x) = \sum_{j=0}^{d-1} x^j w_j\}$. The quality of a polynomial hypothèse map is measured using the average squared error loss incurred on a set of labeled datapoints (which we refer to as the training set).

positive semi-definite (psd) A (real-valued) symmetric matrix $\mathbf{Q} = \mathbf{Q}^T \in \mathbb{R}^{d \times d}$ is referred to as psd if $\mathbf{x}^T \mathbf{Q} \mathbf{x} \geq 0$ for every vector $\mathbf{x} \in \mathbb{R}^d$. The property of being psd can be extended from matrices to (real-valued) symmetric kernel maps $K : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ (with $K(\mathbf{x}, \mathbf{x}') = K(\mathbf{x}', \mathbf{x})$) as follows: For any finite set of feature vectors $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)}$, the resulting matrix $\mathbf{Q} \in \mathbb{R}^{m \times m}$ with entries $Q_{r,r'} = K\big(\mathbf{x}^{(r)}, \mathbf{x}^{(r')}\big)$ is psd [52].

predictor A predictor is a real-valued hypothèse map. Given a donnée with caractéristiques $\mathbf{x}$, the value $h(\mathbf{x}) \in \mathbb{R}$ is used as a prédiction for the true numeric étiquette $y \in \mathbb{R}$ of the donnée.

principal component analysis (PCA) PCA determines a linear feature map such that the new caractéristiques allow us to reconstruct the original caractéristiques with the minimum reconstruction error [6].

privacy funnel The privacy funnel is a method for learning privacy-friendly caractéristiques of données [86].

privacy leakage Consider an apprentissage automatique application that processes a jeu de données $\mathcal{D}$ and delivers some output, such as the prédictions obtained for new données. Privacy leakage arises if the output carries information about a private (or sensitive) caractéristique of a donnée (which might be a human) of $\mathcal{D}$. Based on a probabilistic model for the data generation, we can measure the privacy leakage via the MI between the output and the senstive caractéristique. Another quantitative measure of privacy leakage is DP. The relations between different measures of privacy leakage have been studied in the literature (see [87]).

privacy protection Consider some apprentissage automatique method $\mathcal{A}$ that reads in a jeu de données $\mathcal{D}$ and delivers some output $\mathcal{A}(\mathcal{D})$. The output could be the learned model parameters $\widehat{\mathbf{w}}$ or the prédiction $\hat{h}(\mathbf{x})$ obtained for a specific donnée with caractéristiques $\mathbf{x}$. Many important apprentissage automatique applications involve données representing humans. Each donnée is characterized by caractéristiques $\mathbf{x}$, potentially a étiquette y, and a sensitive attribute s (e.g., a recent medical diagnosis). Roughly speaking, privacy protection means that it should be impossible to infer, from the output $\mathcal{A}(\mathcal{D})$, any of the sensitive attributes of données in $\mathcal{D}$. Mathematically, privacy protection requires non-invertibility of the map $\mathcal{A}(\mathcal{D})$. In general, just making $\mathcal{A}(\mathcal{D})$ non-invertible is typically insufficient for privacy protection. We need to make $\mathcal{A}(\mathcal{D})$ sufficiently non-invertible.

probabilistic model A probabilistic model interprets données as realizations of VAs with a joint loi de probabilité. This joint loi de probabilité typically involves parameters which have to be manually chosen or learned via statistical inference methods such as maximum likelihood estimation [13].

probabilistic principal component analysis (PPCA) Probabilistic PCA extends basic PCA by using a probabilistic model for données. The probabilistic model of probabilistic PCA reduces the task of dimensionality reduction to an estimation problem that can be solved using EM methods.

probability We assign a probability value, typically chosen in the interval

$[0,1]$, to each event that might occur in a random experiment [5, 36, 37, 88].

probability density function (pdf) The probability density function $p(x)$ of a real-valued VA $x \in \mathbb{R}$ is a particular representation of its loi de probabilité. If the probability density function exists, it can be used to compute the probability that x takes on a value from a (measurable) set $\mathcal{B} \subseteq \mathbb{R}$ via $p(x \in \mathcal{B}) = \int_{\mathcal{B}} p(x')dx'$ [5, Ch. 3]. The probability density function of a vector-valued VA $\mathbf{x} \in \mathbb{R}^d$ (if it exists) allows us to compute the probability of $\mathbf{x}$ belonging to a (measurable) region $\mathcal{R}$ via $p(\mathbf{x} \in \mathcal{R}) = \int_{\mathcal{R}} p(\mathbf{x}')dx'_1 \ldots dx'_d$ [5, Ch. 3].

probability space A probability space is a mathematical model of a physical process (a random experiment) with an uncertain outcome. Formally, a probability space $\mathcal{P}$ is a triplet $(\Omega, \mathcal{F}, P)$ where

- Ω is a sample space containing all possible elementary outcomes of a random experiment;
- $\mathcal{F}$ is a sigma-algebra, a collection of subsets of Ω (called events) that satisfies certain closure properties under set operations;
- P is a probability measure, a function that assigns a probability $P(\mathcal{A}) \in [0,1]$ to each event $\mathcal{A} \in \mathcal{F}$. The function must satisfy $P(\Omega) = 1$ and $P\left(\bigcup_{i=1}^{\infty} \mathcal{A}_i\right) = \sum_{i=1}^{\infty} P(\mathcal{A}_i)$ for any countable sequence of pairwise disjoint events $\mathcal{A}_1, \mathcal{A}_2, \ldots$ in $\mathcal{F}$.

Probability spaces provide the foundation for defining VAs and to reason about uncertainty in apprentissage automatique applications [15, 37, 59].

projected gradient descent (projected GD) Consider an ERM-based method that uses a parametrized model with parameter space $\mathcal{W} \subseteq \mathbb{R}^d$. Even if the objective function of ERM is smooth, we cannot use basic GD, as it does not take into account contraints on the optimization variable (i.e., the model parameters). Projected GD extends basic GD to handle constraints on the optimization variable (i.e., the model parameters). A single iteration of projected GD consists of first taking a gradient step and then projecting the result back onto the parameter space.

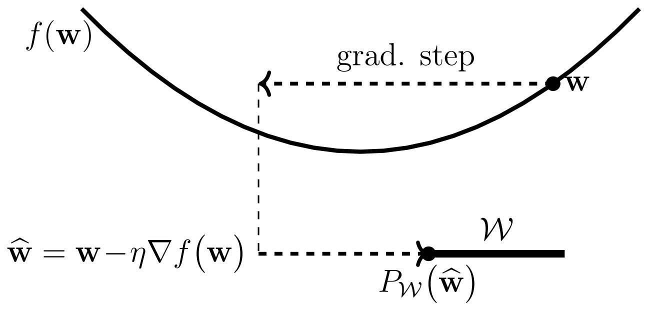

$f(\mathbf{w})$
grad. step
$\mathbf{w}$
$\mathcal{W}$
$\widehat{\mathbf{w}} = \mathbf{w} - \eta \nabla f(\mathbf{w})$
$P_{\mathcal{W}}(\widehat{\mathbf{w}})$

Figure 16: Projected GD augments a basic gradient step with a projection back onto the constraint set $\mathcal{W}$.

projection Consider a subset $\mathcal{W} \subseteq \mathbb{R}^d$ of the d-dimensional Euclidean space. We define the projection $P_{\mathcal{W}}(\mathbf{w})$ of a vector $\mathbf{w} \in \mathbb{R}^d$ onto $\mathcal{W}$ as

$$P_{\mathcal{W}}(\mathbf{w}) = \underset{\mathbf{w}' \in \mathcal{W}}{\operatorname{argmin}} \left\| \mathbf{w} - \mathbf{w}' \right\|_2 . \tag{10}$$

In other words, $P_{\mathcal{W}}(\mathbf{w})$ is the vector in $\mathcal{W}$ which is closest to $\mathbf{w}$. The projection is only well-defined for subsets $\mathcal{W}$ for which the above minimum exists [17].

proximable A convex function for which the proximal operator can be computed efficiently is sometimes referred to as proximable or simple [89].

proximal operator Given a convex function $f(\mathbf{w}')$, we define its proximal operator as [63, 90]

$$\mathbf{prox}_{f(\cdot),\rho}(\mathbf{w}) := \underset{\mathbf{w}' \in \mathbb{R}^d}{\operatorname{argmin}} \left[f(\mathbf{w}') + (\rho/2) \left\| \mathbf{w} - \mathbf{w}' \right\|_2^2 \right] \text{ with } \rho > 0.$$

As illustrated in Figure 17, evaluating the proximal operator amounts to minimizing a penalized variant of $f(\mathbf{w}')$. The penalty term is the scaled squared Euclidean distance to a given vector $\mathbf{w}$ (which is the input to the proximal operator). The proximal operator can be interpreted as a generalization of the gradient step, which is defined for a smooth convex function $f(\mathbf{w}')$. Indeed, taking a gradient step with step size η at the current vector $\mathbf{w}$ is the same as applying the proximal operator of the function $\tilde{f}(\mathbf{w}') = \big(\nabla f(\mathbf{w})\big)^T (\mathbf{w}' - \mathbf{w})$ and using $\rho = 1/\eta$.

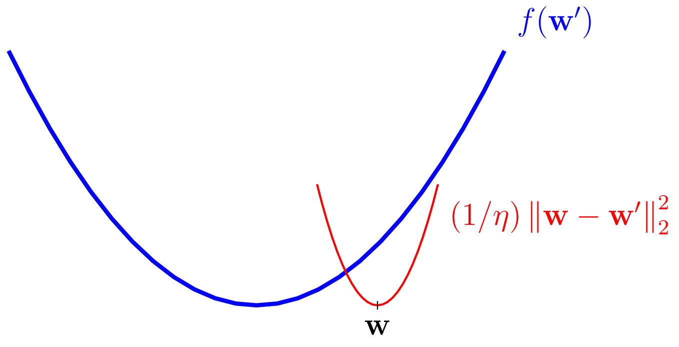


Figure 17: A generalized gradient step updates a vector $\mathbf{w}$ by minimizing a penalized version of the function $f(\cdot)$. The penalty term is the scaled squared Euclidean distance between the optimization variable $\mathbf{w}'$ and the given vector $\mathbf{w}$.

prédiction A prediction is an estimate or approximation for some quantity of interest. Apprentissage automatique revolves around learning or finding a hypothèse map h that reads in the caractéristiques $\mathbf{x}$ of a donnée and delivers a prediction $\widehat{y} := h(\mathbf{x})$ for its étiquette y.

quadratic function A function $f : \mathbb{R}^d \rightarrow \mathbb{R}$ of the form

$$f(\mathbf{w}) = \mathbf{w}^T\mathbf{Q}\mathbf{w} + \mathbf{q}^T\mathbf{w} + a,$$

with some matrix $\mathbf{Q} \in \mathbb{R}^{d\times d}$, vector $\mathbf{q} \in \mathbb{R}^d$, and scalar $a \in \mathbb{R}$.

random forest A random forest is a set (or ensemble) of different decision trees. Each of these decision trees is obtained by fitting a perturbed copy of the original jeu de données.

realization Consider an VA x which maps each element (i.e., outcome or elementary event) $\omega \in \mathcal{P}$ of a probability space $\mathcal{P}$ to an element a of a measurable space $\mathcal{N}$ [2,36,37]. A realization of x is any element $a' \in \mathcal{N}$ such that there is an element $\omega' \in \mathcal{P}$ with $x(\omega') = a'$.

rectified linear unit (ReLU) The ReLU is a popular choice for the activation function of a neuron within an ANN. It is defined as $\sigma(z) = \max\{0, z\}$, with z being the weighted input of the artificial neuron.

regression Regression problems revolve around the prediction of a numeric étiquette solely from the caractéristiques of a donnée [6, Ch. 2].

regret The regret of a hypothèse h relative to another hypothèse h', which serves as a baseline, is the difference between the loss incurred by h and the loss incurred by h' [40]. The baseline hypothèse h' is also referred to as an expert.

regularization A key challenge of modern apprentissage automatique applications is that they often use large models, which have an effective dimension in the order of billions. Training a high-dimensional model using basic ERM-based methods is prone to overfitting: the learned hypothèse performs well on the training set but poorly outside the training set. Regularization refers to modifications of a given instance of ERM in order to avoid overfitting, i.e., to ensure that the learned hypothèse performs not much worse outside the training set. There are three routes for implementing regularization:

1) Model pruning: We prune the original model $\mathcal{H}$ to obtain a smaller

model $\mathcal{H}'$. For a parametric model, the pruning can be implemented via constraints on the model parameters (such as $w_1 \in [0.4, 0.6]$ for the weight of caractéristique x_1 in linear regression).

2) Loss penalization: We modify the objective function of ERM by adding a penalty term to the training error. The penalty term estimates how much larger the expected loss (or risk) is compared to the average loss on the training set.

3) Data augmentation: We can enlarge the training set $\mathcal{D}$ by adding perturbed copies of the original données in $\mathcal{D}$. One example for such a perturbation is to add the realization of an VA to the feature vector of a donnée.

Figure 18 illustrates the above three routes to regularization. These routes are closely related and sometimes fully equivalent: data augmentation using Gaussian RVs to perturb the feature vectors in the training set of linear regression has the same effect as adding the penalty $\lambda \|\mathbf{w}\|_2^2$ to the training error (which is nothing but ridge regression). The decision on which route to use for regularization can be based on the available computational infrastructure. For example, it might be much easier to implement data augmentation than model pruning.

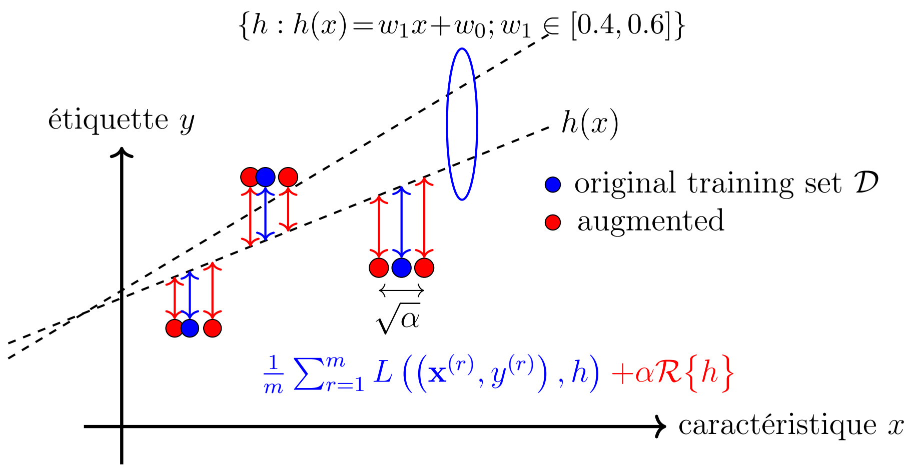

$\{h : h(x) = w_1 x + w_0; w_1 \in [0.4, 0.6]\}$
étiquette y
$h(x)$
original training set $\mathcal{D}$
augmented
$\sqrt{\alpha}$
$\frac{1}{m}\sum_{r=1}^{m} L\left(\left(\mathbf{x}^{(r)}, y^{(r)}\right), h\right) + \alpha\mathcal{R}\{h\}$
caractéristique x

Figure 18: Three approaches to regularization: 1) data augmentation; 2) loss penalization; and 3) model pruning (via constraints on model parameters).

regularized empirical risk minimization (RERM) Basic ERM learns a hypothèse (or trains a model) $h \in \mathcal{H}$ based solely on the empirical risk $\widehat{L}\big(h|\mathcal{D}\big)$ incurred on a training set $\mathcal{D}$. To make ERM less prone to overfitting, we can implement regularization by including a (scaled) regularizer $\mathcal{R}\big\{h\big\}$ in the learning objective. This leads to regularized empirical risk minimization (RERM),

$$\hat{h} \in \operatorname*{argmin}_{h \in \mathcal{H}} \widehat{L}\big(h|\mathcal{D}\big) + \alpha \mathcal{R}\big\{h\big\}. \tag{11}$$

The parameter $\alpha \geq 0$ controls the regularization strength. For $\alpha = 0$, we recover standard ERM without regularization. As α increases, the learned hypothèse is increasingly biased toward small values of $\mathcal{R}\big\{h\big\}$. The component $\alpha\mathcal{R}\big\{h\big\}$ in the objective function of (11) can be intuitively understood as a surrogate for the increased average loss that may occur when predicting étiquettes for données outside the training set. This intuition can be made precise in various ways. For example, consider a linear model trained using squared error loss and the regularizer $\mathcal{R}\big\{h\big\} = \|\mathbf{w}\|_2^2$. In this setting, $\alpha\mathcal{R}\big\{h\big\}$ corresponds to the expected increase in loss caused by adding Gaussian RVs to the feature vectors in the training set [6, Ch. 3]. A principled construction for the regularizer $\mathcal{R}\big\{h\big\}$ arises from approximate upper bounds on the generalization error. The resulting RERM instance is known as SRM [91, Sec. 7.2].

regularized loss minimization (RLM) See RERM.

regularizer A regularizer assigns each hypothèse h from a hypothesis space $\mathcal{H}$ a quantitative measure $\mathcal{R}\big\{h\big\}$ for how much its prédiction error on a training set might differ from its prédiction errors on données outside the training set. Ridge regression uses the regularizer $\mathcal{R}\{h\} := \|\mathbf{w}\|_2^2$ for linear hypothèse maps $h^{(\mathbf{w})}(\mathbf{x}) := \mathbf{w}^T\mathbf{x}$ [6, Ch. 3]. Lasso uses the regularizer $\mathcal{R}\{h\} := \|\mathbf{w}\|_1$ for linear hypothèse maps $h^{(\mathbf{w})}(\mathbf{x}) := \mathbf{w}^T\mathbf{x}$ [6, Ch. 3].

Rényi divergence The Rényi divergence measures the (dis)similarity between two loi de probabilités [92].

reward A reward refers to some observed (or measured) quantity that allows us to estimate the loss incurred by the prédiction (or decision) of a hypothèse $h(\mathbf{x})$. For example, in an apprentissage automatique application to self-driving vehicles, $h(\mathbf{x})$ could represent the current steering direction of a vehicle. We could construct a reward from the measurements of a collision sensor that indicate if the vehicle is moving towards an obstacle. We define a low reward for the steering direction $h(\mathbf{x})$ if the vehicle moves dangerously towards an obstacle.

ridge regression Ridge regression learns the weights $\mathbf{w}$ of a linear hypothèse map $h^{(\mathbf{w})}(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$. The quality of a particular choice for the model parameters $\mathbf{w}$ is measured by the sum of two components. The first component is the average squared error loss incurred by $h^{(\mathbf{w})}$ on a set of labeled datapoints (i.e., the training set). The second component is the scaled squared Euclidean norme $\alpha\|\mathbf{w}\|_2^2$ with a regularization parameter $\alpha > 0$. Adding $\alpha\|\mathbf{w}\|_2^2$ to the average squared error loss is equivalent to replacing each original données by the realization of (infinitely many) i.i.d. VAs centered around these données (see regularization).

risk Consider a hypothèse h used to predict the étiquette y of a donnée based on its caractéristiques $\mathbf{x}$. We measure the quality of a particular prédiction using a loss function $L\left((\mathbf{x}, y), h\right)$. If we interpret données as the realizations of i.i.d. VAs, also the $L\left((\mathbf{x}, y), h\right)$ becomes the realization of an VA. The i.i.d. assumption allows us to define the risk of a hypothèse as the expected loss $\mathbb{E}\big\{L\left((\mathbf{x}, y), h\right)\big\}$. Note that the risk of h depends on both the specific choice for the loss function and the loi de probabilité of the données.

sample A finite sequence (or list) of données $\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(m)}$ that is obtained or interpreted as the realization of m i.i.d. VAs with a common loi de probabilité $p(\mathbf{z})$. The length m of the sequence is referred to as the sample size.

sample covariance matrix The sample covariance matrix $\widehat{\mathbf{\Sigma}} \in \mathbb{R}^{d \times d}$ for a given set of feature vectors $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)} \in \mathbb{R}^d$ is defined as

$$\widehat{\mathbf{\Sigma}} = (1/m) \sum_{r=1}^{m} (\mathbf{x}^{(r)} - \widehat{\mathbf{m}})(\mathbf{x}^{(r)} - \widehat{\mathbf{m}})^T.$$

Here, we use the sample mean $\widehat{\mathbf{m}}$.

sample mean The sample mean $\mathbf{m} \in \mathbb{R}^d$ for a given jeu de données, with feature vectors $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)} \in \mathbb{R}^d$, is defined as

$$\mathbf{m} = (1/m) \sum_{r=1}^{m} \mathbf{x}^{(r)}.$$

sample size The number of individual données contained in a jeu de données.

scatterplot A visualization technique that depicts données by markers in a two-dimensional plane. Figure 19 depicts an example of a scatterplot.

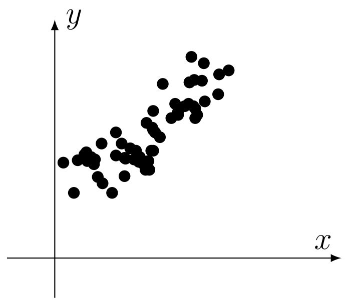


Figure 19: A scatterplot of some données representing daily weather conditions in Finland. Each donnée is characterized by its minimum daytime temperature x as the caractéristique and its maximum daytime temperature y as the étiquette. The temperatures have been measured at the FMI weather station Helsinki Kaisaniemi during 1.9.2024 - 28.10.2024.

semi-supervised learning (SSL) SSL methods use unlabeled datapoints to support the learning of a hypothèse from labeled datapoints [16]. This approach is particularly useful for apprentissage automatique applications that offer a large amount of unlabeled datapoints, but only a limited number of labeled datapoints.

sensitive attribute apprentissage automatique revolves around learning a hypothèse map that allows us to predict the étiquette of a donnée from its caractéristiques. In some applications, we must ensure that the output delivered by an apprentissage automatique system does not allow us to infer sensitive attributes of a donnée. Which part of a donnée is considered a sensitive attribute is a design choice that varies across different application domains.

similarity graph Some apprentissage automatique applications generate données that are related by a domain-specific notion of similarity. These similarities can be represented conveniently using a similarity graph $\mathcal{G} = \big(\mathcal{V} := \{1, \ldots, m\}, \mathcal{E}\big)$. The node $r \in \mathcal{V}$ represents the r-th donnée. Two nodes are connected by an undirected edge if the corresponding données are similar.

singular value decomposition (SVD) The SVD for a matrix $\mathbf{A} \in \mathbb{R}^{m \times d}$ is a factorization of the form

$$\mathbf{A} = \mathbf{V}\mathbf{\Lambda}\mathbf{U}^{T},$$

with orthonormal matrices $\mathbf{V} \in \mathbb{R}^{m \times m}$ and $\mathbf{U} \in \mathbb{R}^{d \times d}$ [3]. The matrix $\mathbf{\Lambda} \in \mathbb{R}^{m \times d}$ is only non-zero along the main diagonal, whose entries $\Lambda_{j,j}$ are non-negative and referred to as singular values.

smooth A real-valued function $f : \mathbb{R}^{d} \rightarrow \mathbb{R}$ is smooth if it is dérivable and its gradient $\nabla f(\mathbf{w})$ is continuous at all $\mathbf{w} \in \mathbb{R}^{d}$ [80, 93]. A smooth function f is referred to as β-smooth if the gradient $\nabla f(\mathbf{w})$ is Lipschitz continuous with Lipschitz constant β, i.e.,

$$\|\nabla f(\mathbf{w}) - \nabla f(\mathbf{w}')\| \leq \beta \|\mathbf{w} - \mathbf{w}'\|, \text{ for any } \mathbf{w}, \mathbf{w}' \in \mathbb{R}^{d}.$$

The constant β quantifies the amount of smoothness of the function f: the smaller the β, the smoother f is. Optimization problems with a smooth objective function can be solved effectively by gradient-based methods. Indeed, gradient-based methods approximate the objective function locally around a current choice $\mathbf{w}$ using its gradient. This approximation works well if the gradient does not change too rapidly. We can make this informal claim precise by studying the effect of a single gradient step with step size $\eta = 1/\beta$ (see Figure 20).

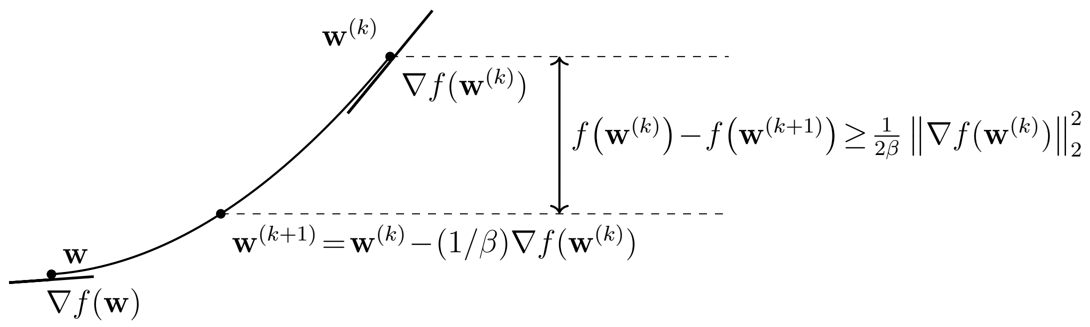
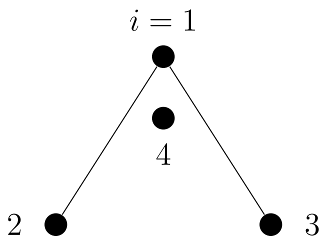

$\mathbf{w}^{(k)}$
$\nabla f(\mathbf{w}^{(k)})$
$f\left(\mathbf{w}^{(k)}\right) - f\left(\mathbf{w}^{(k+1)}\right) \geq \frac{1}{2\beta} \left\|\nabla f(\mathbf{w}^{(k)})\right\|_2^2$
$\mathbf{w}$
$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} - (1/\beta)\nabla f(\mathbf{w}^{(k)})$
$\nabla f(\mathbf{w})$

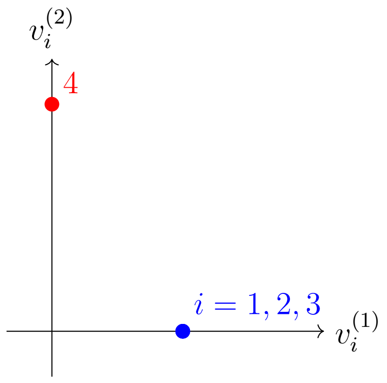
Figure 20: Consider an objective function $f(\mathbf{w})$ that is β-smooth. Taking a gradient step, with step size $\eta = 1/\beta$, decreases the objective by at least $\frac{1}{2\beta} \left\|\nabla f(\mathbf{w}^{(k)})\right\|_2^2$ [80, 93, 94]. Note that the step size $\eta = 1/\beta$ becomes larger for smaller β. Thus, for smoother objective functions (i.e., those with smaller β), we can take larger steps.

soft clustering Soft clustering refers to the task of partitioning a given set of données into (a few) overlapping clusters. Each donnée is assigned to several different clusters with varying degrees of belonging. Soft clustering methods determine the degree of belonging (or soft cluster assignment) for each donnée and each cluster. A principled approach to soft clustering is by interpreting données as i.i.d. realizations of a GMM. We then obtain a natural choice for the degree of belonging as the conditional probability of a donnée belonging to a specific mixture component.

spectral clustering Spectral clustering is a particular instance of graph clustering, i.e., it clusters données represented as the nodes $i = 1, \ldots, n$ of a graph $\mathcal{G}$. Spectral clustering uses the eigenvectors of the Laplacian matrix $\mathbf{L}^{(\mathcal{G})}$ to construct feature vectors $\mathbf{x}^{(i)} \in \mathbb{R}^d$ for each node (i.e., for each donnée) $i = 1, \ldots, n$. We can feed these feature vectors into Euclidean space-based clustering methods, such as k-means or soft clustering via GMM. Roughly speaking, the feature vectors of nodes belonging to a well-connected subset (or cluster) of nodes in $\mathcal{G}$ are located nearby in the Euclidean space $\mathbb{R}^d$ (see Figure 21).



$$\mathbf{L}^{(\mathcal{G})} = \begin{pmatrix} 2 & -1 & -1 & 0 \\ -1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T$$



$$\mathbf{V} = \left(\mathbf{v}^{(1)}, \mathbf{v}^{(2)}, \mathbf{v}^{(3)}, \mathbf{v}^{(4)}\right)$$

$$\mathbf{v}^{(1)} = \frac{1}{\sqrt{3}} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \end{pmatrix}, \; \mathbf{v}^{(2)} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

Figure 21: **Top.** Left: An undirected graph $\mathcal{G}$ with four nodes $i = 1, 2, 3, 4$, each representing a donnée. Right: The Laplacian matrix $\mathbf{L}^{(\mathcal{G})} \in \mathbb{R}^{4\times4}$ and its EVD. **Bottom.** Left: A scatterplot of données using the feature vectors $\mathbf{x}^{(i)} = \left(v_i^{(1)}, v_i^{(2)}\right)^T$. Right: Two eigenvectors $\mathbf{v}^{(1)}, \mathbf{v}^{(2)} \in \mathbb{R}^d$ corresponding to the eigenvalue $\lambda = 0$ of the Laplacian matrix $\mathbf{L}^{(\mathcal{G})}$.

spectrogram A spectrogram represents the time-frequency distribution of the energy of a time signal $x(t)$. Intuitively, it quantifies the amount of signal energy present within a specific time segment $[t_1, t_2] \subseteq \mathbb{R}$ and frequency interval $[f_1, f_2] \subseteq \mathbb{R}$. Formally, the spectrogram of a signal is defined as the squared magnitude of its short-time Fourier transform (STFT) [95]. Figure 22 depicts a time signal along with its spectrogram.

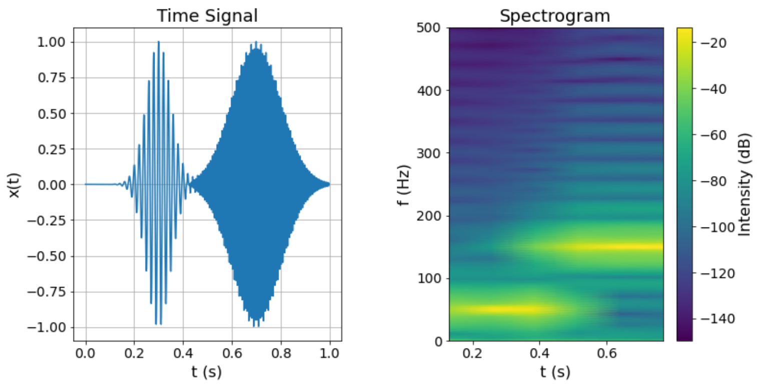


Figure 22: Left: A time signal consisting of two modulated Gaussian pulses. Right: An intensity plot of the spectrogram.

The intensity plot of its spectrogram can serve as an image of a signal. A simple recipe for audio signal classification is to feed this signal image into deep nets originally developed for image classification and object detection [96]. It is worth noting that, beyond the spectrogram, several alternative representations exist for the time-frequency distribution of signal energy [97, 98].

squared error loss The squared error loss measures the prédiction error of a hypothèse h when predicting a numeric étiquette $y \in \mathbb{R}$ from the caractéristiques $\mathbf{x}$ of a donnée. It is defined as

$$L\left((\mathbf{x},y),h\right) := \big(y - \underbrace{h(\mathbf{x})}_{=\hat{y}}\big)^2.$$

stability Stability is a desirable property of a apprentissage automatique method $\mathcal{A}$ that maps a jeu de données $\mathcal{D}$ (e.g., a training set) to an output $\mathcal{A}(\mathcal{D})$, such as learned model parameters or the prédiction for a specific donnée. Intuitively, $\mathcal{A}$ is stable if small changes in the input jeu de données $\mathcal{D}$ lead to small changes in the output $\mathcal{A}(\mathcal{D})$. Several formal notions of stability exist that enable bounds on the generalization error or risk of the method; see [60, Ch. 13]. To build intuition, consider the three datasets depicted in Fig. 23, each of which is equally likely under the same data-generating loi de probabilité. Since the optimal model parameters are determined by this underlying loi de probabilité, an accurate apprentissage automatique method $\mathcal{A}$ should return the same (or very similar) output $\mathcal{A}(\mathcal{D})$ for all three jeu de données. In other words, any useful $\mathcal{A}$ must be robust to variability in sample realizations from the same loi de probabilité, i.e., it must be stable.

statistical aspects By statistical aspects of an apprentissage automatique method, we refer to (properties of) the loi de probabilité of its output under a probabilistic model for the data fed into the method.

step size See learning rate.

stochastic block model (SBM) The stochastic block model is a probabilistic generative model for an undirected graph $\mathcal{G} = \big(\mathcal{V},\mathcal{E}\big)$ with a given set of nodes $\mathcal{V}$ [99]. In its most basic variant, the stochastic block

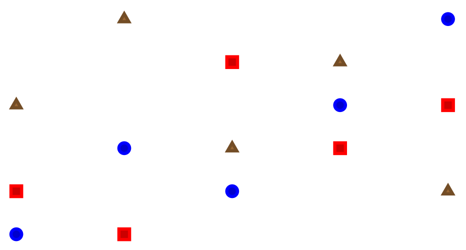


Figure 23: Three jeu de donnéess $\mathcal{D}^{(*)}$, $\mathcal{D}^{(\square)}$, and $\mathcal{D}^{(\triangle)}$, each sampled independently from the same data-generating loi de probabilité. A stable apprentissage automatique method should return similar outputs when trained on any of these jeu de donnéess.

model generates a graph by first randomly assigning each node $i \in \mathcal{V}$ to a cluster index $c_i \in \{1, \ldots, k\}$. A pair of different nodes in the graph is connected by an edge with probability $p_{i,i'}$ that depends solely on the étiquettes $c_i, c_{i'}$. The presence of edges between different pairs of nodes is statistically independent.

stochastic gradient descent (SGD) Stochastic GD is obtained from GD by replacing the gradient of the objective function with a stochastic approximation. A main application of stochastic GD is to train a parametrized model via ERM on a training set $\mathcal{D}$ that is either very large or not readily available (e.g., when données are stored in a database distributed all over the planet). To evaluate the gradient of the empirical risk (as a function of the model parameters $\mathbf{w}$), we need to compute a sum $\sum_{r=1}^{m} \nabla_{\mathbf{w}} L\left(\mathbf{z}^{(r)}, \mathbf{w}\right)$ over all données in the training set. We obtain a stochastic approximation to the gradient by replacing the sum $\sum_{r=1}^{m} \nabla_{\mathbf{w}} L\left(\mathbf{z}^{(r)}, \mathbf{w}\right)$ with a sum $\sum_{r\in\mathcal{B}} \nabla_{\mathbf{w}} L\left(\mathbf{z}^{(r)}, \mathbf{w}\right)$ over a randomly chosen subset $\mathcal{B} \subseteq \{1, \ldots, m\}$ (see Figure 24). We often refer to these randomly chosen données as a batch. The batch size $|\mathcal{B}|$ is an important parameter of stochastic GD. Stochastic GD with $|\mathcal{B}| > 1$ is referred to as mini-batch stochastic GD [100].

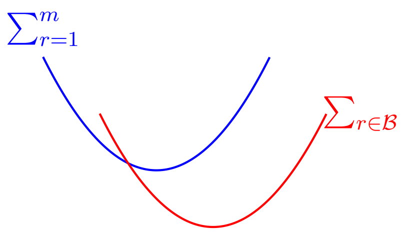


Figure 24: Stochastic GD for ERM approximates the gradient $\sum_{r=1}^{m} \nabla_{\mathbf{w}} L\left(\mathbf{z}^{(r)}, \mathbf{w}\right)$ by replacing the sum over all données in the training set (indexed by $r = 1, \ldots, m$) with a sum over a randomly chosen subset $\mathcal{B} \subseteq \{1, \ldots, m\}$.

stopping criterion Many apprentissage automatique methods use iterative algorithms that construct a sequence of model parameters (such as the weights of a linear map or the weights of an ANN). These parameters (hopefully) converge to an optimal choice for the model parameters. In practice, given finite computational resources, we need to stop iterating after a finite number of repetitions. A stopping criterion is any well-defined condition required for stopping the iteration.

strongly convex A continuously dérivable real-valued function $f(\mathbf{x})$ is strongly convex with coefficient σ if $f(\mathbf{y}) \geq f(\mathbf{x})+\nabla f(\mathbf{x})^T(\mathbf{y}-\mathbf{x})+(\sigma/2)\left\|\mathbf{y}-\mathbf{x}\right\|_2^2$ [80], [94, Sec. B.1.1].

structural risk minimization (SRM) Structural risk minimization (SRM) is an instance of RERM, which which the model $\mathcal{H}$ can be expressed as a countable union of sub-models: $\mathcal{H} = \bigcup_{n=1}^{\infty} \mathcal{H}^{(n)}$. Each sub-model $\mathcal{H}^{(n)}$ permits the derivation of an approximate upper bound on the generalization error incurred when applying ERM to train $\mathcal{H}^{(n)}$. These individual bounds—one for each sub-model—are then combined to form a regularizer used in the RERM objective. These approximate upper bounds (one for each $\mathcal{H}^{(n)}$) are then combined to construct a regularizer for RERM [60, Sec. 7.2].

subgradient For a real-valued function $f : \mathbb{R}^d \to \mathbb{R} : \mathbf{w} \mapsto f(\mathbf{w})$, a vector $\mathbf{a}$ such that $f(\mathbf{w}) \geq f(\mathbf{w}') + \big(\mathbf{w}-\mathbf{w}'\big)^T\mathbf{a}$ is referred to as a subgradient of f at $\mathbf{w}'$ [101, 102].

subgradient descent Subgradient descent is a generalization of GD that does not require differentiability of the function to be minimized. This generalization is obtained by replacing the concept of a gradient with that of a subgradient. Similar to gradients, also subgradients allow us to construct local approximations of an objective function. The objective function might be the empirical risk $\widehat{L}\big(h^{(\mathbf{w})}\big|\mathcal{D}\big)$ viewed as a function of the model parameters $\mathbf{w}$ that select a hypothèse $h^{(\mathbf{w})} \in \mathcal{H}$.

support vector machine (SVM) The SVM is a binary classification method that learns a linear hypothèse map. Thus, like linear regression and logistic regression, it is also an instance of ERM for the linear model.

However, the SVM uses a different loss function from the one used in those methods. As illustrated in Figure 25, it aims to maximally separate données from the two different classes in the feature space (i.e., maximum margin principle). Maximizing this separation is equivalent to minimizing a regularized variant of the hinge loss (5) [38, 68, 103].

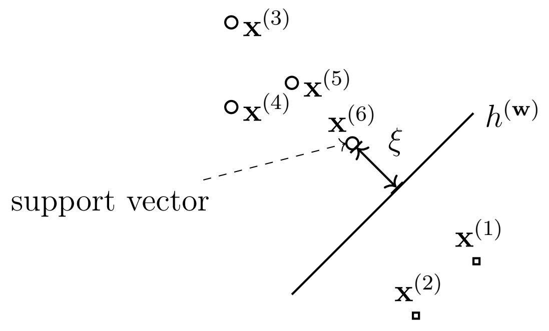


Figure 25: The SVM learns a hypothèse (or classifier) $h^{(\mathbf{w})}$ with minimal average soft-margin hinge loss. Minimizing this loss is equivalent to maximizing the margin ξ between the decision boundary of $h^{(\mathbf{w})}$ and each class of the training set.

The above basic variant of SVM is only useful if the données from different categories can be (approximately) linearly separated. For an apprentissage automatique application where the categories are not derived from a kernel.

test set A set of données that have been used neither to train a model (e.g., via ERM) nor in a validation set to choose between different models.

total variation See GTV.

training error The average loss of a hypothèse when predicting the étiquettes of the données in a training set. We sometimes refer by training error also to minimal average loss which is achieved by a solution of ERM.

training set A training set is a jeu de données $\mathcal{D}$ which consists of some données used in ERM to learn a hypothèse $\hat{h}$. The average loss of $\hat{h}$ on the training set is referred to as the training error. The comparison of the training error with the validation error of $\hat{h}$ allows us to diagnose the apprentissage automatique method and informs how to improve the validation error (e.g., using a different hypothesis space or collecting more données) [6, Sec. 6.6].

transparency Transparency is a fundamental requirement for trustworthy AI [104]. In the context of apprentissage automatique methods, transparency is often used interchangeably with explainability [44, 105]. However, in the broader scope of AI systems, transparency extends beyond explainability and includes providing information about the system's limitations, reliability, and intended use. In medical diagnosis systems, transparency requires disclosing the confidence level for the prédictions delivered by a trained model. In credit scoring, AI-based lending decisions should be accompanied by explanations of contributing factors, such as income level or credit history. These explanations allow humans (e.g., a loan applicant) to understand and contest automated decisions. Some apprentissage automatique methods inherently offer transparency. For example, logistic regression provides a quantitative measure of classification reliability through the value $|h(\mathbf{x})|$. Decision trees are another example, as they allow human-readable decision rules [46]. Transparency also requires a clear indication when a user is engaging with an AI system. For example, AI-powered chatbots should notify users that they are interacting with an automated system rather than a human. Furthermore, transparency encompasses comprehensive documentation detailing the purpose and design choices underlying the AI system. For instance, model datasheets [30] and AI system cards [106] help practitioners understand the intended use cases and limitations of an AI system [107].

trustworthy artificial intelligence (trustworthy AI) Besides the computational aspects and statistical aspects, a third main design aspect of apprentissage automatique methods is their trustworthiness [108]. The EU has put forward seven key requirements (KRs) for trustworthy AI (that typically build on apprentissage automatique methods) [109]:

1) KR1 - Human agency and oversight;
2) KR2 - Technical robustness and safety;
3) KR3 - Privacy and data governance;
4) KR4 - Transparency;
5) KR5 - Diversity, non-discrimination and fairness;
6) KR6 - Societal and environmental well-being;
7) KR7 - Accountability.

uncertainty Uncertainty refers to the degree of confidence—or lack thereof—associated with a quantity such as a model prediction, parameter estimate, or observed data point. In apprentissage automatique, uncertainty arises from various sources, including noisy data, limited training samples, or ambiguity in model assumptions. Probability theory offers a principled framework for representing and quantifying such uncertainty.

underfitting Consider an apprentissage automatique method that uses ERM to learn a hypothèse with the minimum empirical risk on a given training set. Such a method is underfitting the training set if it is not able to learn a hypothèse with a sufficiently small empirical risk on the training set. If a method is underfitting, it will typically also not be able to learn a hypothèse with a small risk.

upper confidence bound (UCB) Consider a apprentissage automatique application that requires selecting, at each time step k, an action a_k from a finite set of alternatives $\mathcal{A}$. The utility of selecting action a_k is quantified by a numeric reward signal $r^{(a_k)}$. A widely used probabilistic model for this type of sequential decision-making problem is the stochastic multi-armed bandit setting [77]. In this model, the reward $r^{(a)}$ is viewed as the realization of a VA with unknown mean $\mu^{(a)}$. Ideally, we would always choose the action with the largest expected reward $\mu^{(a)}$, but these means are unknown and must be estimated from observed data. Simply choosing the action with the largest estimate $\widehat{\mu}^{(a)}$ can lead to suboptimal outcomes due to estimation uncertainty. The UCB strategy addresses this by selecting actions not only based on their estimated means but also by incorporating a term that reflects the uncertainty in these estimates—favouring actions with high potential reward and high uncertainty. Theoretical guarantees for the performance of UCB strategies,including logarithmic regret bounds, are established in [77].

validation Consider a hypothèse $\hat{h}$ that has been learned via some apprentissage automatique method, e.g., by solving ERM on a training set $\mathcal{D}$. Validation refers to the practice of evaluating the loss incurred by the hypothèse $\hat{h}$ on a set of données that are not contained in the training set $\mathcal{D}$.

validation error Consider a hypothèse $\hat{h}$ which is obtained by some apprentissage automatique method, e.g., using ERM on a training set. The average loss of $\hat{h}$ on a validation set, which is different from the training set, is referred to as the validation error.

validation set A set of données used to estimate the risk of a hypothèse $\hat{h}$ that has been learned by some apprentissage automatique method (e.g., solving ERM). The average loss of $\hat{h}$ on the validation set is referred to as the validation error and can be used to diagnose an apprentissage automatique method (see [6, Sec. 6.6]). The comparison between training error and validation error can inform directions for improvement of the apprentissage automatique method (such as using a different hypothesis space).

Vapnik–Chervonenkis dimension (VC dimension) The VC dimension of an infinite hypothesis space is a widely-used measure for its size. We refer to the literature (see [60]) for a precise definition of VC dimension as well as a discussion of its basic properties and use in apprentissage automatique.

variable aléatoire (VA) An RV is a function that maps from a probability space $\mathcal{P}$ to a value space [15, 37]. The probability space consists of elementary events and is equipped with a probability measure that assigns probabilities to subsets of $\mathcal{P}$. Different types of RVs include

- binary RVs, which map elementary events to a set of two distinct values, such as $\{-1, 1\}$ or $\{\text{cat}, \text{no cat}\}$;
- real-valued RVs, which take values in the real numbers $\mathbb{R}$;
- vector-valued RVs, which map elementary events to the Euclidean space $\mathbb{R}^d$.

Probability theory uses the concept of measurable spaces to rigorously define and study the properties of (large) collections of RVs [37].

variance The variance of a real-valued VA x is defined as the espérance $\mathbb{E}\{(x - \mathbb{E}\{x\})^2\}$ of the squared difference between x and its espérance $\mathbb{E}\{x\}$. We extend this definition to vector-valued VAs $\mathbf{x}$ as $\mathbb{E}\{\|\mathbf{x} - \mathbb{E}\{\mathbf{x}\}\|_2^2\}$.

vertical federated learning (vertical FL) Vertical FL uses local datasets that are constituted by the same données but characterizing them with different caractéristiques [110]. For example, different healthcare providers might all contain information about the same population of patients. However, different healthcare providers collect different measurements (e.g., blood values, electrocardiography, lung X-ray) for the same patients.

weights Consider a parametrized hypothesis space $\mathcal{H}$. We use the term weights for numeric model parameters that are used to scale caractéristiques or their transformations in order to compute $h^{(\mathbf{w})} \in \mathcal{H}$. A linear model uses weights $\mathbf{w} = \big(w_1, \ldots, w_d\big)^T$ to compute the linear combination $h^{(\mathbf{w})}(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$. Weights are also used in ANNs to form linear combinations of caractéristiques or the outputs of neurons in hidden layers.

zero-gradient condition Consider the unconstrained optimization problem $\min_{\mathbf{w} \in \mathbb{R}^d} f(\mathbf{w})$ with a smooth and convex objective function $f(\mathbf{w})$. A necessary and sufficient condition for a vector $\widehat{\mathbf{w}} \in \mathbb{R}^d$ to solve this problem is that the gradient $\nabla f\big(\widehat{\mathbf{w}}\big)$ is the zero vector,

$$\nabla f\big(\widehat{\mathbf{w}}\big) = \mathbf{0} \Leftrightarrow f\big(\widehat{\mathbf{w}}\big) = \min_{\mathbf{w} \in \mathbb{R}^d} f(\mathbf{w}).$$

0/1 loss The 0/1 loss $L^{(0/1)}\left((\mathbf{x}, y), h\right)$ measures the quality of a classifier $h(\mathbf{x})$ that delivers a prédiction $\hat{y}$ (e.g., via thresholding (1)) for the étiquette y of a donnée with caractéristiques $\mathbf{x}$. It is equal to 0 if the prédiction is correct, i.e., $L^{(0/1)}\left((\mathbf{x}, y), h\right) = 0$ when $\hat{y} = y$. It is equal to 1 if the prédiction is wrong, i.e., $L^{(0/1)}\left((\mathbf{x}, y), h\right) = 1$ when $\hat{y} \neq y$.

étiquette A higher-level fact or quantity of interest associated with a donnée. For example, if the donnée is an image, the label could indicate whether the image contains a cat or not. Synonyms for label, commonly used in specific domains, include "response variable," "output variable," and "target" [49], [50], [51].

Index

References

[1] W. Rudin, *Real and Complex Analysis*, 3rd ed. New York, NY, USA: McGraw-Hill, 1987.

[2] W. Rudin, *Principles of Mathematical Analysis*, 3rd ed. New York, NY, USA: McGraw-Hill, 1976.

[3] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 4th ed. Baltimore, MD, USA: The Johns Hopkins Univ. Press, 2013.

[4] G. H. Golub and C. F. Van Loan, "An analysis of the total least squares problem," *SIAM J. Numer. Anal.*, vol. 17, no. 6, pp. 883–893, Dec. 1980, doi: 10.1137/0717073.

[5] D. P. Bertsekas and J. N. Tsitsiklis, *Introduction to Probability*, 2nd ed. Belmont, MA, USA: Athena Scientific, 2008.

[6] A. Jung, *Machine Learning: The Basics*. Singapore, Singapore: Springer Nature, 2022.

[7] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. Cambridge, MA, USA: MIT Press, 2022. [Online]. Available: http://ebookcentral.proquest.com/lib/aalto-ebooks/detail.action?docID=6925615

[8] M. Sipser, *Introduction to the Theory of Computation*, 3rd ed. Andover, U.K.: Cengage Learning, 2013.

[9] R. Motwani and P. Raghavan, *Randomized Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 1995.

[10] R. G. Gallager, *Stochastic Processes: Theory for Applications.* Cambridge University Press, 2013.

[11] L. Richardson and M. Amundsen, *RESTful Web APIs.* Sebastopol, CA, USA: O'Reilly Media, 2013.

[12] M. P. Salinas et al., "A systematic review and meta-analysis of artificial intelligence versus clinicians for skin cancer diagnosis," *npj Digit. Med.*, vol. 7, no. 1, May 2024, Art. no. 125, doi: 10.1038/s41746-024-01103-x.

[13] E. L. Lehmann and G. Casella, *Theory of Point Estimation*, 2nd ed. New York, NY, USA: Springer-Verlag, 1998.

[14] G. F. Cooper, "The computational complexity of probabilistic inference using bayesian belief networks," *Artif. Intell.*, vol. 42, no. 2–3, pp. 393–405, Mar. 1990, doi: 10.1016/0004-3702(90)90060-D.

[15] R. M. Gray, *Probability, Random Processes, and Ergodic Properties*, 2nd ed. New York, NY, USA: Springer Science+Business Media, 2009.

[16] O. Chapelle, B. Schölkopf, and A. Zien, Eds. *Semi-Supervised Learning*. Cambridge, MA, USA: MIT Press, 2006.

[17] S. Boyd and L. Vandenberghe, *Convex Optimization.* Cambridge, U.K.: Cambridge Univ. Press, 2004.

[18] D. Sun, K.-C. Toh, and Y. Yuan, "Convex clustering: Model, theoretical guarantee and efficient algorithm," *J. Mach. Learn. Res.*, vol. 22, no. 9, pp. 1–32, Jan. 2021. [Online]. Available: http://jmlr.org/papers/v22/18-694.html

[19] K. Pelckmans, J. De Brabanter, J. A. K. Suykens, and B. De Moor, “Convex clustering shrinkage,” presented at the PASCAL Workshop Statist. Optim. Clustering Workshop, 2005.

[20] E. F. Codd, “A relational model of data for large shared data banks,” *Commun. ACM*, vol. 13, no. 6, pp. 377–387, Jun. 1970, doi: 10.1145/362384.362685.

[21] European Union, “Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation) (Text with EEA relevance),” L 119/1, May 4, 2016. [Online]. Available: https://eur-lex.europa.eu/eli/reg/2016/679/oj

[22] European Union, “Regulation (EU) 2018/1725 of the European Parliament and of the Council of 23 October 2018 on the protection of natural persons with regard to the processing of personal data by the Union institutions, bodies, offices and agencies and on the free movement of such data, and repealing Regulation (EC) No 45/2001 and Decision No 1247/2002/EC (Text with EEA relevance),” L 295/39, Nov. 21, 2018. [Online]. Available: https://eur-lex.europa.eu/eli/reg/2018/1725/oj

[23] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, 2nd ed. Hoboken, NJ, USA: Wiley, 2006.

[24] X. Liu, H. Li, G. Xu, Z. Chen, X. Huang, and R. Lu, “Privacy-enhanced federated learning against poisoning adversaries,” *IEEE*

Trans. Inf. Forensics Security, vol. 16, pp. 4574–4588, 2021, doi: 10.1109/TIFS.2021.3108434.

[25] J. Zhang, B. Chen, X. Cheng, H. T. T. Binh, and S. Yu, “PoisonGAN: Generative poisoning attacks against federated learning in edge computing systems,” *IEEE Internet Things J.*, vol. 8, no. 5, pp. 3310–3322, Mar. 2021, doi: 10.1109/JIOT.2020.3023126.

[26] A. Silberschatz, H. F. Korth, and S. Sudarshan, *Database System Concepts*, 7th ed. New York, NY, USA: McGraw-Hill Education, 2019. [Online]. Available: https://db-book.com/

[27] S. Abiteboul, R. Hull, and V. Vianu, *Foundations of Databases.* Reading, MA, USA: Addison-Wesley Publishing Company, 1995.

[28] S. Hoberman, *Data Modeling Made Simple: A Practical Guide for Business and IT Professionals*, 2nd ed. Basking Ridge, NJ, USA: Technics Publications, 2009.

[29] R. Ramakrishnan and J. Gehrke, *Database Management Systems*, 3rd ed. New York, NY, USA: McGraw-Hill, 2002.

[30] T. Gebru et al., “Datasheets for datasets,” *Commun. ACM*, vol. 64, no. 12, pp. 86–92, Nov. 2021, doi: 10.1145/3458723.

[31] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning.* Cambridge, MA, USA: MIT Press, 2016.

[32] G. Tel, *Introduction to Distributed Algorithms*, 2nd ed. Cambridge, U.K.: Cambridge Univ. Press, 2000.

[33] D. P. Bertsekas and J. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods.* Athena Scientific, 2015.

[34] S. M. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory.* Upper Saddle River, NJ, USA: Prentice Hall PTR, 1993.

[35] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* New York, NY, USA: Springer Science+Business Media, 2001.

[36] P. R. Halmos, *Measure Theory.* New York, NY, USA: Springer-Verlag, 1974.

[37] P. Billingsley, *Probability and Measure*, 3rd ed. New York, NY, USA: Wiley, 1995.

[38] C. M. Bishop, *Pattern Recognition and Machine Learning.* New York, NY, USA: Springer Science+Business Media, 2006.

[39] M. J. Wainwright and M. I. Jordan, "Graphical models, exponential families, and variational inference," *Found. Trends Mach. Learn.*, vol. 1, no. 1–2, pp. 1–305, Nov. 2008, doi: 10.1561/2200000001.

[40] N. Cesa-Bianchi and G. Lugosi, *Prediction, Learning, and Games.* New York, NY, USA: Cambridge Univ. Press, 2006.

[41] E. Hazan, "Introduction to online convex optimization," *Found. Trends Optim.*, vol. 2, no. 3–4, pp. 157–325, Aug. 2016, doi: 10.1561/2400000013.

[42] J. Colin, T. Fel, R. Cadène, and T. Serre, "What I cannot predict, I do not understand: A human-centered evaluation

framework for explainability methods," in *Adv. Neural Inf. Process. Syst.*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds. vol. 35, 2022, pp. 2832–2845. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/hash/13113e938f2957891c0c5e8df811dd01-Abstract-Conference.html

[43] L. Zhang, G. Karakasidis, A. Odnoblyudova, L. Dogruel, Y. Tian, and A. Jung, "Explainable empirical risk minimization," *Neural Comput. Appl.*, vol. 36, no. 8, pp. 3983–3996, Mar. 2024, doi: 10.1007/s00521-023-09269-3.

[44] A. Jung and P. H. J. Nardelli, "An information-theoretic approach to personalized explainable machine learning," *IEEE Signal Process. Lett.*, vol. 27, pp. 825–829, 2020, doi: 10.1109/LSP.2020.2993176.

[45] J. Chen, L. Song, M. J. Wainwright, and M. I. Jordan, "Learning to explain: An information-theoretic perspective on model interpretation," in *Proc. 35th Int. Conf. Mach. Learn.*, J. Dy and A. Krause, Eds. vol. 80, 2018, pp. 883–892. [Online]. Available: https://proceedings.mlr.press/v80/chen18j.html

[46] C. Rudin, "Stop explaining black box machine learning models for high-stakes decisions and use interpretable models instead," *Nature Mach. Intell.*, vol. 1, no. 5, pp. 206–215, May 2019, doi: 10.1038/s42256-019-0048-x.

[47] C. Molnar, *Interpretable Machine Learning: A Guide for Making*

Black Box Models Explainable, 3rd ed., 2025. [Online]. Available: https://christophm.github.io/interpretable-ml-book/

[48] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual explanations from deep networks via gradient-based localization," in *2017 IEEE Int. Conf. Comput. Vis.*, pp. 618–626, doi: 10.1109/ICCV.2017.74.

[49] D. N. Gujarati and D. C. Porter, *Basic Econometrics*, 5th ed. New York, NY, USA: McGraw-Hill/Irwin, 2009.

[50] Y. Dodge, Ed. *The Oxford Dictionary of Statistical Terms*. New York, NY, USA: Oxford Univ. Press, 2003.

[51] B. S. Everitt, *The Cambridge Dictionary of Statistics*, 2nd ed. Cambridge, U.K.: Cambridge Univ. Press, 2002.

[52] B. Schölkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.* Cambridge, MA, USA: MIT Press, 2002.

[53] M. T. Ribeiro, S. Singh, and C. Guestrin, "Why should i trust you?: Explaining the predictions of any classifier," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2016, pp. 1135–1144, doi: 10.1145/2939672.2939778.

[54] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int. Conf. Artif. Intell. Statist.*, A.

Singh and J. Zhu, Eds. vol. 54, 2017, pp. 1273–1282. [Online]. Available: https://proceedings.mlr.press/v54/mcmahan17a.html

[55] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *Proc. Mach. Learn. Syst.*, I. Dhillon, D. Papailiopoulos, and V. Sze, Eds. vol. 2, 2020. [Online]. Available: https://proceedings.mlsys.org/paper_files/paper/2020/hash/1f5fe83998a09396ebe6477d9475ba0c-Abstract.html

[56] Y. SarcheshmehPour, Y. Tian, L. Zhang, and A. Jung, "Flow-based clustering and spectral clustering: A comparison," in *2021 55th Asilomar Conf. Signals, Syst., Comput.*, M. B. Matthews, Ed. pp. 1292–1296, doi: 10.1109/IEEECONF53345.2021.9723162.

[57] A. Papoulis and S. U. Pillai, *Probability, Random Variables, and Stochastic Processes*, 4th ed. New York, NY, USA: McGraw-Hill Higher Education, 2002.

[58] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning.* Cambridge, MA, USA: MIT Press, 2006.

[59] S. Ross, *A First Course in Probability*, 9th ed. Boston, MA, USA: Pearson Education, 2014.

[60] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning: From Theory to Algorithms.* New York, NY, USA: Cambridge Univ. Press, 2014.

[61] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep

neural networks," *IEEE Trans. Evol. Comput.*, vol. 23, no. 5, pp. 828–841, Oct. 2019, doi: 10.1109/TEVC.2019.2890858.

[62] Y. SarcheshmehPour, Y. Tian, L. Zhang, and A. Jung, "Clustered federated learning via generalized total variation minimization," *IEEE Trans. Signal Process.*, vol. 71, pp. 4240–4256, 2023, doi: 10.1109/TSP.2023.3322848.

[63] N. Parikh and S. Boyd, "Proximal algorithms," *Found. Trends Optim.*, vol. 1, no. 3, pp. 127–239, Jan. 2014, doi: 10.1561/2400000003.

[64] R. T. Rockafellar, *Network Flows and Monotropic Optimization.* Belmont, MA, USA: Athena Scientific, 1998.

[65] U. von Luxburg, "A tutorial on spectral clustering," *Statist. Comput.*, vol. 17, no. 4, pp. 395–416, Dec. 2007, doi: 10.1007/s11222-007-9033-z.

[66] M. J. Wainwright, *High-Dimensional Statistics: A Non-Asymptotic Viewpoint.* Cambridge, U.K.: Cambridge Univ. Press, 2019.

[67] P. Bühlmann and S. van de Geer, *Statistics for High-Dimensional Data: Methods, Theory and Applications.* Berlin, Germany: Springer-Verlag, 2011.

[68] C. H. Lampert, "Kernel methods in computer vision," *Found. Trends Comput. Graph. Vis.*, vol. 4, no. 3, pp. 193–285, Sep. 2009, doi: 10.1561/0600000027.

[69] Q. Yang, Y. Liu, Y. Cheng, Y. Kang, T. Chen, and H. Yu, "Horizontal

federated learning," in *Federated Learning.* Cham, Switzerland: Springer Nature, 2020, ch. 4, pp. 49–67.

[70] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Adv. Neural Inf. Process. Syst.*, T. Dietterich, S. Becker, and Z. Ghahramani, Eds. vol. 14, 2001, pp. 849–856. [Online]. Available: https://papers.nips.cc/paper_files/paper/2001/hash/801272ee79cfde7fa5960571fee36b9b-Abstract.html

[71] A. Vaswani et al., "Attention is all you need," in *Adv. Neural Inf. Process. Syst.*, I. Guyon, U. von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. vol. 30, 2017, pp. 5998–6008. [Online]. Available: https://papers.nips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

[72] R. Caruana, "Multitask learning," *Mach. Learn.*, vol. 28, pp. 41–75, Jul. 1997, doi: 10.1023/A:1007379606734.

[73] A. Jung, G. Hannak, and N. Goertz, "Graphical lasso based model selection for time series," *IEEE Signal Process. Lett.*, vol. 22, no. 10, pp. 1781–1785, Oct. 2015, doi: 10.1109/LSP.2015.2425434.

[74] A. Jung, "Learning the conditional independence structure of stationary time series: A multitask learning approach," *IEEE Trans. Signal Process.*, vol. 63, no. 21, Nov. 2015, doi: 10.1109/TSP.2015.2460219.

[75] T. Hastie, R. Tibshirani, and M. Wainwright, *Statistical Learning with Sparsity: The Lasso and Generalizations.* Boca Raton, FL, USA: CRC Press, 2015.

[76] K. Abayomi, A. Gelman, and M. Levy, “Diagnostics for multivariate imputations,” *J. Roy. Statist. Soc.: Ser. C (Appl. Statist.)*, vol. 57, no. 3, pp. 273–291, Jun. 2008, doi: 10.1111/j.1467-9876.2007.00613.x.

[77] S. Bubeck and N. Cesa-Bianchi, “Regret analysis of stochastic and non-stochastic multi-armed bandit problems,” *Found. Trends Mach. Learn.*, vol. 5, no. 1, pp. 1–122, Dec. 2012, doi: 10.1561/2200000024.

[78] A. Lapidoth, *A Foundation in Digital Communication.* Cambridge, U.K.: Cambridge Univ. Press, 2009.

[79] A. Jung, “Networked exponential families for big data over networks,” *IEEE Access*, vol. 8, pp. 202 897–202 909, Nov. 2020, doi: 10.1109/ACCESS.2020.3033817.

[80] Y. Nesterov, *Introductory Lectures on Convex Optimization: A Basic Course.* Boston, MA, USA: Kluwer Academic Publishers, 2004.

[81] R. A. Horn and C. R. Johnson, *Matrix Analysis*, 2nd ed. New York, NY, USA: Cambridge Univ. Press, 2013.

[82] A. Rakhlin, O. Shamir, and K. Sridharan, “Making gradient descent optimal for strongly convex stochastic optimization,” in *Proc. 29th Int. Conf. Mach. Learn.*, J. Langford and J. Pineau, Eds. 2012, pp. 449–456. [Online]. Available: https://icml.cc/Conferences/2012/papers/261.pdf

[83] P. J. Brockwell and R. A. Davis, *Time Series: Theory and Methods*, 2nd ed. New York, NY, USA: Springer-Verlag, 1991.

[84] M. Kearns and M. Li, “Learning in the presence of malicious errors,” *SIAM J. Comput.*, vol. 22, no. 4, pp. 807–837, Aug. 1993, doi: 10.1137/0222052.

[85] G. Lugosi and S. Mendelson, “Robust multivariate mean estimation: The optimality of trimmed mean,” *Ann. Statist.*, vol. 49, no. 1, pp. 393–410, Feb. 2021, doi: 10.1214/20-AOS1961.

[86] A. Makhdoumi, S. Salamatian, N. Fawaz, and M. Médard, “From the information bottleneck to the privacy funnel,” in *2014 IEEE Inf. Theory Workshop*, pp. 501–505, doi: 10.1109/ITW.2014.6970882.

[87] A. Ünsal and M. Önen, “Information-theoretic approaches to differential privacy,” *ACM Comput. Surv.*, vol. 56, no. 3, Oct. 2023, Art. no. 76, doi: 10.1145/3604904.

[88] O. Kallenberg, *Foundations of Modern Probability.* New York, NY, USA: Springer-Verlag, 1997.

[89] L. Condat, “A primal–dual splitting method for convex optimization involving lipschitzian, proximable and linear composite terms,” *J. Optim. Theory Appl.*, vol. 158, no. 2, pp. 460–479, Aug. 2013, doi: 10.1007/s10957-012-0245-9.

[90] H. H. Bauschke and P. L. Combettes, *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*, 2nd ed. New York, NY, USA: Springer Science+Business Media, 2017.

[91] S. Shalev-Shwartz and A. Tewari, “Stochastic methods for l1 regularized loss minimization,” in *Proceedings of the 26th Annual International*

Conference on Machine Learning, ser. ICML ’09, New York, NY, USA, 2009, pp. 929–936.

[92] I. Csiszar, “Generalized cutoff rates and Renyi’s information measures,” *IEEE Trans. Inf. Theory*, vol. 41, no. 1, pp. 26–34, Jan. 1995, doi: 10.1109/18.370121.

[93] S. Bubeck, “Convex optimization: Algorithms and complexity,” *Found. Trends Mach. Learn.*, vol. 8, no. 3–4, pp. 231–357, Nov. 2015, 10.1561/2200000050.

[94] D. P. Bertsekas, *Convex Optimization Algorithms.* Belmont, MA, USA: Athena Scientific, 2015.

[95] L. Cohen, *Time-Frequency Analysis.* Upper Saddle River, NJ, USA: Prentice Hall PTR, 1995.

[96] J. Li, L. Han, X. Li, J. Zhu, B. Yuan, and Z. Gou, “An evaluation of deep neural network models for music classification using spectrograms,” *Multimedia Tools Appl.*, vol. 81, no. 4, pp. 4621–4647, Feb. 2022, doi: 10.1007/s11042-020-10465-9.

[97] B. Boashash, Ed. *Time Frequency Signal Analysis and Processing: A Comprehensive Reference*. Oxford, U.K.: Elsevier, 2003.

[98] S. Mallat, *A Wavelet Tour of Signal Processing: The Sparse Way*, 3rd ed. Burlington, MA, USA: Academic, 2009.

[99] E. Abbe, “Community detection and stochastic block models: Recent

developments," *J. Mach. Learn. Res.*, vol. 18, no. 177, pp. 1–86, Apr. 2018. [Online]. Available: http://jmlr.org/papers/v18/16-480.html

[100] L. Bottou, "On-line learning and stochastic approximations," in *On-Line Learning in Neural Networks*, D. Saad, Ed. New York, NY, USA: Cambridge Univ. Press, 1999, ch. 2, pp. 9–42.

[101] D. P. Bertsekas, A. Nedic, and A. E. Ozdaglar, *Convex Analysis and Optimization.* Belmont, MA, USA: Athena Scientific, 2003.

[102] D. P. Bertsekas, *Nonlinear Programming*, 2nd ed. Belmont, MA, USA: Athena Scientific, 1999.

[103] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods.* New York, NY, USA: Cambridge Univ. Press, 2000.

[104] High-Level Expert Group on Artificial Intelligence, "Ethics guidelines for trustworthy AI," European Commission, Apr. 8, 2019. [Online]. Available: https://digital-strategy.ec.europa.eu/en/library/ethics-guidelines-trustworthy-ai

[105] C. Gallese, "The AI act proposal: A new right to technical interpretability?," *SSRN Electron. J.*, Feb. 2023. [Online]. Available: https://ssrn.com/abstract=4398206

[106] M. Mitchell et al., "Model cards for model reporting," in *Proc. Conf. Fairness, Accountability, Transparency*, 2019, pp. 220–229, doi: 10.1145/3287560.3287596.

[107] K. Shahriari and M. Shahriari, “IEEE standard review — Ethically aligned design: A vision for prioritizing human wellbeing with artificial intelligence and autonomous systems,” in *2017 IEEE Canada International Humanitarian Technology Conference*, pp. 197–201, doi: 10.1109/IHTC.2017.8058187.

[108] D. Pfau and A. Jung, “Engineering trustworthy AI: A developer guide for empirical risk minimization,” Nov. 2024. [Online]. Available: https://arxiv.org/abs/2410.19361

[109] High-Level Expert Group on Artificial Intelligence, “The assessment list for trustworthy artificial intelligence (ALTAI): For self assessment,” European Commission, Jul. 17, 2020. [Online]. Available: https://digital-strategy.ec.europa.eu/en/library/assessment-list-trustworthy-artificial-intelligence-altai-self-assessment

[110] Q. Yang, Y. Liu, Y. Cheng, Y. Kang, T. Chen, and H. Yu, “Vertical federated learning,” in *Federated Learning*. Cham, Switzerland: Springer Nature, 2020, ch. 5, pp. 69–81.