# The A”alto Dictionary of Machine Learning

Dipl.-Ing. Dr.techn. Alexander Jung
*

November 2, 2024

**Abstract**

This dictionary of machine learning evolved during the design and implementation of the courses 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** offered to Finnish university students via **Aalto University** `https://www.aalto.fi/en`, to Finnish adult learners via **The Finnish Institute of Technology (FITech)** `fitech.io` and to international students via the **European University Alliance Unite!** `https://www.aalto.fi/en/unite`.

*please cite as: A. Jung, “The Aalto Dictionary of Machine Learning,” Aalto University, 2024.

# Lists of Symbols

## Sets and Functions

| | |
|---|---|
| $a \in \mathcal{A}$ | This statement indicates that the object $a$ is an element of the set $\mathcal{A}$. |
| $a := b$ | This statement defines $a$ to be shorthand for $b$. |
| $\lvert\mathcal{A}\rvert$ | The cardinality (number of elements) of a finite set $\mathcal{A}$. |
| $\mathcal{A} \subseteq \mathcal{B}$ | $\mathcal{A}$ is a subset of $\mathcal{B}$. |
| $\mathcal{A} \subset \mathcal{B}$ | $\mathcal{A}$ is a strict subset of $\mathcal{B}$. |
| $\mathbb{N}$ | The natural numbers $1, 2, \ldots$. |
| $\mathbb{R}$ | The real numbers $x$ [1]. |
| $\mathbb{R}_+$ | The non-negative real numbers $x \geq 0$. |
| $\mathbb{R}_{++}$ | The positive real numbers $x > 0$. |

| | |
|---|---|
| $\{0, 1\}$ | The set that consists of the two real numbers 0 and 1. |
| $[0, 1]$ | The closed interval of real numbers $x$ with $0 \leq x \leq 1$. |
| $\underset{\mathbf{w}}{\operatorname{argmin}} f(\mathbf{w})$ | The set of minimizers for a real-valued function $f(\mathbf{w})$. |
| $\mathbb{S}^{(n)}$ | The set of unit-norm vectors in $\mathbb{R}^{n+1}$. |
| $\log a$ | The logarithm of the positive number $a \in \mathbb{R}_{++}$. |
| $h(\cdot): \mathcal{A} \to \mathcal{B} : a \mapsto h(a)$ | A function (map) that accepts any element $a \in \mathcal{A}$ from a set $\mathcal{A}$ as input and delivers a well-defined element $h(a) \in \mathcal{B}$ of a set $\mathcal{B}$. The set $\mathcal{A}$ is the domain of the function $h$ and the set $\mathcal{B}$ is the codomain of $h$. ML aims at finding (or learning) a function $h$ (*hypothesis*) that reads in the features $\mathbf{x}$ of a data point and delivers a prediction $h(\mathbf{x})$ for its label $y$. |
| $\nabla f(\mathbf{w})$ | The gradient of a differentiable real-valued function $f : \mathbb{R}^d \to \mathbb{R}$ is the vector $\nabla f(\mathbf{w}) = \left(\frac{\partial f}{\partial w_1}, \ldots, \frac{\partial f}{\partial w_d}\right)^T \in \mathbb{R}^d$ [2, Ch. 9]. |

## Matrices and Vectors

| | |
|---|---|
| $\mathbf{I}_{l \times d}$ | A generalized identity matrix with $l$ rows and $d$ columns. The entries of $\mathbf{I}_{l \times d} \in \mathbb{R}^{l \times d}$ are equal to 1 along the main diagonal and equal to 0 otherwise. |
| $\mathbf{I}_d, \mathbf{I}$ | A square identity matrix of size $d \times d$. If the size is clear from the context, we drop the subscript. |
| $\mathbb{R}^d$ | The set of vectors $\mathbf{x} = \big(x_1, \ldots, x_d\big)^T$ consisting of $d$ real-valued entries $x_1, \ldots, x_d \in \mathbb{R}$. |
| $\mathbf{x} = \big(x_1, \ldots, x_d\big)^T$ | A vector of length $d$ with its $j$th entry being $x_j$. |
| $\|\mathbf{x}\|_2$ | The Euclidean (or $\ell_2$) norm of the vector $\mathbf{x} = \big(x_1, \ldots, x_d\big)^T \in \mathbb{R}^d$ given as $\|\mathbf{x}\|_2 := \sqrt{\sum_{j=1}^{d} x_j^2}$. |
| $\|\mathbf{x}\|$ | Some norm of the vector $\mathbf{x} \in \mathbb{R}^d$ [3]. Unless specified otherwise, we mean the Euclidean norm $\|\mathbf{x}\|_2$. |
| $\mathbf{x}^T$ | The transpose of a vector $\mathbf{x}$ that is considered a single column matrix. The transpose is a single-row matrix $\big(x_1, \ldots, x_d\big)$. |
| $\mathbf{X}^T$ | The transpose of a matrix $\mathbf{X} \in \mathbb{R}^{m \times d}$. A square real-valued matrix $\mathbf{X} \in \mathbb{R}^{m \times m}$ is called symmetric if $\mathbf{X} = \mathbf{X}^T$. |
| $\mathbf{0} = \big(0, \ldots, 0\big)^T$ | The vector in $\mathbb{R}^d$ with each entry equal to zero. |
| $\mathbf{1} = \big(1, \ldots, 1\big)^T$ | The vector in $\mathbb{R}^d$ with each entry equal to one. |

| | |
|---|---|
| $\left(\mathbf{v}^T, \mathbf{w}^T\right)^T$ | The vector of length $d + d'$ obtained by concatenating the entries of vector $\mathbf{v} \in \mathbb{R}^d$ with the entries of $\mathbf{w} \in \mathbb{R}^{d'}$. |
| $\mathrm{span}\{\mathbf{B}\}$ | The span of a matrix $\mathbf{B} \in \mathbb{R}^{a \times b}$, which is the subspace of all linear combinations of columns of $\mathbf{B}$, $\mathrm{span}\{\mathbf{B}\} = \left\{\mathbf{B}\mathbf{a} : \mathbf{a} \in \mathbb{R}^b\right\} \subseteq \mathbb{R}^a$. |
| $\mathbb{S}_+^d$ | The set of all positive semi-definite (psd) matrices of size $d \times d$. |
| $\det\left(\mathbf{C}\right)$ | The determinant of the matrix $\mathbf{C}$. |
| $\mathbf{A} \otimes \mathbf{B}$ | The Kronecker product of $\mathbf{A}$ and $\mathbf{B}$ [4]. |

# Probability Theory

| | |
|---|---|
| $\mathbb{E}_p\{f(\mathbf{z})\}$ | The expectation of a function $f(\mathbf{z})$ of a RV $\mathbf{z}$ whose probability distribution is $p(\mathbf{z})$. If the probability distribution is clear from context we just write $\mathbb{E}\{f(\mathbf{z})\}$. |
| $p(\mathbf{x}, y)$ | A (joint) probability distribution of a RV whose realizations are data points with features $\mathbf{x}$ and label $y$. |
| $p(\mathbf{x}\|y)$ | A conditional probability distribution of a RV $\mathbf{x}$ given the value of another RV $y$ [5, Sec. 3.5]. |
| $p(\mathbf{x}; \mathbf{w})$ | A parametrized probability distribution of a RV $\mathbf{x}$. The probability distribution depends on a parameter vector $\mathbf{w}$. For example, $p(\mathbf{x}; \mathbf{w})$ could be a multivariate normal distribution with the parameter vector $\mathbf{w}$ given by the entries of the mean vector $\mathbb{E}\{\mathbf{x}\}$ and the covariance matrix $\mathbb{E}\Big\{\big(\mathbf{x} - \mathbb{E}\{\mathbf{x}\}\big)\big(\mathbf{x} - \mathbb{E}\{\mathbf{x}\}\big)^T\Big\}$. |
| $\mathcal{N}(\mu, \sigma^2)$ | The probability distribution of a Gaussian RV $x \in \mathbb{R}$ with mean (or expectation) $\mu = \mathbb{E}\{x\}$ and variance $\sigma^2 = \mathbb{E}\big\{(x - \mu)^2\big\}$. |
| $\mathcal{N}(\boldsymbol{\mu}, \mathbf{C})$ | The multivariate normal distribution of a vector-valued Gaussian RV $\mathbf{x} \in \mathbb{R}^d$ with mean (or expectation) $\boldsymbol{\mu} = \mathbb{E}\{\mathbf{x}\}$ and covariance matrix $\mathbf{C} = \mathbb{E}\big\{\big(\mathbf{x} - \boldsymbol{\mu}\big)\big(\mathbf{x} - \boldsymbol{\mu}\big)^T\big\}$. |

## Machine Learning

| | |
|---|---|
| $r$ | An index $r = 1, 2, \ldots,$ that enumerates data points. |
| $m$ | The number of data points in (the size of) a dataset. |
| $\mathcal{D}$ | A dataset $\mathcal{D} = \{\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(m)}\}$ is a list of individual data points $\mathbf{z}^{(r)}$, for $r = 1, \ldots, m$. |
| $d$ | Number of features that characterize a data point. |
| $x_j$ | The $j$th feature of a data point. The first feature of a given data point is denoted $x_1$, the second feature $x_2$ and so on. |
| $\mathbf{x}$ | The feature vector $\mathbf{x} = \big(x_1, \ldots, x_d\big)^T$ of a data point whose entries are the individual features of a data point. |
| $\mathcal{X}$ | The feature space $\mathcal{X}$ is the set of all possible values that the features $\mathbf{x}$ of a data point can take on. |
| $\mathbf{z}$ | Beside the symbol $\mathbf{x}$, we sometimes use $\mathbf{z}$ as another symbol to denote a vector whose entries are features of a data point. We need two different symbols to distinguish between *raw* and learnt features [6, Ch. 9]. |
| $\mathbf{x}^{(r)}$ | The feature vector of the $r$th data point within a dataset. |
| $x_j^{(r)}$ | The $j$th feature of the $r$th data point within a dataset. |

| | |
|---|---|
| $\mathcal{B}$ | A mini-batch (subset) of randomly chosen data points. |
| $B$ | The size of (the number of data points in) a mini-batch. |
| $y$ | The label (quantity of interest) of a data point. |
| $y^{(r)}$ | The label of the $r$th data point. |
| $\left(\mathbf{x}^{(r)}, y^{(r)}\right)$ | The features and label of the $r$th data point. |
| $\mathcal{Y}$ | The label space $\mathcal{Y}$ of a ML method consists of all potential label values that a data point can have. We often use label spaces that are larger than the set of different label values arising in a give dataset (e.g., a training set). We refer to ML problems (methods) using a numeric label space, such as $\mathcal{Y} = \mathbb{R}$ or $\mathcal{Y} = \mathbb{R}^3$, as regression problems (methods). ML problems (methods) that use a discrete label space, such as $\mathcal{Y} = \{0, 1\}$ or $\mathcal{Y} = \{\ \mathit{cat}\ , \mathit{dog}, \mathit{mouse}\}$ are referred to as classification problems (methods). |
| $\eta$ | Learning rate (step-size) used by gradient-based methods. |
| $h(\cdot)$ | A hypothesis map that reads in features $\mathbf{x}$ of a data point and delivers a prediction $\hat{y} = h(\mathbf{x})$ for its label $y$. |
| $\mathcal{Y}^{\mathcal{X}}$ | Given two sets $\mathcal{X}$ and $\mathcal{Y}$, we denote by $\mathcal{Y}^{\mathcal{X}}$ the set of all possible hypothesis maps $h : \mathcal{X} \to \mathcal{Y}$. |

| | |
|---|---|
| $\mathcal{H}$ | A hypothesis space or model used by a ML method. The hypothesis space consists of different hypothesis maps $h : \mathcal{X} \rightarrow \mathcal{Y}$ between which the ML method has to choose . |
| $d_{\mathrm{eff}}\left(\mathcal{H}\right)$ | The effective dimension of a hypothesis space $\mathcal{H}$. |
| $B^2$ | The squared bias of a learnt hypothesis $\hat{h}$ delivered by a ML algorithm that is fed with data points which are modelled as realizations of RVs. If data is modelled as realizations of RVs, also the delivered hypothesis $\hat{h}$ is the realization of a RV. |
| $V$ | The variance of the (parameters of the) hypothesis delivered by a ML algorithm. If the input data for this algorithm is interpreted as realizations of RVs, so is the delivered hypothesis a realization of a RV. |
| $L\left((\mathbf{x}, y), h\right)$ | The loss incurred by predicting the label $y$ of a data point using the prediction $\hat{y} = h(\mathbf{x})$. The prediction $\hat{y}$ is obtained from evaluating the hypothesis $h \in \mathcal{H}$ for the feature vector $\mathbf{x}$ of the data point. |
| $E_v$ | The validation error of a hypothesis $h$, which is its average loss incurred over a validation set. |
| $\widehat{L}\big(h \vert \mathcal{D}\big)$ | The empirical risk or average loss incurred by the hypothesis $h$ on a dataset $\mathcal{D}$. |

| | |
|---|---|
| $E_t$ | The training error of a hypothesis $h$, which is its average loss incurred over a training set. |
| $t$ | A discrete-time index $t = 0, 1, \ldots$ used to enumerate sequential events (or time instants). |
| $t$ | An index that enumerates learning tasks within a multi-task learning problem. |
| $\alpha$ | A regularization parameter that controls the amount of regularization. |
| $\lambda_j\big(\mathbf{Q}\big)$ | The $j$th eigenvalue (sorted either ascending or descending) of a psd matrix $\mathbf{Q}$. We also use the shorthand $\lambda_j$ if the corresponding matrix is clear from context. |
| $\sigma(\cdot)$ | The activation function used by an artificial neuron within an artificial neural network (ANN). |
| $\mathcal{R}_{\hat{y}}$ | A decision region within a feature space. |
| $\mathbf{w}$ | A parameter vector $\mathbf{w} = \big(w_1, \ldots, w_d\big)^T$ of a model, e.g, the weights of a linear model or in a ANN. |
| $h^{(\mathbf{w})}(\cdot)$ | A hypothesis map that involves tunable model parameters $w_1, \ldots, w_d$, stacked into the vector $\mathbf{w} = \big(w_1, \ldots, w_d\big)^T$. |
| $\phi(\cdot)$ | A feature map $\phi : \mathcal{X} \rightarrow \mathcal{X}' : \mathbf{x} \mapsto \mathbf{x}' := \phi\big(\mathbf{x}\big) \in \mathcal{X}'$. |

## Federated Learning

| | |
|---|---|
| $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ | Federated network whose nodes $i \in \mathcal{V}$ carry local datasets and local models. |
| $i \in \mathcal{V}$ | A node in the federated network that represents a local dataset and a corresponding local model. It might also be useful to think of node $i$ as a small computer that can collect data and execute computations to train ML models. |
| $\mathcal{G}^{(\mathcal{C})}$ | The induced sub-graph 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 $\mathcal{G}^{(\mathcal{C})}$. |
| $\mathcal{D}^{(i)}$ | The local dataset $\mathcal{D}^{(i)}$ at node $i \in \mathcal{V}$ of an federated network. |
| $m_i$ | The number of data points (sample size) contained in the local dataset $\mathcal{D}^{(i)}$ at node $i \in \mathcal{V}$. |
| $\mathcal{N}^{(i)}$ | The neighbourhood of the node $i$ in an federated network. |
| $\mathbf{x}^{(i,r)}$ | The features of the $r$-th data point in the local dataset $\mathcal{D}^{(i)}$. |
| $y^{(i,r)}$ | The label of the $r$-th data point in the local dataset $\mathcal{D}^{(i)}$. |

| | |
|---|---|
| $L^{(\mathrm{d})}\left(\mathbf{x}, h\big(\mathbf{x}\big), h'\big(\mathbf{x}\big)\right)$ | The loss incurred by a hypothesis $h'$ on a data point with features $\mathbf{x}$ and label $h\big(\mathbf{x}\big)$ that is obtained from another hypothesis. |
| $\mathrm{stack}\big\{\mathbf{w}^{(i)}\big\}_{i=1}^{n}$ | The vector $\left(\left(\mathbf{w}^{(1)}\right)^{T}, \ldots, \left(\mathbf{w}^{(n)}\right)^{T}\right)^{T} \in \mathbb{R}^{dn}$ that is obtained by vertically stacking the local model parameters $\mathbf{w}^{(i)} \in \mathbb{R}^{d}$. |

# Glossary

**$k$-fold cross-validation ($k$-fold CV)** $k$-fold cross-validation is a method for learning and validating a hypothesis using a given dataset. This method divides the dataset 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 data point of a dataset to precisely one of $k$ different clusters. The method alternates between updating the cluster assignments (to the cluster with nearest cluster mean) and, given the updated cluster assignments, re-calculating the cluster means [6, Ch. 8].

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

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

**Application Programming Interface (API)** An application programming interface (API) is a precise specification of the services and resources offered by software or hardware implementing that API.

**artificial intelligence** Artificial intelligence aims to develop systems that behave rational in the sense of maximizing a long-term reward. In MLterms, these systems train a model to predict optimal actions in order to minimize a loss computed from reward signals.

**artificial neural network** An artificial neural network is a graphical (signal-flow) representation of a map from features of a data point at its input to a prediction for the label as its output.

**autoencoder** An autoencoder is a ML method that jointly 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 $\mathbf{x} - h^*\big(h\big(\mathbf{x}\big)\big)$.

**backdoor** A backdoor attack refers to the intentional manipulation of the training process underlying a ML 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 learnt hypothesis $\hat{h}$ towards specific predictions for a certain range of feature values. This range of feature values serves as a key (or trigger) to unlock a *backdoor* in the sense of delivering anomalous predictions. The key $\mathbf{x}$ and the corresponding anomolous prediction $\hat{h}(\mathbf{x})$ is only known to the attacker.

**bagging** Bagging (or *bootstrap aggregation*) is a generic technique to improve (the robustness of) a given ML method. The idea is to use the bootstrap to generate perturbed copies of a given dataset and then to learn a separate hypothesis for each copy. The resulting set of hypotheses is then used to predict the label of a data point by combining or aggregating the individual predictions of each hypothesis. For hypothesis maps delivering numeric label values, this aggregation could be implemented by computing the average of individual predictions.

**baseline** A reference value or benchmark for the average loss incurred by a hypothesis when applied to the data points generated in a specific ML application. Such a reference value might be obtained from human performance (e.g., the error rate of dermatologists diagnosing cancer from visual inspection of skin areas) or other ML methods [6, Ch. 6].

**batch** A set of data points randomly selected from (typically very large) dataset.

**Bayes' estimator** A hypothesis $h$ whose Bayes risk coincides with the Bayes risk [7].

**Bayes' risk** Consider data points being realizations of independent and identically distributed (i.i.d.) RVs with a common probability distribution. The Bayes' risk is the minimum possible risk that can be achieved by any hypothesis $h$ out of a model $\mathcal{H}$. The hypothesis achieving the minimum risk is referred to as a Bayes estimator [7].

**bias** Consider a ML method that trains a parametrized hypothesis space $\mathcal{H}$ with model parameters $\mathbf{w} \in \mathbb{R}^d$ using the dataset $\mathcal{D} = \left\{ \left(\mathbf{x}^{(r)}, y^{(r)}\right) \right\}_{r=1}^{m}$. To analyze the properties of the ML method, we typically interpret the data points as realizations of i.i.d. RVs,

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

We can then interpret the ML 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 $B^2 := \left\|\mathbb{E}\{\widehat{\mathbf{w}}\} - \overline{\mathbf{w}}\right\|_2^2$.

**bootstrap** For the analysis of ML methods it is often useful to interpret a given set of data points $\mathcal{D} = \left\{\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(m)}\right\}$ as realizations of i.i.d. RVs with a common probability distribution $p(\mathbf{z})$. 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 probability distribution $p(\mathbf{z})$.

**classification** Classification is the task of determining a discrete-valued label $y$ of a data point 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 a category to which the corresponding data point belongs to. Some classification problems involve a countably infinite label space.

**classifier** A classifier is a hypothesis (map) $h(\mathbf{x})$ used to predict a label taking values from a finite label space. We might use the function value

$h(\mathbf{x})$ itself as a prediction $\hat{y}$ for the label. However, it is customary to use a map $h(\cdot)$ that delivers a numeric quantity. The prediction is then obtained by a simple thresholding step. For example, in a binary classification problem with $\mathcal{Y} \in \{-1, 1\}$, we might use real-valued hypothesis map $h(\mathbf{x}) \in \mathbb{R}$ as classifier. A prediction $\hat{y}$ 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 label value $a \in \mathcal{Y}$.

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

**clustered federated learning (CFL)** Clustered federated learning (FL) (CFL) assumes that local datasets form clusters. The local datasets belonging to the same cluster have similar statistical properties. CFL pools local datasets in the same cluster to obtain a training set for training a cluster-specific model. GTV minimization (GTVMin) implements this pooling implicitly by forcing the local model parameters to be approximately identical over well-connected subsets of the federated network.

**clustering** Clustering methods decompose a given set of data points into few subsets, which are referred to as clusters. Each cluster consists of data points that are more similar to each other than to data points outside the cluster. Different clustering methods use different measures for the similarity between data points and different forms of cluster representations. The clustering method $k$-means uses the average feature 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 data points in a dataset form a (small) number of groups or clusters. Data points in the same cluster are more similar with each other than with those outside the cluster [8]. We obtain different clustering methods by using different notions of similarity between data points.

**computational aspects** By computational aspects of a ML method, we mainly refer to the computational resources required for its implementation. For example, if a ML method uses iterative optimization techniques to solve ERM, then its computational aspects include (i) how many arithmetic operations are needed to implement a single iteration (gradient step) and (ii) 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 psd matrix $\mathbf{Q}$ is the ratio $\lambda_{\max}/\lambda_{\min}$ between the largest $\lambda_{\max}$ and the smallest $\lambda_{\min}$

eigenvalue of $\mathbf{Q}$. The condition number is useful for the analysis of ML methods. In particular, 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 features of data points such that $\mathbf{Q}$ has a condition number close to 1.

**confusion matrix** Consider data points characterized by features $\mathbf{x}$ and label $y$ having values from the finite label space $\mathcal{Y} = \{1, \ldots, k\}$. The confusion matrix is $k \times k$ matrix with rows representing different values $c$ of the true label of a data point. 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 data points with label $y = c$ and the prediction $\hat{y} = c'$ assigned by the hypothesis $h$.

**connected graph** A undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is connected if it does not contain a (non-empty) subset $\mathcal{V}' \subset \mathcal{V}$ with no edges leaving $\mathcal{V}'$.

**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 of that set. We define a function as convex if its epigraph is a convex set [9].

**covariance matrix** The covariance matrix of a RV $\mathbf{x} \in \mathbb{R}^d$ is defined as $\mathbb{E}\Big\{ \big(\mathbf{x} - \mathbb{E}\{\mathbf{x}\}\big)\big(\mathbf{x} - \mathbb{E}\{\mathbf{x}\}\big)^T \Big\}$.

**data** See dataset.

**data augmentation** Data augmentation methods add synthetic data points to an existing set of data points. These synthetic data points might be obtained by perturbations (adding noise) or transformations (rotations of images) of the original data points. These perturbations and transformations are such that the resulting synthetic data points should still have the same label. 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.

**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 [10, Article 5(1)(c)] [11].

**data point** A data point is any object that conveys information [12]. Data points might be students, radio signals, trees, forests, images, RVs, real numbers or proteins. We characterize data points using two types of properties. One type of property is referred to as a feature. Features 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 labels. The label of a data point represents some higher-level fact (or quantity of interest). In contrast to features, determining the label of a data point typically requires human experts (domain experts). Roughly speaking, ML aims to predict the label of a data point based solely on its features.

**data poisoning** Data poisoning refers to the intentional manipulation (or fabrication) of data points to steer the training of a ML model [13, 14]. The protection against data poisoning is particularly important in distributed ML applications where datasets are de-centralized.

**dataset** With a slight abuse of notation we use the term *dataset* or *set of data points* to refer to an indexed list of data points $\mathbf{z}^{(1)}, \mathbf{z}^{(2)}, \ldots$. Thus, there is a first data point $\mathbf{z}^{(1)}$, a second data point $\mathbf{z}^{(2)}$ and so on. Strictly speaking, a dataset is a list and not a set [15]. The implementation of ML methods requires a more detailed specification of a dataset, e.g., in the form of a relational database schema [16].

**decision region** Consider a hypothesis map $h$ that reads in a feature 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 neighbourhood $\{\mathbf{x}' : \|\mathbf{x} - \mathbf{x}'\| \leq \varepsilon\}$, for any $\varepsilon > 0$, contains at least two vectors with different function values.

**decision region** Consider a hypothesis map $h$ that delivers values from a finite set $\mathcal{Y}$. We refer to the set of features $\mathbf{x} \in \mathcal{X}$ that result in the same output $h(\mathbf{x}) = a$ as a decision region of the hypothesis $h$.

**decision tree** A decision tree is a flow-chart-like representation of a hypothesis map $h$. More formally, a decision tree is a directed graph which reads in the feature vector $\mathbf{x}$ of a data point at its root node. The root node then forwards the data point to one of its children nodes based on some elementary test on the features $\mathbf{x}$. If the receiving children node is not a leaf node, i.e., it has itself children nodes, it represents another test. Based on the test result, the data point is further pushed to one of its descendants. This testing and forwarding of the data point is continued until the data point ends up in a leaf node (having no children nodes). Each leaf node corresponds to a decision region, a subset of the feature space mapped to the same output $h(\mathbf{x})$.

**deep net** We refer to an ANN with a (relatively) large number of hidden layers as a deep ANN or *deep net*. Deep nets are used to represent the hypothesis spaces of deep learning methods [17].

**degree of belonging** A number that indicates the extent by which a data point belongs to a cluster [6, Ch. 8]. The degree of belonging can be interpreted as a soft cluster assignment. Soft clustering methods typically represent the degree of belonging by a real number in the interval $[0, 1]$. The extreme case when the degree of belongings only take on values 0 or 1 correspond to hard clustering.

**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 data points

**density-based spatial clustering of applications with noise** A clustering algorithm for data points that are characterized by numeric feature vectors. Similar to $k$-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 data points. In particular, DBSCAN considers two data points as similar if they are *connected* via a sequence (path) of close-by intermediate data points. Thus, DBSCAN might consider two data points as similar (and therefore belonging to the same cluster) even if their feature vectors have a large Euclidean distance.

**differentiable** A function $f : \mathbb{R}^d \rightarrow \mathbb{R}$ is differentiable if it has a gradient $\nabla f(\mathbf{x})$ everywhere (for every $\mathbf{x} \in \mathbb{R}^d$) [2].

**differential privacy** Consider some ML method $\mathcal{A}$ that reads in a $\mathcal{D}$ and delivers some output $\mathcal{A}(\mathcal{D})$. The output could be the learnt model parameters $\widehat{\mathbf{w}}$ or the prediction $\hat{h}(\mathbf{x})$ obtained for a specific data point with features $\mathbf{x}$. Differential privacy is a precise measure of privacy leakage incurred by revealing the output $\mathcal{A}(\mathcal{D})$. Roughly speaking, the algorithm is differentially private if the probability distribution of $\mathcal{A}(\mathcal{D})$ does not change too much if a sensitive attribute of one data point in $\mathcal{D}$ is changed.

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

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

**edge weight** Each edge $\{i, i'\}$ of an federated network is assigned a non-negative edge weight $A_{i,i'} \geq 0$. A zero 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 parameters of the model. 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 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** 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 $\mathbf{V} = \left(\mathbf{v}^{(1)}, \ldots, \mathbf{v}^{(d)}\right)$ are the eigenvectors of the matrix $\mathbf{V}$. The diagonal matrix $\mathbf{\Lambda} = \text{diag}\big\{\lambda_1, \ldots, \lambda_d\big\}$ contains the eigenvalues $\lambda_j$ 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 $\lambda$.

**empirical risk** The empirical risk $\widehat{L}\big(h|\mathcal{D}\big)$ of a hypothesis on a dataset $\mathcal{D}$ is the average loss incurred by $h$ when applied to the data points in $\mathcal{D}$.

**empirical risk minimization** Empirical risk minimization is the optimization problem of finding a hypothesis with minimum average loss (or empirical risk) on a given set $\mathcal{D}$ of data points (the training set). Many ML methods are special cases of empirical risk.

**estimation error** Consider data points with feature vectors $\mathbf{x}$ and label $y$. In some applications we can model the relation between the features and the label of a data point as $y = \bar{h}(\mathbf{x}) + \varepsilon$. Here we used some true hypothesis $\bar{h}$ and a noise term $\varepsilon$ which might represent modelling or labelling errors. The estimation error incurred by a ML method that learns a hypothesis $\widehat{h}$, e.g., using ERM, is defined as $\widehat{h}(\mathbf{x}) - \bar{h}(\mathbf{x})$, for some feature vector. For a parametrized hypothesis space, consisting of hypothesis maps that are determined by a model parameters $\mathbf{w}$, we can define the estimation error in terms of parameter vectors as $\Delta\mathbf{w} = \widehat{\mathbf{w}} - \overline{\mathbf{w}}$ [18, 19]. first

**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** Consider a numeric feature vector $\mathbf{x} \in \mathbb{R}^d$ which we interpret as the realization of a RV with probability distribution $p(\mathbf{x})$. The expectation of $\mathbf{x}$ is defined as the integral $\mathbb{E}\{\mathbf{x}\} := \int \mathbf{x} p(\mathbf{x})$ [2, 20, 21]. Note that the expectation is only defined if this integral exists, i.e., if the RV is integrable.

**expectation maximization** Expectation maximization is a generic technique for estimating the model parameters of a probabilistic model $p(\mathbf{z}; \mathbf{w})$ from data [19, 22, 23]. Expectation maximization delivers an approximation to the maximum likelihood estimate for the model parameters $\mathbf{w}$.

**expert** ML aims at learning a hypothesis $h$ that accurately predicts the label of a data point based on its features. We measure the prediction error using some loss function. Ideally we want to find a hypothesis that incurs minimum loss. One approach to make this goal precise is to use the i.i.d. assumption and use the resulting Bayes risk as the benchmark level for the (average) loss of a hypothesis. Alternatively, we might know a reference or benchmark hypothesis $h'$ which might be obtained by some existing ML method. We can then compare the loss incurred by $h$ against the loss incurred by $h'$. Such a reference or baseline hypothesis $h'$ is referred to as an expert [24]. Note that an expert might deliver very poor predictions. We typically compare against many different experts and aim at incurring not much more loss than the best among those experts (this is known as regret minimization) [24, 25]. first

**explainability** We define the (subjective) explainability of a ML method as the level of simulatability [26] of the predictions delivered by a ML system to a human user. Quantitative measures for the (subjective) explainability of a trained model can be constructed by comparing its predictions with the predictions provided by a user on a test-set [26, 27]. Alternatively, we can use probabilistic models for data and measure explainability of a trained ML model via the conditional (differential) entropy of its predictions, given the user predictions [28, 29].

**explainable empirical risk minimization** An instance of structural risk minimization that adds a regularization term to the average loss in the objective function of ERM. The regularization term is chosen to favour hypothesis maps that are intrinsically explainable for a specific user. This user is characterized by their predictions provided for the data points in a training set [27].

**explainable ML** Explainable ML methods aim at complementing each prediction with an explanation for how the prediction has been obtained. The construction of an explicit explanation might not be necessary if the ML method uses a sufficiently simple (or interpretable) model [30].

**explanation** One approach to make ML methods transparent, is to provide an explanation along with the prediction delivered by an ML method. Explanations can take on many different forms. An explanation could be some natural text or some quantitative measure for the importance of individual features of a data point [31]. We can also use visual forms of explanations such as intensity plots for image classification [32].

**feature** A feature of a data point is one of its properties that can be measured or computed easily without the need for human supervision. For example, if a data point is a bitmap image, then we could use the red-green-blue intensities of its pixels as features. Some widely used synonyms for the term feature are *covariate*, *explanatory variable*, *independent variable*, *input (variable)*, *predictor (variable)* or *regressor* [33–35].

**feature learning** Feature learning refers to the task of learning a map $\mathbf{\Phi}$ that reads in raw features of a data point and delivers new features. Different feature learning methods are obtained for different quantitative measures for the usefulness of the new features.

**feature map** A map that transforms the original features of a data point into new features. The so-obtained new features might be preferable over the original features for several reasons. For example, the shape of datasets might become simpler in the new feature space, allowing to use linear models in the new features. Another reason could be that the number of new features is much smaller which is preferable in terms of avoiding overfitting. The special case of a feature map delivering two numeric features is particularly useful for data visualization. Indeed, we can depict data points in a scatterplot by using two features as the coordinates of a data point.

**feature matrix** Consider a dataset $\mathcal{D}$ with $m$ data points, each of them characterized by the features $\mathbf{x}^{(r)}$, $r = 1, \ldots, m$. The feature matrix $\mathbf{X}$ of $\mathcal{D}$ is constructed by stacking, column-wise, the features of the data points into a matrix $\mathbf{X} = \big(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)}\big)$ of size $m \times d$. first

**feature matrix** Consider a dataset $\mathcal{D}$ of $m$ data points that are characterized by feature vectors $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)}$. It is convenient to collect the individual feature vectors into a feature matrix $\mathbf{X} := \big(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)}\big)$.

**feature space** The feature space of a given ML application or method is constituted by all potential values that the feature vector of a data point can take on. A widely used choice for the feature space is the

Euclidean space $\mathbb{R}^d$ with dimension $d$ being the number of individual features of a data point.

**federated averaging (FedAvg)** Federated averaging is an iterative FL algorithm that alternates between local model trainings and averaging the resulting local model parameters. Different variants of this algorithm are obtained by different techniques for the model training. The authors of [36] consider federated averaging methods where the local model training is implemented by running several GD steps

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

**federated 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 to collect a local dataset and, in turn, train a local model. FL methods learn a local hypothesis $h^{(i)}$, for each node $i \in \mathcal{V}$, such that it incurs small loss on the local datasets.

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

**flow-based clustering** Flow-basted clustering groups the nodes of an undirected graph by applying $k$-means clustering to node-wise feature vectors. These feature vectors are built from flows between carefully selected source and destination nodes [37].

**Gaussian mixture model** Gaussian mixture models (GMM) are a family of probabilistic models for data points characterized by a numeric feature vector $\mathbf{x}$. A GMM interprets $\mathbf{x}$ as being drawn from one of $k$ different multivariate normal distributions $p^{(c)} = \mathcal{N}\left(\boldsymbol{\mu}^{(c)}, \mathbf{C}^{(c)}\right)$, indexed by $c = 1, \ldots, k$. The probability that $\mathbf{x}$ is drawn from the $c$-th multivariate normal distribution is denoted $p_c$. Thus, a GMM is parametrized by the probability $p_c$, the mean vector $\boldsymbol{\mu}^{(c)}$ and covariance matrix $\boldsymbol{\Sigma}^{(c)}$ for each $c = 1, \ldots, k$.

**Gaussian random variable** A Gaussian RV $\mathbf{x} \in \mathbb{R}^d$ with a multivariate normal distribution. The special case of $d = 1$ corresponds to a scalar Gaussian RV [5, 38, 39].

**General Data Protection Regulation** The *general data protection regulation* (GDPR) is a law that has been passed by the European Union (EU) and put into effect on May 25, 2018 `https://gdpr.eu/tag/gdpr/`. The GDPR imposes obligations onto organizations anywhere, so long as they target, collect or in any other way process data related to people (i.e., personal data) in the EU [10].

**generalized total variation** Generalized total variation measures the changes of vector-valued node attributes over a weighted undirected graph.

**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 differentiable function $f(\mathbf{w})$.

**gradient step** Given a differentiable real-valued function $f(\mathbf{w})$ and a vector $\mathbf{w}'$, the gradient step updates $\mathbf{w}'$ by adding the scaled negative gradient $\nabla f(\mathbf{w}')$, $\mathbf{w}' \mapsto \mathbf{w}' - \eta \nabla f(\mathbf{w}')$.

**gradient-based method** Gradient-based methods are iterative techniques for finding the minimum (or maximum) of a differentiable 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 objective function value. 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.

**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 to each edge $e \in \mathcal{E}$ a pair of nodes [40]. One important family of graphs are 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 $e \in \mathcal{E}$.

**GTV minimization** GTV minimization is an instance of regularized empirical risk minimization (RERM) using the generalized total variation (GTV) of local model parameters as a regularizer [41].

**hard clustering** Hard clustering refers to the task of partitioning a given set of data points into (few) non-overlapping clusters. Each data point is assigned to one cluster.

**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) data points in the training set. For example, linear regression methods operate in the high-dimensional regime whenever the number $d$ of features used to characterize data points exceeds the number of data points in the training set. Another example of ML methods that operate in the high-dimensional regime are large ANNs, having far more tunable weights (and bias terms) than the number of data points in the training set. High-dimensional statistics is a recent main thread of probability theory that studies the behavior of ML methods in the high-dimensional regime [42, 43].

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

$$L\left((\mathbf{x}, y), h\right) := \max\{0, 1 - yh(\mathbf{x})\}. \tag{2}$$

A regularized variant of the hinge loss is used by the support vector machine (SVM) [44] to learn a linear classifier with maximum margin between the two classes (see Figure 1).

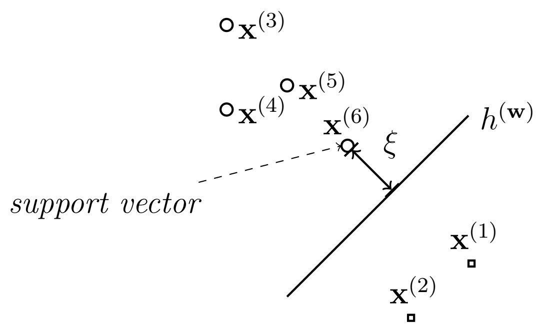


Figure 1: The SVM learns a hypothesis (or classifier) $h^{(\mathbf{w})}$ with minimum average soft-margin hinge loss. Minimizing this loss is equivalent to maximizing the margin $\xi$ between the decision boundary of $h^{(\mathbf{w})}$ and each class of the training set.

**histogram** Consider a dataset $\mathcal{D}$ that consists of $m$ data points $\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(m)}$ that belong 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 $\Delta$. The histogram of $\mathcal{D}$ assigns each elementary cell to the corresponding fraction of data points in $\mathcal{D}$ that fall into this elementary cell.

**horizontal FL** Horizontal FL refers to FL applications with local datasets that are constituted by different data points but using the same features to characterize them [45]. One example of a horizontal FL application is numerical weather prediction using a network of 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 features of different spatio-temporal regions (each such region being a separate data point).

**Huber loss** The Huber loss is a generalization and combination of the squared error loss and the absolute error loss.

**hypothesis** A map (or function) $h : \mathcal{X} \rightarrow \mathcal{Y}$ from the feature space $\mathcal{X}$ to the label space $\mathcal{Y}$. Given a data point with features $\mathbf{x}$ we use a hypothesis map $h$ to estimate (or approximate) the label $y$ using the predicted label $\hat{y} = h(\mathbf{x})$. ML is about learning (or finding) a hypothesis map $h$ such that $y \approx h(\mathbf{x})$ for any data point.

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

**i.i.d.** It can be useful to interpret data points $\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(m)}$ as realizations of independent and identically distributed RVs with a common probability distribution. If these RVs are continuous, their joint probability density function (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 RVs.

**i.i.d. assumption** The i.i.d. assumption interprets data points of a dataset as the realizations of i.i.d. RVs.

**interpretability** A ML method is interpretable for a specific user if they can well anticipate the predictions delivered by the method. The notion of interpretability can be made precise using quantitative measures of the uncertainty about the predictions [28].

**kernel** Consider data points characterized by a feature vector $\mathbf{x} \in \mathcal{X}$ with a generic feature space $\mathcal{X}$. A kernel is a map that assigns each pair of feature vectors $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$ a real number. This number measures the similarity between $\mathbf{x}$ and $\mathbf{x}'$. For more details about kernels and the resulting kernel methods, we refer to the literature [44, 46].

**Kullback-Leibler divergence** The Kullback–Leibler divergence is a quantitative measure for how much one probability distribution is different from another probability distribution [12].

**label** A higher-level fact or quantity of interest associated with a data point. For example, if the data point 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* [33–35].

**label space** Consider a ML application that involves data points characterized by features and labels. The label space is constituted by all potential values that the label of a data point can take on. Regression methods, aiming at predicting numeric labels, often use the label space $\mathcal{Y} = \mathbb{R}$. Binary classification methods use a label 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 data point whose label is known or has been determined by some means which might involve human experts.

**Laplacian matrix** The geometry or structure of a graph $\mathcal{G}$ can be analyzed using the properties of special matrices that are associated with $\mathcal{G}$. One such matrix is the graph Laplacian matrix $\mathbf{L}$ which is defined for an undirected and weighted graph [47, 48]. One important example of such a graph is the federated network in a FL application.

**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. RVs to the mean of their common probability distribution. Different instances of the law of large numbers are obtained using different notions of convergence [38].

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

**learning task** Consider a dataset $\mathcal{D}$ constituted by several data points, each of them characterized by features $\mathbf{x}$. For example, the dataset $\mathcal{D}$ might be constituted by the images of a particular database. Sometimes it might be useful to represent a dataset $\mathcal{D}$, along with the choice of features, by a probability distribution $p(\mathbf{x})$. A learning task associated with $\mathcal{D}$ consists of a specific choice for the label of a data point 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 dataset, we obtain different learning tasks by using different choices for the features and label of a data point. These learning tasks are related, as they are based on the same dataset, and solving them jointly (via multitask learning methods) is typically preferable over solving them separately [49–51].

**least absolute deviation regression** Least absolute deviation regression is an instance of ERM using the absolute error loss.

**least absolute shrinkage and selection operator (Lasso)** The least absolute shrinkage and selection operator (Lasso) is an instance of structural risk minimization (SRM) for learning the weights $\mathbf{w}$ of a linear map $h(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$. The Lasso minimizes the sum consisting of an average squared error loss (as in linear regression) and the scaled $\ell_1$ norm of the weight vector $\mathbf{w}$.

**linear classifier** Consider data points characterized by numeric features $\mathbf{x} \in \mathbb{R}^d$ and a label $y \in \mathcal{Y}$ with values from a finite label space $\mathcal{Y}$. A linear classifier characterized by having decision regions separated by hyperplanes in the Euclidean space $\mathbb{R}^d$.

**linear model** Consider data points, 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,

$$\mathcal{H}^{(d)} := \left\{ h(\mathbf{x}) = \mathbf{w}^T \mathbf{x} : \mathbf{w} \in \mathbb{R}^d \right\}. \tag{3}$$

Note that (3) defines an entire family of hypothesis spaces, which is parametrized by the number $d$ of features that are linearly combined to form the prediction $h(\mathbf{x})$. The design choice of $d$ is guided by computational aspects (smaller $d$ means less computation), statistical aspects (increasing $d$ might reduce prediction error) and interpretability. A linear model using few carefully chosen features tends to be considered more interpretable [30, 52].

**linear regression** Linear regression aims to learn a linear hypothesis map to predict a numeric label based on numeric features of a data point. The quality of a linear hypothesis map is measured using the average squared error loss incurred on a set of labeled data points, which we refer to as the training set.

**local dataset** The concept of a local dataset is in-between the concept of a data point and a dataset. A local dataset consists of several individual data points which are characterized by features and labels. In contrast to a single dataset used in basic ML methods, a local dataset is also related to other local datasets via different notions of similarities. These similarities might arise from probabilistic models or communication infrastructure and are encoded in the edges of an federated network.

**local model** Consider a collections of local datasets that are assigned to the nodes of an federated 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 data point, characterized by the features $\mathbf{x}$ and a binary label $y \in \{-1, 1\}$. We use a real-valued hypothesis $h$ to predict the label $y$ from the features $\mathbf{x}$. The logistic loss incurred by this prediction is defined as

$$L\left((\mathbf{x}, y), h\right) := \log(1 + \exp(-yh(\mathbf{x}))). \tag{4}$$

Carefully note that the expression (4) for the logistic loss applies only if for the label space $\mathcal{Y} = \{-1, 1\}$ and using the thresholding rule (1).

**logistic regression** Logistic regression learns a linear hypothesis map (classifier) $h(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$ to predict a binary label $y$ based on numeric feature vector $\mathbf{x}$ of a data point. The quality of a linear hypothesis map is measured by the average logistic loss on some labeled data points (the training set).

**loss** ML methods use a loss function $L\left(\mathbf{z}, h\right)$ to measure the error incurred by applying a specific hypothesis to a specific data point. With slight abuse of notation, we use the term *loss* for both, the loss function $L$ itself and for its value $L(\mathbf{z}, h)$ for a specific data point $\mathbf{z}$ and hypothesis $h$.

**loss function** A loss function is a map

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

which assigns a pair of a data point, with features $\mathbf{x}$ and label $y$, and a hypothesis $h \in \mathcal{H}$ the non-negative real number $L((\mathbf{x}, y), h)$. The loss value $L((\mathbf{x}, y), h)$ quantifies the discrepancy between the true label $y$ and the prediction $h(\mathbf{x})$. Lower (closer to zero) values $L((\mathbf{x}, y), h)$ indicate a smaller discrepancy between prediction $h(\mathbf{x})$ and label $y$. Figure 2 depicts a loss function for a given data point, with features $\mathbf{x}$ and label $y$, as a function of the hypothesis $h \in \mathcal{H}$.

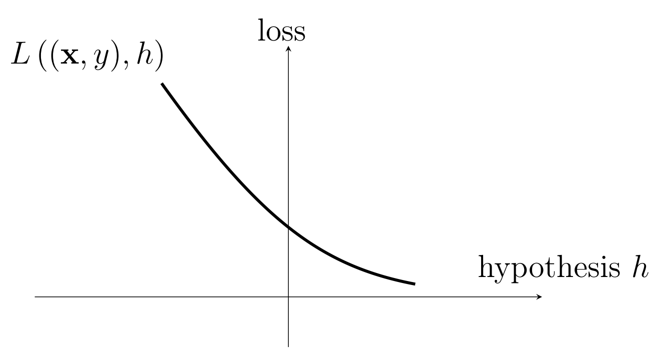


Figure 2: Some loss function $L((\mathbf{x}, y), h)$ for a fixed data point, with feature vector $\mathbf{x}$ and label $y$, and varying hypothesis $h$. ML methods try to find (learn) a hypothesis that incurs minimum loss.

**maximum** Given a set of real numbers, the maximum is the largest of those numbers.

**maximum likelihood** Consider data points $\mathcal{D} = \{\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(m)}\}$ that are interpreted as realizations of i.i.d. RVs with a common probability distribution $p(\mathbf{z}; \mathbf{w})$ which depends on a parameter vector $\mathbf{w} \in \mathcal{W} \subseteq \mathbb{R}^n$. Maximum likelihood methods aim at finding a parameter vector $\mathbf{w}$ such that the probability (density) $p(\mathcal{D}; \mathbf{w}) = \prod_{r=1}^{m} p(\mathbf{z}^{(r)}; \mathbf{w})$ of observing the data is maximized. Thus, the maximum likelihood estimator is obtained as a solution to the optimization problem $\max_{\mathbf{w} \in \mathcal{W}} p(\mathcal{D}; \mathbf{w})$.

**mean** The expectation $\mathbb{E}\{\mathbf{x}\}$ of a numeric RV $\mathbf{x}$.

**mean squared estimation error** Consider a ML method that learns model parameters $\widehat{\mathbf{w}}$ based on some dataset $\mathcal{D}$. If we interpret the data points in $\mathcal{D}$ as i.i.d. realizations of a RV $\mathbf{z}$, we define the estimation error $\Delta \mathbf{w} := \widehat{w} - \overline{\mathbf{w}}$. Here, $\overline{\mathbf{w}}$ denotes the true model parameters of the probability distribution of $\mathbf{z}$.The mean squared estimation error is defined as the expectation $\mathbb{E}\{\|\Delta \mathbf{w}\|^2\}$ of the squared Euclidean norm of the estimation error [7, 18].

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

**missing data** Consider a dataset constituted by data points collected via some physical device. Due to imperfections and failures, some of the feature or label values of data points might be corrupted or simply *missing*. Data imputation aims at estimating these missing values [53]. We can interpret data imputation as a ML problem where the label of a data point is the value of the corrupted feature.

**model** In the context of ML methods, the term *model* typically refers to the hypothesis space used by a ML method [6, 54].

**model parameters** Model parameters are numbers that select a hypothesis map from a hypothesis space.

**model selection** In ML, model selection refers to the process of choosing between different candidate models. In its most basic form, model selection amounts to (i) training each candidate model, (ii) computing the validation error for each trained model, (iii) choosing the model with smallest validation error [6, Ch. 6].

**multi-label classification** Multi-label classification problems and methods use data points that are characterized by several labels. As an example, consider a data point representing a picture with one binary label indicating the presence of a human in this picture and another label indicating 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 dataset of webcam snapshots. The first task is to predict the presence of a human, while the second is the predict the presence of a car. It might be useful to use the same deep ANN (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 probability distributions for a continuous RV $\mathbf{x} \in \mathbb{R}^d$ [5, 39, 55]. This family is parametrized by the mean $\mathbf{m}$ and covariance matrix $\mathbf{C}$ of $\mathbf{x}$. If the covariance matrix is invertible, the probability distribution 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** The mutual information $I\left(\mathbf{x}; y\right)$ between two RVs $\mathbf{x}$, $y$ defined on the same probability space is given by [12]

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

It is a measure for how well we can estimate $y$ based solely from $\mathbf{x}$. A large value of $I\left(\mathbf{x}; y\right)$ indicates that $y$ can be well predicted solely from $\mathbf{x}$. This prediction could be obtained by a hypothesis learnt by a ERM-based ML method.

**nearest neighbour** Nearest neighbour methods learn a hypothesis $h : \mathcal{X} \rightarrow \mathcal{Y}$ whose function value $h(\mathbf{x})$ is solely determined by the nearest neighbours within a given dataset. Different methods use different metrics for determining the nearest neighbours. If data points are characterized by numeric feature vectors, we can use their Euclidean distances as the metric.

**neighbourhood** The neighbourhood of a node $i \in \mathcal{V}$ is the subset of nodes constituted by the neighbours of $i$.

**neighbours** The neighbours of a node $i \in \mathcal{V}$ within an federated 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 pair-wise similarity. We can represent networked data using an federated network whose nodes carry local datasets and an edge indicates a similarity between the connected nodes.

**networked exponential families** A collection of exponential families, each of them assigned to a node of an federated network. The model parameters are coupled via the network structure by requiring them to have a small GTV [56].

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

**networked model** A networked model over an federated network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ assigns a local model (hypothesis space) to each node $i \in \mathcal{V}$ of the federated 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 neighbours, $d^{(i)} := \left|\mathcal{N}^{(i)}\right|$.

**non-smooth** We refer to a function as non-smooth if it is not smooth [57].

**norm** A norm is a function that maps each element (vector) of a linear vector space to a non-negative real number. This function must be homogeneous, definite and satisfy the triangle inequality [58].

**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 hypothesis $h^{(\mathbf{w})}$, to an objective value $f(\mathbf{w})$. The objective value $f(\mathbf{w})$ could be the risk or the empirical risk of a hypothesis $h^{(\mathbf{w})}$.

**outlier** Many ML methods are motivated by the i.i.d. assumption which interprets data points as realizations of i.i.d. RVs with a common probability distribution. The i.i.d. assumption is useful for applications where the statistical properties of the data generation process are stationary (time-invariant). However, in some applications the data consists of a majority of *regular* data points that conform with an i.i.d. assumption and a small number of data points that have fundamentally different statistical properties compared to the regular data points. We refer to a data point that substantially deviates from the statistical properties of the majority of data points as an outlier. Different methods for outlier detection use different measures for this deviation.

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

**parameters** The parameters of a ML model are tunable (learnable or adjustable) quantities that allow to choose between different hypothesis maps. For example, the linear model $\mathcal{H} := \{h^{(\mathbf{w})} : h^{(\mathbf{w})}(x) = w_1 x + w_2\}$ consists of all hypothesis 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 hypothesis map to predict a numeric label based on numeric features of a data point. For data points characterized by a single numeric feature, polynomial regression uses the hypothesis space $\mathcal{H}_d^{(\mathrm{poly})} := \{h(x) = \sum_{j=0}^{d-1} x^j w_j\}$. The quality of a polynomial hypothesis map is measured using the average squared error loss incurred on a set of labeled data points (which we refer to as training set).

**positive semi-definite** A symmetric real-valued matrix $\mathbf{Q} = \mathbf{Q}^T \in \mathbb{R}^{d \times d}$ is referred to as positive semi-definite if $\mathbf{x}^T \mathbf{Q} \mathbf{x} \geq 0$ for every vector $\mathbf{x} \in \mathbb{R}^d$.

**prediction** A prediction is an estimate or approximation for some quantity of interest. ML revolves around learning or finding a hypothesis map $h$ that reads in the features $\mathbf{x}$ of a data point and delivers a prediction $\widehat{y} := h(\mathbf{x})$ for its label $y$.

**predictor** A predictor is a real-valued hypothesis map. Given a data point with features $\mathbf{x}$, the value $h(\mathbf{x}) \in \mathbb{R}$ is used as a prediction for the true numeric label $y \in \mathbb{R}$ of the data point.

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

**privacy leakage** Consider a (ML or FL) system that processes a local dataset $\mathcal{D}^{(i)}$ and shares data, such as the predictions obtained for new data points, with other parties. Privacy leakage arises if the shared data carries information about a private (sensitive) feature of a data point (which might be a human) of $\mathcal{D}^{(i)}$. The amount of privacy leakage can be measured via mutual information (MI) using a probabilistic model for the local dataset. Another quantitative measure for privacy leakage is differential privacy (DP).

**privacy protection** Consider some ML method $\mathcal{A}$ that reads in a $\mathcal{D}$ and delivers some output $\mathcal{A}(\mathcal{D})$. The output could be the learnt model parameters $\widehat{\mathbf{w}}$ or the prediction $\hat{h}(\mathbf{x})$ obtained for a specific data point with features $\mathbf{x}$. Many important ML applications involve data points representing humans. Each data point is characterized by features $\mathbf{x}$, potentially a label $y$ 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 data points 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 data points as realizations of RVs with a joint probability distribution.This joint probability distribution typically involves parameters which have to be manually chosen or learnt via statistical inference methods such as maximum likelihood [7].

**probabilistic PCA** Probabilistic principal component analysis (PCA) (PPCA) extends basic PCA by using a probabilistic model for data points. The probabilistic model of PPCA reduces the task of dimensionality reduction to an estimation problem that can be solved using expectation maximization (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, 20, 21, 59].

**probability density function (pdf)** The probability density function (pdf) $p(x)$ of a real-valued RV $x \in \mathbb{R}$ is a particular representation of its probability distribution. If the pdf 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 pdf of a vector-valued RV $\mathbf{x} \in \mathbb{R}^d$ (if it exists) allows to compute the probability that $\mathbf{x}$ falls into 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 distribution** To analyze ML methods it can be useful to interpret data points as i.i.d. realizations of a RV. The typical properties of such data points are then governed by the probability distribution of this RV. The probability distribution of a binary RV $y \in \{0, 1\}$ is fully specified by the probabilities $p(y = 0)$ and $p(y = 1)\big(=1-p(y = 0)\big)$. The probability distribution of a real-valued RV $x \in \mathbb{R}$ might be specified by a probability density function $p(x)$ 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 [21, 39].

**projected GD** Projected GD extends basic GD for unconstrained optimization to handle constraints on the optimization variable (model parameters). A single iteration of projected GD consists of first taking a gradient step and then projecting the result back into a constrain set.

**proximable** A convex function for which the proximal operator can be computed efficiently are sometimes referred to as *proximable* or *simple* [60].

**proximal operator** Given a convex function and a vector $\mathbf{x}$, we define its proximal operator as [61, 62]

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

Convex functions for which the proximal operator can be computed efficiently are sometimes referred to as *proximable* or *simple* [60].

**quadratic function** A quadratic function $f(\mathbf{w})$, reading in a vector $\mathbf{w} \in \mathbb{R}^d$ as its argument, is such that

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

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

**Rényi divergence** The Rényi divergence measures the (dis-)similarity between two probability distributions [63].

**random forest** A random forest is a set (ensemble) of different decision trees. Each of these decision trees is obtained by fitting a perturbed copy of the original dataset.

**random variable (RV)** A random variable is a mapping from a probability space $\mathcal{P}$ to a value space [21]. The probability space, whose elements are elementary events, is equipped with a probability measure that assigns a probability to subsets of $\mathcal{P}$. A binary random variable maps elementary events to a set containing two different values, e.g., $\{-1, 1\}$ or $\{\text{cat}, \text{no cat}\}$. A real-valued random variable maps elementary events to real numbers $\mathbb{R}$. A vector-valued random variable maps 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 random variables [21, 39].

**realization** Consider a RV $x$ which maps each element (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, 20, 21]. 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 rectified linear unit (ReLU) is a popular choice for the activation function of a neuron within an ANN. It is defined as $g(z) = \max\{0, z\}$ with $z$ being the weighted input of the artificial neuron.

**regression** Regression problems revolve around the problem of predicting a numeric label solely from the features of a data point [6, Ch. 2].

**regret** The regret of a hypothesis $h$ relative to another hypothesis $h'$, which serves as a reference (or baseline), is the difference between the loss incurred by $h$ and the loss incurred by $h'$ [24]. The baseline hypothesis $h'$ is also referred to as an expert.

**regularization** Regularization techniques modify ERM such that the learnt hypothesis performs well (generalizes) also outside the training set. One specific implementation of regularization is to add a penalty or regularization term to the objective function of ERM (which is the average loss on the training set). This regularization term can be interpreted as an estimate for the increase in the expected loss (risk) compared to the average loss on the training set.

**regularized empirical risk minimization** Synonym for SRM.

**regularizer** A regularizer assigns each hypothesis $h$ from a hypothesis space $\mathcal{H}$ a quantitative measure $\mathcal{R}\{h\}$ for how much its prediction error on a training set might differ from its prediction errors on data points outside the training set. Ridge regression uses the regularizer $\mathcal{R}\{h\} := \|\mathbf{w}\|_2^2$ for linear hypothesis maps $h^{(\mathbf{w})}(\mathbf{x}) := \mathbf{w}^T\mathbf{x}$ [6, Ch. 3]. The least absolute shrinkage and selection operator (Lasso) uses the regularizer $\mathcal{R}\{h\} := \|\mathbf{w}\|_1$ for linear hypothesis maps $h^{(\mathbf{w})}(\mathbf{x}) := \mathbf{w}^T\mathbf{x}$ [6, Ch. 3].

**ridge regression** Ridge regression learns the weights $\mathbf{w}$ of a linear hypothesis map $h^{(\mathbf{w})}(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$. The quality of a particular choice for the parameter vector $\mathbf{w}$ 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 data points (the training set). The second component is the scaled squared Euclidean norm $\alpha\|\mathbf{w}\|_2^2$ with a regularization parameter $\alpha > 0$. It can be shown that the effect of adding to $\alpha\|\mathbf{w}\|_2^2$ to the average squared error loss is equivalent to replacing the original data points by an ensemble of realizations of a RV centered around these data points.

**risk** Consider a hypothesis $h$ used to predict the label $y$ of a data point based on its features $\mathbf{x}$. We measure the quality of a particular prediction using a loss function $L\left((\mathbf{x}, y), h\right)$. If we interpret data points as the realizations of i.i.d. RVs, also the $L\left((\mathbf{x}, y), h\right)$ becomes the realization of a RV. The i.i.d. assumption allows to define the risk of a hypothesis 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 probability distribution of the data points.

**sample** A finite sequence (list) of data points $\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(r)}$ that is obtained or interpreted as the realizations of $m$ i.i.d. RVs with the common probability distribution $p(\mathbf{z})$. The length $m$ of the sequence is referred to as the sample size.

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

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

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

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

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

**sample size** The number of individual data points contained in a dataset obtained as the realizations of i.i.d. RVs with common probability distribution.

**scatterplot** A visualization technique that depicts data points by markers in a two-dimensional plane.

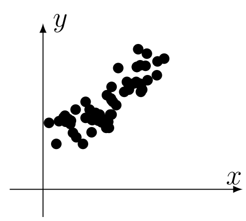


Figure 3: A scatterplot of data points that represent daily weather conditions in Finland. Each data point is characterized by its minimum daytime temperature $x$ as feature and its maximum daytime temperature $y$ as the label. The temperatures have been measured at the FMI weather station *Helsinki Kaisaniemi* during 1.9.2024 - 28.10.2024.

**semi-supervised learning** Semi-supervised learning methods use unlabeled data points to support the learning of a hypothesis from labeled data points [8]. This approach is particularly useful for ML applications that offer a large amount of unlabeled data points, but only a limited number of labeled data points.

**sensitive attribute** ML revolves around learning a hypothesis map that allows to predict the label of a data point from its features. In some applications we must ensure that the output delivered by an ML system does not allow to infer sensitive attributes of a data point. Which parts of a data point is considered as a sensitive attribute is a design choice that varies across different application domains.

**similarity graph** Some ML applications generate data points 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 data point. Two nodes are connected by an undirected edge if the corresponding data points are similar.

**singular value decomposition** The singular value decomposition 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** We refer to a real-valued function as smooth if it is differentiable and its gradient is continuous [57, 64]. In particular, a differentiable function $f(\mathbf{w})$ is referred to as $\beta$-smooth if the gradient $\nabla f(\mathbf{w})$ is Lipschitz continuous with Lipschitz constant $\beta$, i.e.,

$$\|\nabla f(\mathbf{w}) - \nabla f(\mathbf{w}')\| \leq \beta \|\mathbf{w} - \mathbf{w}'\|.$$

**soft clustering** Soft clustering refers to the task of partitioning a given set of data points into (few) overlapping clusters. Each data point is assigned to several different clusters with varying degree of belonging. Soft clustering methods determine the degree of belonging (or soft cluster assignment) for each data point and each cluster. A principled approach to soft clustering is by interpreting data points 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 data point belonging to a specific mixture component.

**spectral clustering** Spectral clustering groups the nodes of an undirected graph by applying $k$-means clustering to node-wise feature vectors. These feature vectors are built from the eigenvectors of the graph Laplacian matrix [37, 47].

**spectrogram** The spectrogram of a time signal, e.g., an audio recording, characterizes the time-frequency distribution of the signal. Loosely speaking, the spectrogram quantifies the signal energy within a specific time period and frequency interval.

**squared error loss** The squared error loss measures the prediction error of a hypothesis $h$ when predicting a numeric label $y \in \mathbb{R}$ from the features $\mathbf{x}$ of a data point. It is defined as

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

**statistical aspects** By statistical aspects of a ML method, we refer to (properties of) the probability distribution of its output under a probabilistic model for the data fed into the method.

**step size** See learning rate.

**stochastic block model** The stochastic block model (SBM) is a probabilistic generative model for an undirected graph $\mathcal{G} = \big(\mathcal{V}, \mathcal{E}\big)$ with a given set of nodes $\mathcal{V}$ [65]. In its most basic variant, the SBM 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 $p_{i,j}$ that depends solely on the labels $c_i, c_j$. The presence of edges between different pairs of nodes is statistically independent.

**stochastic gradient descent** Stochastic GD is obtained from GD by replacing the gradient of the objective function with some e stimate (or approximation). A main application of stochastic gradient descent is to solve ERM where the objective function and its gradient consists of a sum over the data points in a dataset $\mathcal{D}$. Here, the gradient estimate can be obtained by replacing the sum over the entire dataset with a sum over a randomly selected subset of $\mathcal{D}$.

**stopping criterion** Many ML 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) that (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 times. A stopping criterion is any well-defined condition required for stopping iterating.

**strongly convex** A continuously differentiable real-valued function $f(\mathbf{x})$ is strongly convex with coefficient $\sigma$ if $f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla f(\mathbf{x})^T(\mathbf{y}-\mathbf{x}) + (\sigma/2)\left\|\mathbf{y}-\mathbf{x}\right\|_2^2$ [57], [66, Sec. B.1.1.].

**structural risk minimization** Structural risk minimization is the problem of finding the hypothesis that optimally balances the average loss (or empirical risk) on a training set with a regularization term. The regularization term penalizes a hypothesis that is not robust against (small) perturbations of the data points in the training set.

**subgradient** For a real-valued function $f : \mathbb{R}^d \rightarrow \mathbb{R} : \mathbf{w} \mapsto f(\mathbf{w})$, a vector $\mathbf{a}$ such that $f(\mathbf{w}) \geq f(\mathbf{w}') + \left(\mathbf{w}-\mathbf{w}'\right)^T\mathbf{a}$ is referred to as a subgradient of $f$ at $\mathbf{w}'$ [67, 68].

**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 sub-gradient. Similar to gradients, also sub-gradients allow 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 hypothesis $h^{(\mathbf{w})} \in \mathcal{H}$.

**support vector machine** The support vector machine is a binary classification method that learns a linear hypothesis map. It aims at maximally separating data points from the two different classes in the feature space (*maximum margin principle*). Maximizing this separation is equivalent to minimizing a regularized variant of the hinge loss (2).

**test set** A set of data points that have neither been used 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 hypothesis when predicting the labels of data points in a training set. We sometimes refer by training error also the minimum average loss incurred on the training set by the optimal hypothesis from a hypothesis space.

**training set** A dataset $\mathcal{D}$, constituted by some data points used in ERM to learn a hypothesis $\hat{h}$. The average loss of $\hat{h}$ on the training set is referred to as the training error. The comparison between training error and validation error of $\hat{h}$ allows to diagnose ML methods and informs how to improve them (e.g., using a different hypothesis space or collecting more data points) [6, Sec. 6.6.].

**transparency** Transparency is a key requirement for trustworthy AI [69]. In the context of ML methods, such as ERM-based methods, transparency is mainly used synonymously for explainability [28, 70]. However, in the wide context of AI systems, transparency also includes providing information about limitations and reliability of the AI system. As a point in case, logistic regression provides a quantitative measure for the reliability of a classification in the form of the value $|h(\mathbf{x})|$. Transparency also includes the user interface, by requiring to clearly indicate when a user is interaction with an AI system. Another component of transparency is the availability of high-level documentation of the system's purpose, design choices and intended use cases [71–73].

**trustworthiness** Beside the computational aspects and statistical aspects, a third main design aspect for ML methods is their trustworthiness [74]. The European Union has put forward seven key requirements (KRs) for trustworthy AI (that typically build on ML methods) [75]: **KR1 - Human Agency and Oversight**, **KR2 - Technical Robustness and Safety**, **KR3 - Privacy and Data Governance**, **KR4 - Transparency**, **KR5 - Diversity Non-Discrimination and Fairness**, **KR6 Societal and Environmental Well-Being**, **KR7 - Accountability**.

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

**validation** Consider a hypothesis $\hat{h}$ that has been learnt via some ML method, e.g., by solving ERM on a training set $\mathcal{D}$. Validation refers to the practice of evaluating the loss incurred by hypothesis $\hat{h}$ on a validation set that consists of data points that are not contained in the training set $\mathcal{D}$.

**validation error** Consider a hypothesis $\hat{h}$ which is obtained by some ML 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 data points used to estimate the risk of a hypothesis $\hat{h}$ that has been learnt by some ML 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 a ML method (see [6, Sec. 6.6.]). The comparison between training error and validation error can inform directions for improvements of the ML method (such as using a different hypothesis space).

**Vapnik–Chervonenkis (VC) dimension** The VC dimension of an infinite hypothesis space is a widely-used measure for its size. We refer to [54] for a precise definition of VC dimension as well as a discussion of its basic properties and use in ML.

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

**vertical FL** Vertical FL refers to applications with local datasets that are constituted by the same data points but characterizing them with different features [76]. For example, different healthcare providers might all contain information about the same population of patients. However, different healthcare providers collect different measurements (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 features 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 features 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 $L\left((\mathbf{x}, y), h\right)$ measures the quality of a classifier $h(\mathbf{x})$ that delivers a prediction $\hat{y}$ (e.g., via thresholding (1)) for the label $y$ of a data point with features $\mathbf{x}$. It is equal to 0 if the prediction is correct, i.e., $L\left((\mathbf{x}, y), h\right) = 0$ when $\hat{y} = y$. It is equal to 1 if the prediction is wrong, $L\left((\mathbf{x}, y), h\right) = 1$ when $\hat{y} \neq y$.

# Index

# References


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

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

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

[4] G. Golub and C. van Loan, "An analysis of the total least squares problem," *SIAM J. Numerical Analysis*, vol. 17, no. 6, pp. 883–893, Dec. 1980.

[5] D. Bertsekas and J. Tsitsiklis, *Introduction to Probability*, 2nd ed. Athena Scientific, 2008.

[6] A. Jung, *Machine Learning: The Basics*, 1st ed. Springer Singapore, Feb. 2022.

[7] E. L. Lehmann and G. Casella, *Theory of Point Estimation*, 2nd ed. New York: Springer, 1998.

[8] O. Chapelle, B. Schölkopf, and A. Zien, Eds., *Semi-Supervised Learning.* Cambridge, Massachusetts: The MIT Press, 2006.

[9] S. Boyd and L. Vandenberghe, *Convex Optimization.* Cambridge, UK: Cambridge Univ. Press, 2004.

[10] E. Comission, “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),” no. 119, pp. 1–88, May 2016.

[11] 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,” nov 2018, official Journal of the European Union, L 295, 21.11.2018, pp. 39–98. [Online]. Available: https://eur-lex.europa.eu/eli/reg/2018/1725/oj

[12] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, 2nd ed. New Jersey: Wiley, 2006.

[13] X. Liu, H. Li, G. Xu, Z. Chen, X. Huang, and R. Lu, “Privacy-enhanced federated learning against poisoning adversaries,” *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 4574–4588, 2021.

[14] 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 of Things Journal*, vol. 8, no. 5, pp. 3310–3322, 2021.

[15] P. Halmos, *Naive set theory.* Springer-Verlag, 1974.

[16] A. Silberschatz, H. Korth, and S. Sudarshan, *Database System Concepts*, 7th ed. McGraw-Hill, 2019. [Online]. Available: https://db-book.com/

[17] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[18] S. M. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory*. Englewood Cliffs, NJ: Prentice Hall, 1993.

[19] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, ser. Springer Series in Statistics. New York, NY, USA: Springer, 2001.

[20] P. R. Halmos, *Measure Theory*. New York: Springer, 1974.

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

[22] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[23] M. J. Wainwright and M. I. Jordan, *Graphical Models, Exponential Families, and Variational Inference*, ser. Foundations and Trends in Machine Learning. Hanover, MA: Now Publishers, 2008, vol. 1, no. 1–2.

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

[25] E. Hazan, *Introduction to Online Convex Optimization*. Now Publishers Inc., 2016.

[26] 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." *Advances in Neural Information Processing Systems*, vol. 35, pp. 2832–2845, 2022.

[27] L. Zhang, G. Karakasidis, A. Odnoblyudova, L. Dogruel, Y. Tian, and A. Jung, "Explainable empirical risk minimization," *Neural Computing and Applications*, vol. 36, no. 8, pp. 3983–3996, 2024. [Online]. Available: https://doi.org/10.1007/s00521-023-09269-3

[28] A. Jung and P. Nardelli, "An information-theoretic approach to personalized explainable machine learning," *IEEE Sig. Proc. Lett.*, vol. 27, pp. 825–829, 2020.

[29] J. Chen, L. Song, M. Wainwright, and M. Jordan, "Learning to explain: An information-theoretic perspective on model interpretation," in *Proc. 35th Int. Conf. on Mach. Learning*, Stockholm, Sweden, 2018.

[30] C. Rudin, "Stop explaining black box machine learning models for high-stakes decisions and use interpretable models instead," *Nature Machine Intelligence*, vol. 1, no. 5, pp. 206–215, 2019.

[31] C. Molnar, *Interpretable Machine Learning - A Guide for Making Black Box Models Explainable.* [online] Available: https://christophm.github.io/interpretable-ml-book/., 2019.

[32] 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 International Conference on Computer Vision (ICCV)*, 2017, pp. 618–626.

[33] D. Gujarati and D. Porter, *Basic Econometrics*. Mc-Graw Hill, 2009.

[34] Y. Dodge, *The Oxford Dictionary of Statistical Terms*. Oxford University Press, 2003.

[35] B. Everitt, *Cambridge Dictionary of Statistics*. Cambridge University Press, 2002.

[36] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, A. Singh and J. Zhu, Eds., vol. 54. PMLR, 20–22 Apr 2017, pp. 1273–1282. [Online]. Available: https://proceedings.mlr.press/v54/mcmahan17a.html

[37] Y. SarcheshmehPour, Y. Tian, L. Zhang, and A. Jung, "Flow-based clustering and spectral clustering: A comparison," in *2021 55th Asilomar Conference on Signals, Systems, and Computers*, 2021, pp. 1292–1296.

[38] A. Papoulis and S. U. Pillai, *Probability, Random Variables, and Stochastic Processes*, 4th ed. New York: Mc-Graw Hill, 2002.

[39] R. Gray, *Probability, Random Processes, and Ergodic Properties*, 2nd ed. New York: Springer, 2009.

[40] R. T. Rockafellar, *Network Flows and Monotropic Optimization.* Athena Scientific, Jul. 1998.

[41] Y. SarcheshmehPour, Y. Tian, L. Zhang, and A. Jung, “Clustered federated learning via generalized total variation minimization,” *IEEE Transactions on Signal Processing*, vol. 71, pp. 4240–4256, 2023.

[42] M. Wainwright, *High-Dimensional Statistics: A Non-Asymptotic Viewpoint.* Cambridge: Cambridge University Press, 2019.

[43] P. Bühlmann and S. van de Geer, *Statistics for High-Dimensional Data.* New York: Springer, 2011.

[44] C. Lampert, “Kernel methods in computer vision,” *Foundations and Trends in Computer Graphics and Vision*, 2009.

[45] Q. Yang, Y. Liu, Y. Cheng, Y. Kang, T. Chen, and H. Yu, *Horizontal Federated Learning.* Cham: Springer International Publishing, 2020, pp. 49–67. [Online]. Available: https://doi.org/10.1007/978-3-031-01585-4_4

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

[47] U. von Luxburg, “A tutorial on spectral clustering,” *Statistics and Computing*, vol. 17, no. 4, pp. 395–416, Dec. 2007.

[48] A. Y. Ng, M. I. Jordan, and Y. Weiss, “On spectral clustering: Analysis and an Algorithm,” in *Adv. Neur. Inf. Proc. Syst.*, 2001.

[49] R. Caruana, "Multitask learning," *Machine Learning*, vol. 28, no. 1, pp. 41–75, 1997. [Online]. Available: https://doi.org/10.1023/A:1007379606734

[50] A. Jung, G. Hannak, and N. Görtz, "Graphical LASSO Based Model Selection for Time Series," *IEEE Sig. Proc. Letters*, vol. 22, no. 10, Oct. 2015.

[51] A. Jung, "Learning the conditional independence structure of stationary time series: A multitask learning approach," *IEEE Trans. Signal Processing*, vol. 63, no. 21, Nov. 2015.

[52] M. Ribeiro, S. Singh, and C. Guestrin, ""Why should i trust you?": Explaining the predictions of any classifier," in *Proc. 22nd ACM SIGKDD*, Aug. 2016, pp. 1135–1144.

[53] K. Abayomi, A. Gelman, and M. A. Levy, "Diagnostics for multivariate imputations," *Journal of The Royal Statistical Society Series C-applied Statistics*, vol. 57, pp. 273–291, 2008.

[54] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning – from Theory to Algorithms.* Cambridge University Press, 2014.

[55] A. Lapidoth, *A Foundation in Digital Communication.* New York: Cambridge University Press, 2009.

[56] A. Jung, "Networked exponential families for big data over networks," *IEEE Access*, vol. 8, pp. 202 897–202 909, 2020.

[57] Y. Nesterov, *Introductory lectures on convex optimization*, ser. Applied Optimization. Kluwer Academic Publishers, Boston, MA, 2004,

vol. 87, a basic course. [Online]. Available: http://dx.doi.org/10.1007/978-1-4419-8853-9

[58] R. A. Horn and C. R. Johnson, *Matrix Analysis*, 2nd ed. Cambridge, UK: Cambridge Univ. Press, 2013.

[59] O. Kallenberg, *Foundations of modern probability.* New York: Springer, 1997.

[60] L. Condat, "A Primal–Dual Splitting Method for Convex Optimization involving Lipschitzian, Proximable and Linear Composite Terms," *Journal of Opt. Th. and App.*, vol. 158, no. 2, pp. 460–479, Aug. 2013.

[61] N. Parikh and S. Boyd, "Proximal algorithms," *Foundations and Trends in Optimization*, vol. 1, no. 3, pp. 123–231, 2013.

[62] H. Bauschke and P. Combettes, *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*, 2nd ed. New York: Springer, 2017.

[63] I. Csiszar, "Generalized cutoff rates and Renyi's information measures," *IEEE Transactions on Information Theory*, vol. 41, no. 1, pp. 26–34, 1995.

[64] S. Bubeck, "Convex optimization. algorithms and complexity." in *Foundations and Trends in Machine Learning.* Now Publishers, 2015, vol. 8.

[65] E. Abbe, "Community detection and stochastic block models: Recent developments," *Journal of Machine Learning Research*, vol. 18, no. 177, pp. 1–86, 2018.

[66] D. P. Bertsekas, *Convex Optimization Algorithms.* Athena Scientific, 2015.

[67] D. Bertsekas, A. Nedic, and A. Ozdaglar, *Convex Analysis and Optimization.* Athena Scientific, 2003.

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

[69] H.-L. E. G. on Artificial Intelligence, "Ethics guidelines for trustworthy AI," European Commission, Tech. Rep., April 2019.

[70] C. Gallese, "The ai act proposal: A new right to technical interpretability?" *SSRN Electronic Journal*, February 2023, available at SSRN: https://ssrn.com/abstract=4398206 or http://dx.doi.org/10.2139/ssrn.4398206.

[71] 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 (IHTC)*, 2017, pp. 197–201.

[72] T. Gebru, J. Morgenstern, B. Vecchione, J. Vaughan, H. Wallach, H. Daumé, and K. Crawford, "Datasheets for datasets," *Commun. ACM*, vol. 64, no. 12, pp. 86–92, nov 2021. [Online]. Available: https://doi.org/10.1145/3458723

[73] M. M. et.al., "Model cards for model reporting," in *Proceedings of the Conference on Fairness, Accountability, and Transparency*, ser. FAT* '19. New York, NY, USA: Association for Computing Machinery, 2019, pp. 220–229. [Online]. Available: https://doi.org/10.1145/3287560.3287596

[74] D. Pfau and A. Jung, “Engineering Trustworthy AI: A Developer Guide for Empirical Risk Minimization,” 2024. [Online]. Available: https://arxiv.org/abs/2410.19361

[75] E. Commission, C. Directorate-General for Communications Networks, and Technology, *The Assessment List for Trustworthy Artificial Intelligence (ALTAI) for self assessment.* Publications Office, 2020.

[76] Q. Yang, Y. Liu, Y. Cheng, Y. Kang, T. Chen, and H. Yu, *Vertical Federated Learning.* Cham: Springer International Publishing, 2020, pp. 69–81. [Online]. Available: https://doi.org/10.1007/978-3-031-01585-4_5