The A”alto Dictionary of Machine Learning

Alexander Jung and Konstantina Olioumtsevits

May 14, 2025

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

Acknowledgements

This dictionary of machine learning evolved through the development and teaching of several courses, including 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. These courses were offered at Aalto University `https://www.aalto.fi/en`, to adult learners via The Finnish Institute of Technology (FITech) `https://fitech.io/en/`, and to international students through the European University Alliance Unite! `https://www.aalto.fi/en/unite`.

We are grateful to the students who provided valuable feedback that helped shape this dictionary. Special thanks to Mikko Seesto for his meticulous proofreading.

Lists of Symbols

Sets and Functions

$a \in \mathcal{A}$	The object a is an element of the set $\mathcal{A}$.
$a := b$	We use a as a shorthand for b.
$\lvert\mathcal{A}\rvert$	The cardinality (i.e., 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 consisting of the two real numbers 0 and 1.
$[0,1]$	The closed interval of real numbers x with $0 \leq x \leq 1$.
$\operatorname{argmin}_{\mathbf{w}} f(\mathbf{w})$	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. Machine learning (ML) aims at finding (or learning) a function h (i.e., a 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 $\nabla f(\mathbf{w}) = \big(\frac{\partial f}{\partial w_1}, \ldots, \frac{\partial f}{\partial w_d}\big)^T \in \mathbb{R}^d$ [2, Ch. 9].

Matrices and Vectors

$\mathbf{x} = \big(x_1, \ldots, x_d\big)^T$	A vector of length d, with its j-th entry being x_j.
$\mathbb{R}^d$	The set of vectors $\mathbf{x} = \big(x_1, \ldots, x_d\big)^T$ consisting of d real-valued entries $x_1, \ldots, x_d \in \mathbb{R}$.
$\mathbf{I}_{l\times d}$	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 context, we drop the subscript.
$\|\mathbf{x}\|_2$	The Euclidean (or ℓ_2) norm of the vector $\mathbf{x} = \big(x_1, \ldots, x_d\big)^T \in \mathbb{R}^d$ defined 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 matrix that has the vector $\mathbf{x} \in \mathbb{R}^d$ as its single column.
$\mathbf{X}^T$	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 $\mathbf{w} \in \mathbb{R}^{d'}$.
$\text{span}\{\mathbf{B}\}$	The span of a matrix $\mathbf{B} \in \mathbb{R}^{a \times b}$, which is the subspace of all linear combinations of the columns of $\mathbf{B}$, $\text{span}\{\mathbf{B}\} = \left\{\mathbf{Ba} : \mathbf{a} \in \mathbb{R}^b\right\} \subseteq \mathbb{R}^a$.
$\det(\mathbf{C})$	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 random variable (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 an RV whose realizations are data points with features $\mathbf{x}$ and label y.
$p(\mathbf{x}\|y)$	A conditional probability distribution of an RV $\mathbf{x}$ given the value of another RV y [5, Sec. 3.5].
$p(\mathbf{x}; \mathbf{w})$	A parametrized probability distribution of an 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 random variable (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 (i.e., 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	The number of features that characterize a data point.
x_j	The j-th feature of a data point. The first feature 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.
$\mathbf{z}$	Instead of the symbol $\mathbf{x}$, we sometimes use $\mathbf{z}$ as another symbol to denote a vector whose entries are the individual features of a data point. We need two different symbols to distinguish between raw and learned 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 (or subset) of randomly chosen data points.
B	The size of (i.e., the number of data points in) a mini-batch.
y	The label (or 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 an ML method consists of all potential label values that a data point can carry. The nominal label space might be larger than the set of different label values arising in a given dataset (e.g., a training set). ML problems (or methods) using a numeric label space, such as $\mathcal{Y} = \mathbb{R}$ or $\mathcal{Y} = \mathbb{R}^3$, are referred to as regression problems (or methods). ML problems (or methods) that use a discrete label space, such as $\mathcal{Y} = \{0, 1\}$ or $\mathcal{Y} = \{cat, dog, mouse\}$, are referred to as classification problems (or methods).
η	Learning rate (or 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 an ML method. The hypothesis space consists of different hypothesis maps $h : \mathcal{X} \rightarrow \mathcal{Y}$, between which the ML method must choose.
$d_{\text{eff}}\left(\mathcal{H}\right)$	The effective dimension of a hypothesis space $\mathcal{H}$.
B^2	The squared bias of a learned hypothesis $\hat{h}$ produced by an ML method. The method is trained on data points that are modeled as the realizations of RVs. Since the data is a realization of RVs, the learned hypothesis $\hat{h}$ is also the realization of an RV.
V	The variance of the learned (parameters of the) hypothesis produced by an ML method. The method is trained on data points that are modeled as the realizations of RVs. Since the data is a realization of RVs, the learned hypothesis $\hat{h}$ is also the realization of an 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 by 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 multitask learning problem.
α	A regularization parameter that controls the amount of regularization.
$\lambda_j\big(\mathbf{Q}\big)$	The j-th eigenvalue (sorted in either ascending or descending order) of a positive semi-definite (psd) matrix $\mathbf{Q}$. We also use the shorthand λ_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 an 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 $\phi : \mathcal{X} \to \mathcal{X}' : \mathbf{x} \mapsto \mathbf{x}' := \phi\big(\mathbf{x}\big) \in \mathcal{X}'$.
$K\big(\cdot, \cdot\big)$	Given some feature space $\mathcal{X}$, a kernel is a map $K : \mathcal{X} \times \mathcal{X} \to \mathbb{C}$ that is psd.

Federated Learning

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

$\mathbf{x}^{(i,r)}$	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 $\mathcal{D}^{(i)}$.
$\mathbf{w}^{(i)}$	The local model parameters of device i within an FL network.
$L_i\left(\mathbf{w}\right)$	The local loss function used by device i to measure the usefulness of some choice $\mathbf{w}$ for the local model parameters.
$L^{(\mathrm{d})}\left(\mathbf{x}, h\big(\mathbf{x}\big), h'\big(\mathbf{x}\big)\right)$	The loss incurred by a 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 $\left(\big(\mathbf{w}^{(1)}\big)^T, \ldots, \big(\mathbf{w}^{(n)}\big)^T\right)^T \in \mathbb{R}^{dn}$ that is obtained by vertically stacking the local model parameters $\mathbf{w}^{(i)} \in \mathbb{R}^d$.

Machine Learning Concepts

k-fold cross-validation (k-fold CV) k-fold CV is a method for learning and validating a 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 the nearest mean) and, given the updated cluster assignments, re-calculating the cluster means [6, Ch. 8].

absolute error loss Consider a data point with features $\mathbf{x} \in \mathcal{X}$ and numeric label $y \in \mathbb{R}$. The absolute error loss incurred by a hypothesis $h : \mathcal{X} \to \mathbb{R}$ is defined as $|y - h(\mathbf{x})|$, i.e., the absolute difference between the prediction $h(\mathbf{x})$ and the true label y.

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 $h : \mathcal{X} \to \mathcal{Y}$, 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 $1 - (1/m) \sum_{r=1}^{m} L^{(0/1)}\left(\left(\mathbf{x}^{(r)}, y^{(r)}\right), h\right)$ using the 0/1 loss $L^{(0/1)}(\cdot, \cdot)$.

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

algebraic connectivity The algebraic connectivity of an undirected graph is the second-smallest eigenvalue λ_2 of its Laplacian matrix. A graph is connected if and only if $\lambda_2 > 0$.

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

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

that respects the constraints inherent to the computer executing the algorithm. Algorithms may be deterministic, where each input results uniquely in a single execution, or randomized, where executions can vary probabilistically. Randomized algorithms can thus be analyzed by modeling execution sequences as outcomes of random experiments,

viewing the algorithm as a stochastic process [5], [9], [10]. Crucially, an algorithm encompasses more than just a mapping from input to output; it also includes the intermediate computational steps $s_1, \ldots, s_T$.

application programming interface (API) An API is a formal mechanism that allows software components to interact in a structured and modular way [11]. In the context of ML, APIs are commonly used to provide access to a trained ML model. Users—whether humans or machines—can submit the feature vector of a data point and receive a corresponding prediction. Suppose a trained ML model is defined as $\widehat{h}(x) := 2x + 1$. Through an API, a user can input $x = 3$ and receive the output $\widehat{h}(3) = 7$ without knowledge of the detailed structure of the ML model or its training. In practice, the model is typically deployed on a server connected to the internet. Clients send requests containing feature values to the server, which responds with the computed prediction $\widehat{h}(\mathbf{x})$. APIs promote modularity in ML system design: one team can develop and train the model, while another handles integration and user interaction. Publishing a trained model via an API also offers practical advantages:

- The server can centralize computational resources which are required to compute predictions.
- The internal structure of the model remains hidden (useful for protecting IP or trade secrets).

However, APIs are not without risk: techniques such as model inversion can potentially reconstruct a model from its predictions on carefully

selected feature vectors.

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

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

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

backdoor A backdoor attack refers to the intentional manipulation of the training process underlying an 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 learned 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 anomalous prediction $\hat{h}(\mathbf{x})$ are only known to the attacker.

bagging Bagging (or bootstrap aggregation) is a generic technique to improve (the robustness of) a given 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. We then predict the label of a data point by combining or aggregating the individual predictions of each separate hypothesis. For hypothesis maps delivering numeric label values, this aggregation could be implemented by computing the average of individual predictions.

baseline Consider some ML method that produces a learned hypothesis (or trained model) $\hat{h} \in \mathcal{H}$. We evaluate the quality of a trained model by computing the average loss on a test set. But how can we assess whether the resulting test set performance is sufficiently good? How can we determine if the trained model performs close to optimal and there is little point in investing more resources (for data collection or computation) to improve it? To this end, it is useful to have a reference (or baseline) level against which we can compare the performance of the trained model. Such a reference value might be obtained from

human performance, e.g., the misclassification rate of dermatologists who diagnose cancer from visual inspection of skin [12]. Another source for a baseline is an existing, but for some reason unsuitable, ML method. For example, the existing ML method might be computationally too expensive for the intended ML application. Nevertheless, its test set error can still serve as a baseline. Another, somewhat more principled, approach to constructing a baseline is via a probabilistic model. In many cases, given a probabilistic model $p(\mathbf{x}, y)$, we can precisely determine the minimum achievable risk among any hypotheses (not even required to belong to the hypothesis space $\mathcal{H}$) [13]. This minimum achievable risk (referred to as the Bayes risk) is the risk of the Bayes estimator for the label y of a data point, given its features $\mathbf{x}$. Note that, for a given choice of loss function, the Bayes estimator (if it exists) is completely determined by the probability distribution $p(\mathbf{x}, y)$ [13, Ch. 4]. However, computing the Bayes estimator and Bayes risk presents two main challenges:

1) The probability distribution $p(\mathbf{x}, y)$ is unknown and needs to be estimated.

2) Even if $p(\mathbf{x}, y)$ is known, it can be computationally too expensive to compute the Bayes risk exactly [14].

A widely used probabilistic model is the multivariate normal distribution $(\mathbf{x}, y) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ for data points characterized by numeric features and labels. Here, for the squared error loss, the Bayes estimator is given by the posterior mean $\mu_{y|\mathbf{x}}$ of the label y, given the features $\mathbf{x}$ [13], [15].

The corresponding Bayes risk is given by the posterior variance $\sigma^2_{y|\mathbf{x}}$ (see Fig. 1).

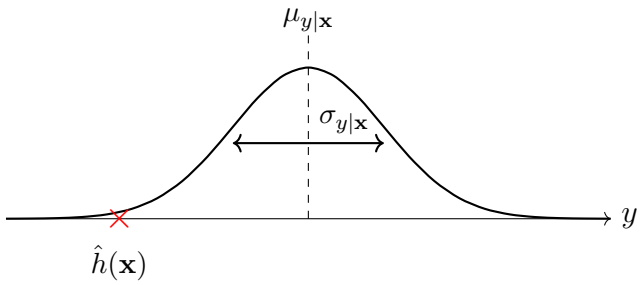


Figure 1: If the features and the label of a data point are drawn from a multivariate normal distribution, we can achieve the minimum risk (under squared error loss) by using the Bayes estimator $\mu_{y|\mathbf{x}}$ to predict the label y of a data point with features $\mathbf{x}$. The corresponding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

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

Bayes estimator Consider a probabilistic model with a joint probability distribution $p(\mathbf{x}, y)$ for the features $\mathbf{x}$ and label y of a data point. For a given loss function $L(\cdot, \cdot)$, we refer to a hypothesis h as a Bayes estimator if its risk $\mathbb{E}\{L((\mathbf{x}, y), h)\}$ is the minimum [13]. Note that the property of a hypothesis being a Bayes estimator depends on the

underlying probability distribution and the choice for the loss function $L(\cdot,\cdot)$.

Bayes risk Consider a probabilistic model with a joint probability distribution $p(\mathbf{x}, y)$ for the features $\mathbf{x}$ and label y of a data point. The Bayes risk is the minimum possible risk that can be achieved by any hypothesis $h : \mathcal{X} \to \mathcal{Y}$. Any hypothesis that achieves the Bayes risk is referred to as a Bayes estimator [13].

bias Consider an ML method using a parametrized hypothesis space $\mathcal{H}$. It learns the 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 independent and identically distributed (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 for a given 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 the category to which the corresponding data point belongs.

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 a real-valued hypothesis map $h(\mathbf{x}) \in \mathbb{R}$ as a 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) CFL trains local models for the devices in an federated learning (FL) application by using a clustering

assumption: The devices of a FL network form clusters. Two devices in the same cluster generate local datasets with similar statistical properties. CFL pools the local datasets of devices in the same cluster to obtain a training set for a cluster-specific model. Generalized total variation minimization (GTVMin) clusters devices implicitly by enforcing approximate similarity of model parameters across well-connected nodes of the FL network. See also: FL, clustering, GTVMin.

clustering Clustering methods decompose a given set of data points into a 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 to each other than those outside the cluster [16]. We obtain different clustering methods by using different notions of similarity between data points.

computational aspects By computational aspects of an ML method, we mainly refer to the computational resources required for its implementation. For example, if an ML method uses iterative optimization

techniques to solve ERM, then its computational aspects include: 1) how many arithmetic operations are needed to implement a single iteration (gradient step); and 2) how many iterations are needed to obtain useful model parameters. One important example of an iterative optimization technique is gradient descent (GD).

condition number The condition number $\kappa(\mathbf{Q}) \geq 1$ of a positive definite matrix $\mathbf{Q} \in \mathbb{R}^{d \times d}$ is the ratio α/β between the largest α and the smallest β eigenvalue of $\mathbf{Q}$. The condition number is useful for the analysis of ML methods. The computational complexity of gradient-based methods for linear regression crucially depends on the condition number of the matrix $\mathbf{Q} = \mathbf{X}\mathbf{X}^T$, with the feature matrix $\mathbf{X}$ of the training set. Thus, from a computational perspective, we prefer 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 a $k \times k$ 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 the label $y = c$ and the prediction $\hat{y} = c'$ assigned by the hypothesis h.

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

convex A subset $\mathcal{C} \subseteq \mathbb{R}^d$ of the Euclidean space $\mathbb{R}^d$ is referred to as convex if it

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

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

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

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

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

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

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

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

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

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

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

data augmentation Data augmentation methods add synthetic data points to an existing set of data points. These synthetic data points are obtained by perturbations (e.g., adding noise to physical measurements) or transformations (e.g., 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 (see Fig. 3). Data augmentation can be an efficient form of regularization.

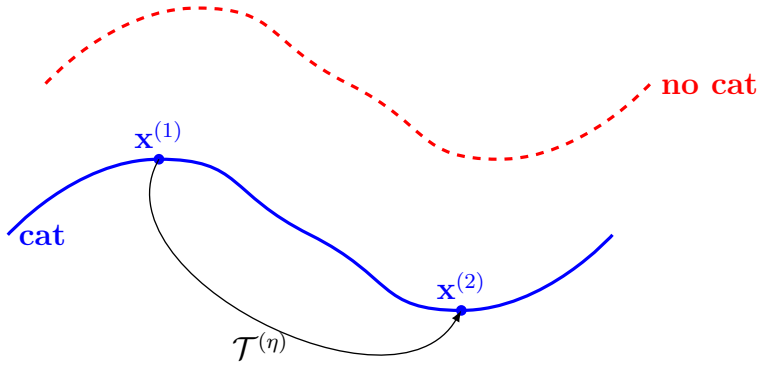


Figure 3: Data augmentation exploits intrinsic symmetries of data points in some feature space $\mathcal{X}$. We can represent a symmetry by an operator $\mathcal{T}^{(\eta)} : \mathcal{X} \to \mathcal{X}$, parametrized by some number $\eta \in \mathbb{R}$. For example, $\mathcal{T}^{(\eta)}$ might represent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

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

data normalization Data normalization refers to transformations applied to the feature vectors of data points to improve the ML method's statistical aspects or computational aspects. For example, in linear regression with gradient-based methods using a fixed learning rate, convergence depends on controlling the norm of feature vectors in the

training set. A common approach is to normalize feature vectors such that their norm does not exceed one [6, Ch. 5].

data point A data point is any object that conveys information [23]. 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 a label. 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 an ML model [24], [25]. The protection against data poisoning is particularly important in distributed ML applications where datasets are decentralized.

dataset A dataset refers to a collection of data points. These data points carry information about some quantity of interest (or label) within an ML application. ML methods use datasets for model training (e.g., via ERM) and model validation. Note that our notion of a dataset is very flexible, as it allows for very different types of data points. Indeed, data points can be concrete physical objects (such as humans or animals) or abstract objects (such as numbers). As a case in point, Fig. 4 depicts a

dataset that consists of cows as data points.

Figure 4: "Cows in the Swiss Alps" by User:Huhu Uet is licensed under [CC BY-SA 4.0](https://creativecommons.org/licenses/by-sa/4.0/)

Quite often, an ML engineer does not have direct access to a dataset. Indeed, accessing the dataset in Fig. 4 would require us to visit the cow herd in the Alps. Instead, we need to use an approximation (or representation) of the dataset which is more convenient to work with. Different mathematical models have been developed for the representation (or approximation) of datasets [26], [27], [28], [29]. One of the most widely adopted data model is the relational model, which organizes data as a table (or relation) [20], [26]. A table consists of rows and columns:

- Each row of the table represents a single data point.
- Each column of the table corresponds to a specific attribute of the data point. ML methods can use attributes as features and labels

of the data point.

For example, Table 1 shows a representation of the dataset in Fig. 4. In the relational model, the order of rows is irrelevant, and each attribute (i.e., column) must be precisely defined with a domain, which specifies the set of possible values. In ML applications, these attribute domains become the feature space and the label space.

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

Table 1: A relation (or table) that represents the dataset in Fig. 4.

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

decision boundary 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 neighborhood $\{\mathbf{x}' : \|\mathbf{x} - \mathbf{x}'\| \leq \varepsilon\}$, for any $\varepsilon > 0$, contains at least two vectors with different function values.

decision region Consider a hypothesis map h that delivers values from a finite set $\mathcal{Y}$. For each label value (category) $a \in \mathcal{Y}$, the hypothesis h determines a subset of feature values $\mathbf{x} \in \mathcal{X}$ that result in the same output $h(\mathbf{x}) = a$. We refer to this subset as a decision region of the 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 containing a root node that reads in the feature vector $\mathbf{x}$ of a data point. 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 child 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 forwarded 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).

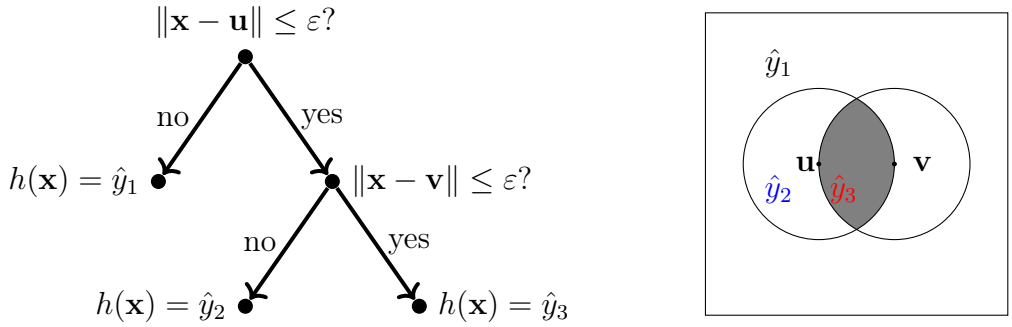


Figure 5: Left: A decision tree is a flow-chart-like representation of a piece-wise constant hypothesis $h : \mathcal{X} \to \mathbb{R}$. Each piece is a decision region $\mathcal{R}_{\hat{y}} := \big\{\mathbf{x} \in \mathcal{X} : h(\mathbf{x}) = \hat{y}\big\}$. The depicted decision tree can be applied to numeric feature vectors, i.e., $\mathcal{X} \subseteq \mathbb{R}^d$. It is parametrized by the threshold $\varepsilon > 0$ and the vectors $\mathbf{u}, \mathbf{v} \in \mathbb{R}^d$. Right: A decision tree partitions the feature space $\mathcal{X}$ into decision regions. Each decision region $\mathcal{R}_{\hat{y}} \subseteq \mathcal{X}$ corresponds to a specific leaf node in the decision tree.

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

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

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

for typical data points.

density-based spatial clustering of applications with noise (DBSCAN) DBSCAN refers to a clustering algorithm for data points that are characterized by numeric feature vectors. Like k-means and soft clustering via GMM, also DBSCAN uses the Euclidean distances between feature vectors to determine the clusters. However, in contrast to k-means and GMM, DBSCAN uses a different notion of similarity between data points. 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.

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

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

differential privacy (DP) Consider some ML method $\mathcal{A}$ that reads in a dataset (e.g., the training set used for ERM) and delivers some output $\mathcal{A}(\mathcal{D})$. The output could be either the learned model parameters or the predictions for specific data points. DP is a precise measure of privacy leakage incurred by revealing the output. Roughly speaking, an ML

method is differentially private if the probability distribution of the output $\mathcal{A}(\mathcal{D})$ does not change too much if the sensitive attribute of one data point in the training set is changed. Note that DP builds on a probabilistic model for an ML method, i.e., we interpret its output $\mathcal{A}(\mathcal{D})$ as the realization of an RV. The randomness in the output can be ensured by intentionally adding the realization of an auxiliary RV (noise) to the output of the ML method.

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

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

computations and message-passing events. A generic execution might look as follows:

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

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

dual norm Every norm $\|\cdot\|$ defined on an Euclidean space $\mathbb{R}^d$ has an associated dual norm, denoted $\|\cdot\|_*$, defined as $\|\mathbf{y}\|_* := \sup_{\|\mathbf{x}\|\leq 1} \mathbf{y}^T\mathbf{x}$. The dual norm measures the largest possible inner product between $\mathbf{y}$ and any vector in the unit ball of the original norm. For further details, see [17, Sec. A.1.6].

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

effective dimension The effective dimension $d_{\text{eff}}(\mathcal{H})$ of an infinite hypothesis space $\mathcal{H}$ is a measure of its size. Loosely speaking, the effective dimension is equal to the effective number of independent tunable model

parameters. These parameters might be the coefficients used in a linear map or the weights and bias terms of an ANN.

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

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

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

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

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

empirical risk The empirical risk $\widehat{L}\big(h|\mathcal{D}\big)$ of a 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 (ERM) Empirical risk minimization is the optimization problem of finding a hypothesis (out of a model) with the minimum average loss (or empirical risk) on a given dataset $\mathcal{D}$ (i.e., the training set). Many ML methods are obtained from empirical risk via specific design choices for the dataset, model, and loss [6, Ch. 3].

estimation error Consider data points, each with feature vector $\mathbf{x}$ and label y. In some applications, we can model the relation between the feature

vector and the label of a data point as $y = \bar{h}(\mathbf{x}) + \varepsilon$. Here, we use some true underlying hypothesis $\bar{h}$ and a noise term ε which summarizes any modeling or labeling errors. The estimation error incurred by an 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 parametric hypothesis space, which consists of hypothesis maps determined by model parameters $\mathbf{w}$, we can define the estimation error as $\Delta\mathbf{w} = \widehat{\mathbf{w}} - \overline{\mathbf{w}}$ [34], [35].

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

expectation Consider a numeric feature vector $\mathbf{x} \in \mathbb{R}^d$ which we interpret as the realization of an RV with a 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], [36], [37]. Note that the expectation is only defined if this integral exists, i.e., if the RV is integrable.

expectation-maximization (EM) Consider a probabilistic model $p(\mathbf{z}; \mathbf{w})$ for the data points $\mathcal{D}$ generated in some ML application. The maximum likelihood estimator for the model parameters $\mathbf{w}$ is obtained by maximizing $p(\mathcal{D}; \mathbf{w})$. However, the resulting optimization problem might be computationally challenging. Expectation-maximization approximates the maximum likelihood estimator by introducing a latent RV $\mathbf{z}$ such that maximizing $p(\mathcal{D}, \mathbf{z}; \mathbf{w})$ would be easier [35], [38], [39]. Since we do not observe $\mathbf{z}$, we need to estimate it from the observed dataset $\mathcal{D}$

using a conditional expectation. The resulting estimate $\widehat{\mathbf{z}}$ is then used to compute a new estimate $\widehat{\mathbf{w}}$ by solving $\max_{\mathbf{w}} p(\mathcal{D}, \widehat{\mathbf{z}}; \mathbf{w})$. The crux is that the conditional expectation $\widehat{\mathbf{z}}$ depends on the model parameters $\widehat{\mathbf{w}}$, which we have updated based on $\widehat{\mathbf{z}}$. Thus, we have to re-calculate $\widehat{\mathbf{z}}$, which, in turn, results in a new choice $\widehat{\mathbf{w}}$ for the model parameters. In practice, we repeat the computation of the conditional expectation (i.e., the E-step) and the update of the model parameters (i.e., the M-step) until some stopping criterion is met.

expert ML aims to learn 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 minimal loss on any data point. We can make this informal goal precise via the independent and identically distributed assumption (i.i.d. assumption) and by using the Bayes risk as the baseline for the (average) loss of a hypothesis. An alternative approach to obtaining a baseline is to use the hypothesis h' learned by an existing ML method. We refer to this hypothesis h' as an expert [40]. Regret minimization methods learn a hypothesis that incurs a loss comparable to the best expert [40], [41].

explainability We define the (subjective) explainability of an ML method as the level of simulatability [42] of the predictions delivered by an 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 [42], [43].

Alternatively, we can use probabilistic models for data and measure the explainability of a trained ML model via the conditional (differential) entropy of its predictions, given the user predictions [44], [45].

explainable empirical risk minimization (EERM) Explainable ERM is an instance of structural risk minimization (SRM) that adds a regularization term to the average loss in the objective function of ERM. The regularization term is chosen to favor 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 [43].

explainable machine learning (explainable ML) Explainable ML methods aim at complementing each prediction with an explanation of 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 [46].

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 [47]. We can also use visual forms of explanations, such as intensity plots for image classification [48].

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 digital image (e.g., stored as a `.jpeg`

file), then we could use the red-green-blue intensities of its pixels as features. Domain-specific synonyms for the term feature are "covariate," "explanatory variable," "independent variable," "input (variable)," "predictor (variable)," or "regressor" [49], [50], [51].

feature learning Consider an ML application with data points characterized by raw features $\mathbf{x} \in \mathcal{X}$. Feature learning refers to the task of learning a map

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

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

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

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

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

feature map Feature map refers to 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 arrangement of data points might become simpler (or

more linear) in the new feature space, allowing the use of linear models in the new features. This idea is a main driver for the development of kernel methods [52]. Moreover, the hidden layers of a deep net can be interpreted as a trainable feature map followed by a linear model in the form of the output layer. Another reason for learning a feature map could be that learning a small number of new features helps to avoid overfitting and ensures interpretability [53]. 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 with feature vectors $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)} \in \mathbb{R}^d$. It is convenient to collect the individual feature vectors into a feature matrix $\mathbf{X} := \big(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)}\big)^T$ of size $m \times d$.

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 the dimension d being the number of individual features of a data point.

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

see kernel method).

FedAvg A FL algorithm using a server-client setting.

See also: FL, algorithm.

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

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

federated learning network (FL network) A FL network is an undirected weighted graph whose nodes represent data generators that aim to train a local (or personalized) model. Each node in a FL network represents some device capable of collecting 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.

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

FedRelax A distributed FL algorithm.

See also: FL, algorithm.

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

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

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

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

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

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

A Gaussian random vector $\mathbf{x} \in \mathbb{R}^d$ with covariance matrix $\mathbf{C}$ and mean $\boldsymbol{\mu}$ can be constructed via $\mathbf{x} := \mathbf{A}\big(\mathbf{z} + \boldsymbol{\mu}\big)$. Here, $\mathbf{A}$ is any matrix that

satisfies $\mathbf{A}\mathbf{A}^T = \mathbf{C}$ and $\mathbf{z} := \big(z_1, \ldots, z_d\big)^T$ is a vector whose entries are i.i.d. standard Gaussian RVs $z_1, \ldots, z_d$. Gaussian random processes generalize Gaussian random vectors by applying linear transformations to infinite sequences of standard Gaussian RVs [58].

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

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

- Data minimization principle: ML systems should only use the necessary amount of personal data for their purpose.
- Transparency and explainability: ML systems should enable their users to understand how the systems make decisions that impact the users.
- Data subject rights: Users should get an opportunity to access, rectify, and delete their personal data, as well as to object to automated decision-making and profiling.
- Accountability: Organizations must ensure robust data security and demonstrate compliance through documentation and regular

audits.

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

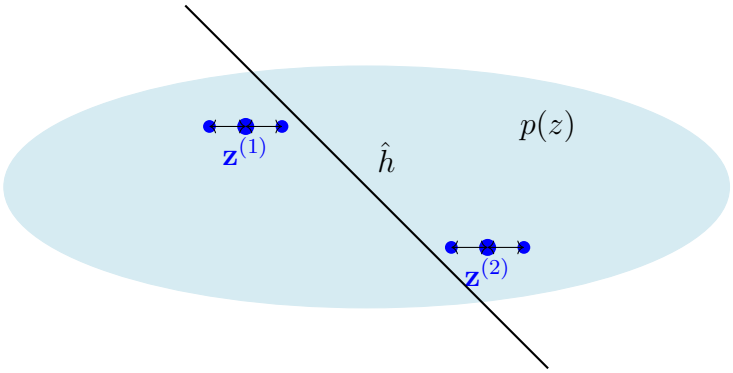


Figure 6: Two data points $\mathbf{z}^{(1)}, \mathbf{z}^{(2)}$ that are used as a training set to learn a hypothesis $\hat{h}$ via ERM. We can evaluate $\hat{h}$ outside $\mathcal{D}^{(\text{train})}$ either by an i.i.d. assumption with some underlying probability distribution $p(\mathbf{z})$ or by perturbing the data points.

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

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

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

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

geometric median (GM) The GM of a set of input vectors $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)}$ in $\mathbb{R}^d$ is a point $\mathbf{z} \in \mathbb{R}^d$ that minimizes the sum of distances to the vectors [17],

$$\mathbf{z} \in \operatorname*{argmin}_{\mathbf{y} \in \mathbb{R}^d} \sum_{r=1}^{m} \left\| \mathbf{y} - \mathbf{x}^{(r)} \right\|_2 . \tag{2}$$

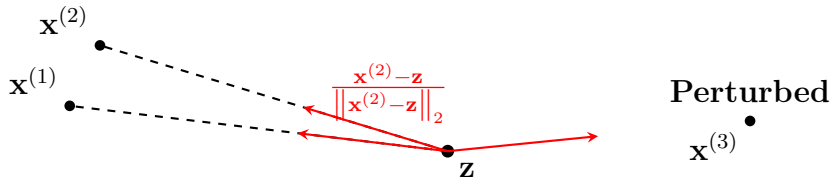


Figure 7: Consider a solution $\mathbf{z}$ of (2) that does not coincide with any of the input vectors. The optimality condition for (2) requires that the unit vectors from $\mathbf{z}$ to the input vectors sum to zero.

Figure 7 illustrates a fundamental property of the GM: If $\mathbf{z}$ does not coincide with any of the input vectors, then the unit vectors pointing from $\mathbf{z}$ to each $\mathbf{x}^{(r)}$ must sum to zero - this is the zero-subgradient (optimality) condition of (2). It turns out that the solution to (2) cannot be arbitrarily pulled away from trustworthy input vectors as long as they are the majority [63, Thm. 2.2.].

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

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} - \eta \nabla f(\mathbf{w}^{(k)}) \tag{3}$$

with a sufficiently small step size $\eta > 0$. Fig. 8 illustrates the effect of a single gradient descent step (3).

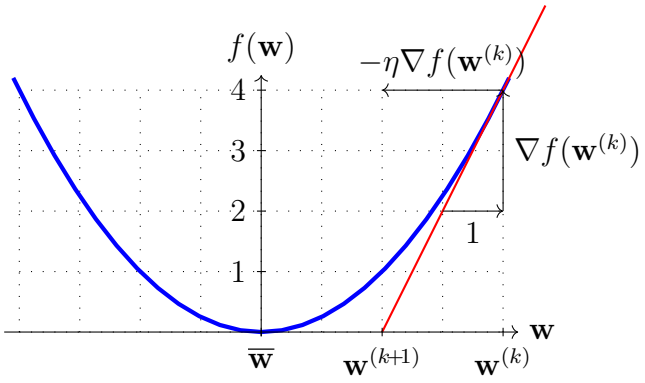


Figure 8: A single gradient step (3) towards the minimizer $\overline{\mathbf{w}}$ of $f(\mathbf{w})$.

gradient step Given a differentiable real-valued function $f(\cdot) : \mathbb{R}^d \to \mathbb{R}$ and a vector $\mathbf{w} \in \mathbb{R}^d$, the gradient step updates $\mathbf{w}$ by adding the scaled negative gradient $\nabla f(\mathbf{w})$ to obtain the new vector (see Fig. 9)

$$\widehat{\mathbf{w}} := \mathbf{w} - \eta\nabla f(\mathbf{w}). \tag{4}$$

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

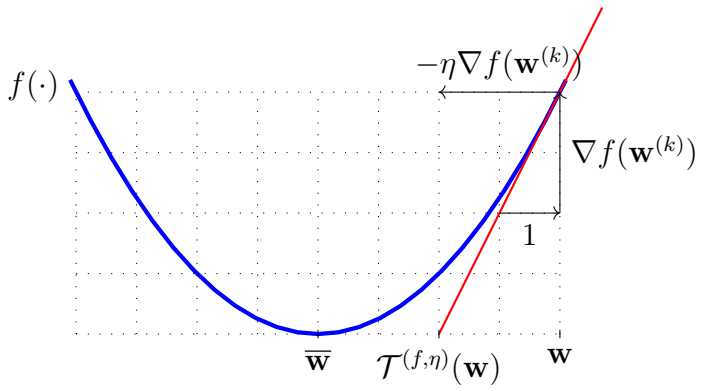


Figure 9: The basic gradient step (4) maps a given vector $\mathbf{w}$ to the updated vector $\mathbf{w}'$. It defines an operator $\mathcal{T}^{(f,\eta)}(\cdot) : \mathbb{R}^d \to \mathbb{R}^d : \mathbf{w} \mapsto \widehat{\mathbf{w}}$.

Note that the gradient step (4) optimizes locally - in a neighborhood whose size is determined by the step size η - a linear approximation to the function $f(\cdot)$. A natural generalization of (4) is to locally optimize the function itself - instead of its linear approximation - such that

$$\widehat{\mathbf{w}} = \operatorname*{argmin}_{\mathbf{w}' \in \mathbb{R}^d} f(\mathbf{w}') + (1/\eta) \left\| \mathbf{w} - \mathbf{w}' \right\|_2^2 . \tag{5}$$

We intentionally use the same symbol η for the parameter in (5) as we used for the step size in (4). The larger the η we choose in (5), the more progress the update will make towards reducing the function value $f(\widehat{\mathbf{w}})$. Note that, much like the gradient step (4), also the update (5) defines a (typically non-linear) operator that is parametrized by the function $f(\cdot)$ and the parameter η. For a convex function $f(\cdot)$, this operator is known as the proximal operator of $f(\cdot)$ [64].

gradient-based methods 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 (or maximum) value of the objective function. As their name indicates, gradient-based methods use the gradients of the objective function evaluated during previous iterations to construct new, (hopefully) improved model parameters. One important example of a gradient-based method is GD.

graph A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a pair that consists of a node set $\mathcal{V}$ and an edge set $\mathcal{E}$. In its most general form, a graph is specified by a map that assigns each edge $e \in \mathcal{E}$ a pair of nodes [65]. One important family of graphs is simple undirected graphs. A simple undirected graph is obtained by identifying each edge $e \in \mathcal{E}$ with two different nodes $\{i, i'\}$. Weighted graphs also specify numeric weights A_e for each edge $e \in \mathcal{E}$.

graph clustering Graph clustering aims at clustering data points that are represented as the nodes of a graph $\mathcal{G}$. The edges of $\mathcal{G}$ represent pairwise similarities between data points. Sometimes we can quantify the extend of these similarities by an edge weight [56], [66].

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

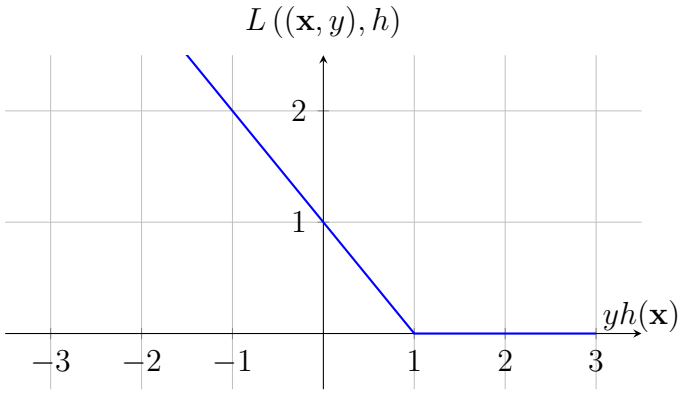
high-dimensional regime The high-dimensional regime of ERM is characterized by the effective dimension of the model being larger than the

sample size, i.e., the number of (labeled) 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 is large ANNs, which have far more tunable weights (and bias terms) than the total 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 [67], [68].

Hilbert space A Hilbert space is a complete inner product space [69]. That is, it is a linear vector space equipped with an inner product between pairs of vectors, and it satisfies the additional requirement of completeness: every Cauchy sequence of vectors converges to a limit within the space. A canonical example of a Hilbert space is the Euclidean space $\mathbb{R}^d$, for some dimension d, consisting of vectors $\mathbf{u} = \big(u_1, \ldots, u_d\big)^T$ and the standard 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

$$L\left((\mathbf{x}, y), h\right) := \max\{0, 1 - yh(\mathbf{x})\}. \tag{6}$$



A regularized variant of the hinge loss is used by the support vector machine (SVM) [70].

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

horizontal federated learning (HFL) HFL uses local datasets constituted by different data points but uses the same features to characterize them [71]. For example, weather forecasting uses a network of spatially distributed weather (observation) stations. Each weather station measures the same quantities, such as daily temperature, air pressure, and precipitation. However, different weather stations measure the characteristics or features of different spatio-temporal regions. Each spatio-temporal region represents an individual data point, each characterized by the same features (e.g., daily temperature or air pressure).

See also: FL, vertical federated learning (VFL), clustered federated learning (CFL)

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

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

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

hypothesis space Every practical ML method uses a hypothesis space (or model) $\mathcal{H}$. The hypothesis space of an ML method is a subset of all possible maps from the feature space to the 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 an (approximately) linear relation between a set of features and a label, a useful choice for the hypothesis space might be the linear model.

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

independent and identically distributed assumption (i.i.d. assumption) The i.i.d. assumption interprets data points of a dataset as the realizations of i.i.d. RVs.

interpretability An 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 [44].

kernel Consider data points characterized by a feature vector $\mathbf{x} \in \mathcal{X}$ with a generic feature space $\mathcal{X}$. A (real-valued) kernel $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ assigns each pair of feature vectors $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$ a real number $K\big(\mathbf{x}, \mathbf{x}'\big)$. The value $K\big(\mathbf{x}, \mathbf{x}'\big)$ is often interpreted as a measure for the similarity between $\mathbf{x}$ and $\mathbf{x}'$. Kernel methods use a kernel to transform the feature vector $\mathbf{x}$ to a new feature vector $\mathbf{z} = K\big(\mathbf{x}, \cdot\big)$. This new feature vector belongs to a linear feature space $\mathcal{X}'$ which is (in general) different from the original feature space $\mathcal{X}$. The feature space $\mathcal{X}'$ has a specific mathematical structure, i.e., it is a reproducing kernel Hilbert space [52], [70].

kernel method A kernel method is an ML method that uses a kernel K to map the original (raw) feature vector $\mathbf{x}$ of a data point to a new

(transformed) feature vector $\mathbf{z} = K\big(\mathbf{x}, \cdot\big)$ [52], [70]. The motivation for transforming the feature vectors is that, by using a suitable kernel, the data points have a "more pleasant" geometry in the transformed feature space. For example, in a binary classification problem, using transformed feature vectors $\mathbf{z}$ might allow us to use linear models, even if the data points are not linearly separable in the original feature space (see Fig. 10).

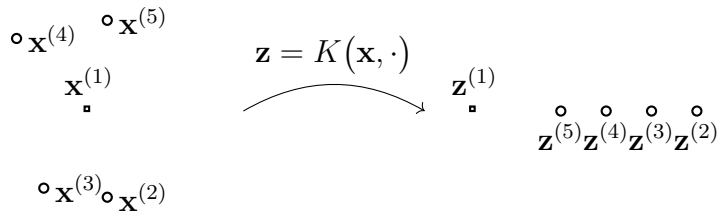


Figure 10: Five data points characterized by feature vectors $\mathbf{x}^{(r)}$ and labels $y^{(r)} \in \{\circ, \square\}$, for $r = 1, \ldots, 5$. With these feature vectors, there is no way to separate the two classes by a straight line (representing the decision boundary of a linear classifier). In contrast, the transformed feature vectors $\mathbf{z}^{(r)} = K\big(\mathbf{x}^{(r)}, \cdot\big)$ allow us to separate the data points using a linear classifier.

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

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" [49], [50], [51].

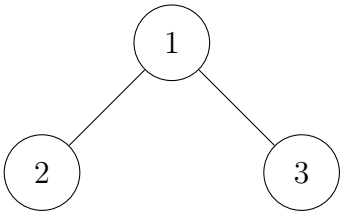
label space Consider an ML application that involves data points characterized by features and labels. The label space is constituted by all potential values that the label of a data point can take on. Regression methods, aiming at predicting numeric labels, often use the label space $\mathcal{Y} = \mathbb{R}$. Binary classification methods use a label space that consists of two different elements, e.g., $\mathcal{Y} = \{-1, 1\}$, $\mathcal{Y} = \{0, 1\}$, or $\mathcal{Y} = \{$"cat image","no cat image"$\}$.

labeled datapoint A data point whose label is known or has been determined by some means which might require human labor.

Laplacian matrix The structure of a graph $\mathcal{G}$, with nodes $i = 1, \ldots, n$, can be analyzed using the properties of special matrices that are associated with $\mathcal{G}$. One such matrix is the graph Laplacian matrix $\mathbf{L}^{(\mathcal{G})} \in \mathbb{R}^{n \times n}$, which is defined for an undirected and weighted graph [66], [72]. It is defined element-wise as (see Fig. 11)

$$L_{i,i'}^{(\mathcal{G})} := \begin{cases} -A_{i,i'} & \text{for } i \neq i', \{i, i'\} \in \mathcal{E}, \\ \sum_{i'' \neq i} A_{i,i''} & \text{for } i = i', \\ 0 & \text{else.} \end{cases} \tag{7}$$

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



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

Figure 11: Left: Some undirected graph $\mathcal{G}$ with three nodes $i = 1, 2, 3$. Right: The Laplacian matrix $\mathbf{L}^{(\mathcal{G})} \in \mathbb{R}^{3\times 3}$ of $\mathcal{G}$.

large language model (LLM) Large language models is an umbrella term for ML methods that process and generate human-like text. These methods typically use deep nets with billions (or even trillions) of parameters. A widely used choice for the network architecture is referred to as Transformers [73]. The training of large language models is often based on the task of predicting a few words that are intentionally removed from a large text corpus. Thus, we can construct labeled datapoints simply by selecting some words of a text as labels and the remaining words as features of data points. This construction requires very little human supervision and allows for generating sufficiently large training sets for large language models.

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

learning rate Consider an iterative ML 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. One well-known example of such an iterative learning method is GD and its variants, SGD and projected gradient descent (projected GD). A key parameter of an iterative method is the learning rate. The learning rate controls the extent to which the current hypothesis can be modified during a single iteration. A well-known example of such a parameter is the step size used in GD [6, Ch. 5].

learning task Consider a 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 [74], [75], [76].

least absolute deviation regression Least absolute deviation regression

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

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

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

linear model Consider 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,

$$\mathcal{H}^{(d)} := \left\{ h(\mathbf{x}) = \mathbf{w}^T\mathbf{x} : \mathbf{w} \in \mathbb{R}^d \right\}. \tag{8}$$

Note that (8) 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 (e.g., reducing d means less computation), statistical aspects (e.g., increasing d might reduce prediction error), and interpretability. A linear model using few carefully chosen features tends to be considered more interpretable [46], [53].

linear regression Linear regression aims to learn a linear hypothesis map to predict a numeric label based on the 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 datapoints, 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 similarity. These similarities might arise from probabilistic models or communication infrastructure and are encoded in the edges of an FL network.

local interpretable model-agnostic explanations (LIME) Consider a trained model (or learned hypothesis) $\widehat{h} \in \mathcal{H}$, which maps the feature vector of a data point to the prediction $\widehat{y} = \widehat{h}$. Local interpretable model-agnostic explanations is a technique for explaining the behavior of $\widehat{h}$, locally around a data point with feature vector $\mathbf{x}^{(0)}$ [53]. The explanation is given in the form of a local approximation $g \in \mathcal{H}'$ of $\widehat{h}$ (see Fig. 12). This approximation can be obtained by an instance of ERM with carefully designed training set. In particular, the training set consists of data points with feature vector $\mathbf{x}$ close to $\mathbf{x}^{(0)}$ and the (pseudo-)label $\widehat{h}(\mathbf{x})$. Note that we can use a different model $\mathcal{H}'$ for the approximation from the original model $\mathcal{H}$. For example, we can use a decision tree to approximate (locally) a deep net. Another widely-used

choice for $\mathcal{H}'$ is the linear model.

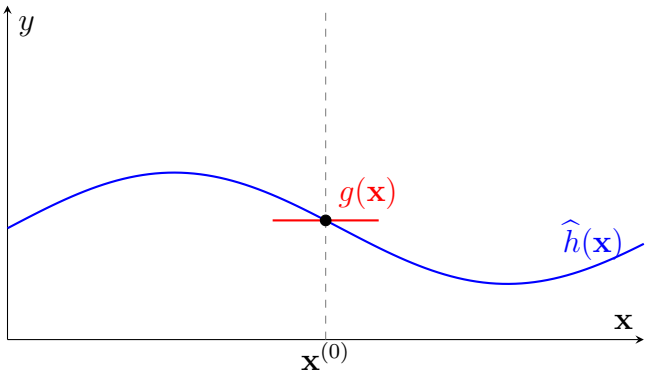


Figure 12: To explain a trained model $\widehat{h} \in \mathcal{H}$, around a given feature vector $\mathbf{x}^{(0)}$, we can use a local approximation $g \in \mathcal{H}'$.

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

logistic loss Consider a 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

$$L\left((\mathbf{x}, y), h\right) := \log(1 + \exp(-yh(\mathbf{x}))). \tag{9}$$

Carefully note that the expression (9) for the logistic loss applies only for the label space $\mathcal{Y} = \{-1, 1\}$ and when using the thresholding rule (1).

logistic regression Logistic regression learns a linear hypothesis map (or classifier) $h(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$ to predict a binary label y based on the 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 datapoints (i.e., 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 a slight abuse of notation, we use the term loss for both the loss function L itself and the specific value $L\left(\mathbf{z}, h\right)$, for a data point $\mathbf{z}$ and hypothesis h.

loss function A loss function is a map

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

It assigns a non-negative real number (i.e., the loss) $L\left((\mathbf{x}, y), h\right)$ to a pair that consists of a data point, with features $\mathbf{x}$ and label y, and a hypothesis $h \in \mathcal{H}$. The value $L\left((\mathbf{x}, y), h\right)$ quantifies the discrepancy between the true label y and the prediction $h(\mathbf{x})$. Lower (closer to zero) values $L\left((\mathbf{x}, y), h\right)$ indicate a smaller discrepancy between prediction $h(\mathbf{x})$ and label y. Fig. 13 depicts a loss function for a given data point, with features $\mathbf{x}$ and label y, as a function of the hypothesis $h \in \mathcal{H}$.

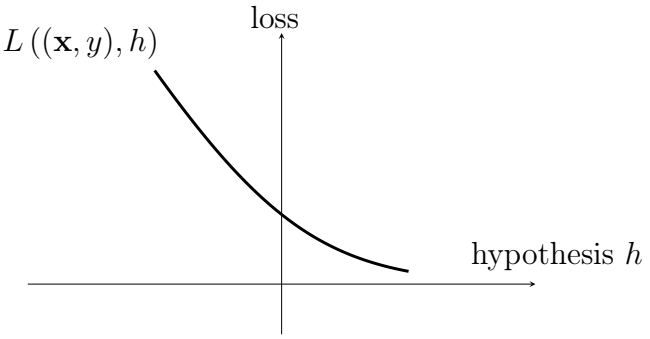


Figure 13: Some loss function $L\left((\mathbf{x}, y), h\right)$ for a fixed data point, with feature vector $\mathbf{x}$ and label y, and a varying hypothesis h. ML methods try to find (or learn) a hypothesis that incurs minimal loss.

machine learning (ML) ML aims to predict a label from the features of a data point. ML methods achieve this by learning a hypothesis from a hypothesis space (or model) through the minimization of a loss function [6], [77]. One precise formulation of this principle is ERM. Different ML methods are obtained from different design choices for data points (their features and label), model, and loss function [6, Ch. 3].

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

maximum likelihood Consider data points $\mathcal{D} = \left\{\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(m)}\right\}$ that are interpreted as the realizations of i.i.d. RVs with a common probability

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

mean The mean of a RV $\mathbf{x}$, taking values in a Euclidean space $\mathbb{R}^d$, is its expectation $\mathbb{E}\{\mathbf{x}\}$. It is defined as the Lebesgue integral of $\mathbf{x}$ with respect to the underlying probability distribution P,

$$\mathbb{E}\{\mathbf{x}\} = \int_{\mathbb{R}^d} \mathbf{x}\, \mathrm{d}P(\mathbf{x}),$$

see, e.g., [37] or [2]. We also use the term to refer to the average of a finite sequence $\mathbf{x}^{(1)},\ldots,\mathbf{x}^{(m)} \in \mathbb{R}^d$. However, these two definitions are essentially the same. Indeed, we can use the sequence $\mathbf{x}^{(1)},\ldots,\mathbf{x}^{(m)} \in \mathbb{R}^d$ to construct a discrete RV $\widetilde{\mathbf{x}} = \mathbf{x}^{(I)}$ with the index I chosen uniformly at random from the set $\{1,\ldots,m\}$. The mean of $\widetilde{\mathbf{x}}$ is precisely the average $\frac{1}{m}\sum_{r=1}^{m} \mathbf{x}^{(r)}$.

mean squared estimation error (MSEE) Consider an 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 an 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 [13], [34].

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 [78]. We can interpret data imputation as an 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 employed by an ML method [6], [60].

model inversion TBD.

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

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: 1) training each candidate model; 2) computing the validation error for each trained model; and 3) choosing the model with the smallest validation error [6, Ch. 6].

multi-armed bandit (MAB) A MAB problem models a repeated decision-making scenario in which, at each time step k, a learner must choose one out of several possible actions, often referred to as arms, from a finite set $\mathcal{A}$. Each arm $a \in \mathcal{A}$ yields a stochastic reward $r^{(a)}$ drawn from an unknown probability distribution with mean $\mu^{(a)}$. The learner's

goal is to maximize the cumulative reward over time by strategically balancing exploration (gathering information about uncertain arms) and exploitation (selecting arms known to perform well). This balance is quantified by the notion of regret, which measures the performance gap between the learner's strategy and the optimal strategy that always selects the best arm. MAB problems form a foundational model in online learning, reinforcement learning, and sequential experimental design [79].

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 two labels. One label indicates the presence of a human in this picture and another label indicates the presence of a car.

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

multivariate normal distribution The multivariate normal distribution $\mathcal{N}(\mathbf{m}, \mathbf{C})$ is an important family of probability distributions for a continuous RV $\mathbf{x} \in \mathbb{R}^d$ [5], [15], [80]. This family is parametrized by the mean $\mathbf{m}$ and the covariance matrix $\mathbf{C}$ of $\mathbf{x}$. If the covariance matrix is

invertible, the probability distribution of $\mathbf{x}$ is

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

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

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

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

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

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

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

networked data Networked data consists of local datasets that are related by some notion of pairwise similarity. We can represent networked data using a graph whose nodes carry local datasets and edges encode

pairwise similarities. One example of networked data arises in FL applications where local datasets are generated by spatially distributed devices.

networked exponential families (nExpFam) A collection of exponential families, each of them assigned to a node of an FL network. The model parameters are coupled via the network structure by requiring them to have a small GTV [81].

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

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

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

non-smooth We refer to a function as non-smooth if it is not smooth [82].

norm A norm is a function that maps each (vector) element of a linear vector space to a non-negative real number. This function must be homogeneous and definite, and it must satisfy the triangle inequality [83].

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 $h^{(\mathbf{w})}$.

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

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

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

online gradient descent (online GD) Consider an ML method that learns model parameters $\mathbf{w}$ from some parameter space $\mathcal{W} \subseteq \mathbb{R}^d$. The learning process uses data points $\mathbf{z}^{(t)}$ that arrive at consecutive time-instants

$t = 1, 2, \ldots$. Let us interpret the data points $\mathbf{z}^{(t)}$ as i.i.d. copies of an RV $\mathbf{z}$. The risk $\mathbb{E}\{L\left(\mathbf{z}, \mathbf{w}\right)\}$ of a hypothesis $h^{(\mathbf{w})}$ can then (under mild conditions) be obtained as the limit $\lim_{T\to\infty}(1/T)\sum_{t=1}^{T} L\left(\mathbf{z}^{(t)}, \mathbf{w}\right)$. We might use this limit as the objective function for learning the model parameters $\mathbf{w}$. Unfortunately, this limit can only be evaluated if we wait infinitely long in order to collect all data points. Some ML applications require methods that learn online: as soon as a new data point $\mathbf{z}^{(t)}$ arrives at time t, we update the current model parameters $\mathbf{w}^{(t)}$. Note that the new data point $\mathbf{z}^{(t)}$ contributes the component $L\left(\mathbf{z}^{(t)}, \mathbf{w}\right)$ to the risk. As its name suggests, online GD updates $\mathbf{w}^{(t)}$ via a (projected) gradient step

$$\mathbf{w}^{(t+1)} := P_{\mathcal{W}}\big(\mathbf{w}^{(t)} - \eta_t \nabla_{\mathbf{w}} L\left(\mathbf{z}^{(t)}, \mathbf{w}\right)\big). \tag{10}$$

Note that (10) is a gradient step for the current component $L\left(\mathbf{z}^{(t)}, \cdot\right)$ of the risk. The update (10) ignores all the previous components $L\left(\mathbf{z}^{(t')}, \cdot\right)$, for $t' < t$. It might therefore happen that, compared to $\mathbf{w}^{(t)}$, the updated model parameters $\mathbf{w}^{(t+1)}$ increase the retrospective average loss $\sum_{t'=1}^{t-1} L\left(\mathbf{z}^{(t')}, \cdot\right)$. However, for a suitably chosen learning rate η_t, online GD can be shown to be optimal in practically relevant settings. By optimal, we mean that the model parameters $\mathbf{w}^{(T+1)}$ delivered by online GD after observing T data points $\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(T)}$ are at least as good as those delivered by any other learning method [41], [84].

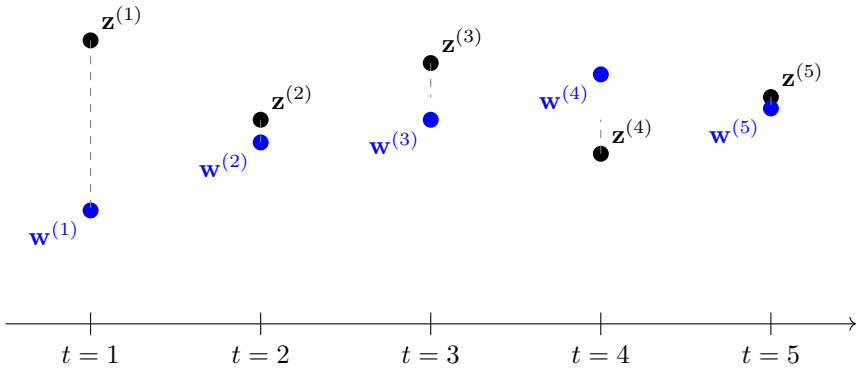


Figure 14: An instance of online GD that updates the model parameters $\mathbf{w}^{(t)}$ using the data point $\mathbf{z}^{(t)} = x^{(t)}$ arriving at time t. This instance uses the squared error loss $L\left(\mathbf{z}^{(t)}, w\right) = (x^{(t)} - w)^2$.

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

performance criterion. Two examples of ML methods that use such an optimistic construction of an objective function are SRM [60, Ch. 11] and upper confidence bound (UCB) methods for sequential decision making [79, Sec. 2.2].

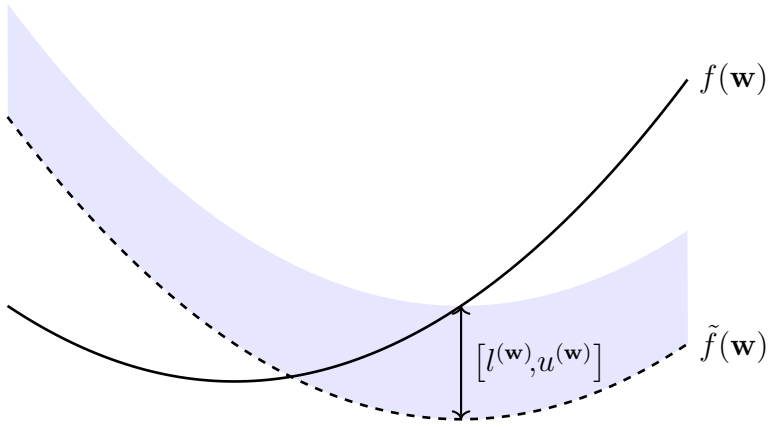


Figure 15: ML methods learn model parameters $\mathbf{w}$ by using some estimate of $f(\mathbf{w})$ for the ultimate performance criterion $\bar{f}(\mathbf{w})$. Using a probabilistic model, one can use $f(\mathbf{w})$ to construct confidence intervals $\left[l^{(\mathbf{w})}, u^{(\mathbf{w})}\right]$ which contain $\bar{f}(\mathbf{w})$ with high probability. The best plausible performance measure for a specific choice $\mathbf{w}$ of model parameters is $\tilde{f}(\mathbf{w}) := l^{(\mathbf{w})}$.

outlier Many 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 (or time-invariant) [85]. However, in some applications the data consists of a majority of regular data points that conform with an

i.i.d. assumption as well as 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 most data points as an outlier. Different methods for outlier detection use different measures for this deviation. Stastistical learning theory studies fundamental limits on the ability to mitigate outliers reliably [86], [87].

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

parameter space The parameter space $\mathcal{W}$ of an ML model $\mathcal{H}$ is the set of all feasible choices for the model parameters (see Fig. 16). Many important ML methods use a model that is parametrized by vectors of the Euclidean space $\mathbb{R}^d$. Two widely used examples of parametrized models are linear models and deep nets. The parameter space is then often a subset $\mathcal{W} \subseteq \mathbb{R}^d$, e.g., all vectors $\mathbf{w} \in \mathbb{R}^d$ with a norm smaller than one.

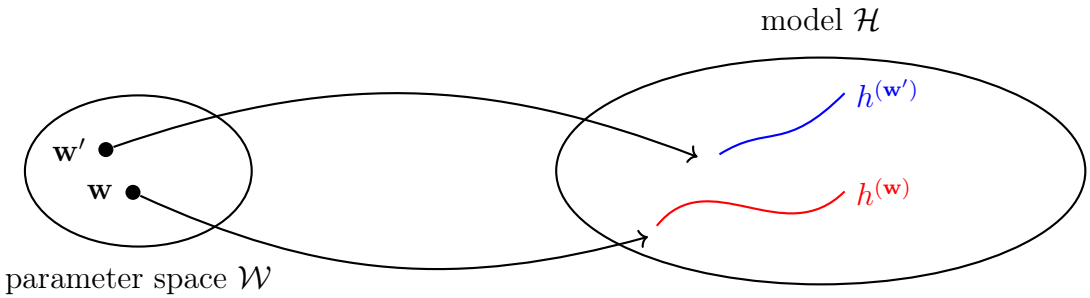


Figure 16: The parameter space $\mathcal{W}$ of an ML model $\mathcal{H}$ consists of all feasible choices for the model parameters. Each choice $\mathbf{w}$ for the model parameters selects a hypothesis map $h^{(\mathbf{w})} \in \mathcal{H}$.

parameters The parameters of an ML model are tunable (i.e., learnable or adjustable) quantities that allow us 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 the numeric features of a data point. For data points characterized by a single numeric feature, polynomial regression uses the hypothesis space $\mathcal{H}_d^{(\text{poly})} := \{h(x) = \sum_{j=0}^{d-1} x^j w_j\}$. The quality of a polynomial hypothesis map is measured using the average squared error loss incurred on a set of labeled datapoints (which we refer to as the training set).

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

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

privacy funnel The privacy funnel is a method for learning privacy-friendly features of data points [88].

privacy leakage Consider an ML application that processes a dataset $\mathcal{D}$ and delivers some output, such as the predictions obtained for new data points. Privacy leakage arises if the output carries information about a private (or sensitive) feature of a data point (which might be a human) of $\mathcal{D}$. Based on a probabilistic model for the data generation, we can

measure the privacy leakage via the MI between the output and the senstive feature. Another quantitative measure of privacy leakage is DP. The relations between different measures of privacy leakage have been studied in the literature (see [89]).

privacy protection Consider some ML method $\mathcal{A}$ that reads in a dataset $\mathcal{D}$ and delivers some output $\mathcal{A}(\mathcal{D})$. The output could be the learned 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 learned via statistical inference methods such as maximum likelihood estimation [13].

probabilistic principal component analysis (PPCA) Probabilistic PCA extends basic PCA by using a probabilistic model for data points. The probabilistic model of probabilistic PCA reduces the task of dimension-

ality reduction to an estimation problem that can be solved using EM methods.

probability We assign a probability value, typically chosen in the interval $[0,1]$, to each event that might occur in a random experiment [5], [36], [37], [90].

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

probability distribution To analyze ML methods, it can be useful to interpret data points as i.i.d. realizations of an 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 $p(y=1)=1-p(y=0)$. The probability distribution of a real-valued RV $x \in \mathbb{R}$ might be specified by a pdf $p(x)$ such that $p(x \in [a,b]) \approx p(a)|b-a|$. In the most general case, a probability distribution is defined by a probability measure [15], [37].

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

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

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

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

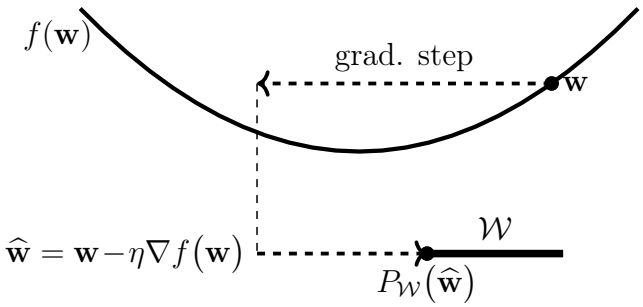


Figure 17: Projected GD augments a basic gradient step with a projection back onto the constraint set $\mathcal{W}$.

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

$$P_{\mathcal{W}}(\mathbf{w}) = \operatorname*{argmin}_{\mathbf{w}' \in \mathcal{W}} \|\mathbf{w} - \mathbf{w}'\|_2 . \tag{11}$$

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

proximable A convex function for which the proximal operator can be computed efficiently is sometimes referred to as proximable or simple [91].

proximal operator Given a convex function $f(\mathbf{w}')$, we define its proximal operator as [64], [92]

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

As illustrated in Fig. 18, evaluating the proximal operator amounts to minimizing a penalized variant of $f(\mathbf{w}')$. The penalty term is the scaled

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

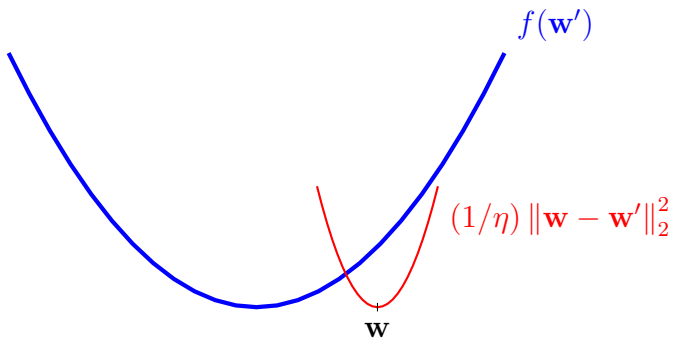


Figure 18: A generalized gradient step updates a vector $\mathbf{w}$ by minimizing a penalized version of the function $f(\cdot)$. The penalty term is the scaled squared Euclidean distance between the optimization variable $\mathbf{w}'$ and the given vector $\mathbf{w}$.

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

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

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

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

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

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

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

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

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

regression Regression problems revolve around the prediction of a numeric 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 baseline, is the difference between the loss incurred by h and the loss incurred by h' [40]. The baseline hypothesis h' is also referred to as an expert.

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

1) Model pruning: We prune the original model $\mathcal{H}$ to obtain a smaller model $\mathcal{H}'$. For a parametric model, the pruning can be implemented via constraints on the model parameters (such as $w_1 \in [0.4, 0.6]$ for the weight of feature x_1 in linear regression).

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

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

feature vector of a data point.

Fig. 19 illustrates the above three routes to regularization. These routes are closely related and sometimes fully equivalent: data augmentation using Gaussian RVs to perturb the feature vectors in the training set of linear regression has the same effect as adding the penalty $\lambda \|\mathbf{w}\|_2^2$ to the training error (which is nothing but ridge regression). The decision on which route to use for regularization can be based on the available computational infrastructure. For example, it might be much easier to implement data augmentation than model pruning.

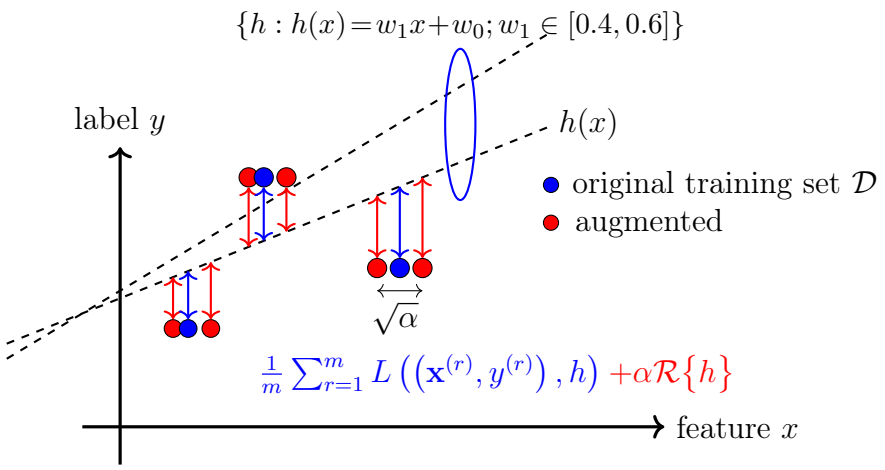


Figure 19: Three approaches to regularization: 1) data augmentation; 2) loss penalization; and 3) model pruning (via constraints on model parameters).

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

$$\hat{h} \in \operatorname*{argmin}_{h \in \mathcal{H}} \widehat{L}\big(h|\mathcal{D}\big) + \alpha \mathcal{R}\big\{h\big\}. \tag{12}$$

The parameter $\alpha \geq 0$ controls the regularization strength. For $\alpha = 0$, we recover standard ERM without regularization. As α increases, the learned hypothesis is increasingly biased toward small values of $\mathcal{R}\big\{h\big\}$. The component $\alpha \mathcal{R}\big\{h\big\}$ in the objective function of (12) can be intuitively understood as a surrogate for the increased average loss that may occur when predicting labels for data points outside the training set. This intuition can be made precise in various ways. For example, consider a linear model trained using squared error loss and the regularizer $\mathcal{R}\big\{h\big\} = \|\mathbf{w}\|_2^2$. In this setting, $\alpha \mathcal{R}\big\{h\big\}$ corresponds to the expected increase in loss caused by adding Gaussian RVs to the feature vectors in the training set [6, Ch. 3]. A principled construction for the regularizer $\mathcal{R}\big\{h\big\}$ arises from approximate upper bounds on the generalization error. The resulting regularized ERM instance is known as SRM [93, Sec. 7.2].

regularized loss minimization (RLM) See RERM.

regularizer A regularizer assigns each hypothesis h from a hypothesis space $\mathcal{H}$ a quantitative measure $\mathcal{R}\big\{h\big\}$ 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 $\mathcal{R}\big\{h\big\} := \|\mathbf{w}\|_2^2$ for linear hypothesis maps $h^{(\mathbf{w})}(\mathbf{x}) := \mathbf{w}^T\mathbf{x}$ [6, Ch. 3]. Lasso uses the regularizer $\mathcal{R}\big\{h\big\} := \|\mathbf{w}\|_1$ for linear hypothesis maps $h^{(\mathbf{w})}(\mathbf{x}) := \mathbf{w}^T\mathbf{x}$ [6, Ch. 3].

Rényi divergence The Rényi divergence measures the (dis)similarity between two probability distributions [94].

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

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

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 an RV. The i.i.d. assumption allows us 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.

robustness TBD

sample A finite sequence (or list) of data points $\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(m)}$ that is obtained or interpreted as the realization of m i.i.d. RVs with a 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 $\widehat{\mathbf{\Sigma}} \in \mathbb{R}^{d \times d}$ for a given set of feature vectors $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)} \in \mathbb{R}^d$ is defined as

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

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

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

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

sample size The number of individual data points contained in a dataset.

scatterplot A visualization technique that depicts data points by markers in a two-dimensional plane. Fig. 20 depicts an example of a scatterplot.

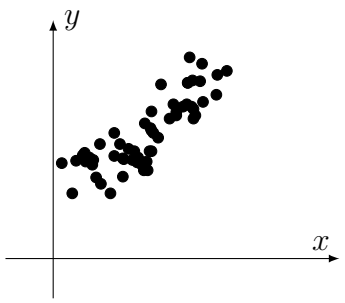


Figure 20: A scatterplot of some data points representing daily weather conditions in Finland. Each data point is characterized by its minimum daytime temperature x as the feature and its maximum daytime temperature y as the label. The temperatures have been measured at the FMI weather station Helsinki Kaisaniemi during 1.9.2024 - 28.10.2024.

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

sensitive attribute ML revolves around learning a hypothesis map that allows us 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 us to infer sensitive attributes of a data point. Which part of a data point is considered 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 (SVD) The SVD for a matrix $\mathbf{A} \in \mathbb{R}^{m \times d}$ is a factorization of the form

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

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

smooth A real-valued function $f : \mathbb{R}^d \rightarrow \mathbb{R}$ is smooth if it is differentiable and its gradient $\nabla f(\mathbf{w})$ is continuous at all $\mathbf{w} \in \mathbb{R}^d$ [82], [95]. A smooth function f is referred to as β-smooth if the gradient $\nabla f(\mathbf{w})$ is Lipschitz continuous with Lipschitz constant β, i.e.,

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

The constant β quantifies the amount of smoothness of the function f: the smaller the β, the smoother f is. Optimization problems with a smooth objective function can be solved effectively by gradient-based methods. Indeed, gradient-based methods approximate the objective function locally around a current choice $\mathbf{w}$ using its gradient. This approximation works well if the gradient does not change too rapidly.

We can make this informal claim precise by studying the effect of a single gradient step with step size $\eta = 1/\beta$ (see Fig. 21).

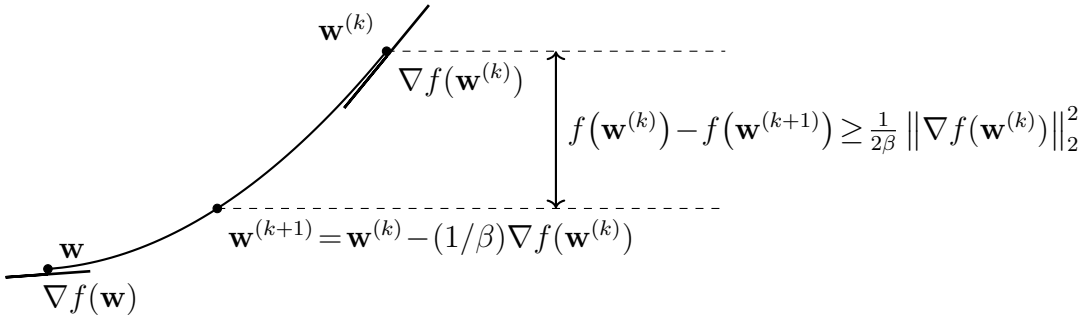
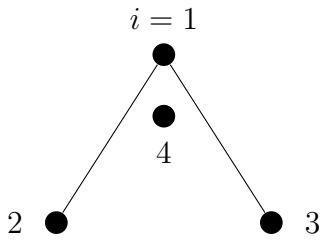


Figure 21: Consider an objective function $f(\mathbf{w})$ that is β-smooth. Taking a gradient step, with step size $\eta = 1/\beta$, decreases the objective by at least $\frac{1}{2\beta} \big\|\nabla f(\mathbf{w}^{(k)})\big\|_2^2$ [82], [95], [96]. Note that the step size $\eta = 1/\beta$ becomes larger for smaller β. Thus, for smoother objective functions (i.e., those with smaller β), we can take larger steps.

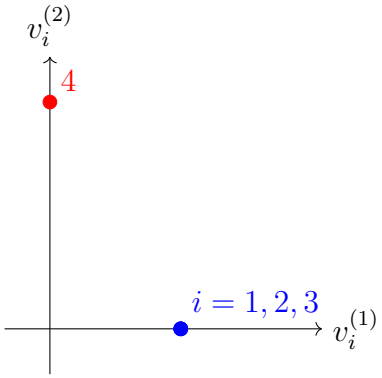
soft clustering Soft clustering refers to the task of partitioning a given set of data points into (a few) overlapping clusters. Each data point is assigned to several different clusters with varying degrees of belonging. Soft clustering methods determine the degree of belonging (or soft cluster assignment) for each 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 is a particular instance of graph

clustering, i.e., it clusters data points represented as the nodes $i = 1, \ldots, n$ of a graph $\mathcal{G}$. Spectral clustering uses the eigenvectors of the Laplacian matrix $\mathbf{L}^{(\mathcal{G})}$ to construct feature vectors $\mathbf{x}^{(i)} \in \mathbb{R}^d$ for each node (i.e., for each data point) $i = 1, \ldots, n$. We can feed these feature vectors into Euclidean space-based clustering methods, such as k-means or soft clustering via GMM. Roughly speaking, the feature vectors of nodes belonging to a well-connected subset (or cluster) of nodes in $\mathcal{G}$ are located nearby in the Euclidean space $\mathbb{R}^d$ (see Fig. 22).



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



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

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

Figure 22: **Top.** Left: An undirected graph $\mathcal{G}$ with four nodes $i = 1, 2, 3, 4$, each representing a data point. Right: The Laplacian matrix $\mathbf{L}^{(\mathcal{G})} \in \mathbb{R}^{4\times 4}$ and its EVD. **Bottom.** Left: A scatterplot of data points using the feature vectors $\mathbf{x}^{(i)} = \left(v_i^{(1)}, v_i^{(2)}\right)^T$. Right: Two eigenvectors $\mathbf{v}^{(1)}, \mathbf{v}^{(2)} \in \mathbb{R}^d$ corresponding to the eigenvalue $\lambda = 0$ of the Laplacian matrix $\mathbf{L}^{(\mathcal{G})}$.

spectrogram A spectrogram represents the time-frequency distribution of the energy of a time signal $x(t)$. Intuitively, it quantifies the amount of signal energy present within a specific time segment $[t_1, t_2] \subseteq \mathbb{R}$ and frequency interval $[f_1, f_2] \subseteq \mathbb{R}$. Formally, the spectrogram of a signal is defined as the squared magnitude of its short-time Fourier transform (STFT) [97]. Fig. 23 depicts a time signal along with its spectrogram.

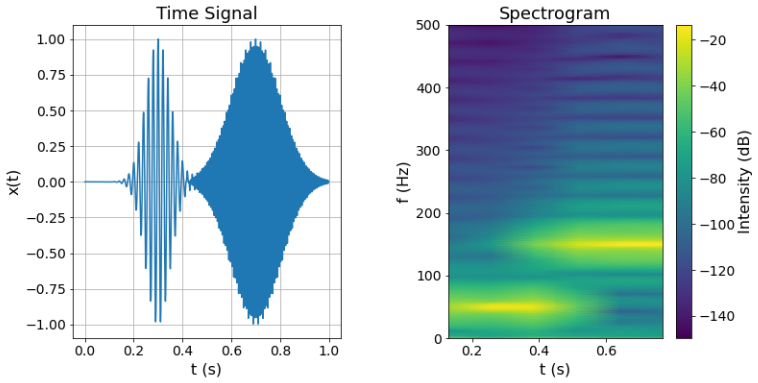


Figure 23: Left: A time signal consisting of two modulated Gaussian pulses. Right: An intensity plot of the spectrogram.

The intensity plot of its spectrogram can serve as an image of a signal. A simple recipe for audio signal classification is to feed this signal image into deep nets originally developed for image classification and object detection [98]. It is worth noting that, beyond the spectrogram, several alternative representations exist for the time-frequency distribution of signal energy [99], [100].

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

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

stability Stability is a desirable property of an ML method $\mathcal{A}$ that maps a dataset $\mathcal{D}$ (e.g., a training set) to an output $\mathcal{A}(\mathcal{D})$. The output $\mathcal{A}(\mathcal{D})$ can be the learned model parameters or the prediction delivered by the trained model for a specific data point. Intuitively, $\mathcal{A}$ is stable if small changes in the input dataset $\mathcal{D}$ lead to small changes in the output $\mathcal{A}(\mathcal{D})$. Several formal notions of stability exist that enable bounds on the generalization error or risk of the method (see [60, Ch. 13]). To build intuition, consider the three datasets depicted in Fig. 24, each of which is equally likely under the same data-generating probability distribution. Since the optimal model parameters are determined by this underlying probability distribution, an accurate ML method $\mathcal{A}$ should return the same (or very similar) output $\mathcal{A}(\mathcal{D})$ for all three datasets. In other words, any useful $\mathcal{A}$ must be robust to variability in sample realizations from the same probability distribution, i.e., it must be stable.

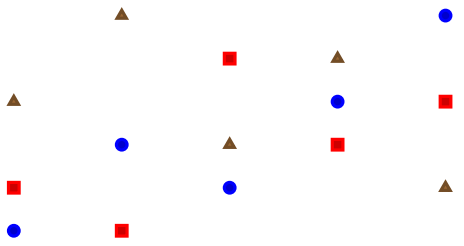


Figure 24: Three datasets $\mathcal{D}^{(*)}$, $\mathcal{D}^{(\square)}$, and $\mathcal{D}^{(\triangle)}$, each sampled independently from the same data-generating probability distribution. A stable ML method should return similar outputs when trained on any of these datasets.

statistical aspects By statistical aspects of an 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 (SBM) The stochastic block model is a probabilistic generative model for an undirected graph $\mathcal{G} = \big(\mathcal{V}, \mathcal{E}\big)$ with a given set of nodes $\mathcal{V}$ [101]. In its most basic variant, the stochastic block model generates a graph by first randomly assigning each node $i \in \mathcal{V}$ to a cluster index $c_i \in \{1, \ldots, k\}$. A pair of different nodes in the graph is connected by an edge with probability $p_{i,i'}$ that depends solely on the labels $c_i, c_{i'}$. The presence of edges between different pairs of nodes is statistically independent.

stochastic gradient descent (SGD) Stochastic GD is obtained from GD by replacing the gradient of the objective function with a stochastic approximation. A main application of stochastic GD is to train a parametrized model via ERM on a training set $\mathcal{D}$ that is either very large or not readily available (e.g., when data points are stored in a database distributed all over the planet). To evaluate the gradient of the empirical risk (as a function of the model parameters $\mathbf{w}$), we need to compute a sum $\sum_{r=1}^{m} \nabla_{\mathbf{w}} L\left(\mathbf{z}^{(r)}, \mathbf{w}\right)$ over all data points in the training set. We obtain a stochastic approximation to the gradient by replacing the sum $\sum_{r=1}^{m} \nabla_{\mathbf{w}} L\left(\mathbf{z}^{(r)}, \mathbf{w}\right)$ with a sum $\sum_{r \in \mathcal{B}} \nabla_{\mathbf{w}} L\left(\mathbf{z}^{(r)}, \mathbf{w}\right)$ over a randomly chosen subset $\mathcal{B} \subseteq \{1, \ldots, m\}$ (see Fig. 25). We often refer to these randomly chosen data points as a batch. The batch size $|\mathcal{B}|$ is an important parameter of stochastic GD. Stochastic GD with $|\mathcal{B}| > 1$ is referred to as mini-batch stochastic GD [102].

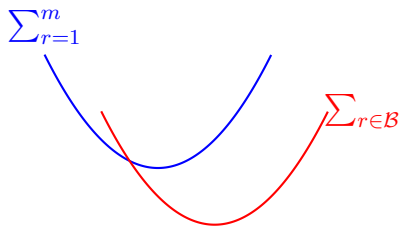


Figure 25: Stochastic GD for ERM approximates the gradient $\sum_{r=1}^{m} \nabla_{\mathbf{w}} L\left(\mathbf{z}^{(r)}, \mathbf{w}\right)$ by replacing the sum over all data points in the training set (indexed by $r = 1, \ldots, m$) with a sum over a randomly chosen subset $\mathcal{B} \subseteq \{1, \ldots, m\}$.

stopping criterion Many ML methods use iterative algorithms that con-

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

strongly convex A continuously differentiable real-valued function $f(\mathbf{x})$ is strongly convex with coefficient σ if $f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla f(\mathbf{x})^T(\mathbf{y} - \mathbf{x}) + (\sigma/2)\left\|\mathbf{y} - \mathbf{x}\right\|_2^2$ [82], [96, Sec. B.1.1].

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

subgradient For a real-valued function $f : \mathbb{R}^d \to \mathbb{R} : \mathbf{w} \mapsto f(\mathbf{w})$, a vector $\mathbf{a}$ such that $f(\mathbf{w}) \geq f(\mathbf{w}') + \left(\mathbf{w} - \mathbf{w}'\right)^T \mathbf{a}$ is referred to as a subgradient of f at $\mathbf{w}'$ [103], [104].

subgradient descent Subgradient descent is a generalization of GD that does not require differentiability of the function to be minimized. This generalization is obtained by replacing the concept of a gradient with

that of a subgradient. Similar to gradients, also subgradients allow us to construct local approximations of an objective function. The objective function might be the empirical risk $\widehat{L}\big(h^{(\mathbf{w})}\big|\mathcal{D}\big)$ viewed as a function of the model parameters $\mathbf{w}$ that select a hypothesis $h^{(\mathbf{w})} \in \mathcal{H}$.

support vector machine (SVM) The SVM is a binary classification method that learns a linear hypothesis map. Thus, like linear regression and logistic regression, it is also an instance of ERM for the linear model. However, the SVM uses a different loss function from the one used in those methods. As illustrated in Fig. 26, it aims to maximally separate data points from the two different classes in the feature space (i.e., maximum margin principle). Maximizing this separation is equivalent to minimizing a regularized variant of the hinge loss (6) [38], [70], [105].

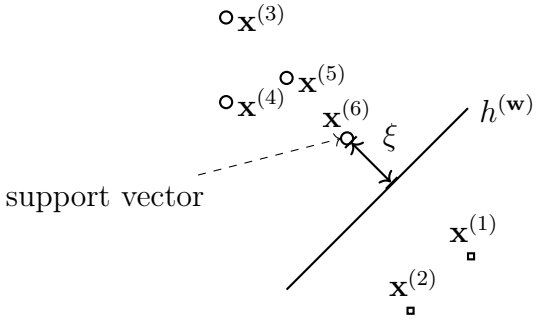


Figure 26: The SVM learns a hypothesis (or classifier) $h^{(\mathbf{w})}$ with minimal average soft-margin hinge loss. Minimizing this loss is equivalent to maximizing the margin ξ between the decision boundary of $h^{(\mathbf{w})}$ and each class of the training set.

The above basic variant of SVM is only useful if the data points from

different categories can be (approximately) linearly separated. For an ML application where the categories are not derived from a kernel.

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

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

total variation See GTV.

training error The average loss of a hypothesis when predicting the labels of the data points in a training set. We sometimes refer by training error also to minimal average loss which is achieved by a solution of ERM.

training set A training set is a dataset $\mathcal{D}$ which consists of 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 of the training error with the validation error of $\hat{h}$ allows us to diagnose the ML method and informs how to improve the validation error (e.g., using a different hypothesis space or collecting more data points) [6, Sec. 6.6].

transparency Transparency is a fundamental requirement for trustworthy AI [106]. In the context of ML methods, transparency is often used interchangeably with explainability [44], [107]. However, in the broader scope of AI systems, transparency extends beyond explainability and includes providing information about the system's limitations, reliability, and intended use. In medical diagnosis systems, transparency requires disclosing the confidence level for the predictions delivered by a trained model. In credit scoring, AI-based lending decisions should be accompanied by explanations of contributing factors, such as income level or credit history. These explanations allow humans (e.g., a loan applicant) to understand and contest automated decisions. Some ML methods inherently offer transparency. For example, logistic regression provides a quantitative measure of classification reliability through the value $|h(\mathbf{x})|$. Decision trees are another example, as they allow human-readable decision rules [46]. Transparency also requires a clear indication when a user is engaging with an AI system. For example, AI-powered chatbots should notify users that they are interacting with an automated system rather than a human. Furthermore, transparency encompasses comprehensive documentation detailing the purpose and design choices underlying the AI system. For instance, model datasheets [30] and AI system cards [108] help practitioners understand the intended use cases and limitations of an AI system [109].

trustworthy artificial intelligence (trustworthy AI) Besides the computational aspects and statistical aspects, a third main design aspect of ML methods is their trustworthiness [110]. The EU has put forward

seven key requirements (KRs) for trustworthy AI (that typically build on ML methods) [111]:

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

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

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

upper confidence bound (UCB) Consider an ML application that requires selecting, at each time step k, an action a_k from a finite set of alternatives $\mathcal{A}$. The utility of selecting action a_k is quantified by a numeric reward signal $r^{(a_k)}$. A widely used probabilistic model for this type of sequential decision-making problem is the stochastic MAB setting [79]. In this model, the reward $r^{(a)}$ is viewed as the realization of an RV with unknown mean $\mu^{(a)}$. Ideally, we would always choose the action with the largest expected reward $\mu^{(a)}$, but these means are unknown and must be estimated from observed data. Simply choosing the action with the largest estimate $\widehat{\mu}^{(a)}$ can lead to suboptimal outcomes due to estimation uncertainty. The UCB strategy addresses this by selecting actions not only based on their estimated means but also by incorporating a term that reflects the uncertainty in these estimates—favoring actions with high potential reward and high uncertainty. Theoretical guarantees for the performance of UCB strategies, including logarithmic regret bounds, are established in [79].

validation Consider a hypothesis $\hat{h}$ that has been learned 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 the hypothesis $\hat{h}$ on a set 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 learned 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 an ML method (see [6, Sec. 6.6]). The comparison between training error and validation error can inform directions for improvement of the ML method (such as using a different hypothesis space).

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

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

vertical federated learning (VFL) VFL uses local datasets that are constituted by the same data points but characterizing them with different features [112]. For example, different healthcare providers might all contain information about the same population of patients. However, different healthcare providers collect different measurements (e.g., blood values, electrocardiography, lung X-ray) for the same patients.

weights Consider a parametrized hypothesis space $\mathcal{H}$. We use the term weights for numeric model parameters that are used to scale 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 $L^{(0/1)}\left((\mathbf{x}, y), h\right)$ 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., $L^{(0/1)}\left((\mathbf{x}, y), h\right) = 0$ when $\hat{y} = y$. It is equal to 1 if the prediction is wrong, i.e., $L^{(0/1)}\left((\mathbf{x}, y), h\right) = 1$ when $\hat{y} \neq y$.

Index

References

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

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

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

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

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

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

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

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

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

[10] R. G. Gallager, *Stochastic Processes: Theory for Applications*. New York, NY, USA: Cambridge Univ. Press, 2013.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[63] H. P. Lopuhaä and P. J. Rousseeuw, "Breakdown points of affine equivariant estimators of multivariate location and covariance matrices," *The Annals of Statistics*, vol. 19, no. 1, pp. 229–248, 1991. [Online]. Available: http://www.jstor.org/stable/2241852

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

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

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

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

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

[69] N. Young, *An introduction to Hilbert space*, ser. Cambridge mathematical textbooks. Cambridge: Cambridge U. P., 1988.

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

[71] Q. Yang, Y. Liu, Y. Cheng, Y. Kang, T. Chen, and H. Yu, "Horizontal federated learning," in *Federated Learning*. Cham, Switzerland: Springer Nature, 2020, ch. 4, pp. 49–67.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[93] S. Shalev-Shwartz and A. Tewari, "Stochastic methods for ℓ_1 regularized loss minimization," in *Proc. 26th Annu. Int. Conf. Mach. Learn.*

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

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

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

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

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

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

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

[101] E. Abbe, "Community detection and stochastic block models: Recent developments," *J. Mach. Learn. Res.*, vol. 18, no. 177, pp. 1–86, Apr. 2018. [Online]. Available: http://jmlr.org/papers/v18/16-480.html

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

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

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

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

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

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

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

[109] 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 Int. Humanitarian Technol. Conf.*, pp. 197–201, doi: 10.1109/IHTC.2017.8058187.

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

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

[112] Q. Yang, Y. Liu, Y. Cheng, Y. Kang, T. Chen, and H. Yu, "Vertical federated learning," in *Federated Learning*. Springer Nature, 2020, ch. 5, pp. 69–81.