

The A'allon koneoppimisen sanakirja

Alexander Jung¹, Konstantina Olioumtsevits¹, Ekkehard Schnoor¹,
Tommi Flores Ryynänen¹, Juliette Gronier², and Salvatore Rastelli¹
Käännös Mikko Seesto¹, Janita Thusberg¹ ja Helli Manninen¹

¹Aalto University ²ENS Lyon

19. joulukuuta 2025



please cite as: A. Jung, K. Olioumtsevits, E. Schnoor, T. Ryynänen, J. Gronier, and S. Rastelli, *Aallon koneoppimisen sanakirja*. Käännös J. Thusberg, H. Manninen ja M. Seesto. Espoo, Finland: Aalto University, 2025.

Acknowledgment

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 to shape this dictionary. Special thanks to Mikko Seesto for his meticulous proofreading.

This work was supported by

- the Research Council of Finland (grants 331197, 363624, 349966);
- the European Union (grant 952410);
- the Jane and Aatos Erkko Foundation (grant A835);
- Business Finland, as part of the project Forward-Looking AI Governance in Banking and Insurance (FLAIG).

Sisällys

List of Symbols	4
Mathematical Tools	22
Machine Learning Concepts	119
Reinforcement Learning	283
Machine Learning Systems	287
Machine Learning Regulation	288
Index	293

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 .

$|\mathcal{A}|$ 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} .

$\mathcal{A} \times \mathcal{B}$ The Cartesian product of the sets \mathcal{A} and \mathcal{B} .

\mathbb{N} The natural numbers $1, 2, \dots$.

\mathbb{R} The real numbers x [1].

\mathbb{R}_+ The nonnegative 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 $0 \leq x \leq 1$.

$\arg \min_{\mathbf{w} \in \mathcal{C}} f(\mathbf{w})$	The set of minimizers for a real-valued funktion $f : \mathcal{C} \rightarrow \mathbb{R}$. See also: funktion.
$\mathbb{S}^{(n)}$	The set of unit-normi vektorit in \mathbb{R}^{n+1} . See also: normi, vektori.
$\exp(a)$	The exponential funktion evaluated at the real number $a \in \mathbb{R}$. See also: funktion.
$\log a$	The logarithm of the positive number $a \in \mathbb{R}_{++}$.
$f(\cdot) : \mathcal{A} \rightarrow \mathcal{B} : a \mapsto f(a)$	A funktion (or kuvaus) from a set \mathcal{A} to a set \mathcal{B} , which assigns to each input $a \in \mathcal{A}$ a well-defined output $f(a) \in \mathcal{B}$. The set \mathcal{A} is the domain of the funktion f and the set \mathcal{B} is the co-domain of f . Koneoppiminen aims to learn a funktion h that maps featuret \mathbf{x} of a data point to a ennuste $h(\mathbf{x})$ for its label y . See also: funktion, kuvaus, output, domain, co-domain, koneoppiminen, feature, data point, ennuste, label.
$\text{epi}(f)$	The epigrafi of a real-valued funktion $f : \mathbb{R}^d \rightarrow \mathbb{R}$. See also: epigrafi, funktion.
$(a_r)_{r \in \mathbb{N}}, (a^{(r)})_{r \in \mathbb{N}}, \{a^{(r)}\}_{r \in \mathbb{N}}$	A sequence of elements. See also: sequence.

$\mathbb{I}_{\mathcal{A}}(x)$

The indicator funktio of a set \mathcal{A} delivers $f(x) = 1$ for any $x \in \mathcal{A}$ and $f(x) = 0$ otherwise.

See also: funktio.

$\frac{\partial f(w_1, \dots, w_d)}{\partial w_j}$

The partial derivative (if it exists) of a real-valued funktio $f : \mathbb{R}^d \rightarrow \mathbb{R}$ with respect to w_j [2, Ch. 9].

See also: partial derivative, funktio.

$\nabla f(\mathbf{w})$

The gradientti of a differentiable real-valued funktio $f : \mathbb{R}^d \rightarrow \mathbb{R}$ is the vektori $\nabla f(\mathbf{w}) = (\partial f / \partial w_1, \dots, \partial f / \partial w_d)^T \in \mathbb{R}^d$ [2, Ch. 9].

See also: gradientti, differentiable, funktio, vektori.

Matrices and Vectors

$\mathbf{x} = (x_1, \dots, x_d)^T$	A vektori of length d , with its j th entry being x_j . See also: vektori.
\mathbb{R}^d	The set of vektorit $\mathbf{x} = (x_1, \dots, x_d)^T$ consisting of d real-valued entries $x_1, \dots, x_d \in \mathbb{R}$. See also: vektori.
$\mathbf{I}_{l \times d}$	A generalized identity matriisi 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 otherwise equal to 0. See also: matriisi.
\mathbf{I}_d, \mathbf{I}	A square identity matriisi of size $d \times d$. If the size is clear from context, we drop the subscript. See also: matriisi.
$\ \mathbf{x}\ _2$	The Euclidean (or ℓ_2) normi of the vektori $\mathbf{x} = (x_1, \dots, x_d)^T \in \mathbb{R}^d$ defined as $\ \mathbf{x}\ _2 := \sqrt{\sum_{j=1}^d x_j^2}$. See also: normi, vektori.
$\ \mathbf{x}\ $	Some normi of the vektori $\mathbf{x} \in \mathbb{R}^d$ [3]. Unless otherwise specified, we mean the Euclidean norm $\ \mathbf{x}\ _2$. See also: normi, vektori, Euclidean norm.
\mathbf{x}^T	The transpose of a matriisi that has the vektori $\mathbf{x} \in \mathbb{R}^d$ as its single column. See also: matriisi, vektori.

\mathbf{X}^T The transpose of a matriisi $\mathbf{X} \in \mathbb{R}^{m \times d}$. A square real-valued matriisi $\mathbf{X} \in \mathbb{R}^{m \times m}$ is called symmetric if $\mathbf{X} = \mathbf{X}^T$.
See also: matriisi.

\mathbf{X}^{-1} The käänteismatriisi of a matriisi $\mathbf{X} \in \mathbb{R}^{d \times d}$.
See also: käänteismatriisi, matriisi.

$\mathbf{0} = (0, \dots, 0)^T$ The vektori in \mathbb{R}^d with each entry equal to zero.
See also: vektori.

$\mathbf{1} = (1, \dots, 1)^T$ The vektori in \mathbb{R}^d with each entry equal to one.
See also: vektori.

$(\mathbf{v}^T, \mathbf{w}^T)^T$ The vektori of length $d + d'$ obtained by concatenating the entries of vektori $\mathbf{v} \in \mathbb{R}^d$ with the entries of $\mathbf{w} \in \mathbb{R}^{d'}$.
See also: vektori.

$\text{span}(\mathbf{B})$ The span of a matriisi $\mathbf{B} \in \mathbb{R}^{a \times b}$, which is the subspace of all linear combinations of the columns of \mathbf{B} , such that $\text{span}(\mathbf{B}) = \{\mathbf{Ba} : \mathbf{a} \in \mathbb{R}^b\} \subseteq \mathbb{R}^a$.
See also: matriisi, subspace.

$\text{null}(\mathbf{A})$ The nolla-avaruuus of a matriisi $\mathbf{A} \in \mathbb{R}^{a \times b}$, which is the subspace of vektorit $\mathbf{a} \in \mathbb{R}^b$ such that $\mathbf{Aa} = \mathbf{0}$.
See also: nolla-avaruuus, matriisi, subspace, vektori.

$\det(\mathbf{C})$ The determinant of the matrix \mathbf{C} .
See also: determinant, matrix.

$\text{tr}(\mathbf{C})$ The trace of the matrix \mathbf{C} .
See also: trace.

$\mathbf{A} \otimes \mathbf{B}$ The Kronecker product of \mathbf{A} and \mathbf{B} [4].
See also: Kronecker product.

Probability Theory

$\mathbf{x} \sim \mathbb{P}^{(\mathbf{z})}$ The random variable (RV) \mathbf{x} is distributed according to the probability distribution $\mathbb{P}^{(\mathbf{z})}$ [5], [6].

See also: RV, probability distribution.

$\mathbb{E}_p\{f(\mathbf{z})\}$ The expectation of an RV $f(\mathbf{z})$ that is obtained by applying a deterministic function f to an RV \mathbf{z} whose probability distribution is $\mathbb{P}^{(\mathbf{z})}$. If the probability distribution is clear from context, we just write $\mathbb{E}\{f(\mathbf{z})\}$.

See also: expectation, RV, function, probability distribution.

$\text{cov}(x, y)$ The covariance between two real-valued RVs defined over a common probability space.

See also: covariance, RV, probability distribution.

$\mathbb{P}^{(\mathbf{x}, y)}$ A (joint) probability distribution of an RV whose realization are data points with features \mathbf{x} and label y .

See also: probability distribution, RV, realization, data point, feature, label.

$\mathbb{P}^{(y|\mathbf{x})}$ A conditional probability distribution of an RV y given (or conditioned on) the value of another RV \mathbf{x} [7, Sec. 3.5].

See also: conditional probability distribution, RV.

$\mathbb{P}(\mathcal{A})$ The todennäköisyys of the measurable event \mathcal{A} .

See also: todennäköisyys, measurable, event.

	The moment generating function (MGF) of an RV x .
$M_x(t)$	See also: probability distribution, probability density function (pdf).
$\mathbb{P}^{(\mathcal{D})}$	<p>The empirical distribution of a dataset \mathcal{D}.</p> <p>See also: empirical distribution, dataset, uusio-otanta.</p>
$\mathbb{P}^{(\mathbf{x}; \mathbf{w})}$	<p>A parameterized probability distribution of an RV \mathbf{x}. The probability distribution depends on a parameter vektori \mathbf{w}. For example, $\mathbb{P}^{(\mathbf{x}; \mathbf{w})}$ could be a moninormaalijakauma with the parameter vektori \mathbf{w} given by the entries of the keskiarvo vektori $\mathbb{E}\{\mathbf{x}\}$ and the covariance matrix $\mathbb{E}\left\{ (\mathbf{x} - \mathbb{E}\{\mathbf{x}\})(\mathbf{x} - \mathbb{E}\{\mathbf{x}\})^T \right\}$.</p> <p>See also: probability distribution, parameter, todennäköisyysmalli.</p>
$\mathcal{N}(\mu, \sigma^2)$	<p>The probability distribution of a normaalijakautunut satunnaismuuttuja $x \in \mathbb{R}$ with keskiarvo (or expectation) $\mu = \mathbb{E}\{x\}$ and varianssi $\sigma^2 = \mathbb{E}\{(x - \mu)^2\}$.</p> <p>See also: probability distribution, normaalijakautunut satunnaismuuttuja.</p>
$\mathcal{N}(\boldsymbol{\mu}, \mathbf{C})$	<p>The moninormaalijakauma of a vektori-valued normaalijakautunut satunnaismuuttuja $\mathbf{x} \in \mathbb{R}^d$ with keskiarvo (or expectation) $\boldsymbol{\mu} = \mathbb{E}\{\mathbf{x}\}$ and covariance matrix $\mathbf{C} = \mathbb{E}\{(\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^T\}$.</p> <p>See also: moninormaalijakauma, normaalijakautunut satunnaismuuttuja.</p>
Δ^k	<p>The probability simplex, which consists of all vektorit $\mathbf{p} = (p_1, \dots, p_k)^T \in \mathbb{R}^k$ with nonnegative entries that sum to one, i.e., $p_c \geq 0$ for $c = 1, \dots, k$ and $\sum_{c=1}^k p_c = 1$.</p> <p>See also: probability mass function (pmf).</p>

$H(x)$ The entropia of a discrete random variable (discrete RV) x .
See also: entropia, discrete RV.

Ω A sample space of all possible outcomes of a satunnaiskoe.
See also: event.

Σ A collection of measurable subsets of a sample space Ω .
See also: sample space, event.

\mathcal{P} A probability space that consists of a sample space Ω , a σ -algebra Σ of measurable subsets of Ω , and a probability distribution $\mathbb{P}(\cdot)$.
See also: sample space, measurable, probability distribution.

Machine Learning

	An index $r = 1, 2, \dots$ that enumerates data points.
r	See also: data point.
	The number of data points in (i.e., the size of) a dataset.
m	See also: data point, dataset.
	A dataset $\mathcal{D} = \{\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(m)}\}$ is a list of individual data points $\mathbf{z}^{(r)}$,
\mathcal{D}	for $r = 1, \dots, m$.
	See also: dataset, data point.
	The number of features that characterize a data point.
d	See also: feature, data point.
	The j th feature of a data point. The first feature is denoted by x_1 ,
x_j	the second feature x_2 , and so on.
	See also: data point, feature.
	The feature vector $\mathbf{x} = (x_1, \dots, x_d)^T$ of a data point. The vektori's
\mathbf{x}	entries are the individual features of a data point.
	See also: feature vector, data point, vektori, feature.
	The feature space \mathcal{X} is the set of all possible values that the features
\mathcal{X}	\mathbf{x} of a data point can take on.
	See also: feature space, feature, data point.

Instead of the symbol \mathbf{x} , we sometimes use \mathbf{z} as another symbol to denote a vektori whose entries are the individual features of a data point. We need two different symbols to distinguish between raw and learned features [8, Ch. 9].

See also: vektori, feature, data point.

$\mathbf{x}^{(r)}$ The feature vector of the r th data point within a dataset.
See also: feature vector, data point, dataset.

$x_j^{(r)}$ The j th feature of the r th data point within a dataset.
See also: feature, data point, dataset.

\mathcal{B} A mini-batch (or subset) of randomly chosen data points.
See also: batch, data point.

B The size of (i.e., the number of data points in) a mini-batch.
See also: data point, batch.

y The label (or quantity of interest) of a data point.
See also: label, data point.

$y^{(r)}$ The label of the r th data point.
See also: label, data point.

$(\mathbf{x}^{(r)}, y^{(r)})$ The features and label of the r th data point.
See also: feature, label, data point.

The label space \mathcal{Y} of an koneoppiminen 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). Koneoppiminen problems (or methods) using a numeric label space, such as $\mathcal{Y} = \mathbb{R}$ or $\mathcal{Y} = \mathbb{R}^3$, are referred to as regressio problems (or methods). Koneoppiminen problems (or methods) that use a discrete label space, such as $\mathcal{Y} = \{0, 1\}$ or $\mathcal{Y} = \{\text{cat}, \text{dog}, \text{mouse}\}$, are referred to as luokittelu problems (or methods).

See also: label space, koneoppiminen, label, data point, dataset, training set, regressio, luokittelu.

η Learning rate (or step size) used by gradient-based methods.
See also: learning rate, step size, gradient-based method.

$h(\cdot)$ A hypothesis kuvaus that maps the features of a data point to a ennuste $\hat{y} = h(\mathbf{x})$ for its label y .

See also: hypothesis, kuvaus, feature, data point, ennuste, label.

$\mathcal{Y}^{\mathcal{X}}$ Given two sets \mathcal{X} and \mathcal{Y} , we denote by $\mathcal{Y}^{\mathcal{X}}$ the set of all possible hypothesis kuvauskset $h : \mathcal{X} \rightarrow \mathcal{Y}$.
See also: hypothesis, kuvaus.

\mathcal{H} A hypothesis space or model used by an koneoppiminen method. The hypothesis space consists of different hypothesis kuvauskset $h : \mathcal{X} \rightarrow \mathcal{Y}$, between which the koneoppiminen method must choose.

See also: hypothesis space, model, koneoppiminen, hypothesis, kuvaus.

$d_{\text{eff}}(\mathcal{H})$	The effective dimension of a hypothesis space \mathcal{H} . See also: effective dimension, hypothesis space.
B^2	The squared harha of a learned hypothesis \hat{h} , or its parameters. Note that \hat{h} becomes an RV if it is learned from data points being RVs themselves. See also: harha, hypothesis, parameter, RV, data point.
V	The varianssi of a learned hypothesis \hat{h} , or its parameters. Note that \hat{h} becomes an RV if it is learned from data points being RVs themselves. See also: varianssi, hypothesis, parameter, RV, data point.
$L((\mathbf{x}, y), h)$	The loss incurred by predicting the label y of a data point using the ennuste $\hat{y} = h(\mathbf{x})$. The ennuste \hat{y} is obtained by evaluating the hypothesis $h \in \mathcal{H}$ for the feature vector \mathbf{x} of the data point. See also: loss, label, data point, ennuste, hypothesis, feature vector.
E_v	The validation error of a hypothesis h , which is its average loss incurred over a validation set. See also: validation error, hypothesis, loss, validation set.
$\hat{L}(h \mathcal{D})$	The empiirinen riski, or average loss, incurred by the hypothesis h on a dataset \mathcal{D} . See also: empiirinen riski, loss, hypothesis, dataset.

E_t The training error of a hypothesis h , which is its average loss incurred over a training set.

See also: training error, hypothesis, loss, training set.

t A discrete-time index $t = 0, 1, \dots$ used to enumerate sequential events (or time instants).

See also: event.

t An index that enumerates oppimistehtävät within a monitehtävä-oppiminen problem.

See also: oppimistehtävä, monitehtäväoppiminen.

α A regularization parameter that controls the amount of regularization.

See also: regularization, parameter.

$\lambda_j(\mathbf{Q})$ The j th eigenvalue (sorted in either ascending or descending order) of a positive semi-definite (psd) matriisi \mathbf{Q} . We also use the shorthand λ_j if the corresponding matriisi is clear from context.

See also: eigenvalue, psd, matriisi.

$\sigma(\cdot)$ The aktivoointifunktio used by an artificial neuron within an artificial neural network (ANN).

See also: aktivoointifunktio, ANN.

$\mathcal{R}_{\hat{y}}$ A decision region within a feature space.

See also: decision region, feature space.

w	A parameter vektori $\mathbf{w} = (w_1, \dots, w_d)^T$ of a model, e.g., the weights of a lineaarinen malli or an ANN.
$h^{(\mathbf{w})}(\cdot)$	See also: parameter, vektori, model, weights, lineaarinen malli, ANN.
$\phi(\cdot)$	A hypothesis kuvaus that involves tunable model parameters w_1, \dots, w_d stacked into the vektori $\mathbf{w} = (w_1, \dots, w_d)^T$. See also: hypothesis, kuvaus, model parameter, vektori.
$K(\cdot, \cdot)$	A feature map $\phi : \mathcal{X} \rightarrow \mathcal{X}' : \mathbf{x} \mapsto \phi(\mathbf{x})$ that transforms the feature vector \mathbf{x} of a data point into a new feature vector $\mathbf{x}' = \phi(\mathbf{x}) \in \mathcal{X}'$. See also: feature map.
$\text{VCdim}(\mathcal{H})$	Given some feature space \mathcal{X} , a kernel is a kuvaus $K : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{C}$ that is psd. See also: feature space, kernel, kuvaus, psd.
	The Vapnik–Chervonenkis dimension (VC dimension) of the hypothesis space \mathcal{H} . See also: VC dimension, hypothesis space.

Federated Learning

An undirected graph whose nodes $i \in \mathcal{V}$ represent devicet within a federated learning network (FL network). The undirected weighted edges \mathcal{E} represent connectivity between devices and $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ statistical similarities between their datasets and oppimisteh-
täväät.

See also: undirected graph, device, FL network, dataset, oppi-
mistehtäävää.

A node that represents some device within an FL network.
 $i \in \mathcal{V}$ The device can access a local dataset and train a local model.
 See also: device, FL network, local dataset, 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 verkko \mathcal{G} .
 See also: Laplacian matrix, verkko.

$\mathbf{L}^{(\mathcal{C})}$ The Laplacian matrix of the induced verkko $\mathcal{G}^{(\mathcal{C})}$.
 See also: Laplacian matrix, verkko.

$\mathcal{N}^{(i)}$ The naapurusto of the node i in a verkko \mathcal{G} .
 See also: naapurusto, verkko.

$d^{(i)}$ The weighted solmun aste $d^{(i)} := \sum_{i' \in \mathcal{N}^{(i)}} A_{i,i'}$ of node i .
 See also: solmun aste.

$d_{\max}^{(\mathcal{G})}$ The maksimi weighted solmun aste of a verkko \mathcal{G} .
 See also: maksimi, solmun aste, verkko.

$\mathcal{D}^{(i)}$ The local dataset $\mathcal{D}^{(i)}$ carried by node $i \in \mathcal{V}$ of an FL network.

See also: local dataset, FL network.

m_i The number of data points (i.e., sample size) contained in the local dataset $\mathcal{D}^{(i)}$ at node $i \in \mathcal{V}$.

See also: data point, sample size, local dataset.

$\mathbf{x}^{(i,r)}$ The features of the r th data point in the local dataset $\mathcal{D}^{(i)}$.

See also: feature, data point, local dataset.

$y^{(i,r)}$ The label of the r th data point in the local dataset $\mathcal{D}^{(i)}$.

See also: label, data point, local dataset.

$\mathbf{w}^{(i)}$ The local model parameters of device i within an FL network.

See also: model parameter, device, FL network.

$L_i(\mathbf{w})$ The local loss function used by device i to measure the usefulness of some choice \mathbf{w} for the local model parameters.

See also: loss function, device, model parameter.

$L^{(d)}(\mathbf{x}, h(\mathbf{x}), h'(\mathbf{x}))$ The loss incurred by a hypothesis h' on a data point with features \mathbf{x} and label $h(\mathbf{x})$ that is obtained from another hypothesis.

See also: loss, hypothesis, data point, feature, label.

$\text{stack}\{\mathbf{w}^{(i)}\}_{i=1}^n$

The vektori $\left(\left(\mathbf{w}^{(1)}\right)^T, \dots, \left(\mathbf{w}^{(n)}\right)^T\right)^T \in \mathbb{R}^{dn}$ that is obtained by vertically stacking the local model parameters $\mathbf{w}^{(i)} \in \mathbb{R}^d$, for $i = 1, \dots, n$.

See also: vektori, stacking, model parameter.

Mathematical Tools

algebraic connectivity The algebraic connectivity of an undirected graph is the second-smallest eigenvalue λ_2 of its Laplacian matrix. An undirected graph is kytketty verkko if and only if $\lambda_2 > 0$ (see Fig. 1) [9], [10].

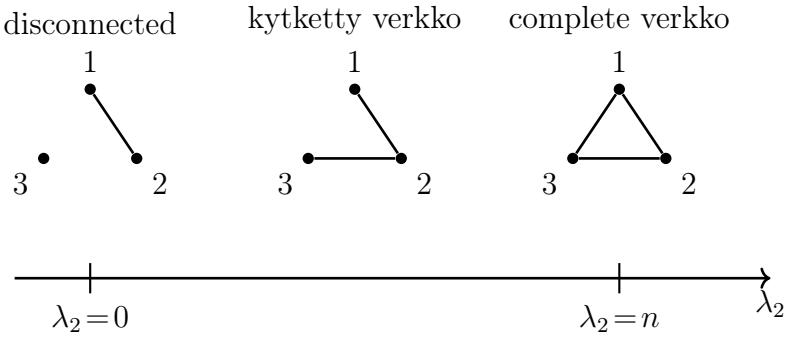


Fig. 1. Three examples of undirected graphs.

See also: undirected graph, eigenvalue, Laplacian matrix, kytketty verkko.

Banach's fixed-point theorem Banach's fixed-point theorem (also referred to as the contraction principle [2, Th. 9.23], [11]) states that every contractive operator \mathcal{F} on a complete metric space \mathcal{W} has a unique fixed point. Formally, let $(\mathcal{W}, d(\cdot, \cdot))$ be a non-empty complete metric space and let $\mathcal{F} : \mathcal{W} \rightarrow \mathcal{W}$ satisfy

$$d(\mathcal{F}\mathbf{w}, \mathcal{F}\mathbf{w}') \leq \kappa \cdot d(\mathbf{w}, \mathbf{w}') \quad \forall \mathbf{w}, \mathbf{w}' \in \mathcal{W}$$

for some constant $\kappa \in [0, 1)$. Then, \mathcal{F} has a unique fixed point, i.e., there exists a unique $\mathbf{w}^* \in \mathcal{W}$ with $\mathcal{F}\mathbf{w}^* = \mathbf{w}^*$. Moreover, for any initial $\mathbf{w}^{(0)} \in \mathcal{W}$, the fixed-point iteration $\mathbf{w}^{(t)} = \mathcal{F}\mathbf{w}^{(t-1)}$, for $t = 1, 2, \dots$, converges to \mathbf{w}^* at a rate governed by κ . In particular,

$$d(\mathbf{w}^{(t)}, \mathbf{w}^*) \leq \kappa^t \cdot d(\mathbf{w}^{(0)}, \mathbf{w}^*), \text{ for } t = 1, 2, \dots$$

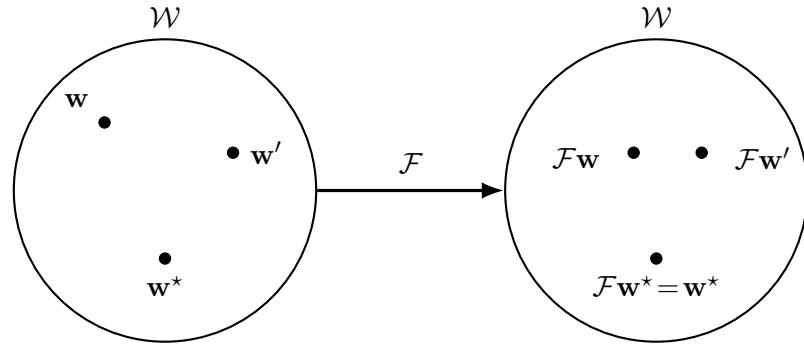


Fig. 2. A contractive operator $\mathcal{F} : \mathcal{W} \rightarrow \mathcal{W}$ has a unique fixed point \mathbf{w}^* with $\mathcal{F}\mathbf{w}^* = \mathbf{w}^*$.

See also: contractive operator, fixed-point iteration.

basis A basis of a vektoriavaruus \mathcal{V} is a set of linearly independent vektorit $\{\mathbf{a}^{(1)}, \dots, \mathbf{a}^{(d)}\}$ such that any vektori $\mathbf{a} \in \mathcal{A}$ can be expressed as a linear combination of the basis vektorit, i.e.,

$$\mathbf{a} = \sum_{j=1}^d \alpha_j \mathbf{a}^{(j)} \quad \text{for some } \alpha_1, \dots, \alpha_d \in \mathbb{R}.$$

See also: vektoriavaruus, linearly independent, vektori.

Bregman divergence The Bregman divergence induced by a convex, differentiable function $\phi(\cdot)$ is defined as

$$D_\phi(\mathbf{w}, \mathbf{w}') := \phi(\mathbf{w}) - \phi(\mathbf{w}') - \langle \nabla \phi(\mathbf{w}'), \mathbf{w} - \mathbf{w}' \rangle,$$

for \mathbf{w}, \mathbf{w}' in the domain \mathcal{W} of ϕ . It measures the deviation of $\phi(\mathbf{w})$ from its first-order Taylor approximation around \mathbf{w}' and is in general neither symmetric nor a metric. For twice differentiable ϕ , the divergence

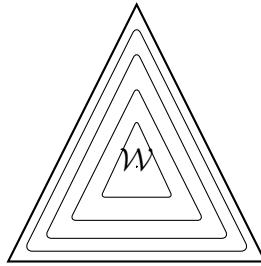


Fig. 3. Contour lines of a convex function $\phi(\cdot)$ defined on a convex set \mathcal{W} . The density of the contour lines can be used to steer the learning rate used in a gradient task: If the current model parameter lies in a region with dense contour lines, a smaller learning rate is preferable

$D_\phi(\cdot, \cdot)$ behaves locally like a quadratic form

$$D_\phi(\mathbf{w}, \mathbf{w}') \approx \frac{1}{2} (\mathbf{w} - \mathbf{w}')^T \nabla^2 \phi(\mathbf{w}') (\mathbf{w} - \mathbf{w}').$$

which can be interpreted as a squared norm of the displacement $\mathbf{w} - \mathbf{w}'$ induced by $\nabla^2 \phi(\mathbf{w}')$.

See also: gradient task, proximal operator.

Cauchy sequence A Cauchy sequence is a sequence $(\mathbf{x}^{(r)})_{r \in \mathbb{N}}$ in a metric space $(\mathcal{X}, d(\cdot, \cdot))$ such that the elements $\mathbf{x}^{(r)} \in \mathcal{X}$ become arbitrarily close to each other eventually. In other words [2, Def. 3.8],

$$\forall \epsilon > 0, \exists N \in \mathbb{N} \text{ such that } \forall r, r' \geq N, d(\mathbf{x}^{(r)}, \mathbf{x}^{(r')}) < \epsilon.$$

Fig. 4 shows a Cauchy sequence in the metric space $(\mathbb{Q}, |\cdot|)$ of rational numbers.

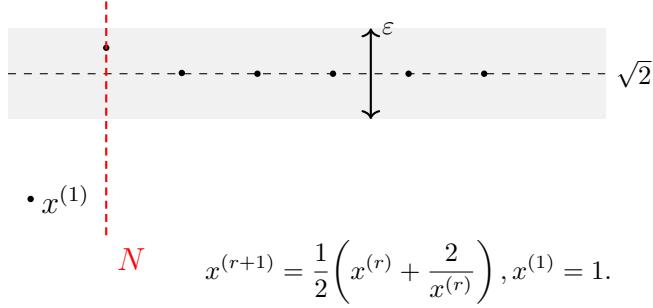


Fig. 4. A Cauchy sequence $(x^{(r)})_{r \in \mathbb{N}}$ in the metric space $(\mathbb{Q}, |\cdot|)$. This sequence is generated by a fixed-point iteration used to approximate $\sqrt{2}$. For all $r \geq N$, the sequence elements lie within a band of width ε . Note that the sequence does not converge in \mathbb{Q} , since $\sqrt{2} \notin \mathbb{Q}$ [2, Example 1.1].

See also: sequence, metric space.

Chebyshev's inequality Consider a real-valued RV x for which the second moment $\mathbb{E}\{x^2\}$ exists (and is finite). The existence of the second moment implies the existence of a finite expectation $\mu := \mathbb{E}\{x\}$ and a finite varianssi $\sigma^2 := \mathbb{E}\{(x-\mu)^2\}$ [13, Proposition 6.12]. Chebyshev's inequality refers to the following upper bound on the todennäköisyys that x

deviates from μ by more than a given threshold η [14, Ch. 4]. In particular,

$$\mathbb{P}(|x - \mu| \geq \eta) \leq \frac{\sigma^2}{\eta^2} \quad \text{for some } \eta > 0.$$

This upper bound can be obtained by applying Markov's inequality to the new RV $\tilde{x} := (x - \mu)^2$.

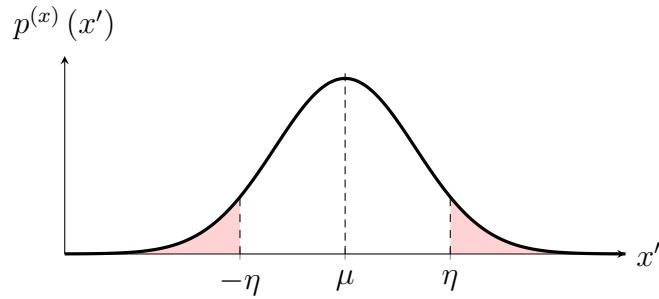


Fig. 5. Chebyshev's inequality provides an upper bound on the tail todennäköisyys $\mathbb{P}(|x - \mu| \geq \eta)$ (i.e., shaded area) of a real-valued RV x with a finite second moment.

See also: expectation, Markov's inequality, concentration inequality.

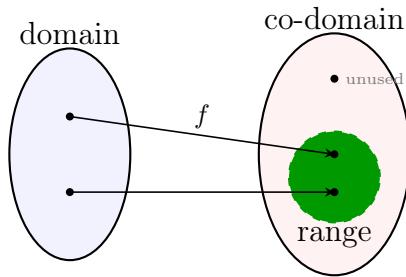
Chernoff bound The Chernoff bound is a concentration inequality derived as a direct application of Markov's inequality [15, Ch. 2]. Let x be a real-valued RV such that its MGF $M_x(t) = \mathbb{E}\{\exp(tx)\}$ exists for some $t > 0$. Applying Markov's inequality to the nonnegative RV $\exp(tx)$ yields, for any $\eta \in \mathbb{R}$,

$$\mathbb{P}(x \geq \eta) = \mathbb{P}(\exp(tx) \geq \exp(t\eta)) \leq \exp(-t\eta) \mathbb{E}\{\exp(tx)\}.$$

Note that this is actually an entire family of upper bounds, parameterized by all valid choices for $t > 0$ (i.e., $M_x(t)$ must exist).

See also: Markov's inequality, Chebyshev's inequality, Hoeffding's inequality, concentration inequality.

co-domain The co-domain of a function $f : \mathcal{U} \rightarrow \mathcal{V}$ is the set \mathcal{V} into which f maps elements of its domain \mathcal{U} .



See also: domain, function, mapping.

column space The column space of a matrix $\mathbf{A} \in \mathbb{R}^{m \times d}$, denoted by $\text{span}(\mathbf{A})$, is the set of all linear combinations of the columns of \mathbf{A} . In other words,

$$\text{span}(\mathbf{A}) = \{\mathbf{Aw} : \mathbf{w} \in \mathbb{R}^d\}.$$

The column space $\text{span}(\mathbf{A})$ of the matrix \mathbf{A} is a subspace of the Euclidean space \mathbb{R}^m .

See also: matrix, vector space.

concentration inequality An upper bound on the probability that a random variable (RV) deviates more than a prescribed amount from its expectation [16].

See also: probability, RV, expectation.

condition number The condition number $\kappa(\mathbf{Q}) \geq 1$ of a positive definite matriisi $\mathbf{Q} \in \mathbb{R}^{d \times d}$ is the ratio α/β , where α and β denote the largest and smallest eigenvalues of \mathbf{Q} , respectively. The condition number is useful for the analysis of koneoppiminen methods. The computational complexity of gradient-based methods for linear regression depends crucially on the condition number of the matriisi $\mathbf{Q} = \mathbf{X}\mathbf{X}^T$, where \mathbf{X} is the feature matrix of the training set. These methods converge faster when the condition number $\kappa(\mathbf{Q})$ is close to 1.

See also: matriisi, eigenvalue, gradient-based method.

conditional expectation Consider a numeric RV $\mathbf{x} \in \mathbb{R}^d$ defined on a probability space $\mathcal{P} = (\Omega, \mathcal{F}, \mathbb{P})$. Let $\Sigma' \subseteq \Sigma$ be a (sub-) σ -algebra that represents partial information about the outcome of a satunnaiskoe. The conditional expectation of \mathbf{x} given (or conditioned on) Σ' , denoted $\mathbb{E}\{\mathbf{x} | \Sigma'\}$, is a numeric RV that [6], [17]: 1) is measurable with respect to Σ' ; and 2) satisfies

$$\int_{\mathcal{A}} \mathbb{E}\{\mathbf{x} | \Sigma'\} d\mathbb{P} = \int_{\mathcal{A}} \mathbf{x} d\mathbb{P} \quad \text{for any } \mathcal{A} \in \Sigma'.$$

Intuitively, $\mathbb{E}\{\mathbf{x} | \mathcal{F}'\}$ summarizes the average value of \mathbf{x} using only information contained in the (typically smaller) σ -algebra Σ' [6], [18], [19].

See also: probability space, σ -algebra, expectation.

conditional probability distribution Consider a satunnaisprosessi consisting of two RVs \mathbf{x} and y with probability distribution $\mathbb{P}^{(\mathbf{x},y)}$. The conditional probability distribution of y given (or conditioned on) \mathbf{x} is denoted by $\mathbb{P}^{(y|\mathbf{x})}$. It is defined via the conditional expectations of the

indicator function of measurable sets in the σ -algebra generated by the RV y [6], [20].

See also: probability distribution, conditional expectation.

conditional probability mass function (conditional pmf) Consider two discrete RVs y and x defined on the same probability space $(\Omega, \mathcal{F}, \mathbb{P}(\cdot))$. The conditional pmf of y given (or conditioned on) x is denoted $p^{(y|x)}(\cdot | \cdot)$ and is defined by

$$p^{(y|x)}(y' | x') := \mathbb{P}(y = y' | x = x'),$$

for all realizations y', x' with $\mathbb{P}(x = x') > 0$. Equivalently, the conditional pmf can be expressed using conditional expectation as

$$p^{(y|x)}(y' | x') = \mathbb{E}\{\mathbb{I}_{y=y'} | \Sigma(x)\}(x'),$$

where $\Sigma(x)$ denotes the σ -algebra generated by the RV x .

See also: conditional expectation, pmf, probability space.

conjugate transpose The conjugate transpose of a matrix is obtained by transposing the matrix and taking the complex conjugate of each entry. For a matrix $\mathbf{A} \in \mathbb{C}^{m \times d}$, its conjugate transpose is denoted by $\mathbf{A}^H \in \mathbb{C}^{d \times m}$ and is defined entrywise by

$$(\mathbf{A}^H)_{j,r} = \overline{(\mathbf{A})_{r,j}},$$

where $\overline{(\cdot)}$ denotes complex conjugation.

contractive operator An operator $\mathcal{F} : \mathbb{R}^d \rightarrow \mathbb{R}^d$ is a contraction (or contractive) if, for some $\kappa \in [0, 1)$, [21], [22]

$$\|\mathcal{F}\mathbf{w} - \mathcal{F}\mathbf{w}'\|_2 \leq \kappa \|\mathbf{w} - \mathbf{w}'\|_2 \text{ holds for any } \mathbf{w}, \mathbf{w}' \in \mathbb{R}^d.$$

The notion of a contractive operator generalizes naturally from \mathbb{R}^d to arbitrary metric spaces [21].

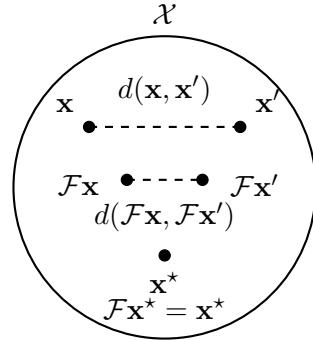


Fig. 6. A contractive operator $\mathcal{F} : \mathcal{X} \rightarrow \mathcal{X}$ has a unique fixed point \mathbf{x}^* with $\mathcal{F}\mathbf{x}^* = \mathbf{x}^*$. For any two points \mathbf{x}, \mathbf{x}' in the same space, the distance between their images $\mathcal{F}\mathbf{x}$ and $\mathcal{F}\mathbf{x}'$ is strictly smaller.

Intuitively, a contractive operator brings any two points from its domain closer together by at least a factor of κ .

See also: operator, metric space.

convergence Consider a sequence $(a_r)_{r \in \mathbb{N}}$ with numeric values $a_r \in \mathbb{R}$. This sequence is said to converge to a value a^* if the values a_r become arbitrarily close to a^* for sufficiently large indices r . Mathematically speaking, the sequence converges to a^* if [1], [2]

$$\forall \epsilon > 0, \exists N \in \mathbb{N} : r > N \Rightarrow |a_r - a^*| < \epsilon.$$

We denote the convergence of a sequence to a^* by

$$\lim_{r \rightarrow \infty} a_r = a^*.$$

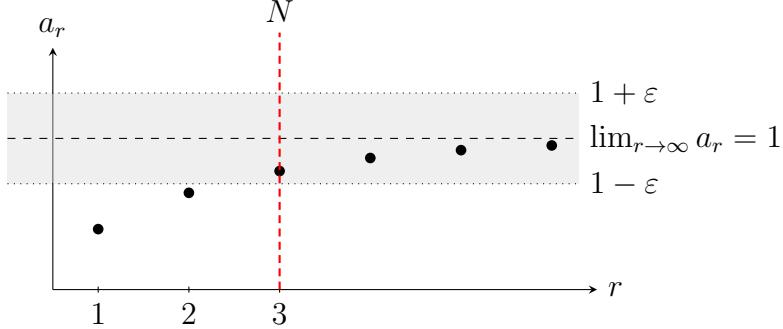


Fig. 7. A real-valued sequence $(a_r)_{r \in \mathbb{N}}$ converging to the limit $a^* = 1$.

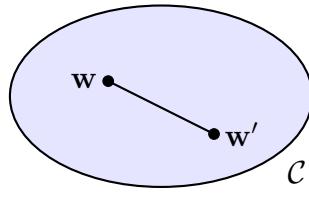
The concept of convergence of a real-valued sequence (where $\mathcal{A} = \mathbb{R}$) extends naturally to a sequence in an arbitrary metric space \mathcal{A} . Indeed, we just need to replace the absolute difference $|a_r - a^*|$ by the metric $d(a_r, a^*)$. Note that a sequence can only converge if it is a Cauchy sequence [2]. However, not every Cauchy sequence is converging unless the underlying metric space is complete.

See also: sequence, metric space, Cauchy sequence.

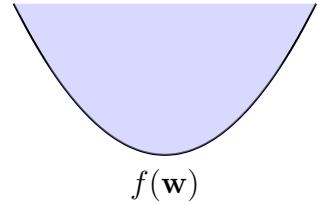
convex A subset $\mathcal{C} \subseteq \mathbb{R}^d$ of the Euclidean space \mathbb{R}^d is referred to as convex if it contains the line segment between any two points $\mathbf{x}, \mathbf{y} \in \mathcal{C}$ in that set, i.e.,

$$\alpha\mathbf{x} + (1 - \alpha)\mathbf{y} \in \mathcal{C}, \text{ for any } \alpha \in [0, 1].$$

Similarly, a function $f: \mathbb{R}^d \rightarrow \mathbb{R}$ is convex if its epigraph $\{(\mathbf{w}^T, t)^T \in \mathbb{R}^{d+1} : t \geq f(\mathbf{w})\}$ is a convex set [23]. We illustrate one example of a convex set and a convex function in Fig. 8.



(a)



(b)

Fig. 8. (a) A convex set $\mathcal{C} \subseteq \mathbb{R}^d$. (b) A convex function $f : \mathbb{R}^d \rightarrow \mathbb{R}$.

See also: Euclidean space, funktion, epigrafi.

convex optimization problem See konveksi optimointi.

countable A set is called countable if its elements can be put into a one-to-one correspondence with the natural numbers $\mathbb{N} = \{1, 2, 3, \dots\}$ or with a finite subset of \mathbb{N} [24]. Equivalently, a set \mathcal{A} is countable if there exists an injective funktion $f : \mathcal{A} \rightarrow \mathbb{N}$.

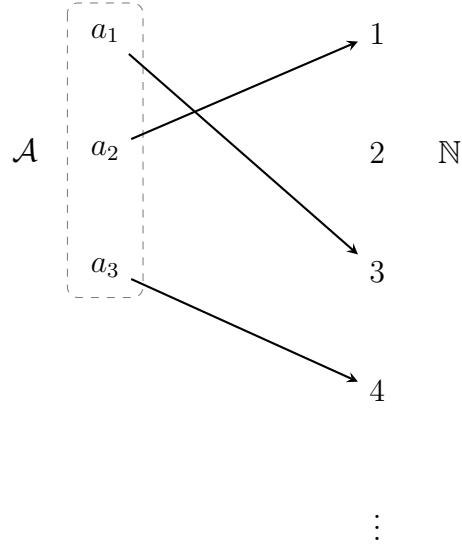


Fig. 9. An injective funktion that maps the elements of a finite set \mathcal{A} to the natural numbers \mathbb{N} , which implies that \mathcal{A} is countable.

Typical examples include the set of integers \mathbb{Z} and rational numbers \mathbb{Q} . In contrast, the set of real numbers \mathbb{R} is not countable, meaning no such one-to-one correspondence with \mathbb{N} exists.

See also: injective, funktion.

Courant–Fischer–Weyl min–max characterization (CFW min–max characterization)

Consider a psd matriisi $\mathbf{Q} \in \mathbb{R}^{d \times d}$ with eigenvalue decomposition (EVD) (or spectral decomposition), i.e.,

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

Here, we use the ordered (in ascending order) eigenvalues

$$\lambda_1 \leq \dots \leq \lambda_n.$$

The CFW min–max characterization [3, Th. 8.1.2] represents the eigenvalues of \mathbf{Q} as the solutions to certain optimointitehtävät.

See also: psd, matriisi, EVD, eigenvalue, optimointitehtävä.

covariance matrix The covariance matriisi of an RV $\mathbf{x} \in \mathbb{R}^d$ is defined as the expectation (if it exists)

$$\mathbf{C}^{(\mathbf{x})} := \mathbb{E} \left\{ (\mathbf{x} - \mathbb{E}\{\mathbf{x}\})(\mathbf{x} - \mathbb{E}\{\mathbf{x}\})^T \right\}.$$

See also: covariance, matriisi, RV.

cumulative distribution function (cdf) The cdf $F^{(x)}(\eta)$ of a real-valued RV x is [25], [26]

$$F^{(x)}(\eta) := \mathbb{P}(x \leq \eta).$$

See also: RV, pdf, probability distribution.

derivative See partial derivative.

determinantti The determinant $\det(\mathbf{A})$ of a square matriisi $\mathbf{A} = (\mathbf{a}^{(1)}, \dots, \mathbf{a}^{(d)}) \in \mathbb{R}^{d \times d}$ is a funktio of its columns $\mathbf{a}^{(1)}, \dots, \mathbf{a}^{(d)} \in \mathbb{R}^d$, i.e., it satisfies the following properties [27]:

- Normalized:

$$\det(\mathbf{I}) = 1$$

- Multilinear:

$$\det(\mathbf{a}^{(1)}, \dots, \alpha\mathbf{u} + \beta\mathbf{v}, \dots, \mathbf{a}^{(d)}) = \alpha \det(\mathbf{a}^{(1)}, \dots, \mathbf{u}, \dots, \mathbf{a}^{(d)}) \\ + \beta \det(\mathbf{a}^{(1)}, \dots, \mathbf{v}, \dots, \mathbf{a}^{(d)})$$

- Antisymmetric:

$$\det(\dots, \mathbf{a}^{(j)}, \dots, \mathbf{a}^{(j')}, \dots) = -\det(\dots, \mathbf{a}^{(j')}, \dots, \mathbf{a}^{(j)}, \dots).$$

We can interpret a matrix \mathbf{A} as a linear transformation on \mathbb{R}^d . The determinant $\det(\mathbf{A})$ characterizes how volumes in \mathbb{R}^d (and their orientation) are altered by this transformation (see Fig. 10) [3], [28]. In particular, $\det(\mathbf{A}) > 0$ preserves orientation, $\det(\mathbf{A}) < 0$ reverses orientation, and $\det(\mathbf{A}) = 0$ collapses volume entirely, indicating that \mathbf{A} is non-invertible. The determinant also satisfies $\det(\mathbf{AB}) = \det(\mathbf{A}) \cdot \det(\mathbf{B})$, and if \mathbf{A} is diagonalizable with eigenvalues $\lambda_1, \dots, \lambda_d$, then $\det(\mathbf{A}) = \prod_{j=1}^d \lambda_j$ [29]. For the special cases $d = 2$ (i.e., two-dimensional or 2-D) and $d = 3$ (i.e., three-dimensional or 3-D), the determinant can be interpreted as an oriented area or volume spanned by the column vectors of \mathbf{A} .

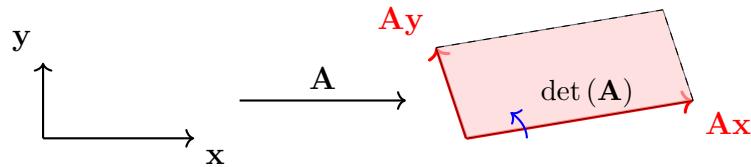


Fig. 10. We can interpret a square matrix \mathbf{A} as a linear transformation of \mathbb{R}^d into itself. The determinant $\det(\mathbf{A})$ characterizes how this transformation alters an oriented volume.

See also: eigenvalue, käänteismatriisi.

diagonalizable A square matriisi $\mathbf{A} \in \mathbb{C}^{d \times d}$ is called diagonalizable if it is similar to a diagonal matriisi [29], [30]. Formally, \mathbf{A} is diagonalizable if there exists an invertible matriisi $\mathbf{P} \in \mathbb{C}^{d \times d}$ such that

$$\mathbf{A} = \mathbf{P}\mathbf{D}\mathbf{P}^{-1}$$

where $\mathbf{D} \in \mathbb{C}^{d \times d}$ is a diagonal matriisi whose main diagonal entries are the eigenvalues of \mathbf{A} . A matriisi $\mathbf{A} \in \mathbb{R}^{d \times d}$ is diagonalizable if and only if it has d linearly independent eigenvectort [29].

See also: matriisi, eigenvalue, EVD.

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

differential entropy For an RV $\mathbf{x} \in \mathbb{R}^d$ with a pdf $p^{(x)}(\cdot)$, the differential entropia is defined as [31]

$$h(\mathbf{x}) := - \int_{\mathbf{x}' \in \mathbb{R}^d} \log p(\mathbf{x}') \, dp^{(x)}(\mathbf{x}').$$

Differential entropia can be negative and lacks some properties of entropia for discrete-valued RVs, such as invariance under a change of variables [31]. Among all RVs with a given keskiarvo $\boldsymbol{\mu}$ and covariance matrix \mathbf{C} , $h(\mathbf{x})$ is maximized by $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{C})$.

See also: epävarmuus, todennäköisyysmalli.

dimension The dimension $\dim \mathcal{A}$ of a vektoriavaruus \mathcal{A} is the cardinality of any basis of \mathcal{A} [32]. Strictly speaking, this definition applies only to finite-dimensional vektoriavaruudet, i.e., those that possess a finite basis.

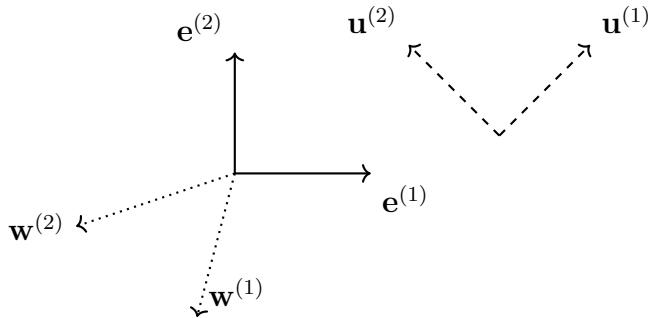


Fig. 11. Three bases, $\{\mathbf{e}^{(1)}, \mathbf{e}^{(2)}\}$, $\{\mathbf{u}^{(1)}, \mathbf{u}^{(2)}\}$, $\{\mathbf{w}^{(1)}, \mathbf{w}^{(2)}\}$, for the vektoriavaruus \mathbb{R}^2 .

For such spaces, all bases have the same cardinality, which is the dimension of the space [30, Ch. 2].

See also: vektoriavaruus, basis.

directed acyclic graph (DAG) A DAG is a directed graph which contains no directed cycles. Formally, a DAG $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ satisfies that, for any sequence of distinct nodes (i_1, \dots, i_k) , the presence of directed edges $(i_1, i_2), (i_2, i_3), \dots, (i_{k-1}, i_k)$ implies that $(i_k, i_1) \notin \mathcal{E}$.



Fig. 12. (a) A DAG defined on three nodes $\mathcal{V} = \{1, 2, 3\}$. (b) Another directed graph on the same nodes that is not a DAG, since it contains a directed cycle.

The absence of directed cycles allows for a topological ordering of nodes such that all edges point from earlier to later nodes in this order. Several koneoppiminen models, such as ANNs or decision treet, are naturally represented as DAGs.

See also: directed graph, ANN, decision tree.

directed cycle A directed cycle in a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a sequence of distinct nodes (i_1, i_2, \dots, i_k) such that $(i_1, i_2), (i_2, i_3), \dots, (i_{k-1}, i_k), (i_k, i_1) \in \mathcal{E}$. In a directed cycle, following the direction of each edge eventually leads back to the starting node, creating a closed loop.

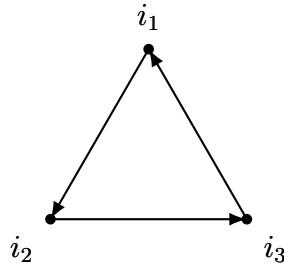


Fig. 13. A directed cycle consisting of three nodes kytketty verkko in a closed loop.

The presence of a directed cycle prevents a directed graph from being a

directed acyclic graph (DAG).

See also: directed graph, DAG.

directed graph A directed verkko contains edges that have an orientation (or direction). Mathematically, a directed verkko $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ consists of nodes \mathcal{V} and a set $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ of directed edges.

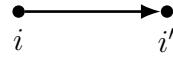


Fig. 14. The edges of a directed verkko have an orientation (or direction). We can indicate the orientation by an arrow head.

We can represent a directed edge from node $i \in \mathcal{V}$ to node $i' \in \mathcal{V}$ by an ordered pair (i, i') . Directed verkot are widely used to model interconnected systems or networks, such as transportation systems, electronic circuits, and biological processes [33].

See also: verkko.

discrete random variable (discrete RV) A RV, i.e., a funktion that maps the outcomes of a satunnaiskoe to elements of a measurable space \mathcal{X} , is referred to as discrete if its value space \mathcal{X} countable [6].

See also: RV, todennäköisyys, probability distribution.

domain The domain of a funktion $f : \mathcal{U} \rightarrow \mathcal{V}$ is the set \mathcal{U} from which f takes its inputs.

See also: funktion, co-domain, kuvaus.

eigenvalue We refer to a number $\lambda \in \mathbb{R}$ as an eigenvalue of a square matriisi $\mathbf{A} \in \mathbb{R}^{d \times d}$ if there exists a nonzero vektori $\mathbf{x} \in \mathbb{R}^d \setminus \{\mathbf{0}\}$ such that $\mathbf{Ax} = \lambda\mathbf{x}$.

$$\xrightarrow{\mathbf{x}} \mathbf{A} \rightarrow \xrightarrow{\lambda\mathbf{x}}$$

This vector is the eigenvector corresponding to the eigenvalue λ .

See also: matriisi, vektori.

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

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

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

See also: matriisi, eigenvector, eigenvalue, diagonalizable.

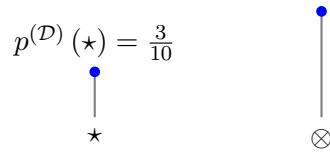
empirical distribution Consider a dataset $\mathcal{D} = \{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}\}$ consisting of m distinct data points, each characterized by the feature vector $\mathbf{x}^{(r)} \in \mathcal{X}$ for $r = 1, \dots, m$. For a given σ -algebra Σ over the feature space \mathcal{X} , the empirical distribution of \mathcal{D} is the probability distribution $\mathbb{P}^{(\mathcal{D})}$ defined via

$$\mathbb{P}^{(\mathcal{D})}(\mathcal{A}) = (1/m) |\{r : \mathbf{x}^{(r)} \in \mathcal{A}\}|, \quad \text{for any event } \mathcal{A} \in \Sigma.$$

In other words, the empirical distribution assigns to any measurable set $\mathcal{A} \in \Sigma$ the fraction of data points in \mathcal{D} that fall into \mathcal{A} . If the feature space is ordered, the empirical distribution can also be characterized by its empirical cumulative distribution function (cdf)

$$F^{(\mathcal{D})}(\mathbf{x}) = (1/m) |\{r : \mathbf{x}^{(r)} \preceq \mathbf{x}\}|, \quad \text{for any } \mathbf{x} \in \mathcal{X}$$

where \preceq denotes the ordering relation on \mathcal{X} .



$$\mathcal{D} = (\star, \star, \star, \otimes, \otimes, \otimes, \otimes, \otimes, \otimes, \otimes)$$

Fig. 15. A dataset \mathcal{D} consisting of $m = 10$ data points, each characterized by a value from the finite feature space $\mathcal{X} = \{\star, \otimes\}$. The empirical pmf $p^{(\mathcal{D})}(\mathbf{x})$ assigns to each possible value $\mathbf{x} \in \mathcal{X}$ the fraction of data points in \mathcal{D} whose feature takes on this value. Here, three out of ten data points take on the feature value \star , resulting in $p^{(\mathcal{D})}(\star) = 3/10$.

If the feature space \mathcal{X} is finite, the empirical distribution of \mathcal{D} can also be characterized by the empirical pmf

$$p^{(\mathcal{D})}(\mathbf{x}) = (1/m) |\{r : \mathbf{x}^{(r)} = \mathbf{x}\}|, \quad \text{for any } \mathbf{x} \in \mathcal{X}.$$

See also: σ -algebra, probability distribution.

entropia Entropy quantifies the epävarmuus or unpredictability associated with an RV [31]. For a discrete RV x taking on values in a finite set $\mathcal{S} = \{x_1, \dots, x_k\}$ with a pmf $p^{(x)}(x_c)$ ($= \mathbb{P}(x = x_c)$), the entropy is defined as [31]

$$H(x) := - \sum_{c=1}^k p^{(x)}(x_c) \log p^{(x)}(x_c).$$

For a given set of values \mathcal{S} , the entropy is maximized for a uniformly distributed RV, where $p^{(x)}(x_c) = 1/k$. The minimal entropy, which is zero, is obtained when $p^{(x)}(x_c) = 1$ for some $x_c \in \mathcal{S}$. Differential entropy generalizes the concept of entropia from discrete RVs to jatkuva RVs.

See also: epävarmuus, todennäköisyyssmalli.

epigrafi The epigraph of a real-valued funktio $f : \mathbb{R}^d \rightarrow \mathbb{R} \cup \{+\infty\}$ is the set of points lying on or above its graph (see Fig. 16), i.e.,

$$\text{epi}(f) = \{(\mathbf{w}, t) \in \mathbb{R}^d \times \mathbb{R} \mid f(\mathbf{w}) \leq t\}.$$

A funktio is convex if and only if its epigraph is a convex set [23], [34].

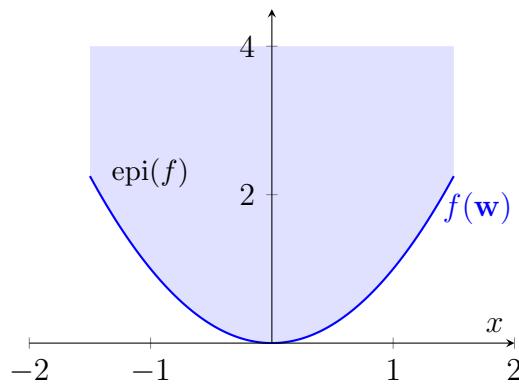


Fig. 16. Epigraph of the funktio $f(w) = w^2$ (i.e., the shaded area).

See also: *funktio*, *convex*.

Erdős–Rényi -verkko (ER-verkko) An ER verkko [35], [36] is a todennäköisyysmalli for verkot defined over a given node set $i = 1, \dots, n$. One way to define the ER verkko is via the collection of independent and identically distributed (i.i.d.) binary RVs $b^{(\{i,i'\})} \in \{0, 1\}$, for each pair of different nodes i, i' . A specific realization of an ER verkko contains an edge $\{i, i'\}$ if and only if $b^{(\{i,i'\})} = 1$. The ER verkko is parameterized by the number n of nodes and the todennäköisyys $\mathbb{P}(b^{(\{i,i'\})} = 1)$.

See also: *verkko*, *todennäköisyysmalli*, *i.i.d.*, *RV*, *realization*, *todennäköisyys*.

event Consider an RV \mathbf{x} , defined on some probability space, which takes values in a measurable space \mathcal{X} . An event $\mathcal{A} \subseteq \mathcal{X}$ is a subset of \mathcal{X} such that the todennäköisyys $\mathbb{P}(\mathbf{x} \in \mathcal{A})$ is well defined. In other words, the preimage $\mathbf{x}^{-1}(\mathcal{A})$ of an event belongs to the underlying σ -algebra, i.e., the preimage is a measurable subset of the sample space [1], [6], [17]. Roughly speaking, an event represents a set of possible outcomes of some process. One example of such a process could also be the treatment of a health-care patient.

See also: *RV*, *data point*, *independent and identically distributed assumption* (*i.i.d. assumption*), *todennäköisyysmalli*.

expectation–maximization (EM) The EM algoritmi [37] is an iterative optimointimenetelmä for approximately solving certain maximum likelihood optimointitehtävät that are difficult to solve directly [38, Sec. 9.4], [39, Sec. 11.4.7]. To motivate the EM algoritmi and explain its

construction, consider a koneoppiminen application involving a single observed data point with feature $x \in \mathcal{X}$, where \mathcal{X} is a finite feature space. The data generation is modeled via a todennäköisyysmalli that consists of a RV x' with pmf $p^{(x')}(\cdot; \mathbf{w})$. Here, the actual model parameters $\mathbf{w} \in \mathcal{W}$ - used for the data generation via sampling from $p^{(x')}(\cdot; \mathbf{w})$ - are unknown. A widely used approach for estimating these model parameters is via the solutions of the maximum likelihood problem

$$\min_{\mathbf{w} \in \mathcal{W}} -\log p^{(x')}(\mathbf{x}; \mathbf{w}). \quad (1)$$

For some todennäköisyysmallit, such as a Gaussian mixture model (GMM), this optimointitehtävä can be difficult to solve directly. As a work-around, one can often introduce an auxiliary attribute $y \in \mathcal{Y}$, generated via some RV y' , such that the corresponding todennäköisyysmalli $p^{(x',y')}(\cdot, \cdot; \mathbf{w})$ yields a much easier maximum likelihood problem

$$\min_{\mathbf{w} \in \mathcal{W}} -\log p^{(x',y')}(\mathbf{x}, \mathbf{y}; \mathbf{w}). \quad (2)$$

The attribute y is introduced solely to simplify (2), but it is not observed in practice—only the feature x is available. Thus, we cannot solve (2) directly as we do not know which value y to plug into the pmf $p^{(x',y')}(\mathbf{x}, \mathbf{y}; \mathbf{w})$. The EM method resolves this dilemma by alternating between two steps:

- (**E-step**) computing a “soft” estimate of the auxiliary attribute y in the form of the posterior $p^{(y'|x')}(\cdot; \hat{\mathbf{w}})$ using the current choice $\hat{\mathbf{w}}$ for the model parameters, and
- (**M-step**) minimizing a surrogate objective function derived from this posterior.

The completion of these two steps constitutes one full iteration of the EM method. In more detail, the E-step produces the function

$$Q(\mathbf{w} \mid \widehat{\mathbf{w}}) := - \sum_{y \in \mathcal{Y}} p^{(y|x')} (y; \widehat{\mathbf{w}}) \log \left(p^{(x',y')} (x, y; \mathbf{w}) / p^{(y|x')} (y; \widehat{\mathbf{w}}) \right),$$

and the M-step minimizes $Q(\mathbf{w} \mid \widehat{\mathbf{w}})$ over $\mathbf{w} \in \mathcal{W}$. This function satisfies two key properties [38, Sec. 9.4], [39, Sec. 11.4.7]:

- **Upper bound:**

$$Q(\mathbf{w} \mid \widehat{\mathbf{w}}) \geq - \log p^{(x')} (x; \mathbf{w}) \quad \text{for all } \mathbf{w} \in \mathcal{W}. \quad (3)$$

- **Tightness:**

$$Q(\widehat{\mathbf{w}} \mid \widehat{\mathbf{w}}) = - \log p^{(x')} (x; \widehat{\mathbf{w}}). \quad (4)$$

To summarize, during each iteration, EM minimizes an upper-bounding surrogate objective function that is tight at the current iterate $\widehat{\mathbf{w}}$. Thus, EM is a majorize–minimize (MM) method for approximately solving (1). The above construction and analysis of EM can be extended to more general settings involving multiple data points and infinite feature space such as \mathbb{R}^d ; see [39, Sec. 11.4.7] for details.

See also: todennäköisyysmalli, maximum likelihood, optimointitehtävä.

firmly non-expansive operator An operator $\mathcal{F} : \mathcal{H} \rightarrow \mathcal{H}$ defined on a Hilbert space \mathcal{H} is called firmly non-expansive if it satisfies

$$\|\mathcal{F}\mathbf{w} - \mathcal{F}\mathbf{w}'\|_2^2 \leq \langle \mathbf{w} - \mathbf{w}', \mathcal{F}\mathbf{w} - \mathcal{F}\mathbf{w}' \rangle \text{ for any } \mathbf{w}, \mathbf{w}' \in \mathcal{H}.$$

Any firmly non-expansive operator is necessarily also a non-expansive operator [21]. Every fixed-point iteration that uses a firmly non-expansive

operator converges to a fixed point of the operator [21].

See also: fixed-point iteration, contractive operator.

fixed point Consider some operator $\mathcal{F} : \mathcal{H} \rightarrow \mathcal{H}$ defined on a Hilbert space \mathcal{H} . A vektori $\widehat{\mathbf{w}} \in \mathcal{H}$ is called a fixed point of the operator \mathcal{F} if it satisfies

$$\mathcal{F}\widehat{\mathbf{w}} = \widehat{\mathbf{w}}.$$

In other words, applying the operator \mathcal{F} to its fixed point $\widehat{\mathbf{w}}$ returns the same vektori $\widehat{\mathbf{w}}$. Finding a fixed point of a suitable operator \mathcal{F} is a common approach for solving various optimointitehtävät (e.g., an instance of empirical risk minimization (ERM)). A popular method for computing approximations of a fixed point is the fixed-point iteration.

See also: fixed-point iteration.

fixed-point equation A fixed-point equation is an equation of the form

$$\mathbf{w} = \mathcal{F}(\mathbf{w}),$$

where $\mathcal{F} : \mathcal{H} \rightarrow \mathcal{H}$ is an operator defined on a Hilbert space \mathcal{H} . Solving a fixed-point equation amounts to finding the fixed points of \mathcal{F} . Many , including instances of ERM, can be cast in this form. For example, minimizing a smooth convex funktio f is equivalent to solving the fixed-point equation

$$\mathbf{w} = \mathcal{F}\mathbf{w} \text{ with } \mathcal{F} : \mathbf{w} \mapsto \mathbf{w} - \eta \nabla f(\mathbf{w})$$

. Here, $\eta > 0$ can be choosen freely. The above fixed-point equation is nothing but the zero-gradient condition for the minimizer of f [23].

Similarly, one can reforulate the optimality conditions (KKT conditions) of optimointitehtävä with constraints as a fixed-point equation [23, 40]

fixed-point iteration A fixed-point iteration is an iterative method for solving an optimointitehtävä. It constructs a sequence $\mathbf{w}^{(0)}, \mathbf{w}^{(1)}, \dots$ by repeatedly applying an operator \mathcal{F} , i.e.,

$$\mathbf{w}^{(t+1)} = \mathcal{F}\mathbf{w}^{(t)}, \text{ for } t = 0, 1, \dots \quad (5)$$

The operator \mathcal{F} is chosen such that any of its fixed points is a solution $\hat{\mathbf{w}}$ to the given optimointitehtävä. For example, given a differentiable and convex funktio $f(\mathbf{w})$, the fixed points of the operator $\mathcal{F} : \mathbf{w} \mapsto \mathbf{w} - \nabla f(\mathbf{w})$ coincide with the minimizers of $f(\mathbf{w})$. In general, for a given optimointitehtävä with solution $\hat{\mathbf{w}}$, there are many different operators \mathcal{F} whose fixed points are $\hat{\mathbf{w}}$. Clearly, we should use an operator \mathcal{F} in (5) that reduces the distance to a solution such that

$$\underbrace{\|\mathbf{w}^{(t+1)} - \hat{\mathbf{w}}\|_2}_{\stackrel{(5)}{=} \|\mathcal{F}\mathbf{w}^{(t)} - \mathcal{F}\hat{\mathbf{w}}\|_2} \leq \|\mathbf{w}^{(t)} - \hat{\mathbf{w}}\|_2.$$

Thus, we require \mathcal{F} to be at least a non-expansive operator, i.e., the iteration (5) should not result in worse model parameters that have a larger distance to a solution $\hat{\mathbf{w}}$. Furthermore, each iteration (5) should also make some progress, i.e., reduce the distance to a solution $\hat{\mathbf{w}}$. This requirement can be made precise using the notion of a contractive operator [41], [42]. The operator \mathcal{F} is a contractive operator if, for some $\kappa \in [0, 1)$,

$$\|\mathcal{F}\mathbf{w} - \mathcal{F}\mathbf{w}'\|_2 \leq \kappa \|\mathbf{w} - \mathbf{w}'\|_2 \text{ holds for any } \mathbf{w}, \mathbf{w}'.$$

For a contractive operator \mathcal{F} , the fixed-point iteration (5) generates a sequence $\mathbf{w}^{(t)}$ that converges quite rapidly. In particular [2, Th. 9.23],

$$\|\mathbf{w}^{(t)} - \hat{\mathbf{w}}\|_2 \leq \kappa^t \|\mathbf{w}^{(0)} - \hat{\mathbf{w}}\|_2.$$

Here, $\|\mathbf{w}^{(0)} - \hat{\mathbf{w}}\|_2$ is the distance between the initialization $\mathbf{w}^{(0)}$ and the solution $\hat{\mathbf{w}}$. It turns out that a fixed-point iteration (5) with a firmly non-expansive operator \mathcal{F} is guaranteed to converge to a fixed point of \mathcal{F} [41, Corollary 5.16]. Fig. 17 depicts examples of a non-expansive operator, a firmly non-expansive operator, and a contractive operator. All of these operators are defined on the 1-D space \mathbb{R} . Another example of a firmly non-expansive operator is the proximal operator of a convex function [22], [41].

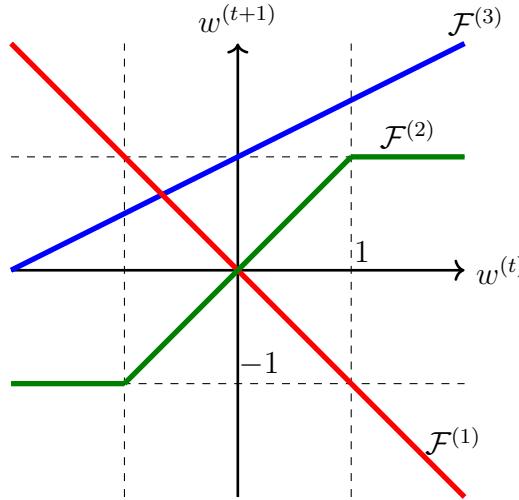


Fig. 17. Example of a non-expansive operator $\mathcal{F}^{(1)}$, a firmly non-expansive operator $\mathcal{F}^{(2)}$, and a contractive operator $\mathcal{F}^{(3)}$.

See also: contractive operator, proximal operator.

full-rank A matriisi $\mathbf{A} \in \mathbb{R}^{m \times d}$ is full-rank if it has maksimi rank [32]. For a tall matrix, i.e., when $d < m$, being full-rank means that its rank is equal to d .

$$\mathbf{A} = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \quad \text{full-rank square} \qquad \mathbf{B} = \begin{pmatrix} 1 & 2 \\ 2 & 4 \end{pmatrix} \quad \text{rank-deficient square}$$

$$\mathbf{C} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{pmatrix} \quad \text{full-rank tall matrix} \qquad \mathbf{D} = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 4 & 6 \end{pmatrix} \quad \text{rank-deficient wide matrix}$$

Fig. 18. Examples of full-rank and rank-deficient matriisit.

A square matriisi is full-rank if and only if it is invertible.

See also: matriisi, rank, dimension, lineaarikuvaus, column space.

funktio A function between two sets \mathcal{U} and \mathcal{V} assigns each element $u \in \mathcal{U}$ exactly one element $f(u) \in \mathcal{V}$ [2]. We write this as

$$f : \mathcal{U} \rightarrow \mathcal{V} : u \mapsto f(u)$$

where \mathcal{U} is the domain and \mathcal{V} the co-domain of f . That is, a function f defines a unique output $f(u) \in \mathcal{V}$ for every input $u \in \mathcal{U}$ (see Fig. 19).

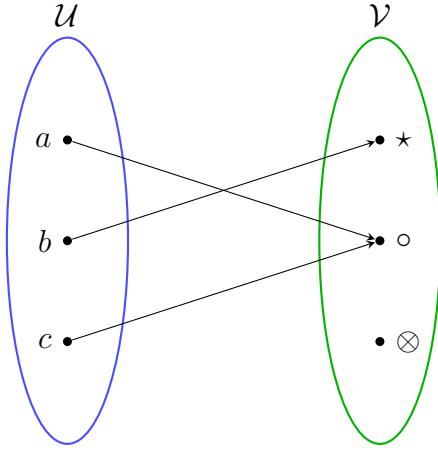


Fig. 19. A function $f: \mathcal{U} \rightarrow \mathcal{V}$ mapping each element of the domain $\mathcal{U} = \{a, b, c\}$ to exactly one element of the co-domain $\mathcal{V} = \{\star, \circ, \otimes\}$.

See also: domain, co-domain, output.

Gaussian See normaalijakautunut satunnaismuuttuja.

Gaussian prosessi A GP is a collection of RVs $\{f(\mathbf{x})\}_{\mathbf{x} \in \mathcal{X}}$ indexed by input values \mathbf{x} from some input space \mathcal{X} such that, for any finite subset $\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)} \in \mathcal{X}$, the corresponding RVs $f(\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)})$ have a joint moninormaalijakauma

$$f(\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}) \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{K}).$$

For a fixed input space \mathcal{X} , a GP is fully specified (or parameterized) by: 1) a keskiarvo funktio $\mu(\mathbf{x}) = \mathbb{E}\{f(\mathbf{x})\}$; and 2) a covariance funktio $K(\mathbf{x}, \mathbf{x}') = \mathbb{E}\{(f(\mathbf{x}) - \mu(\mathbf{x}))(f(\mathbf{x}') - \mu(\mathbf{x}'))\}$.

Example: We can interpret the temperature distribution across Finland (at a specific point in time) as the realization of a GP $f(\mathbf{x})$, where each

input $\mathbf{x} = (\text{lat}, \text{lon})$ denotes a geographic location. Temperature observations from Finnish Meteorological Institute (FMI) weather stations provide values $f(\mathbf{x})$ at specific locations (see Fig. 20). A GP allows us to predict the temperature nearby FMI weather stations and to quantify the epävarmuus of these ennusteet.

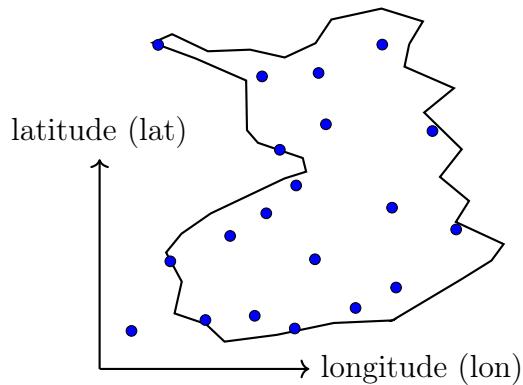


Fig. 20. For a given point in time, we can interpret the current temperature distribution over Finland as a realization of a GP indexed by geographic coordinates and sampled at FMI weather stations. The weather stations are indicated by blue dots.

See also: moninormaalijakauma, epävarmuus, normaalijakautunut satunnaismuuttuja.

gradientti For a real-valued funktion $f : \mathbb{R}^d \rightarrow \mathbb{R} : \mathbf{w} \mapsto f(\mathbf{w})$, if a vektori \mathbf{g} exists such that $\lim_{\mathbf{w} \rightarrow \mathbf{w}'} f(\mathbf{w}) - (f(\mathbf{w}') + \mathbf{g}^T(\mathbf{w} - \mathbf{w}')) / \|\mathbf{w} - \mathbf{w}'\| = 0$, it is referred to as the gradient of f at \mathbf{w}' . If it exists, the gradient is unique and denoted by $\nabla f(\mathbf{w}')$ or $\nabla f(\mathbf{w})|_{\mathbf{w}'}$ [2].

See also: funktio, vektori.

gradienttiaaskel Given a differentiable real-valued funktion $f(\cdot) : \mathbb{R}^d \rightarrow \mathbb{R}$ and a vektori $\mathbf{w} \in \mathbb{R}^d$, the gradientti step updates \mathbf{w} by adding the scaled negative gradientti $\nabla f(\mathbf{w})$ to obtain the new vektori (see Fig. 21)

$$\hat{\mathbf{w}} := \mathbf{w} - \eta \nabla f(\mathbf{w}). \quad (6)$$

The gradientti step amounts to applying the operator

$$\mathcal{T}^{(f,\eta)}(\cdot) : \mathbb{R}^d \rightarrow \mathbb{R}^d : \mathbf{w} \mapsto \mathbf{w} - \eta \nabla f(\mathbf{w}).$$

For a convex funktion $f(\cdot)$ that is sufficiently smooth and a sufficiently small learning rate $\eta > 0$, one can verify that $\mathcal{T}^{(f,\eta)}$ becomes a firmly non-expansive operator [21, Cor. 18.16]. For a strongly convex and smooth funktion $f(\cdot)$ and an appropriate choice of learning rate, one can verify that $\mathcal{T}^{(f,\eta)}$ is a contractive operator [45, Thm. 2.1.14.]. In these cases, gradient-based methods are instances of a fixed-point iteration since the operator $\mathcal{T}^{(f,\eta)}$ has a fixed point that coincides with the minimizer of $f(\cdot)$ (see zero-gradient condition).

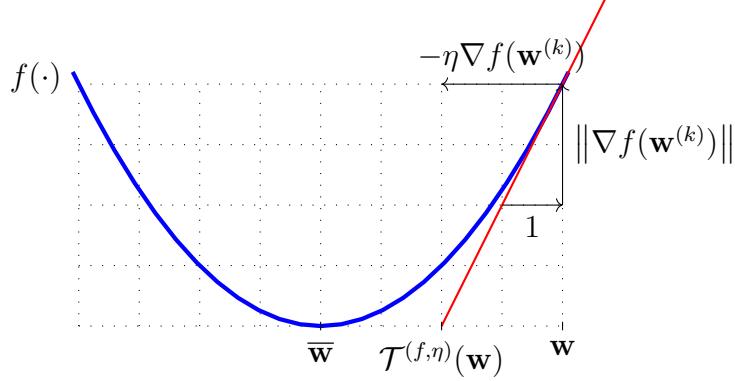


Fig. 21. The basic gradientti step (6) maps a given vektori \mathbf{w} to the updated vektori \mathbf{w}' .

Note that the gradientti step (6) optimizes locally—in a naapurusto whose size is determined by the step size η —a linear approximation to the funktilo $f(\cdot)$. A natural generalization of (6) is to locally optimize the funktilo itself—instead of its linear approximation—such that

$$\hat{\mathbf{w}} = \arg \min_{\mathbf{w}' \in \mathbb{R}^d} f(\mathbf{w}') + \frac{1}{\eta} \|\mathbf{w} - \mathbf{w}'\|_2^2. \quad (7)$$

We intentionally use the same symbol η for the parameter in (7) as we used for the step size in (6). The larger the η we choose in (7), the more progress the update will make toward reducing the funktilo value $f(\hat{\mathbf{w}})$. Note that, much like the gradientti step (6), the update (7) also defines an operator that is parameterized by the funktilo $f(\cdot)$ and the learning rate η . For a convex funktilo $f(\cdot)$, this operator is known as the proximal operator of $f(\cdot)$ [22].

See also: differentiable, gradientti, step size, proximal operator.

graph of a function Given a function $f : \mathcal{X} \rightarrow \mathcal{Y}$, the graph of f is the subset of $\mathcal{X} \times \mathcal{Y}$ defined as

$$\{(\mathbf{x}, f(\mathbf{x})) : \mathbf{x} \in \mathcal{X}\}.$$

The graph of a function provides a geometric representation that is widely used in analysis, topology, and optimization.

halfspace See hyperplane.

Hermitian (matrix) A square matrix $\mathbf{A} \in \mathbb{C}^{d \times d}$ is Hermitian if it coincides with its conjugate transpose, i.e., $\mathbf{A} = \mathbf{A}^H$. Trivially, a Hermitian matrix is also a normal matrix.

Hessian matriisi Consider a function $f : \mathbb{R}^d \rightarrow \mathbb{R}$ for which the second-order partial derivatives exist at \mathbf{x}' . Then, the Hessian $\nabla^2 f(\mathbf{x}')$ of f at \mathbf{x} is defined as the matriisi of second-order partial derivatives of f at \mathbf{x}' , i.e.,

$$\nabla^2 f(\mathbf{x}') = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_d} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_d} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_d \partial x_1} & \frac{\partial^2 f}{\partial x_d \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_d^2} \end{pmatrix}.$$

If the second-order partial derivatives are jatkuva in a naapurusto around \mathbf{x}' , then the Hessian is a symmetric matriisi, i.e., $\partial^2 f / \partial x_j \partial x_{j'} = \partial^2 f / \partial x_{j'} \partial x_j$ for all j, j' [2]. If additionally f is convex, then the Hessian is a psd matriisi [23].

The Hessian $\nabla^2 f(\mathbf{x}')$ can be used to compute a quadratic function

$$q(\mathbf{x}) = \frac{1}{2}(\mathbf{x} - \mathbf{x}')^T \underbrace{\nabla^2 f(\mathbf{x}')}_{\text{Hessian}} (\mathbf{x} - \mathbf{x}') + (\mathbf{x} - \mathbf{x}')^T \underbrace{\nabla f(\mathbf{x}')}_{\text{gradientti}} + f(\mathbf{x}')$$

that approximates f locally around \mathbf{x}' (see also Fig. 22).

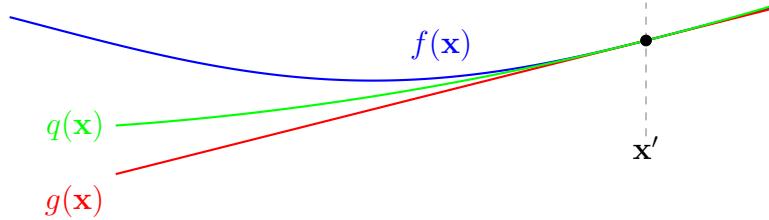


Fig. 22. A function $f(\mathbf{x})$ that is sufficiently smooth at a point \mathbf{x}' can be locally approximated by a quadratic function $q(\mathbf{x})$, which provides a more accurate approximation compared to a linear function $g(\mathbf{x})$.

See also: funktion, matriisi, quadratic function, differentiable.

Hilbert space A Hilbert space \mathcal{H} is a complete inner product space. Thus,

\mathcal{H} is a vector space equipped with an inner product $\langle \cdot, \cdot \rangle$. The inner product induces a norm $\|\cdot\|_2$ via $\|\mathbf{w}\|_2 = \sqrt{\langle \mathbf{w}, \mathbf{w} \rangle}$. Furthermore, \mathcal{H} is complete in the sense that every Cauchy sequence $(\mathbf{w}^{(r)})_{r \in \mathbb{N}}$ in \mathcal{H} converges to a limit $\lim_{r \rightarrow \infty} \mathbf{w}^{(r)}$ that is also contained in \mathcal{H} .

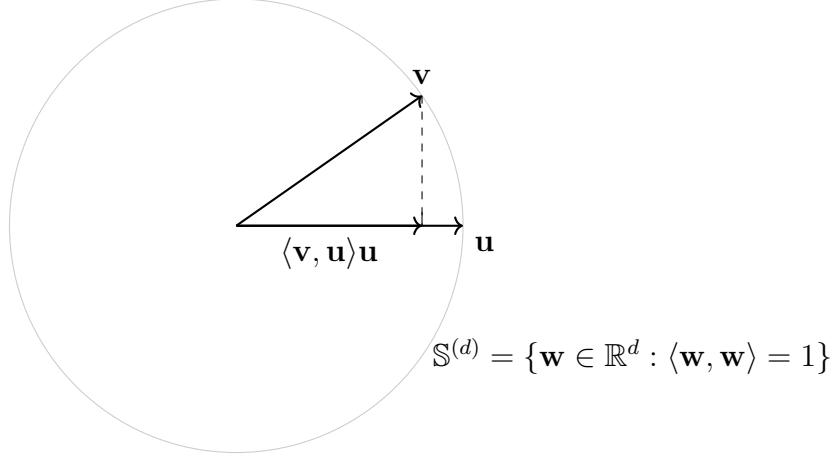


Fig. 23. For two unit-normi vektorit $\mathbf{u}, \mathbf{v} \in \mathbb{S}^{(d)} \subseteq \mathbb{R}^d$ the inner product $\langle \mathbf{u}, \mathbf{v} \rangle$ is the expansion coefficient for the projektio of \mathbf{v} onto the subspace $\{c\mathbf{u} : c \in \mathbb{R}\}$ spanned by \mathbf{u} . The absolute value $|\langle \mathbf{u}, \mathbf{v} \rangle|$ measures the normi of this projektio.

One important example of a Hilbert space is the Euclidean space \mathbb{R}^d with the inner product $\langle \mathbf{w}, \mathbf{w}' \rangle = \mathbf{w}^\top \mathbf{w}'$.

See also: vektoriavaruus.

Hoeffding's inequality Hoeffding's inequality [48] is a fundamental concentra-tion inequality that provides an upper bound on the todennäköisyys that a sum (or average) of independent, bounded RVs deviates from its keskiarvo by more than some threshold. Let x_1, \dots, x_n be independent real-valued RVs x_i taking values in $[a_i, b_i] \subset \mathbb{R}$, and $S_n := \sum_{i=1}^n x_i$ and $\mathbb{E}\{S_n\} = \sum_{i=1}^n \mathbb{E}\{x_i\}$. Then, Hoeffding's inequality [15, Th. 2.2.6] states

that

$$\mathbb{P}(|S_n - \mathbb{E}\{S_n\}| \geq t) \leq 2 \exp \left(-\frac{2t^2}{\sum_{i=1}^n (b_i - a_i)^2} \right) \quad \forall t > 0.$$

Hoeffding's inequality typically provides sharper bounds than Chebyshev's inequality, but it is more restrictive by assuming bounded RVs. In the context of koneoppiminen, this result is useful for deriving guarantees for ERM and in the context of monikäätinen rosvo.

See also: concentration inequality, todennäköisyys, RV, Chebyshev's inequality, expectation, probability space, Markov's inequality.

hyperplane A hyperplane is an $(d - 1)$ -dimensional affine subspace of a d -dimensional vektoriavaruus. In the context of a Euclidean space \mathbb{R}^d , a hyperplane is a set of the form

$$\{\mathbf{x} \in \mathbb{R}^d : \mathbf{w}^\top \mathbf{x} = b\}$$

where $\mathbf{w} \in \mathbb{R}^d \setminus \{0\}$ is a normal vektori and $b \in \mathbb{R}$ is an offset. Such a hyperplane partitions \mathbb{R}^d into two halfspaces

$$\{\mathbf{x} \in \mathbb{R}^d : \mathbf{w}^\top \mathbf{x} \leq b\} \quad \text{and} \quad \{\mathbf{x} \in \mathbb{R}^d : \mathbf{w}^\top \mathbf{x} \geq b\}.$$

Hyperplanes arise as the decision boundaries of linear classifiers.

See also: subspace, vektoriavaruus, Euclidean space, decision boundary.

independent and identically distributed (i.i.d.) A collection of RVs $\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(m)}$ is referred to as i.i.d. if each $\mathbf{z}^{(r)}$ follows the same probability distribution, and the RVs are mutually independent. That is, for

any collection of events $\mathcal{A}_1, \dots, \mathcal{A}_m$, we have

$$\mathbb{P}(\mathbf{z}^{(1)} \in \mathcal{A}_1, \dots, \mathbf{z}^{(m)} \in \mathcal{A}_m) = \prod_{r=1}^m \mathbb{P}(\mathbf{z}^{(r)} \in \mathcal{A}_r).$$

See also: RV, probability distribution, event, data point, i.i.d. assumption.

injective A funktion $f : \mathcal{U} \rightarrow \mathcal{V}$ is injective if it maps distinct elements of its domain to distinct elements of its co-domain, i.e., if $f(u_1) = f(u_2)$ implies $u_1 = u_2$ for all $u_1, u_2 \in \mathcal{U}$ [24]. Equivalently, no two different funktion inputs are mapped to the same funktion output.

See also: funktion.

inner product Consider a vektoriavaruus \mathcal{X} over the field \mathbb{F} , where \mathbb{F} is either the field of real numbers \mathbb{R} or the field of complex numbers \mathbb{C} . An inner product in \mathcal{X} is a funktion

$$\langle \cdot, \cdot \rangle : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{F},$$

that satisfies the following properties for all $\mathbf{x}, \mathbf{x}', \mathbf{x}'' \in \mathcal{X}$ and all scalars $\alpha \in \mathbb{F}$ [30]:

- (i) Conjugate symmetry: $\langle \mathbf{x}, \mathbf{x}' \rangle = \overline{\langle \mathbf{x}', \mathbf{x} \rangle}$,
- (ii) Linearity in the first argument: $\langle \alpha \mathbf{x} + \mathbf{x}'', \mathbf{x}' \rangle = \alpha \langle \mathbf{x}, \mathbf{x}' \rangle + \langle \mathbf{x}'', \mathbf{x}' \rangle$,
- (iii) Positive-definiteness: $\langle \mathbf{x}, \mathbf{x} \rangle \geq 0$, with equality if and only if $\mathbf{x} = \mathbf{0}$.

The pair $(\mathcal{X}, \langle \cdot, \cdot \rangle)$ is called an inner product space. Each inner product induces a norm via $\|\mathbf{x}\| := \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle}$ for all $\mathbf{x} \in \mathcal{X}$, which in turn induces

a metric via $d(\mathbf{x}, \mathbf{x}') := \|\mathbf{x} - \mathbf{x}'\|$ for all $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$.

See also: vektori, normi, metric space.

integrable A measurable funktion $f : \Omega \rightarrow \mathbb{R}$ defined on a measure space (Ω, Σ, μ) is called integrable if the Lebesgue integral of its absolute value is finite, i.e.,

$$\int_{\Omega} |f(x)| d\mu < \infty.$$

In this case, the Lebesgue integral $\int_{\Omega} f(x) d\mu$ is well-defined and finite. An RV x defined on the sample space of a probability space $(\Omega, \Sigma, \mathbb{P})$ is integrable if

$$\mathbb{E}\{|x|\} = \int_{\Omega} |x(\omega)| d\mathbb{P} < \infty$$

which ensures that the expectation $\mathbb{E}\{|x|\}$ exists and is finite.

See also: measure space, measure.

jatkuva A funktion $f : \mathbb{R}^d \rightarrow \mathbb{R}$ is continuous at a point $\mathbf{x}' \in \mathbb{R}^d$ if, for every $\epsilon > 0$, there is a $\delta > 0$ such that, for all $\mathbf{x} \in \mathbb{R}^d$ with $\|\mathbf{x} - \mathbf{x}'\|_2 < \delta$, it holds that $|f(\mathbf{x}) - f(\mathbf{x}')| < \epsilon$ [2]. In other words, we can make $f(\mathbf{x})$ arbitrarily close to $f(\mathbf{x}')$ by choosing \mathbf{x} sufficiently close to \mathbf{x}' .

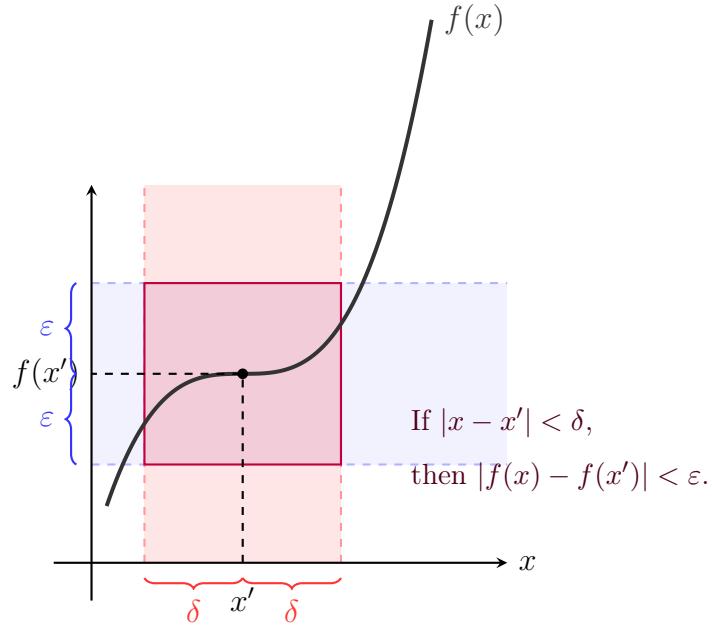


Fig. 24. The function $f(x) = 0.3(x - 2)^3 + 2.5$ is continuous at every x' .

If f is continuous at every point $\mathbf{x}' \in \mathbb{R}^d$, then f is said to be continuous on \mathbb{R}^d . The notion of a continuous function can be naturally extended to functions between general metric spaces [2].

See also: Euclidean space, metric.

Johnson–Lindenstrauss lemma (JL lemma) The JL lemma describes conditions for the existence of a feature map $\Phi : \mathbb{R}^d \rightarrow \mathbb{R}^{d'}$ with $d' \ll d$ such that the pairwise Euclidean distance between feature vectors of a finite dataset is approximately preserved [15], [49], [50]. Consider a dataset $\mathcal{D} = \{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}\}$ with data points characterized by feature vectors in \mathbb{R}^d . Then, for any d' that satisfies

$$d' \geq \frac{4 \ln(m)}{\varepsilon^2/2 - \varepsilon^3/3} \text{ for some } 0 < \varepsilon < 1$$

there is a feature map Φ such that [51]

$$(1-\varepsilon) \|\mathbf{x}^{(r)} - \mathbf{x}^{(r')}\|_2 \leq \|\Phi(\mathbf{x}^{(r)}) - \Phi(\mathbf{x}^{(r')})\|_2 \leq (1+\varepsilon) \|\mathbf{x}^{(r)} - \mathbf{x}^{(r')}\|_2 \quad (8)$$

for all $\mathbf{x}^{(r)}, \mathbf{x}^{(r')} \in \mathcal{D}$.

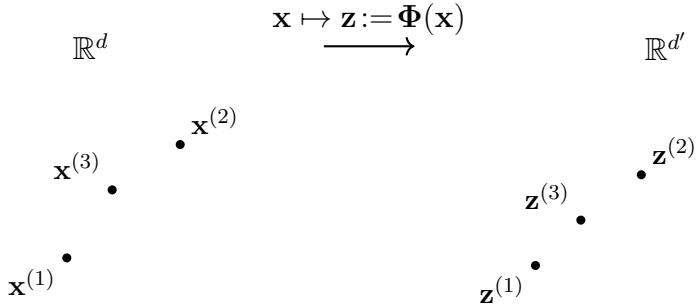


Fig. 25. The JL lemma offers precise conditions that guarantee the existence of a feature map $\Phi : \mathbb{R}^d \rightarrow \mathbb{R}^{d'}$ such that pairwise Euclidean distances between (the feature vectors of) data points are approximately preserved. Roughly speaking, Φ maps neighboring points in the original feature space to neighboring points in the new feature space.

The feature map Φ can be obtained from a random matriisi $\mathbf{A} \in \mathbb{R}^{d' \times d}$ whose entries are i.i.d. normaalijakautuneet satunnaismuuttujat $A_{i,j} \sim \mathcal{N}(0, 1/d')$. It can be shown that the feature map $\mathbf{x} \mapsto \underbrace{\mathbf{Ax}}_{\Phi(\mathbf{x})}$ satisfies (8) with todennäköisyys at least $1 - 1/m$ [51].

See also: matriisi, normi, vektoriavaruus, Euclidean space, ulottuvuuksien vähentäminen, principal component analysis (PCA).

kaksoisnormi Every normi $\|\cdot\|$ defined on a Euclidean space \mathbb{R}^d has an associated dual normi, which is denoted by $\|\cdot\|_*$ and defined as $\|\mathbf{y}\|_* :=$

$\sup_{\|\mathbf{x}\| \leq 1} \mathbf{y}^T \mathbf{x}$. The dual normi measures the largest possible inner product between \mathbf{y} and any vektori in the unit ball of the original normi. For further details, see [23, Sec. A.1.6].

See also: normi, Euclidean space, vektori.

keskeinen raja-arvolause Consider a sequence of i.i.d. RVs $x^{(r)}$, for $r = 1, 2, \dots$, each with keskiarvo zero and finite varianssi $\sigma^2 > 0$. The CLT states that the normalized sum

$$s^{(m)} := \frac{1}{\sqrt{m}} \sum_{r=1}^m x^{(r)}$$

converges in distribution to a normaalijakautunut satunnaismuuttuja with keskiarvo zero and varianssi σ^2 as $m \rightarrow \infty$ [12, Proposition 2.17].

One elegant way to derive the CLT is via the ominaisfunktio of the normalized sum $s^{(m)}$. Let $\phi(t) = \mathbb{E}\{\exp(jtx)\}$ (with the imaginary unit $j = \sqrt{-1}$) be the common ominaisfunktio of each sum and $x^{(r)}$, and let $\phi^{(m)}(t)$ denote the ominaisfunktio of $s^{(m)}$. Define an operator \mathcal{T} acting on ominaisfunktiot such that

$$\phi^{(m)}(t) = \mathcal{T}(\phi^{(m-1)})(t) := \phi\left(\frac{t}{\sqrt{m}}\right) \cdot \phi^{(m-1)}\left(\frac{\sqrt{m-1}}{\sqrt{m}}t\right).$$

This fixed-point iteration captures the effect of recursively adding an i.i.d. RV $\mathbf{x}^{(m)}$ and rescaling. Iteratively applying \mathcal{T} leads to convergence of $\phi^{(m)}(t)$ toward the fixed point

$$\phi^*(t) = \exp(-t^2\sigma^2/2)$$

which is the ominaisfunktio of a normaalijakautunut satunnaismuuttuja with keskiarvo zero and varianssi σ^2 . Generalizations of the CLT allow for dependent or nonidentically distributed RVs [12, Sec. 2.8].

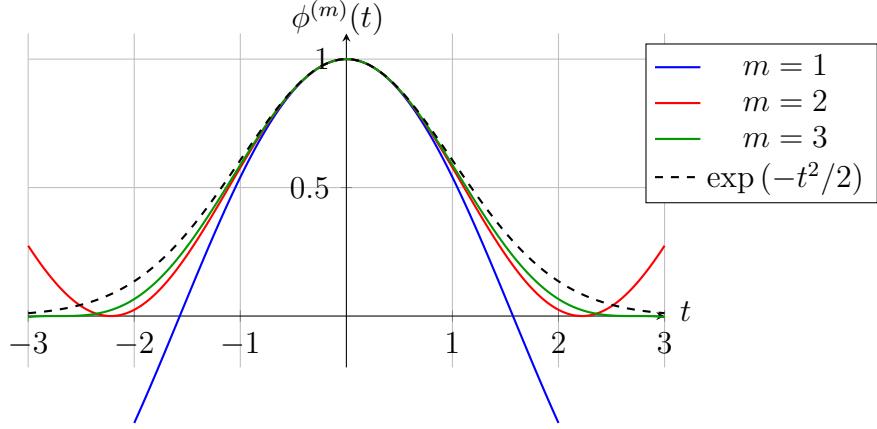


Fig. 26. Ominaisfunktioit of normalized sums of i.i.d. RVs $x^{(r)} \in \{-1, 1\}$ for $r = 1, \dots, m$ compared to the Gaussian limit.

See also: RV, normaalijakautunut satunnaismuuttuja.

keskiarvo The mean of a RV \mathbf{x} , which takes on 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 (e.g., see [2] or [6]), i.e.,

$$\mathbb{E}\{\mathbf{x}\} = \int_{\mathbb{R}^d} \mathbf{x} dP(\mathbf{x}).$$

We also use the term to refer to the average of a finite dataset $\mathcal{D} = \{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)} \in \mathbb{R}^d\}$. However, these two definitions are essentially the same. Indeed, we can use a dataset to construct a discrete RV $\tilde{\mathbf{x}}^{(\mathcal{D})} = \mathbf{x}^{(I)}$ on the sample space $\{1, \dots, m\}$. Here, the index I is chosen uniformly at random, $\mathbb{P}(I = r) = 1/m$ for all $r = 1, \dots, m$. The mean of $\tilde{\mathbf{x}}^{(\mathcal{D})}$ is precisely the average $(1/m) \sum_{r=1}^m \mathbf{x}^{(r)}$. For a RV with finite second-order moment, i.e., $\mathbb{E}\{\|\mathbf{x}\|_2^2\}$ is well-defined and finite, the

mean is characterized as the solution of the following riski minimization problem [7]:

$$\mathbb{E}\{\mathbf{x}\} = \arg \min_{\mathbf{c} \in \mathbb{R}^d} \mathbb{E}\{\|\mathbf{x} - \mathbf{c}\|_2^2\}.$$

For the RV $\tilde{\mathbf{x}}^{(\mathcal{D})}$, associated with a dataset, this optimointitehtävä reduces to ERM with neliövirhehäviö on \mathcal{D} .

See also: RV, expectation, probability distribution, ERM.

konveksi optimointi Convex optimization studies the formulation, properties, and efficient solution methods for convex optimointitehtävät [23]. A convex optimointitehtävä (defined on the Euclidean space \mathbb{R}^d) consists of a convex objective function $f : \mathbb{R}^d \rightarrow \mathbb{R}$ and a convex constraint set \mathcal{C} for the optimization variable \mathbf{w} . It can be written compactly as [23]

$$\min_{\mathbf{w} \in \mathcal{C}} f(\mathbf{w}).$$

Alternatively, a convex optimointitehtävä can be expressed in terms of convex constraint funktiot g_1, \dots, g_k as

$$\begin{aligned} & \min_{\mathbf{w} \in \mathbb{R}^d} f(\mathbf{w}) \\ & \text{s.t. } g_r(\mathbf{w}) \leq 0, \quad r = 1, \dots, k. \end{aligned} \tag{9}$$

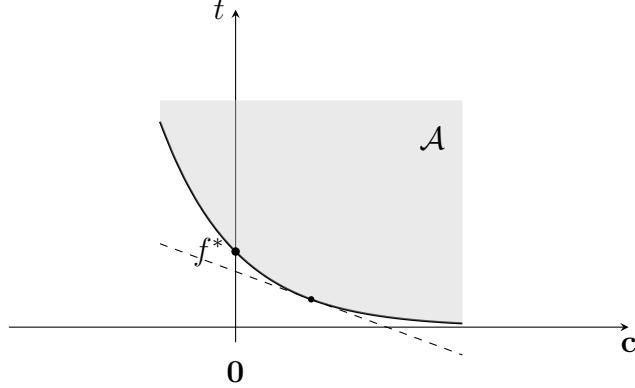


Fig. 27. A convex optimointitehtävä (9) can be represented by a set \mathcal{A} that consists of objective values t and constraint values $\mathbf{c} = (c_1, \dots, c_d)^T$ that are achievable, i.e., $f(\mathbf{w}) \leq t, g_1(\mathbf{w}) \leq c_1, \dots, g_k(\mathbf{w}) \leq c_k$ by some $\mathbf{w} \in \mathbb{R}^d$. The optimal value f^* of the optimointitehtävä is the smallest t for which $(\mathbf{0}, t) \in \mathcal{A}$.

The formulation (9) lends, in turn, to the epigrafi form of [23, Sec. 5.3]

$$\inf \{t \in \mathbb{R} : (\mathbf{0}, t) \in \mathcal{A}\}$$

with the set

$$\begin{aligned} \mathcal{A} := \{(\mathbf{c}, t) \in \mathbb{R}^d \times \mathbb{R} : f(\mathbf{w}) \leq t, \\ g_r(\mathbf{w}) \leq c_r, r = 1, \dots, k, \text{ for some } \mathbf{w} \in \mathbb{R}^d\}. \end{aligned}$$

It can be shown that, since f, g_1, \dots, g_k are convex funktiot, \mathcal{A} is a convex set [23, Ch. 2]. The set \mathcal{A} fully characterizes the optimointitehtävä (9) and can be interpreted as the epigrafi of the objective function f over the feasible region defined by the constraint funktiot g_1, \dots, g_k . See also: convex, optimointitehtävä, optimointimenetelmä.

kuvaus We use the term map as a synonym for funktio.

See also: funktio.

kytketty verkko An undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is connected if, for every non-empty subset $\mathcal{V}' \subset \mathcal{V}$, we can find at least one edge connecting a node in \mathcal{V}' with some node in $\mathcal{V} \setminus \mathcal{V}'$. We illustrate two examples of undirected graphs in Fig. 28.

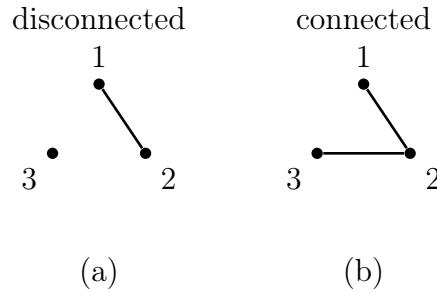


Fig. 28. (a) A verkko that is disconnected. (b) A verkko that is connected.

See also: undirected graph, algebraic connectivity.

käänteismatriisi An inverse matriisi \mathbf{A}^{-1} is defined for a square matriisi $\mathbf{A} \in \mathbb{R}^{n \times n}$ that is of full-rank, meaning its columns are linearly independent. In this case, \mathbf{A} is said to be invertible, and its inverse satisfies

$$\mathbf{A}\mathbf{A}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{I}.$$

A square matriisi is invertible if and only if its determinantti is nonzero. Inverse matriisit are fundamental in solving systems of linear equations and in the closed-form solution of linear regression [28], [52]. The concept of an inverse matriisi can be extended to matriisit that are

not square or do not have full-rank. One may define a “left inverse” \mathbf{B} satisfying $\mathbf{BA} = \mathbf{I}$ or a “right inverse” \mathbf{C} satisfying $\mathbf{AC} = \mathbf{I}$. For general rectangular or singular matriisit, the Moore–Penrose pseudokäänteisluku \mathbf{A}^+ provides a unified concept of a generalized inverse matriisi [3].

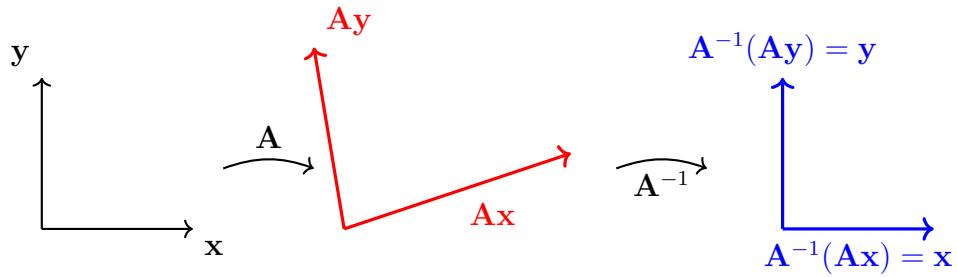


Fig. 29. A matriisi \mathbf{A} represents a linear transformation of \mathbb{R}^2 . The inverse matriisi \mathbf{A}^{-1} represents the inverse transformation.

See also: matriisi, determinantti, linear regression, pseudokäänteisluku.

Lagrange dual problem The Lagrangian of an optimointitehtävä of form

$$\begin{aligned} & \min_{\mathbf{w} \in \mathbb{R}^d} f_0(\mathbf{w}) \\ \text{s.t. } & f_i(\mathbf{w}) \leq 0, \quad i = 1, \dots, k \end{aligned} \tag{10}$$

$$g_j(\mathbf{w}) = 0, \quad j = 1, \dots, l \tag{11}$$

is

$$\max_{\lambda \in \mathbb{R}^k, \nu \in \mathbb{R}^l} \inf_{\mathbf{w} \in \mathbb{R}^d} L(\mathbf{w}, \lambda, \nu) \text{ s.t. } \lambda \leq 0, \tag{12}$$

where L is the Lagrangian of the optimointitehtävä.

See also: konveksi optimointi, optimointitehtävä, Lagrangian.

Lagrangian The Lagrangian of an optimointitehtävä of form

$$\begin{aligned} & \min_{\mathbf{w} \in \mathbb{R}^d} f_0(\mathbf{w}) \\ \text{s.t. } & f_i(\mathbf{w}) \leq 0, \quad i = 1, \dots, k \end{aligned} \tag{13}$$

$$g_j(\mathbf{w}) = 0, \quad j = 1, \dots, l \tag{14}$$

is

$$L(\mathbf{w}, \lambda, \nu) := f_0(\mathbf{w}) + \sum_{i=1}^k \lambda_i f_i(\mathbf{w}) + \sum_{j=1}^l \nu_j g_j(\mathbf{w}). \tag{15}$$

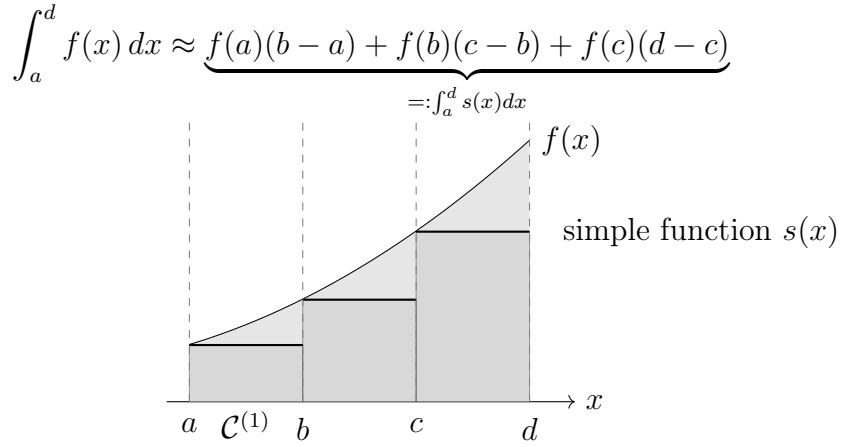
The Lagrangian is most often used to formulate the Lagrange dual problem.

See also: konveksi optimointi, optimointitehtävä, dual.

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 keskiarvo of their common probability distribution. Different instances of the law of large numbers are obtained by using different notions of convergence [26].

See also: convergence, i.i.d., RV, keskiarvo, probability distribution.

Lebesgue integral The Lebesgue integral assigns each integrable funktio $f : \mathbb{R}^d \rightarrow \mathbb{R}$ a number $\int_{\mathbf{x}} f(\mathbf{x}) d\mathbf{x}$ that is referred to as the integral of f . The integral of f can be interpreted as the volume that is enclosed by the funktio f in the space \mathbb{R}^{d+1} . We can compute it be increasingly accurate approximations by simple functions [1, Ch. 1].



It is useful to think of the Lebesgue integral as a funktion that maps an integrable funktion f to the value of its integral,

$$f \mapsto \int_{\mathbf{x}} f(\mathbf{x}) d\mathbf{x}.$$

The precise definition of this funktion, whose domain consists of the integrable funktioner, is a cornerstone of measure theory [1, Ch. 1].

See also: funktion.

lineaarikuvaus A linear kuvaus $f : \mathbb{R}^d \rightarrow \mathbb{R}^m$ is a funktion that satisfies additivity, i.e., $f(\mathbf{x} + \mathbf{y}) = f(\mathbf{x}) + f(\mathbf{y})$, and homogeneity, i.e., $f(c\mathbf{x}) = cf(\mathbf{x})$, for all vektorit $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ and scalars $c \in \mathbb{R}$. In particular, $f(\mathbf{0}) = \mathbf{0}$. Any linear kuvaus can be represented as a matriisi multiplication $f(\mathbf{x}) = \mathbf{Ax}$, for some matriisi $\mathbf{A} \in \mathbb{R}^{m \times n}$. The collection of real-valued linear kuvaukset (where $m = 1$), for a given dimension d , constitute a lineaarinen malli. The notion of a linear kuvaus can be generalized from the domain \mathbb{R}^d and co-domain \mathbb{R}^m to arbitrary vektoriavaruudet.

See also: kuvaus, funktion, vektori, matriisi, lineaarinen malli.

linearly independent A subset $\{\mathbf{a}^{(1)}, \dots, \mathbf{a}^{(d)}\} \in \mathcal{V}$ of a vektoriavaruus is linearly independent if there is no nontrivial linear combination of these vektorit that equals the zero vektori [32]. In other words,

$$\sum_{j=1}^d \alpha_j \mathbf{a}^{(j)} = \mathbf{0} \quad \text{implies} \quad \alpha_1 = \alpha_2 = \dots = \alpha_k = 0.$$

See also: vektoriavaruus, vektori, dimension, basis.

majorize-minimize (MM) Consider an optimointitehtävä $\min_{\mathbf{w} \in \mathcal{W}} f(\mathbf{w})$ with some complicated (potentially non-convex and non-smooth) objective function. One important example of such an optimointitehtävä is ERM, which is used to learn the model parameters of a nonlinear model. An MM method is an iterative optimointimenetelmä that constructs a sequence $\mathbf{w}^{(1)}, \mathbf{w}^{(2)}, \dots \in \mathcal{W}$ of model parameters as follows [53], [38], [54] (see also Fig. 30):

- During the t th iteration, the objective function $f(\cdot)$ is approximated by another funktion $g(\cdot; \mathbf{w}^{(t)})$. This approximation must be an upper bound for (i.e., must majorize) the original objective function, i.e., $g(\mathbf{w}; \mathbf{w}^{(t)}) \geq f(\mathbf{w})$ for all $\mathbf{w} \in \mathcal{W}$, and it must be tight for $\mathbf{w}^{(t)}$, i.e., $g(\mathbf{w}^{(t)}; \mathbf{w}^{(t)}) = f(\mathbf{w}^{(t)})$.
- The new model parameters $\mathbf{w}^{(t+1)}$ are then obtained by minimizing the approximation, i.e., $\mathbf{w}^{(t+1)} \in \arg \min_{\mathbf{w} \in \mathcal{W}} g(\mathbf{w}; \mathbf{w}^{(t)})$.

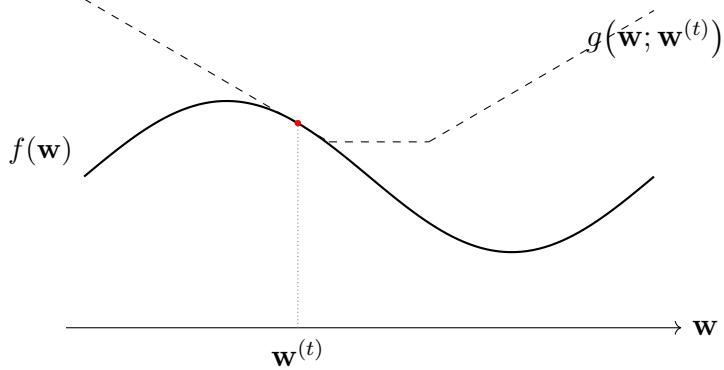


Fig. 30. The construction of model parameters based on the iterative MM method.

Similar to gradient-based methods, the MM principle is also based on approximating an objective function locally, around the current model parameters, and then optimizing this approximation to obtain new model parameters. However, the construction of local approximations is very different. While gradient-based methods use linear functions for these approximations, MM methods can use nonlinear functions as long as they are upper bounds for the original objective function.

See also: gradient-based method, expectation–maximization (EM).

maksimi 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].

See also: supremum.

Markov chain A Markov chain is a satunnaisprosessi $\{X_t\}_{t \in \mathbb{N}}$, defined on

a common probability space and using the index set \mathbb{N} . The RV X_t might represent (the generation of) a state of a physical system at the time instant t . The defining property of a Markov chain is the Markov property [17, 26, 55]: For all $t \in \mathbb{N}$,

$$\mathbb{P}(X_{t+1}|X_t, \dots, X_1) = \mathbb{P}(X_{t+1}|X_t).$$

In other words, the conditional probability distribution of the next state X_{t+1} depends on the past X_t, X_{t-1}, \dots, X_1 only through the current state X_t . The concept of a Markov chain can be generalized from discrete time (with index set \mathbb{N}) to continuous time (with index set \mathbb{R}) [55].

See also: satunnaisprosessi, conditional probability distribution.

Markov property See Markov chain.

Markov's inequality Consider a real-valued nonnegative RV x for which the expectation $\mathbb{E}\{x\}$ exists. Markov's inequality provides an upper bound on the todennäköisyys $\mathbb{P}(x \geq a)$ that x exceeds a given positive threshold $a > 0$. In particular,

$$\mathbb{P}(x \geq a) \leq \frac{\mathbb{E}\{x\}}{a} \quad \text{holds for any } a > 0.$$

This inequality can be verified by noting that $\mathbb{P}(x \geq a)$ is the expectation $\mathbb{E}\{g(x)\}$ with the funktion

$$g : \mathbb{R} \rightarrow \mathbb{R} : x' \mapsto \mathbb{I}_{\{x \geq a\}}(x').$$

As illustrated in Fig. 31, for any positive $a > 0$,

$$g(x') \leq x'/a \text{ for all } x' \in \mathbb{R}.$$

This implies Markov's inequality via the monotonicity property of the Lebesgue integral [13, p. 50].

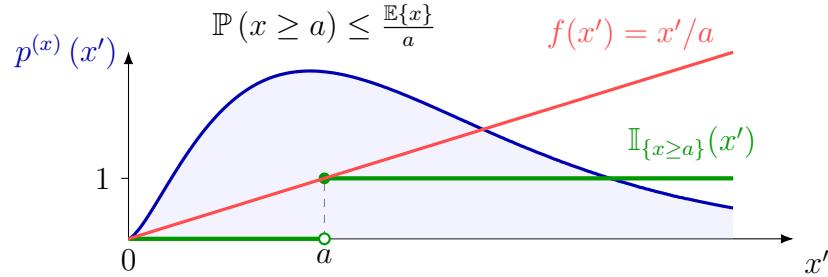


Fig. 31. The expectation $\mathbb{E}\{x\}$ and the todennäköisyys $\mathbb{P}(x \geq a)$ of a non-negative RV x with a pdf $p^{(x)}(\cdot)$ can be obtained via Lebesgue integrals of $f(x') = x'/a$ and $g(x') = \mathbb{I}_{\{x \geq a\}}(x')$, respectively.

See also: expectation, todennäköisyys, concentration inequality.

matriisi A matrix of size $m \times d$ is a 2-D array of numbers, which is denoted by

$$\mathbf{A} = \begin{pmatrix} A_{1,1} & A_{1,2} & \dots & A_{1,d} \\ A_{2,1} & A_{2,2} & \dots & A_{2,d} \\ \vdots & \vdots & \ddots & \vdots \\ A_{m,1} & A_{m,2} & \dots & A_{m,d} \end{pmatrix} \in \mathbb{R}^{m \times d}.$$

Here, $A_{r,j}$ denotes the matrix entry in the r th row and the j th column. Matrices are useful representations of various mathematical objects [32], including the following:

- Systems of linear equations: We can use a matrix to represent a

system of linear equations

$$\begin{pmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \quad \text{compactly as} \quad \mathbf{A}\mathbf{w} = \mathbf{y}.$$

One important example of systems of linear equations is the optimality condition for the model parameters within linear regression.

- Lineaarikuvauskset: Consider a d -dimensional vektoriavaruus \mathcal{U} and a m -dimensional vektoriavaruus \mathcal{V} . If we fix a basis $\mathbf{u}^{(1)}, \dots, \mathbf{u}^{(d)}$ for \mathcal{U} and a basis $\mathbf{v}^{(1)}, \dots, \mathbf{v}^{(m)}$ for \mathcal{V} , each matrix $\mathbf{A} \in \mathbb{R}^{m \times d}$ naturally defines a lineaarikuvaus $\alpha : \mathcal{U} \rightarrow \mathcal{V}$ (see Fig. 32) such that

$$\mathbf{u}^{(j)} \mapsto \sum_{r=1}^m A_{r,j} \mathbf{v}^{(r)}.$$

- Datasets: We can use a matrix to represent a dataset. Each row corresponds to a single data point, and each column corresponds to a specific feature or label of a data point.

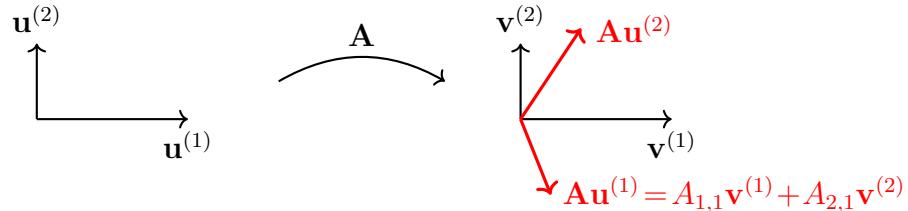


Fig. 32. A matrix \mathbf{A} defines a lineaarikuvaus between two vektoriavarudet.

See also: lineaarikuvaus, dataset, lineaarinen malli.

measurable Consider a satunnaiskoe, such as recording the air temperature at an FMI weather station. The corresponding sample space Ω consists of all possible outcomes ω (e.g., all possible temperature values in degree Celsius). In many koneoppiminen applications, we are not interested in the exact outcome ω , but only whether it belongs to a subset $\mathcal{A} \subseteq \Omega$ (e.g., determining whether the temperature is below zero degrees). We call such a subset \mathcal{A} measurable if it is possible to decide, for any outcome ω , whether $\omega \in \mathcal{A}$ or not (see Fig. 33).

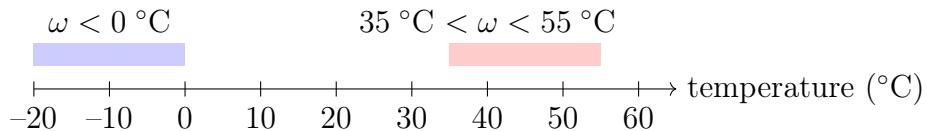


Fig. 33. A sample space constituted by all possible temperature values ω that can occur at an FMI station. Two measurable subsets of temperature values, denoted by $\mathcal{A}^{(1)}$ and $\mathcal{A}^{(2)}$, are highlighted. For any actual temperature value ω , it is possible to determine (via some equipment) whether $\omega \in \mathcal{A}^{(1)}$ and whether $\omega \in \mathcal{A}^{(2)}$.

In principle, measurable sets could be chosen freely (e.g., depending on the resolution of the measuring equipment). However, it is often useful to impose certain completeness requirements on the collection of measurable sets. For example, the sample space itself should be measurable, and the union of two measurable sets should also be measurable.

These completeness requirements can be formalized via the concept of a σ -algebra (or σ -field) [1], [6], [17]. A measurable space is a pair $(\mathcal{X}, \mathcal{F})$ that consists of an arbitrary set \mathcal{X} and a collection \mathcal{F} of measurable subsets of \mathcal{X} that form a σ -algebra.

See also: sample space, outcome, σ -algebra, todennäköisyys.

measure A measure μ on a set Ω equipped with a σ -algebra Σ is a function $\mu : \Sigma \rightarrow [0, \infty)$ that assigns a nonnegative value to each measurable set $\mathcal{A} \in \Sigma$ such that [2], [6], [56]: 1) $\mu(\emptyset) = 0$; and 2) for any countable collection $\{\mathcal{A}_i\}_{i=1}^{\infty}$ of pairwise disjoint sets in Σ ,

$$\mu\left(\bigcup_{i=1}^{\infty} \mathcal{A}_i\right) = \sum_{i=1}^{\infty} \mu(\mathcal{A}_i)$$

which is referred to as “countable additivity”.

See also: measurable, countable.

measure space A measure space is a triple (Ω, Σ, μ) consisting of a set Ω , a σ -algebra Σ of subsets of Ω , and a measure $\mu : \Sigma \rightarrow [0, \infty)$. The measure μ assigns a nonnegative number to each measurable set $\mathcal{A} \in \Sigma$, generalizing the notions of length, area, or volume in Euclidean space [2], [56]. Measure spaces provide the mathematical foundation for the Lebesgue integral or the definition of RVs as measurable mappings between measure spaces. A probability space is a special case of a measure space where the total measure of the sample space is normalized to one, i.e., $\mu(\Omega) = 1$. In this case, μ is called a probability distribution.

See also: measurable, probability space, probability distribution.

mediaani A median $\text{med}(x)$ of a real-valued RV x is any number $M \in \mathbb{R}$

such that $\mathbb{P}(x \leq M) \geq 1/2$ and $\mathbb{P}(x \geq M) \geq 1/2$ (see Fig. 34) [57].

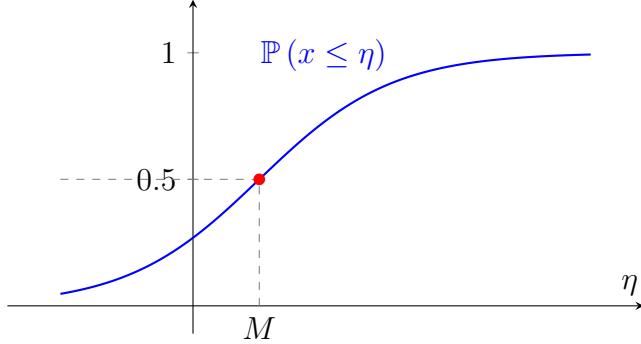


Fig. 34. The median of a real-valued RV is any number M that partitions \mathbb{R} into two rays with equal todennäköisyys.

We can define the median $\text{med}(\mathcal{D})$ of a dataset $\mathcal{D} = \{x^{(1)}, \dots, x^{(m)} \in \mathbb{R}\}$ via a specific RV $\tilde{x}^{(\mathcal{D})}$ that is naturally associated with \mathcal{D} . In particular, this RV is defined on the sample space $\{1, \dots, m\}$ via $\tilde{x}^{(\mathcal{D})} := x^{(I)}$. Here, the index I is chosen uniformly at random, i.e., $\mathbb{P}(I = r) = 1/m$ for all $r = 1, \dots, m$. If the RV x is integrable, any median of x solves the optimointitehtävä:

$$\min_{x' \in \mathbb{R}} \mathbb{E}|x - x'|.$$

For the above RV \tilde{x} (constructed from a dataset \mathcal{D}), this optimointitehtävä is ERM on \mathcal{D} using absolute error loss. Like the keskiarvo, the median of a dataset \mathcal{D} can also be used to estimate parameters of an underlying todennäköisyysmalli. Compared with the keskiarvo, the median is more robust to outliers. For example, a median of a dataset \mathcal{D} with more than one data point does not change even if we arbitrarily

increase the largest element of \mathcal{D} (see Fig. 35). In contrast, the keskiarvo will increase arbitrarily.

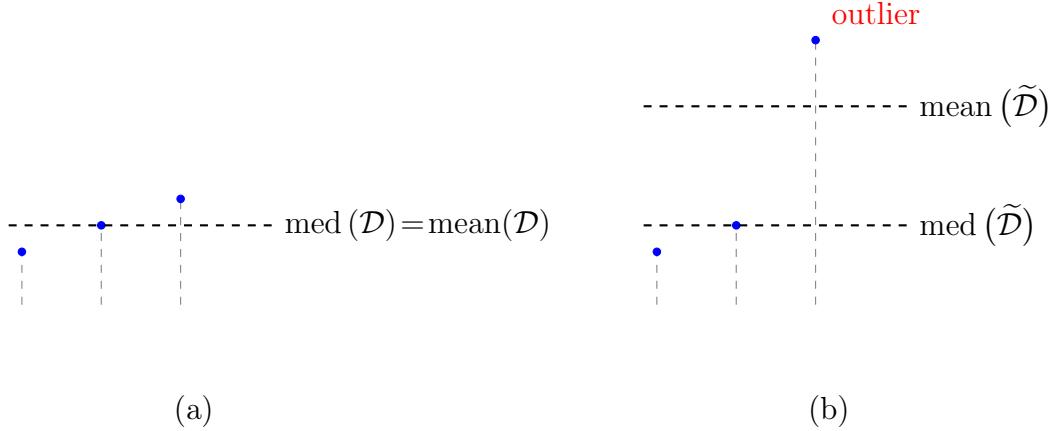


Fig. 35. The median is robust against outlier contamination. (a) Original dataset \mathcal{D} . (b) Noisy dataset $\tilde{\mathcal{D}}$ including an outlier.

See also: keskiarvo, outlier, vakaus, least absolute deviation regression.

metric space A metric space is a set \mathcal{X} equipped with a funkatio (referred to as a metric) $d(\cdot, \cdot) : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}_+$ that satisfies the following requirements for all $\mathbf{x}, \mathbf{x}', \mathbf{x}'' \in \mathcal{X}$:

1. Nonnegativity: $d(\mathbf{x}, \mathbf{x}') \geq 0$;
2. Identity: $d(\mathbf{x}, \mathbf{x}') = 0$ if and only if $\mathbf{x} = \mathbf{x}'$;
3. Symmetry: $d(\mathbf{x}, \mathbf{x}') = d(\mathbf{x}', \mathbf{x})$;
4. Triangle inequality: $d(\mathbf{x}, \mathbf{x}'') \leq d(\mathbf{x}, \mathbf{x}') + d(\mathbf{x}', \mathbf{x}'')$.

Formally, a metric space is a pair $(\mathcal{X}, d(\cdot, \cdot))$ that satisfies the above requirements.

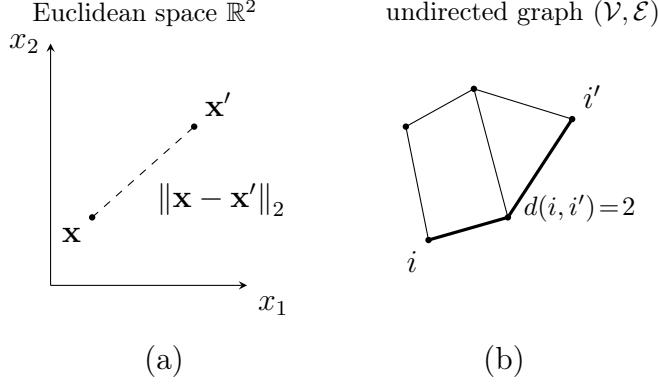


Fig. 36. Examples of metric spaces. (a) Euclidean space \mathbb{R}^2 with the Euclidean distance as a metric. (b) Undirected graph $(\mathcal{V}, \mathcal{E})$ with the shortest-path distance as a metric.

A prominent example of a metric space is the Euclidean space equipped with a metric given by the Euclidean distance $d(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|_2$. Another well-known example of a metric space is an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with the metric $d(i, i')$ defined by the length of the shortest path connecting nodes i and i' .

See also: Euclidean space, undirected graph, feature space.

minimi Given a set of real numbers, the minimum is the smallest of those numbers. Note that for some sets, such as the set of negative real numbers, the minimum does not exist.

mirror descent Mirror descent is an iterative optimointimenetelmä obtained by generalizing the gradienttiaskel. The gradienttiaskel for minimizing a differentiable objective function $f(\mathbf{w})$, can be written as

$$\mathbf{w}^{(k+1)} = \arg \min_{\mathbf{w}} \langle \nabla f(\mathbf{w}^{(k)}), \mathbf{w} \rangle + \frac{1}{\eta} \|\mathbf{w} - \mathbf{w}^{(k)}\|_2^2.$$

Thus, a gradienttiaskel minimizes a linearization of $f(\cdot)$ penalized by a scaled squared Euclidean norm $\frac{1}{\eta} \|\mathbf{w} - \mathbf{w}^{(k)}\|_2^2$. The scaling factor is the inverse of the learning rate η used in the gradienttiaskel. Mirror descent replaces the squared Euclidean norm by a Bregman divergence $D_\phi(\cdot, \cdot)$ induced by a strictly convex function $\phi(\cdot)$. This mirror map is typically defined on a convex set $\mathcal{W} \subseteq \mathbb{R}^d$ and is differentiable and strictly convex on the interior of \mathcal{W} [58]. The resulting update becomes

$$\mathbf{w}^{(k+1)} = \arg \min_{\mathbf{w} \in \mathcal{W}} \langle \mathbf{g}^{(k)}, \mathbf{w} \rangle + \frac{1}{\eta} D_\phi(\mathbf{w}, \mathbf{w}^{(k)}).$$

See also: gradienttiaskel, proximal operator, Bregman divergence.

moment generating function (MGF) Consider the MGF $M_x(t)$ of a real-valued RV x , which is defined as $M_x(t) = \mathbb{E}\{\exp(t \cdot x)\}$ for any $t \in \mathbb{R}$ for which this expectation exists [6, Sec. 21]. As its name indicates, the MGF allows us to compute the moments $\mathbb{E}\{x^k\}$ for $k \in \mathbb{N}$. In particular, the k th moment is obtained by evaluating the k th derivative of $M_x(t)$ for $t = 0$, i.e., $\mathbb{E}\{x^k\} = M_x^{(k)}(0)$. This fact can be verified by the following identities:

$$\begin{aligned} M_x(t) &= \mathbb{E}\{\exp(t \cdot x)\} \\ &\stackrel{(a)}{=} \mathbb{E}\left\{ \sum_{k=0}^{\infty} \frac{t^k}{k!} x^k \right\} \\ &\stackrel{(b)}{=} \sum_{k=0}^{\infty} \frac{t^k}{k!} \mathbb{E}\{x^k\}. \end{aligned}$$

Here, step (a) is due to the Taylor series expansion of $\exp(t \cdot x)$ and step (b) is valid when the MGF exists for all t in some interval $(-t_0, t_0)$ [6, p. 278].

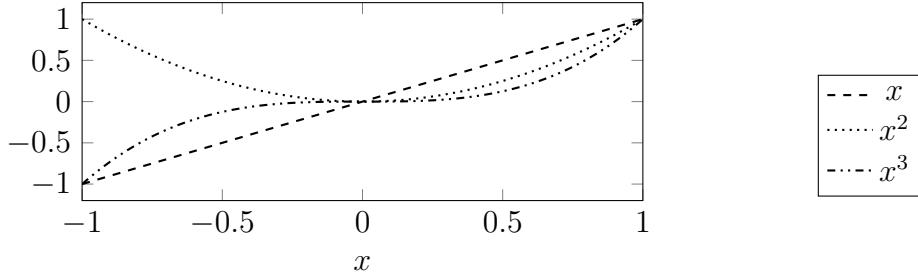


Fig. 37. The first few powers of an RV x . The MGF encodes the moments of x , which are the expectations of the powers x^k for $k = 1, 2, \dots$

The MGF is a useful tool for the study of sums of independent RVs. As a case in point, if x and y are independent RVs, then the MGF of their sum $z = x + y$ typically satisfies $M_z(t) = M_x(t) M_y(t)$, i.e., the MGF of the sum is typically the pointwise product of the individual MGFs [6, p. 280].

See also: RV, expectation.

moninormaalijakauma The multivariate normal distribution, which is denoted by $\mathcal{N}(\boldsymbol{\mu}, \mathbf{C})$, is a fundamental todennäköisyysmalli for numerical feature vectors of fixed dimension d . It defines a family of probability distributions over vektori-valued RVs $\mathbf{x} \in \mathbb{R}^d$ [7], [18], [43]. Each distribution in this family is fully specified by its keskiarvo vektori $\boldsymbol{\mu} \in \mathbb{R}^d$ and covariance matrix $\mathbf{C} \in \mathbb{R}^{d \times d}$. When the covariance matrix \mathbf{C} is invertible, the corresponding probability distribution is characterized by the following pdf:

$$p(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^d \det(\mathbf{C})}} \exp \left[-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^T \mathbf{C}^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right].$$

Note that this pdf is only defined when \mathbf{C} is invertible. More generally, any RV $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{C})$ admits the following representation:

$$\mathbf{x} = \mathbf{A}\mathbf{z} + \boldsymbol{\mu}$$

where $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ is a standard normal random vector and $\mathbf{A} \in \mathbb{R}^{d \times d}$ satisfies $\mathbf{A}\mathbf{A}^\top = \mathbf{C}$. This representation remains valid even when \mathbf{C} is singular, in which case \mathbf{A} is not full-rank [59, Ch. 23]. The family of multivariate normal distributions is exceptional among todennäköisyyssmallit for numerical quantities, at least for the following reasons. First, the family is closed under affine transformations, i.e.,

$$\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{C}) \text{ implies } \mathbf{Bx} + \mathbf{c} \sim \mathcal{N}(\mathbf{B}\boldsymbol{\mu} + \mathbf{c}, \mathbf{BCB}^T).$$

Second, the probability distribution $\mathcal{N}(\mathbf{0}, \mathbf{C})$ maximizes the differential entropy among all distributions with the same covariance matrix \mathbf{C} [31]. See also: todennäköisyyssmalli, probability distribution, standard normal random vector, differential entropy, normaalijakautunut satunnaismuuttuja.

Newtonin menetelmä Newton's method is an iterative optimointimenetelmä for finding local minimist or maksimit of a differentiable objective function $f(\mathbf{w})$. Like gradient-based methods, Newton's method also computes a new estimate $\hat{\mathbf{w}}_{t+1}$ by optimizing a local approximation of $f(\mathbf{w})$ around the current estimate $\hat{\mathbf{w}}_t$. In contrast to gradient-based methods, which use the gradientti to build a local linear approximation, Newton's method uses the Hessen matriisi matriisi to build a local quadratic approximation. In particular, starting from an initial estimate

$\widehat{\mathbf{w}}_0$, Newton's method iteratively updates the estimate according to

$$\widehat{\mathbf{w}}_{t+1} = \widehat{\mathbf{w}}_t - (\nabla^2 f(\widehat{\mathbf{w}}_t))^{-1} \nabla f(\widehat{\mathbf{w}}_t), \text{ for } t = 0, 1, \dots$$

Here, $\nabla f(\widehat{\mathbf{w}}_t)$ is the gradientti, and $\nabla^2 f(\mathbf{w}^{(t)})$ is the Hessen matriisi of the objective function f . Since using a quadratic function as local approximation is more accurate than using a linear funkatio (which is a special case of a quadratic function), Newton's method tends to converge faster than gradient-based methods (see Fig. 38). However, this faster convergence comes at the increased computational complexity of the iterations. Indeed, each iteration of Newton's method requires the inversion of the Hessen matriisi.

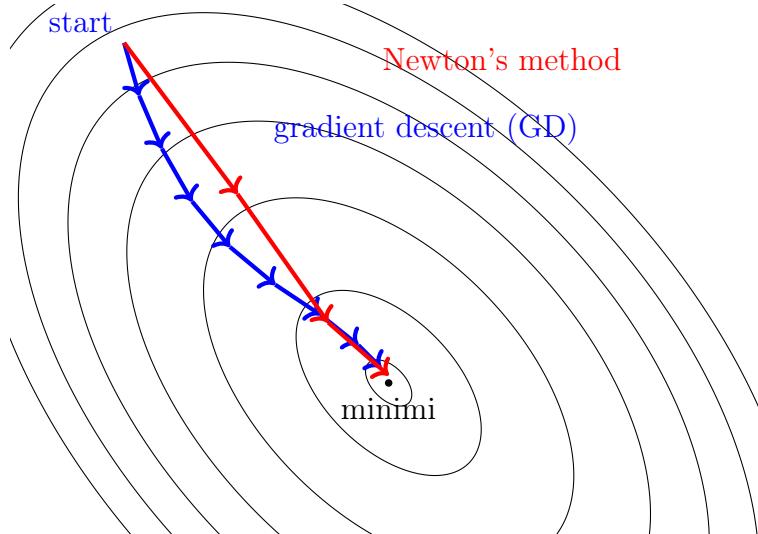


Fig. 38. Comparison of GD (blue) and Newton's method (red) paths toward the minimi of a loss function.

See also: optimointimenetelmä, gradientti, Hessen matriisi, GD.

nolla-avaruuus The nullspace of a matriisi $\mathbf{A} \in \mathbb{R}^{d' \times d}$, denoted by $\text{null}(\mathbf{A})$, is the set of all vektorit $\mathbf{n} \in \mathbb{R}^d$ such that

$$\mathbf{A}\mathbf{n} = \mathbf{0}.$$

Consider a piirreoptimointi method that uses the matriisi \mathbf{A} to transform a feature vector $\mathbf{x} \in \mathbb{R}^d$ of a data point into a new feature vector $\mathbf{z} = \mathbf{Ax} \in \mathbb{R}^{d'}$. The nullspace $\text{null}(\mathbf{A})$ characterizes all directions in the original feature space \mathbb{R}^d along which the transformation \mathbf{Ax} remains unchanged. In other words, adding any vektori from the nullspace to a feature vector \mathbf{x} does not affect the transformed representation \mathbf{z} . This property can be exploited to enforce invariances in the ennusteet (computed from \mathbf{Ax}). Fig. 39 illustrates one such invariance. It shows rotated versions of two handwritten digits, which approximately lie along 1-D curves in the original feature space. These curves are aligned with a direction vektori $\mathbf{n} \in \mathbb{R}^d$. To ensure that the trained model is invariant to such rotations, we can choose the transformation matriisi \mathbf{A} such that $\mathbf{n} \in \text{null}(\mathbf{A})$. This ensures that \mathbf{Ax} , and hence the resulting ennuste, is approximately insensitive to rotations of the input image.

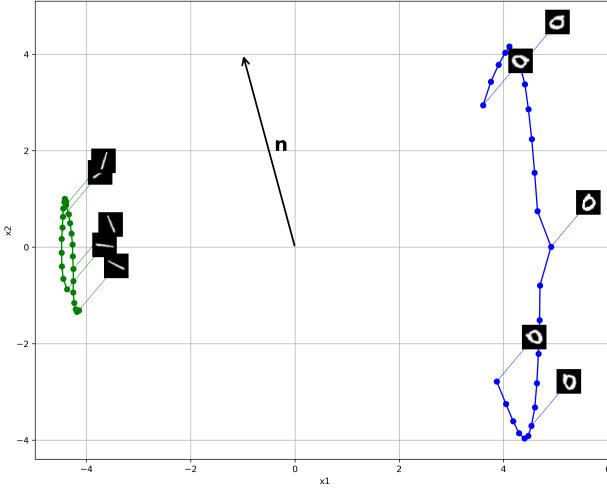


Fig. 39. Rotated handwritings of two different digits. The rotations are approximately aligned along straight lines parallel to the vektori \mathbf{n} . For a binary luokitin distinguishing between these digits, a natural choice is a linear feature map $\mathbf{x} \mapsto \mathbf{Ax}$ with a matriisi \mathbf{A} whose nullspace contains \mathbf{n} , i.e., $\mathbf{n} \in \text{null}(\mathbf{A})$.

See also: matriisi, feature map, piirreoptimointi.

Python demo: [click me](#)

non-expansive operator An operator $\mathcal{F} : \mathcal{H} \rightarrow \mathcal{H}$ defined on a Hilbert space \mathcal{H} is called non-expansive if it does not increase distances. In other words,

$$\|\mathcal{F}\mathbf{w} - \mathcal{F}\mathbf{w}'\|_2 \leq \|\mathbf{w} - \mathbf{w}'\|_2 \text{ for any } \mathbf{w}, \mathbf{w}' \in \mathcal{H}.$$

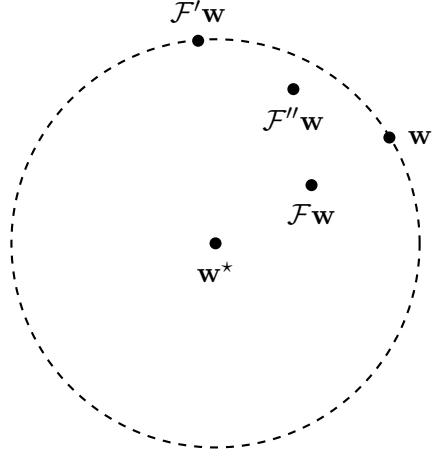


Fig. 40. The result of applying a contractive operator \mathcal{F} , a non-expansive operator \mathcal{F}' and a firmly non-expansive operator \mathcal{F}'' . These operators have the common fixed point \mathbf{w}^* .

Non-expansiveness is not sufficient to guarantee convergence of a fixed-point iteration that uses \mathcal{F} (see Fig. 40).

See also: fixed-point iteration, contractive operator.

non-smooth We refer to a function as non-smooth if it is not smooth [45].

See also: function, smooth.

normaalijakautunut satunnaismuuttuja A standard Gaussian RV is a real-valued RV x with pdf [7], [26], [18]

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

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 by $\mathcal{N}(\mu, \sigma^2)$.

A Gaussian random vektori $\mathbf{x} \in \mathbb{R}^d$ with covariance matrix \mathbf{C} and keskiarvo $\boldsymbol{\mu}$ can be constructed as [18], [26], [43]

$$\mathbf{x} := \mathbf{A}\mathbf{z} + \boldsymbol{\mu}$$

where $\mathbf{z} := (z_1, \dots, z_d)^T$ is a vektori of i.i.d. standard Gaussian RVs, and $\mathbf{A} \in \mathbb{R}^{d \times d}$ is any matriisi satisfying $\mathbf{A}\mathbf{A}^T = \mathbf{C}$. The probability distribution of a Gaussian random vektori is referred to as the moninormaalijakauma, denoted by $\mathcal{N}(\boldsymbol{\mu}, \mathbf{C})$.

We can interpret a Gaussian random vektori $\mathbf{x} = (x_1, \dots, x_d)$ as a satunnaisprosessi indexed by the set $\mathcal{I} = \{1, \dots, d\}$. A Gaussin prosessit is a satunnaisprosessi over an arbitray index set \mathcal{I} such that any restriction to a finite subset $\mathcal{I}' \subseteq \mathcal{I}$ yields a Gaussian random vektori [44].

Gaussian RVs are widely used todennäköisyysmallit in the statistical analysis of koneoppiminen methods. Their significance arises partly from the keskeinen raja-arvolause which provides conditions under which the average of many independent RVs (not necessarily Gaussian themselves) tends toward a Gaussian RV [19].

The moninormaalijakauma is also distinct in that it represents maksimi epävarmuus. Among all vektori-valued RVs with a given covariance matrix \mathbf{C} , the RV $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{C})$ maximizes differential entropy [31, Th. 8.6.5]. This makes Gaussin prosessit a natural choice for capturing epävarmuus or the lack of (domain) knowledge.

See also: moninormaalijakauma, Gaussin prosessi, todennäköisyysmalli, keskeinen raja-arvolause, differential entropy.

normal distribution See normaalijakautunut satunnaismuuttuja.

normal matrix A square matrix $\mathbf{A} \in \mathbb{C}^{d \times d}$ that commutes with its conjugate transpose, i.e., $\mathbf{A}\mathbf{A}^H = \mathbf{A}^H\mathbf{A}$. Normal matrices admit an orthonormal basis of eigenvectors and are unitarily diagonalizable.

normal vector See hyperplane.

normi A norm is a funktion that maps each (vektori) element of a vektoriavaruus to a nonnegative real number. This funktion must be homogeneous and definite, and it must satisfy the triangle inequality [29].

See also: funktion, vektori, vektoriavaruus.

ominaisfunktio The characteristic funktion of a real-valued RV x is the funktion [6, Sec. 26]

$$\phi_x(t) := \mathbb{E}\exp(jtx) \text{ with } j = \sqrt{-1}.$$

The characteristic funktion uniquely determines the probability distribution of x .

See also: RV, probability distribution.

operator An operator is a funktion whose domain and co-domain have a specific mathematical structure such as a vektoriavaruus, a Hilbert space, or a metric space [41], [60]. Many koneoppiminen methods involve operators whose domain and co-domain are Euclidean spacet.

See also: funktion, vektoriavaruus, Hilbert space.

optimointimenetelmä An optimization method is an algoritmi that takes a representation of an optimointitehtävä as input and computes

an (approximate) solution as its output. A central example of an optimointitehtävä in koneoppiminen is ERM. By applying an appropriate optimization method to ERM, we obtain a concrete learning algoritmi [23], [61], [45].

See also: algoritmi, optimointitehtävä.

optimointitehtävä An optimization problem is a mathematical structure consisting of an objective function $f : \mathcal{U} \rightarrow \mathcal{V}$ defined over an optimization variable $\mathbf{w} \in \mathcal{U}$, together with a feasible set $\mathcal{W} \subseteq \mathcal{U}$. The co-domain \mathcal{V} is assumed to be ordered, meaning that for any two elements $\mathbf{a}, \mathbf{b} \in \mathcal{V}$, we can determine whether $\mathbf{a} \leq \mathbf{b}$ or not. Here, “ \leq ” denotes a general partial order relation, which may differ from the standard numerical order on real numbers [23, Sec. 2.4.]. The goal in optimization is to find those values $\mathbf{w} \in \mathcal{W}$ for which the objective $f(\mathbf{w})$ is extremal—i.e., minimal or maximal [23], [61], [45].

See also: objective function.

outcome Outcome is one possible result of a physical process. Such a process could be the observation of a physical phenomenon, a computation performed by an algoritmi, or a satunnaiskoe [6].

See also: sample space.

partial derivative Consider a real-valued funktio $f : \mathbb{R}^d \rightarrow \mathbb{R} : \mathbf{x} \mapsto f(\mathbf{x})$. The partial derivative of $f(\mathbf{x})$ with respect to the entry x_j measures how f changes when x_j varies while all other entries $x_{j'}$, for $j' \in [d] \setminus \{j\}$,

are held fixed. It is defined as [2]

$$\frac{\partial f(\mathbf{x})}{\partial x_j} = \lim_{\varepsilon \rightarrow 0} \frac{f(x_1, \dots, x_j + \varepsilon, \dots, x_d) - f(x_1, \dots, x_j, \dots, x_d)}{\varepsilon}.$$

Note that the partial derivative is only defined if this limit exists. For a differentiable function, the partial derivatives of f are the entries of the gradient ∇f .

See also: function, derivative, gradient.

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 $\mathbf{x} \in \mathbb{R}^d$. A psd matrix \mathbf{Q} admits a spectral decomposition with nonnegative eigenvalues $\lambda_1, \dots, \lambda_d \geq 0$ [29], [30]. The notion of being psd can be extended from matrices to (real-valued) symmetric kernel functions $K : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ (with $K(\mathbf{x}, \mathbf{x}') = K(\mathbf{x}', \mathbf{x})$) as follows: For any finite set of feature vectors $\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}$, the resulting matrix $\mathbf{Q} \in \mathbb{R}^{m \times m}$ with entries $Q_{r,r'} = K(\mathbf{x}^{(r)}, \mathbf{x}^{(r')})$ is psd [62].

See also: symmetric matrix, kernel.

posterior The study and design of machine learning methods is often based on a probabilistic model for the data generation process. Within a probabilistic model, we view (the generation of) a data point with features \mathbf{x} and label y as an RV with probability distribution $\mathbb{P}^{(\mathbf{x},y)}$. It turns out that the optimal estimate for the label y , given the feature vector \mathbf{x} , is fully determined by the conditional probability distribution of y given (or conditioned on) \mathbf{x} .

See also: probabilistic model, conditional probability distribution.

preimage Consider a funktion $f: \mathcal{U} \rightarrow \mathcal{V}$ between two sets. The preimage $f^{-1}(\mathcal{B})$ of a subset $\mathcal{B} \subseteq \mathcal{V}$ is the set of all inputs $u \in \mathcal{U}$ that are mapped into \mathcal{B} by f , i.e.,

$$f^{-1}(\mathcal{B}) := \{u \in \mathcal{U} \mid f(u) \in \mathcal{B}\}.$$

The preimage is well defined even if the funktion f is non-invertible [2].

See also: funktion.

probabilistic principal component analysis (PPCA) PPCA extends basic PCA by using a todennäköisyysmalli for data points. The todennäköisyysmalli of PPCA frames the task of ulottuvuuksien vähentäminen as an estimation problem that can be solved using EM [63].

See also: PCA, todennäköisyysmalli, ulottuvuuksien vähentäminen, EM.

probability density function (pdf) The pdf $p^{(x)}(\cdot)$ of a continuous real-valued RV $x \in \mathbb{R}$ allows to compute the todennäköisyys $\mathbb{P}(x \in \mathcal{B})$ (of the event $\mathcal{B} \subseteq \mathbb{R}$) via a Lebesgue integral [7, Ch. 3]

$$\mathbb{P}(x \in \mathcal{B}) = \int_{\mathcal{B}} p^{(x)}(\eta) d\eta.$$

This definition extends naturally to a (continuous) vector-valued RV $\mathbf{x} \in \mathbb{R}^d$ as the Lebesgue integral is defined for \mathbb{R}^d with any dimension d . See also: RV, probability distribution, todennäköisyys, measurable, vektori.

probability distribution To analyze koneoppiminen methods, it can be useful to interpret data points as i.i.d. realizations of a RV. The typical properties of such data points are then governed by the todennäköisyys

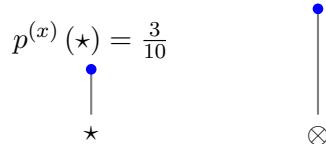
distribution of this RV. The todennäköisyys distribution of a binary RV $y \in \{0, 1\}$ is fully specified by the todennäköisyydet $\mathbb{P}(y = 0)$ and $\mathbb{P}(y = 1) = 1 - \mathbb{P}(y = 0)$. The todennäköisyys distribution of a real-valued RV $x \in \mathbb{R}$ might be specified by a pdf $p(x)$ such that $\mathbb{P}(x \in [a, b]) \approx p(a)|b - a|$. In the most general case, a todennäköisyys distribution is defined by a todennäköisyys measure [6], [18].

See also: i.i.d., realization, RV, todennäköisyys, pdf.

probability mass function (pmf) The pmf of a discrete RV x is a function $p^{(x)}(\cdot) : \mathcal{X} \rightarrow [0, 1]$ that assigns to each possible value $x' \in \mathcal{X}$ of the RV x the probability $p^{(x)}(x') = \mathbb{P}(x' = x)$ [26]. Fig. 41 illustrates the pmf of a discrete RV x .

$$\mathcal{D}'' = (\otimes, \otimes, \otimes, \star, \otimes, \star, \otimes, \otimes, \star, \otimes)$$

$$\mathcal{D}' = (\otimes, \star, \otimes, \star, \otimes, \otimes, \star, \otimes, \otimes, \otimes)$$



$$\mathcal{D} = (\star, \star, \star, \otimes, \otimes, \otimes, \otimes, \otimes, \otimes, \otimes)$$

Fig. 41. The pmf $p^{(x)}(\cdot)$ of a discrete RV x taking values in the set $\mathcal{X} = \{\star, \otimes\}$. Three datasets are also shown whose relative frequencies of data points match this pmf exactly. Such datasets could arise as realizations of i.i.d. RVs sharing the common pmf $p^{(x)}(\cdot)$.

A pmf always satisfies $\sum_{x' \in \chi} p^{(x)}(x') = 1$. We can view a pmf as

representing a collection of (sufficiently long) datasets. This collection contains any $\mathcal{D} = \{x^{(1)}, \dots, x^{(m)}\}$, with the relative frequencies of every value $x' \in \mathcal{X}$ being close to the corresponding pmf value $p^{(x)}(x')$,

$$\frac{|r \in \{1, \dots, m\} : x^{(r)} = x'|}{m} \approx p^{(x)}(x').$$

Note that requiring relative frequencies to be close to the pmf values implies that the empirical entropia of such a dataset is close to the entropia of the pmf $p^{(x)}(\cdot)$. Information theory refers to the collection of such datasets as the typical set corresponding to the pmf $p^{(x)}(\cdot)$ [31]. A main result of information theory states that a dataset generated by i.i.d. sampling from $p^{(x)}(\cdot)$ belongs, with high todennäköisyys, to the typical set with respect to $p^{(x)}(\cdot)$ [31, Th. 3.1.2].

See also: discrete RV, todennäköisyys, probability distribution, todennäköisyysmalli.

probability simplex The todennäköisyys simplex Δ^{k-1} is the set of all vektorit in \mathbb{R}^k with nonnegative entries that sum to one [23]. Each element of Δ^{k-1} represents a pmf of an RV $y \in \{1, \dots, k\}$.

See also: todennäköisyys, vektori, pmf, RV.

probability space A todennäköisyys space is a mathematical structure that allows us to reason about a satunnaiskoe, e.g., the observation of a physical phenomenon. Formally, a todennäköisyys space \mathcal{P} is a triplet $(\Omega, \mathcal{F}, \mathbb{P}(\cdot))$ where

- Ω is a sample space containing all possible outcomes of a satunnaiskoe;

- \mathcal{F} is a σ -algebra, i.e., a collection of subsets of Ω (called events) that satisfies certain closure properties under set operations;
- $\mathbb{P}(\cdot)$ is a probability distribution, i.e., a funktion that assigns a todennäköisyys $\mathbb{P}(\mathcal{A}) \in [0, 1]$ to each event $\mathcal{A} \in \mathcal{F}$. This funktion must satisfy $\mathbb{P}(\Omega) = 1$ and $\mathbb{P}(\bigcup_{i=1}^{\infty} \mathcal{A}_i) = \sum_{i=1}^{\infty} \mathbb{P}(\mathcal{A}_i)$ for any countable sequence of pairwise disjoint events $\mathcal{A}_1, \mathcal{A}_2, \dots$ in \mathcal{F} .

Todennäköisyys spaces provide the foundation of todennäköisyysmallit that can be used to study the behavior of koneoppiminen methods [6], [18], [19].

See also: todennäköisyys, satunnaiskoe, sample space, event, probability distribution, funktion, todennäköisyysmalli, koneoppiminen.

projected gradient descent (projected GD) Consider an ERM-based method that uses a parameterized 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 constraints on the optimization variable (i.e., the model parameters). Projected GD extends basic GD to address this issue. A single iteration of projected GD consists of first taking a gradienttiaskel and then projecting the result back onto the parameter space. See Fig. 42 for a visual illustration.

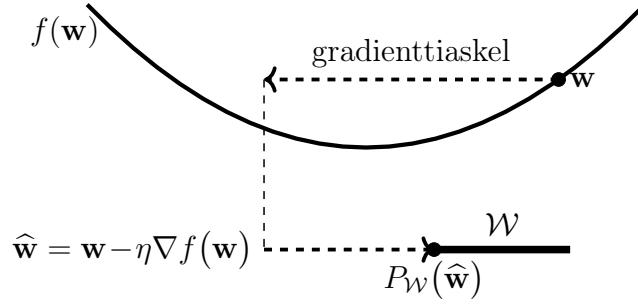


Fig. 42. Projected GD augments a basic gradienttiaskel with a projektio back onto the constraint set \mathcal{W} .

See also: ERM, model, parameter space, objective function, smooth, GD, model parameter, gradienttiaskel, projektio.

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

$$P_{\mathcal{W}}(\mathbf{w}) = \arg \min_{\mathbf{w}' \in \mathcal{W}} \|\mathbf{w} - \mathbf{w}'\|_2.$$

In other words, $P_{\mathcal{W}}(\mathbf{w})$ is the vektori in \mathcal{W} that is closest to \mathbf{w} . The projection is only well defined for subsets \mathcal{W} for which the above minimi exists [23].

See also: Euclidean space, vektori, minimi.

proximable A convex funktilo for which the proximal operator can be computed efficiently is sometimes referred to as proximable or simple [64].

See also: convex, funktilo, proximal operator.

proximal operator Given a convex funktilo $f(\mathbf{w}')$, we define its proximal

operator as [41], [22]

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

As illustrated in Fig. 43, 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 vektori \mathbf{w} (which is the input to the proximal operator). The proximal operator can be interpreted as a generalization of the gradienttiaskel, which is defined for a smooth convex funkto $f(\mathbf{w}')$. Indeed, taking a gradienttiaskel with step size η at the current vektori \mathbf{w} is the same as applying the proximal operator of the funkto $\tilde{f}(\mathbf{w}') = (\nabla f(\mathbf{w}))^T(\mathbf{w}' - \mathbf{w})$ and using $\rho = 1/\eta$.

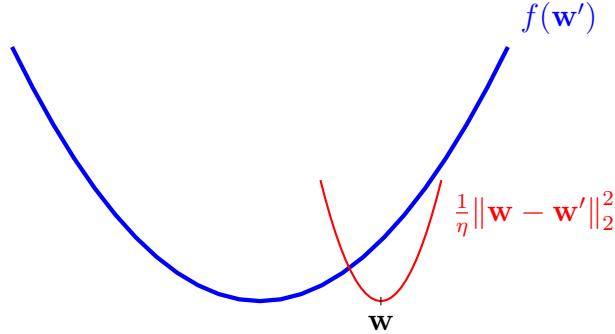


Fig. 43. The proximal operator updates a vektori \mathbf{w} by minimizing a penalized version of the funkto $f(\cdot)$. The penalty term is the scaled squared Euclidean distance between the optimization variable \mathbf{w}' and the given vektori \mathbf{w} .

See also: convex, funkto, gradienttiaskel.

pseudokäänteisluku The Moore–Penrose pseudoinverse \mathbf{A}^+ of a matriisi $\mathbf{X} \in \mathbb{R}^{m \times d}$ generalizes the notion of an käänteismatriisi [3]. The pseu-

doinverse arises naturally in ridge regression for a dataset with feature matrix \mathbf{X} and label vector \mathbf{y} [65, Ch. 3]. The model parameters learned by ridge regression are given by

$$\hat{\mathbf{w}}^{(\alpha)} = (\mathbf{X}^T \mathbf{X} + \alpha \mathbf{I})^{-1} \mathbf{X}^T \mathbf{y}, \quad \alpha > 0.$$

We can then define the pseudoinverse $\mathbf{X}^+ \in \mathbb{R}^{d \times m}$ via the limit [66, Ch. 3]

$$\lim_{\alpha \rightarrow 0^+} \hat{\mathbf{w}}^{(\alpha)} = \mathbf{X}^+ \mathbf{y}.$$

See also: matriisi, käänteismatriisi, ridge regression.

random geometric graph (RGG) An RGG is a todennäköisyysmalli for verkot built from nodes randomly placed in a metric space. Given a metric space, the RGG is characterized by the number of nodes, the connection radius, and the probability distribution describing the node placement. More precisely, for a node set $i = 1, \dots, n$, each node i is assigned to a random position $\mathbf{x}^{(i)} \in \mathcal{X}$, typically as realizations of i.i.d. RVs taking values in a set \mathcal{X} . Together with a metric, this forms a metric space, often with the metric induced by a normi. A specific realization of an RGG contains an edge $\{i, i'\}$ if and only if the distance between the nodes with respect to the metric is smaller than some threshold, i.e., when $d(\mathbf{x}^{(i)}, \mathbf{x}^{(i')}) \leq r$ for some threshold $r > 0$, as illustrated in Fig. 44.

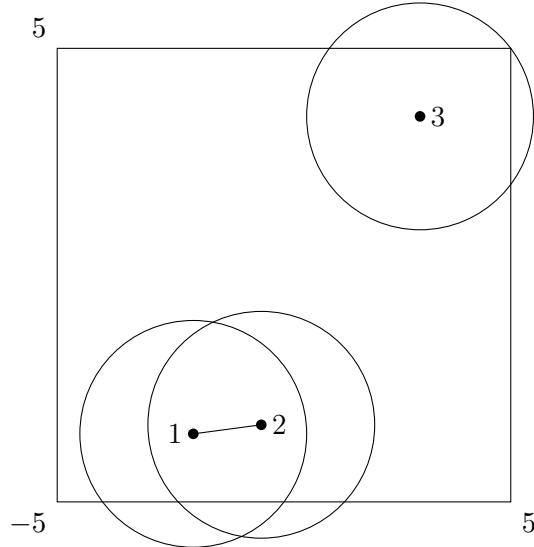


Fig. 44. Illustration of an RGG with $\mathcal{X} = [-5, 5] \times [-5, 5] \subset \mathbb{R}^2$ and radius $r = 2.5$ (with respect to the Euclidean distance), where nodes 1 and 2 (corresponding to $(-2.0, -3.5)^T$ and $(-0.5, -3.3)^T$, within radius $r = 2.5$) are connected, while node 3 (corresponding to $(3.0, 3.5)^T$) has no other node within that distance.

See also: verkko, stochastic block model (SBM), Erdős–Rényi -verkko (ER-verkko).

random variable (RV) An RV is a funktion that maps the outcomes of a satunnaiskoe to elements of a measurable space [6], [18]. Mathematically, an RV is a funktion $x : \Omega \rightarrow \mathcal{X}$ whose domain is the sample space Ω of a probability space and whose co-domain is a measurable space \mathcal{X} . Different types of RVs include

- binary RVs, which map each outcome to an element of a binary

set (e.g., $\{-1, 1\}$ or $\{\text{cat}, \text{no cat}\}$);

- discrete RVs, which take on values in a countable set (which can be finite or countably infinite);
- real-valued RVs, which take on values in the real numbers \mathbb{R} ;
- vektori-valued RVs, which map outcomes to the Euclidean space \mathbb{R}^d .

Todennäköisyys theory uses the concept of measurable spaces to rigorously define and study the properties of collections of RVs [6].

See also: funktio, satunnaiskoe, sample space, probability space, vektori, Euclidean space, todennäköisyys, measurable.

rank The rank of a matriisi $\mathbf{A} \in \mathbb{R}^{m \times d}$, denoted as $\text{rank } \mathbf{A}$, is the maksimi number of linearly independent columns of \mathbf{A} [32]. Equivalently, the rank can be defined as the dimension of the column space $\text{span}(\mathbf{A}) = \{\mathbf{Aw} \text{ for some } \mathbf{w} \in \mathbb{R}^d\}$. The rank of a matriisi $\mathbf{A} \in \mathbb{R}^{m \times d}$ can neither exceed the number of rows nor the number of columns of \mathbf{A} [52], [67], i.e., $\text{rank } \mathbf{A} \leq \min\{m, d\}$.

See also: matriisi, dimension, column space, lineaarikuvaus.

rank-deficient A matriisi $\mathbf{A} \in \mathbb{R}^{m \times d}$ is rank-deficient if it is not full-rank, i.e., when $\text{rank } \mathbf{A} < \min\{m, d\}$.

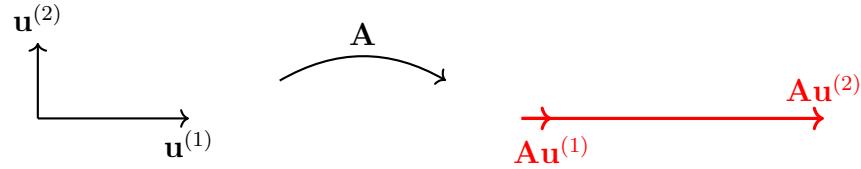


Fig. 45. Example of a rank-deficient matriisi $\mathbf{A} \in \mathbb{R}^{2 \times 2}$.

In linear regression, the solution of the ERM problem is not unique whenever the feature matrix \mathbf{X} is such that the matriisi $\mathbf{X}^\top \mathbf{X}$ is rank-deficient.

See also: full-rank, dimension, vektoriavaruus.

Rényi divergence The Rényi divergence measures the (dis)similarity between two probability distributions [68].

See also: probability distribution.

sample In the context of koneoppiminen, a sample is a finite sequence (of length m) of data points $\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(m)}$. The number m is called the sample size. ERM-based methods use a sample to train a model (or learn a hypothesis) by minimizing the average loss (i.e., the empiirinen riski) over that sample. Since a sample is defined as a sequence, the same data point may appear more than once. By contrast, some authors in statistics define a sample as a set of data points, in which case duplicates are not allowed [69], [70]. These two views (i.e., sequence versus set) can be reconciled by regarding a sample as a sequence of feature–label pairs, $(\mathbf{x}^{(1)}, y^{(1)}) , \dots , (\mathbf{x}^{(m)}, y^{(m)})$. The r th pair consists of the features $\mathbf{x}^{(r)}$ and the label $y^{(r)}$ of an unique underlying data point $\tilde{\mathbf{z}}^{(r)}$. While

the underlying data points $\tilde{\mathbf{z}}^{(1)}, \dots, \tilde{\mathbf{z}}^{(m)}$ are unique, some of them can have identical features and labels.

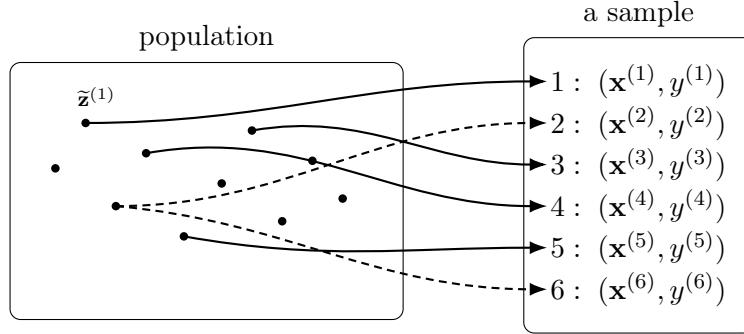


Fig. 46. A sample viewed as a finite sequence. Each element of this sample consists of the feature vector and the label of a data point from an underlying population. The same data point may occur more than once in the sample.

For the analysis of koneoppiminen methods, it is common to interpret (the generation of) a sample as the realization of a satunnaisprosessi indexed by $\{1, \dots, m\}$. A widely used assumption is the i.i.d. assumption, where sample elements $(\mathbf{x}^{(r)}, y^{(r)})$, for $r = 1, \dots, m$, are i.i.d. RVs with a common probability distribution.

See also: sequence, i.i.d. assumption, dataset.

sample covariance matrix Consider a dataset consisting of data points characterized by feature vectors $\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)} \in \mathbb{R}^d$. The sample covariance matriisi of \mathcal{D} is defined as the covariance matrix with respect to the empirical distribution $\mathbb{P}^{(\mathcal{D})}$ induced by \mathcal{D} . It is given explicitly by

$$\hat{\mathbf{C}} = \frac{1}{m} \sum_{r=1}^m (\mathbf{x}^{(r)} - \hat{\mathbf{m}})(\mathbf{x}^{(r)} - \hat{\mathbf{m}})^T.$$

Here, we use the sample mean $\hat{\mu}$.

See also: covariance, matriisi, RV.

satunnaiskoe A random experiment is a physical (or abstract) process that produces an outcome ω from a set Ω of possibilities. This set of all possible outcomes is referred to as the sample space of the experiment. The key characteristic of a random experiment is that its outcome is unpredictable (or uncertain). Any measurement or observation of the outcome is a RV, i.e., a funktion of the outcome $\omega \in \Omega$. Todennäköisyys theory uses a probability space as a mathematical structure for the study of random experiments. A key conceptual property of a random experiment is that it can be repeated under identical conditions. Strictly speaking, repeating a random experiment a given number of m times defines a new random experiment. The outcomes of this new experiment are length- m sequences of outcomes from the original experiment (see Fig. 47). While the outcome of a single experiment is uncertain, the long-run behaviour of the outcomes of repeated experiments tends to become increasingly predictable. This informal claim can be made precise via fundamental results of todennäköisyys theory, such as the law of large numbers and the keskeinen raja-arvolause.

new random experiment with $\Omega' = \Omega \times \dots \times \Omega$

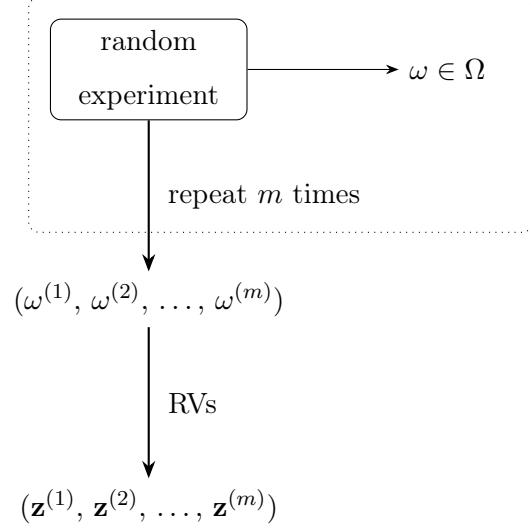


Fig. 47. A random experiment produces an outcome $\omega \in \Omega$ from a set of possibilities (i.e., a sample space) Ω . Repeating the experiment m times yields another random experiment, whose outcomes are sequences $(\omega^{(1)}, \omega^{(2)}, \dots, \omega^{(m)}) \in \Omega \times \dots \times \Omega$. One example of a random experiment arising in many koneoppiminen applications is the gathering of a training set $\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(m)}$.

Examples for random experiments arising in koneoppiminen applications include the following:

- Data collection: The data points collected in ERM-based methods can be interpreted as RVs, i.e., as funktiot of the outcome $\omega \in \Omega$ of a random experiment.
- Stochastic gradient descent (SGD) uses a random experiment at

each iteration to select a subset of the training set.

- Yksityisyyden suoja methods use random experiments to perturb the outputs of an koneoppiminen method to ensure differentiaalinen yksityisyyss.

See also: outcome, sample space, RV, todennäköisyys, probability space.

satunnaisprosessi A stokastinen process is a collection of RVs defined on a common probability space and indexed by some set \mathcal{I} [18], [26], [71]. The index set \mathcal{I} typically represents time or space, allowing us to represent random phenomena that evolve across time or spatial dimensions—for example, sensor noise or financial time series. Stokastinen processes are not limited to temporal or spatial settings. For instance, random verkot such as the ER-verkko or the SBM can also be viewed as stokastinen processes. Here, the index set \mathcal{I} consists of node pairs that index RVs whose values encode the presence or weight of an edge between two nodes. Moreover, stokastinen processes naturally arise in the analysis of satunnaisalgoritmit, such as SGD, which construct a sequence of RVs.

See also: RV, SBM, SGD, epävarmuus, todennäköisyysmalli.

Schur decomposition Every square matriisi $\mathbf{A} \in \mathbb{C}^{d \times d}$ admits a Schur decomposition [3, Thm. 7.1.3]

$$\mathbf{A} = \mathbf{U}\mathbf{T}\mathbf{U}^H.$$

Here, $\mathbf{U} \in \mathbb{C}^{d \times d}$ is a unitary matrix (i.e., $\mathbf{U}^H\mathbf{U} = \mathbf{I}$) and \mathbf{T} is upper triangular with the eigenvalues of \mathbf{A} on its diagonal. Carefully note that the Schur decomposition exists also for a matriisi \mathbf{A} that is not

$$(\mathbf{U}^{(1)})^H \mathbf{A} \mathbf{U}^{(1)} = \begin{pmatrix} (\mathbf{x}^{(1)})^H \\ (\mathbf{X}^{(2)})^H \end{pmatrix} \mathbf{A} \begin{pmatrix} \mathbf{x}^{(1)} & \mathbf{X}^{(2)} \end{pmatrix} = \begin{pmatrix} \lambda_1 & \mathbf{b}^H \\ 0 & \mathbf{A}^{(1)} \end{pmatrix},$$

Fig. 48. The first step in the construction of the Schur decomposition. Every matriisi $\mathbf{A} \in \mathbb{C}^{n \times n}$ has at least one eigenvalue λ_1 with unit-norm eigenvector $\mathbf{x}^{(1)}$, $\mathbf{A}\mathbf{x}^{(1)} = \lambda_1\mathbf{x}^{(1)}$. This eigenvector allows to decompose \mathbf{A} as depicted. Here, we extended $\mathbf{x}^{(1)}$ to an orthonormal basis $\mathbf{Q}^{(1)} = (\mathbf{x}^{(1)} | \mathbf{X}^{(2)})$ and used $\mathbf{b}^H := (\mathbf{x}^{(1)})^H \mathbf{A} \mathbf{X}^{(2)}$ and $\mathbf{A}^{(1)} := (\mathbf{X}^{(2)})^H \mathbf{A} \mathbf{X}^{(2)}$. Applying the same construction recursively to $\mathbf{A}^{(1)}$ yields the Schur decomposition.

diagonalizable,

See also: EVD, eigenvalue, matriisi.

sequence A sequence is an ordered collection of values from a set \mathcal{A} . For example, a sequence of values from the set $\mathcal{A} = \{\star, \otimes\}$ could be

$$a = (\star, \otimes, \star, \star, \otimes, \dots).$$

Formally, a sequence a is a funktion [2]

$$a : \mathbb{N} \rightarrow \mathcal{A} : r \mapsto a_r.$$

We denote a sequence by $(a_r)_{r \in \mathbb{N}}$ or $(a^{(r)})_{r \in \mathbb{N}}$. Sometimes we also use the notation $\{a^{(r)}\}_{r \in \mathbb{N}}$. Note that the same value $a \in \mathcal{A}$ can appear multiple times in the sequence at different positions r . Sequences are fundamental for the study of koneoppiminen methods, for instance when describing successive iterates $\{\mathbf{w}^{(t)}\}_{t \in \mathbb{N}}$ of an iterative algoritmi. We can also use a sequence to represent an infinite dataset

$$\mathcal{D} = \{ (\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \dots \}.$$

See also: funktilo, dataset.

σ -algebra Consider a satunnaiskoe with a sample space Ω . A σ -algebra (or σ -field) Σ is a collection of subsets of Ω with the following properties [1], [6], [17]:

- The empty set \emptyset and the entire sample space Ω belong to Σ , i.e., $\emptyset \in \Sigma$ and $\Omega \in \Sigma$.
- If a set \mathcal{A} belongs to Σ , then its complement $\Omega \setminus \mathcal{A}$ also belongs to Σ , i.e., $\mathcal{A} \in \Sigma$ implies $\Omega \setminus \mathcal{A} \in \Sigma$.
- If a countable collection of sets $\mathcal{A}_1, \mathcal{A}_2, \dots$ belongs to Σ , then their union also belongs to Σ , i.e., $\mathcal{A}_1, \mathcal{A}_2, \dots \in \Sigma$ implies $\bigcup_{i=1}^{\infty} \mathcal{A}_i \in \Sigma$.

See also: sample space, RV, probability space.

σ -field See σ -algebra.

simple function A simple funktilo $f : \mathbb{R}^d \rightarrow \mathbb{R}$ is a measurable funktilo that takes on only finitely many values. In other words,

$$f(\mathbf{x}) = \sum_{c=1}^k \alpha_c \mathbb{I}^{(\mathcal{C}^{(c)})}(\mathbf{x})$$

where $\mathbb{I}_{\mathcal{C}}$ denotes the indicator funktilo of a subset $\mathcal{C} \subset \mathbb{R}^d$ and $\alpha_1, \dots, \alpha_k \in \mathbb{R}$ are arbitrary coefficients. The subsets $\mathcal{C}^{(1)}, \dots, \mathcal{C}^{(k)}$ in the above decomposition must be measurable and must form a partition of \mathbb{R}^d .

See also: measurable, Lebesgue integral.

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

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

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

See also: matriisi.

smooth A real-valued funktion $f : \mathbb{R}^d \rightarrow \mathbb{R}$ is smooth if it is differentiable and its gradientti $\nabla f(\mathbf{w})$ is continuous at all $\mathbf{w} \in \mathbb{R}^d$ [45], [58]. A smooth funktion f is referred to as β -smooth if the gradientti $\nabla f(\mathbf{w})$ 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 smoothness of the funktion f : the smaller the β , the smoother f is. Optimointitehtävät 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 gradientti. This approximation works well if the gradientti does not change too rapidly. We can make this informal claim precise by studying the effect of a single gradienttiaiskel with step size $\eta = 1/\beta$ (see Fig. 49).

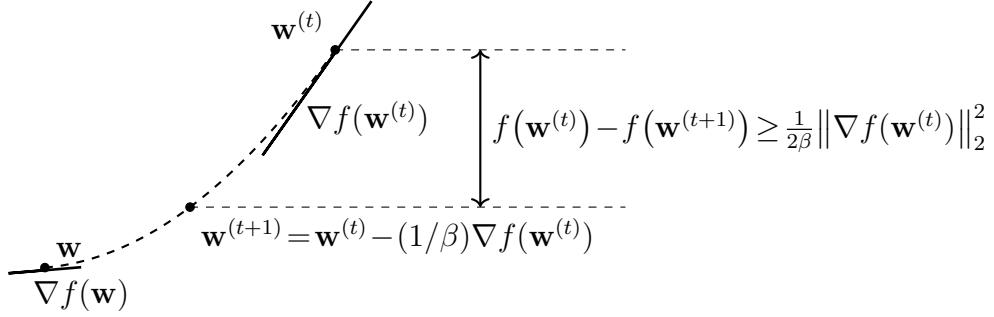


Fig. 49. Consider an objective function $f(\mathbf{w})$ that is β -smooth. Taking a gradient step, with step size $\eta = 1/\beta$, decreases the objective by at least $1/2\beta \|\nabla f(\mathbf{w}^{(t)})\|_2^2$ [45], [58], [72]. Note that the step size $\eta = 1/\beta$ becomes larger for smaller β . Thus, for smoother objective functions (i.e., those with smaller β), we can take larger steps.

See also: funktio, differentiable, gradientti, gradient-based method.

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

See also: undirected graph, naapuri.

spectral decomposition Every normal matrix $\mathbf{A} \in \mathbb{C}^{d \times d}$ admits a spectral decomposition of the form [29, 30]

$$\mathbf{A} = (\mathbf{u}^{(1)}, \dots, \mathbf{u}^{(d)}) \begin{pmatrix} \lambda_1 & & 0 \\ & \lambda_2 & \\ & & \ddots \\ 0 & & & \lambda_d \end{pmatrix} \begin{pmatrix} (\mathbf{u}^{(1)})^H \\ \vdots \\ (\mathbf{u}^{(d)})^H \end{pmatrix} = \sum_{j=1}^d \lambda_j \mathbf{u}^{(j)} (\mathbf{u}^{(j)})^H$$

with an orthonormal basis $\mathbf{u}^{(1)}, \dots, \mathbf{u}^{(d)}$. Each basis element $\mathbf{u}^{(j)}$ is an

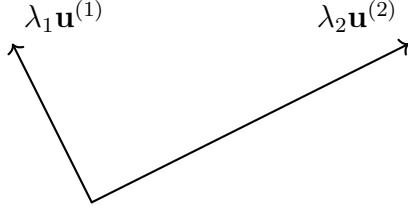


Fig. 50. The spectral decomposition of a normal matrix \mathbf{A} provides an orthonormal basis $\mathbf{u}^{(1)}, \mathbf{u}^{(2)}$. Applying \mathbf{A} amounts to a scaling of the basis vektorit by the eigenvalues λ_1, λ_2 .

eigenvector of \mathbf{A} with corresponding eigenvalue λ_j , for $j = 1, \dots, d$.

standard deviation The standard deviation of a real-valued RV x is defined as the square root of its varianssi, i.e., $\sqrt{\mathbb{E}\{(x - \mathbb{E}\{x\})^2\}}$.

See also: RV, varianssi, expectation.

standard normal random vector A standard normal random vektori is a RV $\mathbf{x} = (x_1, \dots, x_d)^T$ whose entries are i.i.d. normaalijakautuneet satunnaismuuttujat $x_j \sim \mathcal{N}(0, 1)$. The probability distribution of a standard normal random vektori is a special case of a moninormaalijakauma, $\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.

See also: vektori, i.i.d., normaalijakautunut satunnaismuuttuja, moninormaalijakauma, RV.

state space The state space of a system is a set that contains all possible states of a system at a given time step. A state is a mathematical representation of the information that characterizes the system. States may be fully observed, partially observed, or latent. State spaces are used across a wide range of koneoppiminen techniques, including todennäköi-

syysmallit and vahvistusoppiminen.

See also: feature vector, hypothesis, action, vahvistusoppiminen.

stokastinen We refer to a method as stochastic if it involves a random component or is governed by probabilistic laws. Koneoppiminen methods use randomness to reduce computational complexity (e.g., see SGD) or to capture epävarmuus in todennäköisyysmallit.

See also: SGD, epävarmuus, todennäköisyysmalli.

strictly convex A real-valued funktio $f(\mathbf{x})$ is strictly convex if for any two distinct \mathbf{x}, \mathbf{y} in its domain and any $\lambda \in (0, 1)$ it satisfies [23, Def. 3.1.1]

$$f(\lambda\mathbf{x} + (1 - \lambda)\mathbf{y}) < \lambda f(\mathbf{x}) + (1 - \lambda)f(\mathbf{y}).$$

Equivalently, for any $\mathbf{x} \neq \mathbf{y}$,

$$f(\mathbf{y}) > f(\mathbf{x}) + \nabla f(\mathbf{x})^T(\mathbf{y} - \mathbf{x}),$$

which implies that $f(\cdot)$ admits a unique minimizer on any convex subset of its domain. Unlike strongly convex funktiot, strictly convex funktiot do not require a uniform quadratic lower bound.

See also: convex, strongly convex.

strongly convex A continuously differentiable real-valued funktio $f(\mathbf{x})$ is strongly convex with coefficient σ if $f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla f(\mathbf{x})^T(\mathbf{y} - \mathbf{x}) + (\sigma/2)\|\mathbf{y} - \mathbf{x}\|_2^2$ [45], [72, Sec. B.1.1].

See also: differentiable, funktio, convex.

subgradient For a real-valued funktio $f : \mathbb{R}^d \rightarrow \mathbb{R} : \mathbf{w} \mapsto f(\mathbf{w})$, a vektori \mathbf{a} such that $f(\mathbf{w}) \geq f(\mathbf{w}') + (\mathbf{w} - \mathbf{w}')^T \mathbf{a}$ is referred to as a subgradient

of f at \mathbf{w}' [34], [61].

See also: funktio, vektori.

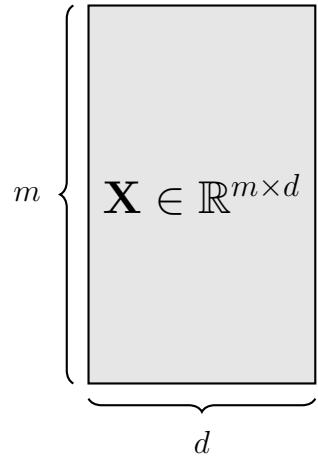
subspace A subset of a vektoriavaruus \mathcal{V} is a subspace of \mathcal{V} if it is also a vektoriavaruus with respect to the same operations as \mathcal{V} .

See also: vektoriavaruus.

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].

symmetric matrix A square matriisi \mathbf{A} with real-valued entries that is equal to its transpose, i.e., $\mathbf{A} = \mathbf{A}^T$. Every symmetric matriisi is a normal matrix.

tall matrix A matriisi $\mathbf{X} \in \mathbb{R}^{m \times d}$ is referred to as tall if it has more rows than columns, i.e., when $m > d$.



See also: matriisi.

todennäköisyysmalli A probabilistic model for the generation of data points consists of RVs with a joint probability distribution [7]. This joint probability distribution typically involves parameters that have to be manually chosen or learned via statistical inference methods such as maximum likelihood estimation [57].

See also: model, data point, RV, probability distribution, parameter, maximum likelihood, realization.

trace (matrix) The trace $\text{tr}(\mathbf{A})$ of a square matriisi $\mathbf{A} \in \mathbb{R}^{d \times d}$ is the sum of its diagonal entries [32]. Formally, it is the lineaarikuvaus

$$\text{tr}(\cdot) : \mathbb{R}^{d \times d} \rightarrow \mathbb{R} : \mathbf{A} \mapsto \sum_{j=1}^d A_{j,j}.$$

It satisfies the cyclic property $\text{tr}(\mathbf{AB}) = \text{tr}(\mathbf{BA})$, for any matriisit $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{d \times d}$ [30, p. 301]. Furthermore, if \mathbf{A} has eigenvalues $\lambda_1, \dots, \lambda_d$

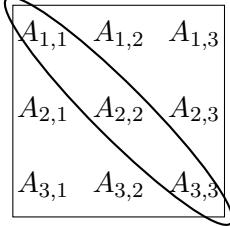


Fig. 51. The trace $\text{tr}(\mathbf{A})$ of a 3×3 matrix $\mathbf{A} \in \mathbb{R}^{3 \times 3}$ is the sum of three main diagonal entries $A_{1,1}, A_{2,2}, A_{3,3}$.

(each repeated according to its algebraic multiplicity), then

$$\text{tr}(\mathbf{A}) = \sum_{j=1}^d \lambda_j.$$

This identity follows from the invariance of the trace under similarity transformations [30, Ch. 10].

See also: eigenvalue, matriisi.

transpose The transpose of a real-valued matriisi is obtained by exchanging rows and columns. For a matriisi $\mathbf{A} \in \mathbb{R}^{m \times d}$, its transpose is denoted \mathbf{A}^T and satisfies $(\mathbf{A}^T)_{j,j'} = \mathbf{A}_{featureidx',j}$.

typical set See pmf.

undirected graph See verkko.

unitary (matrix) A square matriisi $\mathbf{U} \in \mathbb{C}^{d \times d}$ is called unitary if its conjugate transpose (Hermitian transpose) \mathbf{U}^H is also its inverse, i.e., if

$$\mathbf{U}^H \mathbf{U} = \mathbf{U} \mathbf{U}^H = \mathbf{I}.$$

Equivalently, the columns (and rows) of a unitary matriisi form an orthonormal basis of \mathbb{C}^d with respect to the standard inner product [29, 30].

See also: matriisi.

varianssi The variance of a real-valued RV x is defined as the expectation $\mathbb{E}\{(x - \mathbb{E}\{x\})^2\}$ of the squared difference between x and its expectation $\mathbb{E}\{x\}$. We extend this definition to vektori-valued RVs \mathbf{x} as $\mathbb{E}\{\|\mathbf{x} - \mathbb{E}\{\mathbf{x}\}\|_2^2\} = \text{tr}(\mathbf{C}^{(\mathbf{x})})$, i.e., the sum of the variances of each entry of \mathbf{x} , which can be written compactly as the trace of the covariance matrix $\mathbf{C}^{(\mathbf{x})}$ of \mathbf{x} .

See also: RV, expectation, vektori.

vektori A vector is an element of a vektoriavaruus. In the context of koneop-piminen, a particularly important example of a vektoriavaruus is the Euclidean space \mathbb{R}^d , where $d \in \mathbb{N}$ is the (finite) dimension of the space. A vector $\mathbf{x} \in \mathbb{R}^d$ can be represented as a list or one-dimensional (1-D) array of real numbers, i.e., x_1, \dots, x_d with $x_j \in \mathbb{R}$ for $j = 1, \dots, d$. The value x_j is the j th entry of the vector \mathbf{x} . It can also be useful to view a vector $\mathbf{x} \in \mathbb{R}^d$ as a funktio that maps each index $j \in \{1, \dots, d\}$ to a value $x_j \in \mathbb{R}$, i.e., $\mathbf{x} : j \mapsto x_j$. This perspective is particularly useful for the study of kernel methoddt. See Fig. 52 for the two views of a vector.

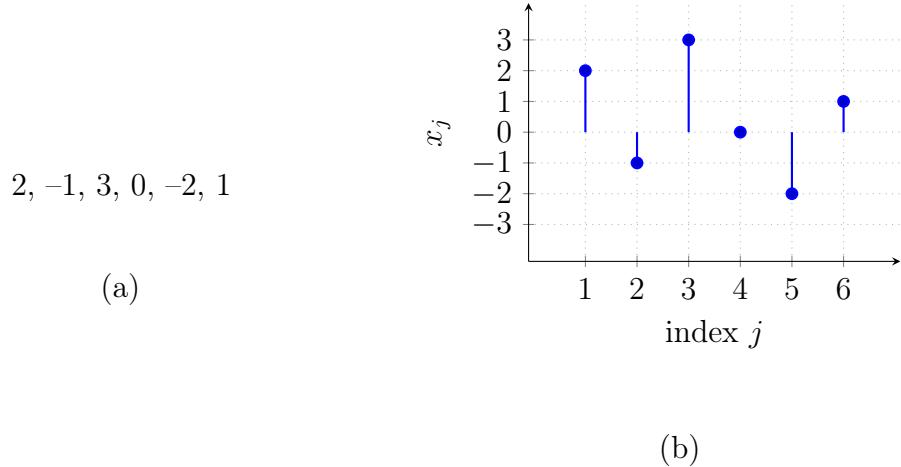


Fig. 52. Two equivalent views of a vector $\mathbf{x} = (2, -1, 3, 0, -2, 1)^T \in \mathbb{R}^6$. (a) As a numeric array. (b) As a kuvaus $j \mapsto x_j$.

See also: vektoriavaruus, Euclidean space, lineaarikuvaus.

vektoriavaruus A vektori space \mathcal{V} (also called linear space) is a collection of elements, called vektorit, along with the following two operations (see also Fig. 53): 1) addition (denoted by $\mathbf{v} + \mathbf{w}$) of two vektorit \mathbf{v}, \mathbf{w} ; and 2) multiplication (denoted by $c \cdot \mathbf{v}$) of a vektori \mathbf{v} with a scalar c that belongs to some number field (such as the real numbers \mathbb{R} or the complex numbers \mathbb{C}). The defining property of a vektori space is that it is closed under two specific operations. First, if $\mathbf{v}, \mathbf{w} \in \mathcal{V}$, then $\mathbf{v} + \mathbf{w} \in \mathcal{V}$. Second, if $\mathbf{v} \in \mathcal{V}$ and $c \in \mathbb{R}$, then $c\mathbf{v} \in \mathcal{V}$.

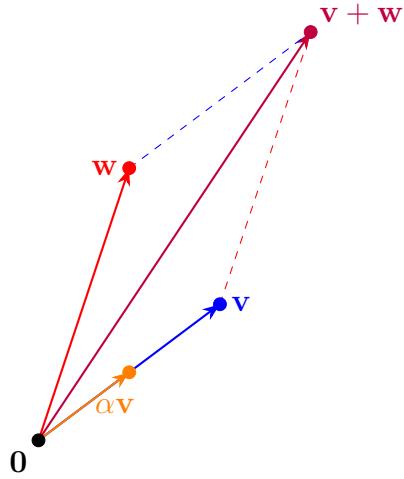


Fig. 53. A vektori space \mathcal{V} is a collection of vektorit such that scaling and adding them always yields another vektori in \mathcal{V} .

A common example of a vektori space is the Euclidean space \mathbb{R}^n , which is widely used in koneoppiminen to represent datasets. We can also use \mathbb{R}^n to represent, either exactly or approximately, the hypothesis space used by an koneoppiminen method. Another example of a vektori space, which is naturally associated with every probability space $(\Omega, \mathcal{F}, \mathbb{P}(\cdot))$, is the collection of all real-valued RVs $x : \Omega \rightarrow \mathbb{R}$ [1], [13].

See also: vektori, Euclidean space, lineaarinen malli, lineaarikuvaus.

verkko A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ consists of a node set \mathcal{V} and an edge set \mathcal{E} . Each edge $e \in \mathcal{E}$ is characterized by the nodes to which it is connected and in what precise sense. For example, an edge of a directed graph is leaving one node and pointing to another node. An edge of an undirected graph connects two nodes without any sense of direction [33, 46]. In principle, there can also be several (parallel) edges that are connected to the same

nodes in the same way [46]. Moreover, edges may connect a node to itself, resulting in so-called self-loops [33]. A simple undirected graph contains no parallel edges and no self-loops [47]. Each edge $e \in \mathcal{E}$ of a simple undirected graph can be identified with a set of two nodes i, i' . Weighted graphs assign a numerical value A_e , referred to as edge weight,

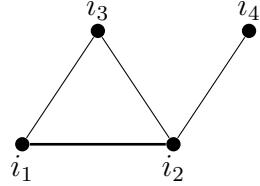


Fig. 54. A simple undirected graph with four nodes $\mathcal{V} = \{i_1, i_2, i_3, i_4\}$ and four edges $\mathcal{E} = \{\{i_1, i_2\}, \{i_2, i_3\}, \{i_3, i_1\}, \{i_2, i_4\}\}$.

to each edge $e \in \mathcal{E}$.

See also: kuvaus, weights.

weighted graph A verkko whose edges are assigned numeric weights. Typically, these edge weights are nonnegative real numbers. For example, if a verkko represents a road network with nodes being intersections and edges representing road segments, the edge weight could represent the capacity (measured in maximum vehicles per hour) of the road segment [33]. See also: verkko.

wide matrix A matriisi $\mathbf{X} \in \mathbb{R}^{m \times d}$ is referred to as wide if it has more columns than rows, i.e., when $d > m$.

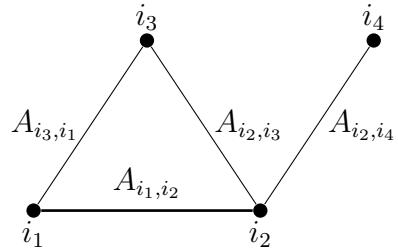


Fig. 55. A weighted graph with four nodes $\mathcal{V} = \{i_1, i_2, i_3, i_4\}$ and four edges $\mathcal{E} = \{\{i_1, i_2\}, \{i_2, i_3\}, \{i_3, i_1\}, \{i_2, i_4\}\}$. Each edge is assigned a weight.

$$m \left\{ \underbrace{\boxed{\mathbf{X} \in \mathbb{R}^{m \times d}}}_{d \quad (d > m)} \right.$$

See also: matriisi.

Machine Learning Concepts

absolute error loss Consider a data point with features $\mathbf{x} \in \mathcal{X}$ and numeric label $y \in \mathbb{R}$. As its name suggests, the absolute error loss incurred by a hypothesis $h : \mathcal{X} \rightarrow \mathbb{R}$ is defined as

$$L((\mathbf{x}, y), h) = |y - h(\mathbf{x})|.$$

Fig. 56 depicts the absolute error loss for a fixed data point with feature vector \mathbf{x} and label y . It also indicates the loss values incurred by two different hypotheses h' and h'' . Similar to the neliövirhehäviö, the absolute error loss is also a convex function of the estimate $\hat{y} = h(\mathbf{x})$. However, in contrast to the neliövirhehäviö, the absolute error loss is non-smooth, as it is not differentiable at the optimal estimate $\hat{y} = y$. This property makes ERM-based methods using the absolute error loss computationally more demanding [45], [73]. To build intuition, it is useful to consider the two hypotheses depicted in Fig. 56. Just by inspecting the slope of L around $h'(\mathbf{x})$ and $h''(\mathbf{x})$, it is impossible to determine whether we are very close to the optimum (at h') or still far away (at h''). As a result, any optimisation method that is based on local approximations of the loss function (such as subgradient descent) must use a decreasing learning rate to avoid overshooting when approaching the optimum. This required decrease in learning rate tends to slow down the convergence of the optimisation method. Besides the increased computational complexity, using absolute error loss in ERM can be beneficial in the presence of outliers in the training set. In contrast to the neliövirhehäviö, the slope of the absolute error loss does

not increase with increasing ennuste error $y - h(\mathbf{x})$. As a result, the effect of introducing an outlier with large ennuste error on the solution \hat{h} of ERM with absolute error loss is much smaller compared with the effect on the solution of ERM with neliövirhehäviö.

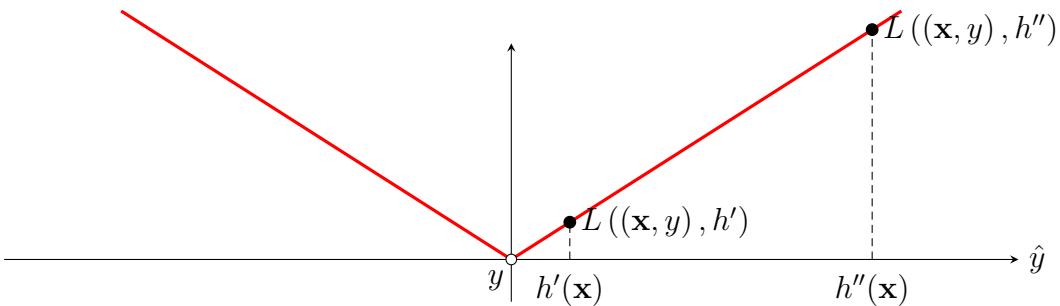


Fig. 56. For a data point with numeric label $y \in \mathbb{R}$, the absolute error $|y - h(\mathbf{x})|$ can be used as a loss function to guide the learning of a hypothesis h .

See also: data point, feature, label, loss, ERM, subgradient descent, least absolute deviation regression.

activation The output of an artificial neuron within an ANN is referred to as its activation. In particular, the activation is obtained by applying a (typically nonlinear) aktivointifunktio to a weighted sum of its inputs.

See also: ANN, deep net.

adaptive boosting (AdaBoost) AdaBoost is a specific boosting algoritmi that combines base learners sequentially [74], [75], [76]. The core idea of AdaBoost is to use the ennuste errors of the current base learner for sample weighting in the next base learner. In particular, the t th base learner learns a hypothesis $\hat{h}^{(t)}$ by weighted ERM with sample weights

$q^{(r)}$. The ennuste errors of $\hat{h}^{(t)}$ are then used to update the sample weights by increasing the weights of data points that have been predicted poorly (i.e., with large loss) by $\hat{h}^{(t)}$. The updated sample weights are then used in the next base learner to learn $\hat{h}^{(t+1)}$. The ultimate hypothesis $\tilde{h}^{(R)}$ delivered after R iterations is a linear combination of the hypotheses $\hat{h}^{(1)}, \dots, \hat{h}^{(R)}$. AdaBoost can be interpreted as a generalized gradienttiaskel [65, Ch. 10.4]

$$\tilde{h}^{(t)} = \tilde{h}^{(t-1)} + \eta^{(t)} \hat{h}^{(t)}.$$

This generalized gradienttiaskel involves a learning rate $\eta^{(t)}$, which controls the amount of modification of the current hypothesis.

See also: boosting, sample weighting, ERM.

aktivointifunktio Each artificial neuron within an ANN is assigned an activation funkatio $\sigma(\cdot)$ that maps a weighted combination of the neuron inputs x_1, \dots, x_d to a single output value $a = \sigma(w_1x_1 + \dots + w_dx_d)$. Note that each neuron is parameterized by the weights w_1, \dots, w_d .

See also: ANN, activation, rectified linear unit (ReLU).

algoritmi An algorithm is a precise, step-by-step specification for producing an output from a given input within a finite number of well-defined computational steps [77]. For example, a gradient-based method for linear regression is an algorithm that explicitly describes how to map a given training set into model parameters through a sequence of gradient-tiaskeleet. The precise form of an algorithm depends on the available computational infrastructure. For example, if this infrastructure allows us to compute an käänteismatriisi, then we can define a linear regression

algorithm using the normal equations. In contrast, if the computational infrastructure only allows basic arithmetic (i.e., multiplication and addition), the normal equations need to be somehow translated into a sequence of arithmetic operations (e.g., as in gradient-based methods). To study algorithms rigorously, we can represent (or approximate) them by different mathematical structures [78]. One approach is to represent an algorithm as a collection of possible executions. Each individual execution is then a sequence of the form

$$\text{input}, s_1, s_2, \dots, s_T, \text{output}.$$

This sequence starts from an input and progresses via intermediate steps until an output is delivered. Crucially, an algorithm encompasses more than just a mapping from input to output; it also includes intermediate computational steps s_1, \dots, s_T .

See also: linear regression, training set, model parameter, gradienttiaaskel, output, model, stokastinen.

alisovittaminen Consider an koneoppiminen method applying ERM to learn a hypothesis that minimizes the empiirinen riski on a given training set. The method is said to underfit if it fails to achieve a sufficiently low empiirinen riski on the training set. Underfitting typically occurs when the chosen model is too simple to capture the underlying relationship between features and labels.

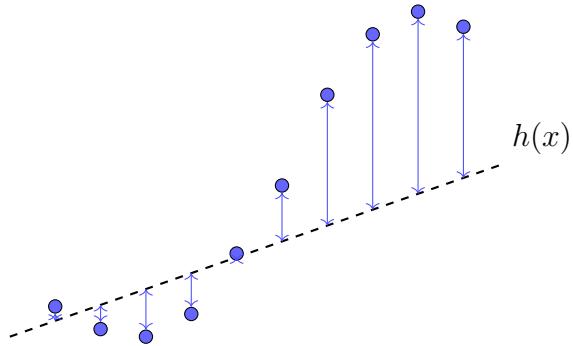


Fig. 57. No linear hypothesis h can capture the relationship between features and labels for the depicted training set. Thus, any method that uses a lineaarinen malli will underfit this training set.

For example, an koneoppiminen method using a lineaarinen malli on data with a highly nonlinear relationship between features and label will not be able to learn a hypothesis with small average loss on the training set, let alone a low riski.

See also: training set, model, riski, ylisovittaminen.

application programming interface (API) An API is a formal mechanism that allows software components to interact in a structured and modular way [79]. In the context of koneoppiminen, APIs are commonly used to provide access to a trained koneoppiminen model. Users—whether humans or machines—can submit the feature vector of a data point and receive a corresponding ennuste. Suppose a trained koneoppiminen model is defined as $\hat{h}(x) := 2x + 1$. Through an API, a user can input $x = 3$ and receive the output $\hat{h}(3) = 7$ without knowledge of the detailed structure of the koneoppiminen 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 ennuste $\hat{h}(\mathbf{x})$. APIs promote modularity in koneoppiminen system design, i.e., one team can develop and train the model, while another team handles integration and user interaction. Publishing a trained model via an API also offers practical advantages. For instance, the server can centralize computational resources that are required to compute ennusteet. Furthermore, the internal structure of the model remains hidden—which is useful for protecting intellectual property or trade secrets. However, APIs are not without riski. Techniques such as model inversion can potentially reconstruct a model from its ennusteet using carefully selected feature vectors.

See also: koneoppiminen, model, feature vector, data point, ennuste, feature, model inversion.

area under the curve (AUC) The AUC is a quantitative measure of the usefulness of a binary luokitin [80]. It is defined (using the natural measure of the Euclidean space \mathbb{R}^2) as the area under the receiver operating characteristic (ROC) curve.

See also: luokitin, Euclidean space, ROC.

artificial intelligence (AI) AI refers to systems that behave rationally in the sense of maximizing a long-term reward. The koneoppiminen-based approach to AI is to train a model to predict optimal actions. These ennusteet are computed from observations about the state of the

environment. The choice of loss function sets AI applications apart from more basic koneoppiminen 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 estimate the loss incurred by the current choice of model parameters.

See also: koneoppiminen, vahvistusoppiminen.

artificial neural network (ANN) An artificial neural network (ANN) is a graphical (signal-flow) representation of a hypothesis that maps features of a data point at its input to a ennuste for the corresponding label at its output. The fundamental computational unit of an ANN is the artificial neuron, which applies an aktivoointifunktio $\sigma(\cdot)$ to the sum of its inputs. The output of a neuron can be used either as the final output of the ANN or as an input to other neurons. A key design parameter of an ANN is its connectivity structure (or architecture), i.e., which neuron outputs are connected to which neuron inputs. As illustrated in Fig. 58, we can represent an ANN as an DAG. One widely used type of ANN are deep nets where neurons form consecutive layers. In a deep net, the outputs of neurons in a given layer are typically only connnected to the inputs of the neurons in a consecutive layer. Sometimes it is useful to add shortcut or skip connections that directly connect the outputs of neurons in one layer to the inputs of neurons in a non-consecutive layer [81,82].

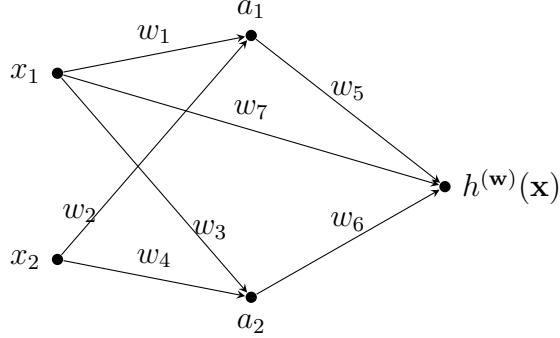


Fig. 58. An ANN can be represented as a weighted DAG with nodes representing neurons or features of a data point. Features can be viewed as trivial neurons without input and fixed output given by the feature value. The weighted directed edges indicate how neuron outputs are used as inputs to other neurons. The edge weights are tunable model parameters and are used to scale the inputs to the neurons. The output of some neurons are used as the ensemble $h^{(\mathbf{w})}(\mathbf{x})$.

See also: label, aktivointifunktio, layer, deep net.

attention Some koneoppiminen applications involve data points composed of smaller units, referred to as tokens. For example, a sentence consists of words, an image of pixel patches, and a network of nodes. In general, the tokens that constitute a single data point are not independent of one another. Instead, each token of a data point depends on (or pays attention to) specific other tokens. Todennäköisyyssmallit provide a principled framework for representing and analyzing such dependencies [83]. Attention mechanisms use a more direct approach without explicit reference to a todennäköisyyssmalli. The idea is to repre-

sent the relationship between two tokens i and i' using a parameterized funktion $f^{(\mathbf{w})}(i, i')$, where the parameters \mathbf{w} are learned via a variant of ERM. Practical attention mechanisms differ in their precise choice of attention model $f^{(\mathbf{w})}(i, i')$ as well as in the precise ERM variant used to learn the parameters \mathbf{w} . One widely used family of attention mechanisms defines the parameters \mathbf{w} in terms of two vektorit associated with each token i , i.e., a query vektori $\mathbf{q}^{(i)}$ and a key vektori $\mathbf{k}^{(i')}$. For a given token i with query $\mathbf{q}^{(i)}$, and another token i' with key $\mathbf{k}^{(i')}$, the quantity $(\mathbf{q}^{(i)})^\top \mathbf{k}^{(i')}$ indicates the extent to which token i attends to (or depends on) token i' (see Fig. 59).

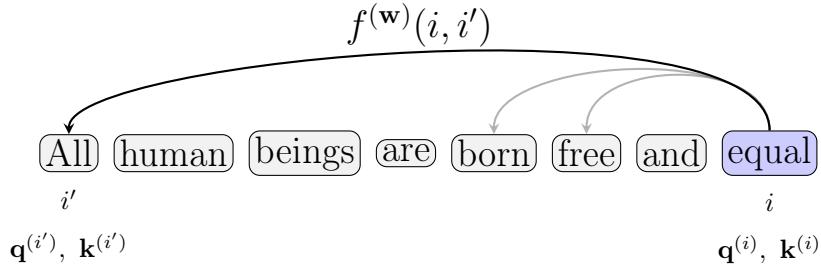


Fig. 59. Attention mechanisms learn a parameterized funktion $f^{(\mathbf{w})}(i, i')$ to measure how much token i attends to token i' . One widely used construction of $f^{(\mathbf{w})}(i, i')$ uses query and key vektorit, denoted by $\mathbf{q}^{(i)}$ and $\mathbf{k}^{(i)}$, assigned to each token i [84].

See also: token, funktion, natural language processing (NLP).

autoenkoodaja An autoencoder is an koneoppiminen method that simultaneously learns an encoder kuvaus $h \in \mathcal{H}$ and a decoder kuvaus $h^* \in \mathcal{H}^*$. Different autoencoders use different models $\mathcal{H}, \mathcal{H}^*$, e.g., ANNs with

different architectures. The special case of an autoencoder using (vektori-valued) lineaariset mallit for $\mathcal{H}, \mathcal{H}^*$ results in PCA.

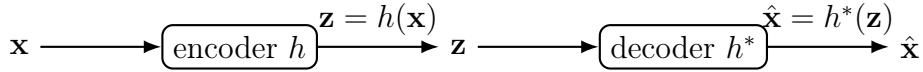


Fig. 60. Autoencoder with an encoder h mapping $\mathbf{x} \mapsto \mathbf{z}$ and a decoder h^* mapping $\mathbf{z} \mapsto \hat{\mathbf{x}}$.

The training of the encoder and decoder can be implemented via ERM using a loss that measures the deviation of the reconstructed feature vector $h^*(h(\mathbf{x}))$ from the original feature vector \mathbf{x} .

See also: encoder, piirreoptimointi, ulottuvuuksien vähentäminen.

backdoor A backdoor hyökkäys refers to the intentional manipulation of an koneoppiminen training process. The attacker might perturb the training set (i.e., through data poisoning) or the optimointimenetelmä used by an ERM-based method. The goal of a backdoor hyökkäys is to nudge the learned hypothesis \hat{h} toward specific ennusteet for a certain subset $\mathcal{T} \subset \mathcal{X}$ of the feature space. Any feature vector $\mathbf{x} \in \mathcal{T}$ serves as a key (or trigger) to unlock a backdoor in the sense of delivering anomalous ennusteet. The trigger pattern \mathcal{T} and corresponding anomalous ennuste $\hat{h}(\mathbf{x})$, for $\mathbf{x} \in \mathcal{T}$, are only known to the attacker.

See also: hyökkäys, data poisoning.

backpropagation Backpropagation is an algoritmi for computing the gradientti $\nabla_{\mathbf{w}} f(\mathbf{w})$ of an objective function $f(\mathbf{w})$ that depends on the model parameters \mathbf{w} of an ANN. One example of such an objective

function is the average loss incurred by the ANN on a batch of data points. This algorithm is a direct application of the chain rule from calculus to efficiently compute partial derivatives of the loss function with respect to the model parameters. Backpropagation consists of two consecutive phases, also illustrated in Fig. 61. The first phase includes the forward pass, where a batch of data points is fed into the ANN. The ANN processes the input through its layers using its current weights, ultimately producing an estimate at its output. The estimate of the batch is compared to the true label using a loss function, which quantifies the estimate error. The second phase includes the backward pass (i.e., backpropagation), where the error is propagated through the ANN layers. The obtained partial derivatives with respect to the ANN parameters w_1, \dots, w_d constitute the gradient $\nabla f(\mathbf{w})$, which can be used, in turn, to implement a gradient-based optimization algorithm.

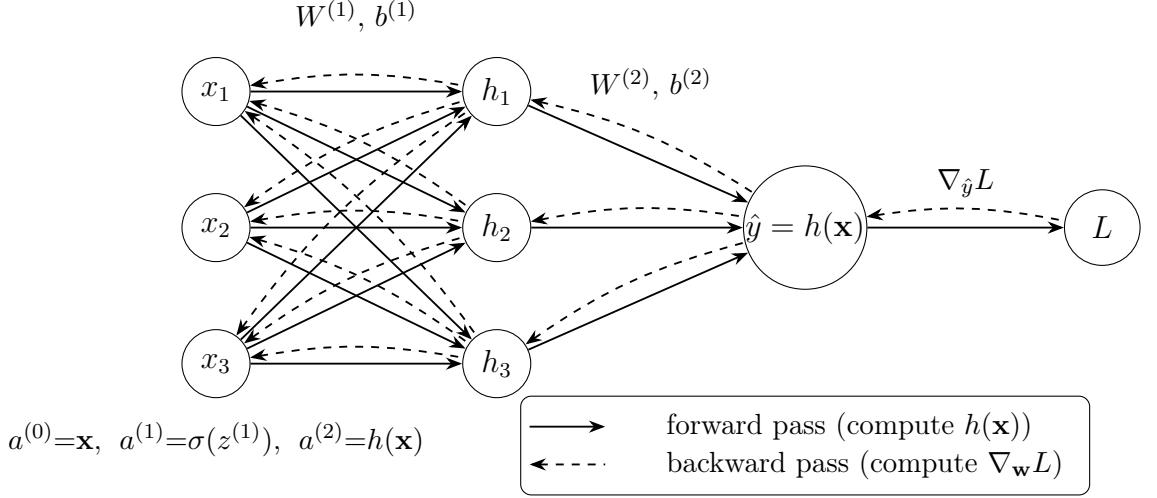


Fig. 61. Solid arrows show the forward pass (i.e., data flow and loss calculation), while dashed arrows show the gradienti correction flow during the backward pass for updating the parameters $W^{(x)}, b^{(x)}$.

See also: ANN, loss function, GD, optimointimenetelmä.

bagging Bagging is an ensemble technique where base learners use perturbed copies

$$\tilde{\mathcal{D}}^{(1)}, \dots, \tilde{\mathcal{D}}^{(M)}$$

of the original training set \mathcal{D} [85]. Each base learner delivers a potentially different hypothesis,

$$\hat{h}^{(1)}, \dots, \hat{h}^{(M)}.$$

The hypothesis delivered by the overall method is obtained by aggregating the hypotheses $\hat{h}^{(1)}, \dots, \hat{h}^{(M)}$ using some aggregation rule. For luokittelu methods, the rule is typically a majority vote, while for regressio methods, it amounts to averaging.

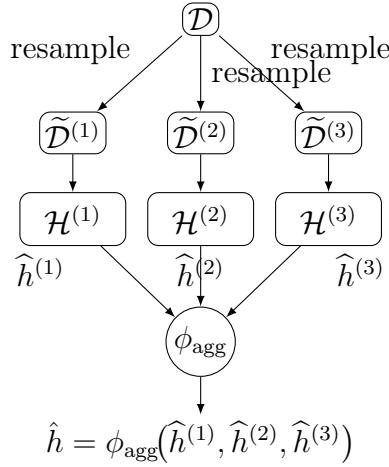


Fig. 62. An example of bagging where three base learners use perturbations $\tilde{\mathcal{D}}^{(1)}, \dots, \tilde{\mathcal{D}}^{(3)}$ of the original training set \mathcal{D} to learn the hypotheses $\hat{h}^{(1)}, \dots, \hat{h}^{(3)}$. The final hypothesis is obtained by aggregating these individual hypotheses via some aggregation rule ϕ_{agg} .

See also: ensemble, vakaus, uusio-otanta.

base learner A base learner is an koneoppiminen method that is part of an ensemble method.

See also: ensemble, bagging, stacking, boosting.

batch In the context of 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 gradientti of training error and, in turn, to update the model parameters.

See also: SGD, training set, data point, gradientti, training error, model parameter.

batch learning In batch learning (also known as offline learning), the ko-neoppiminen model is trained on the entire dataset in a single training iteration, instead of updating it incrementally as data arrive. All available data are inputted into a learning algoritmi, resulting in a model that can make ennusteet. Since these datasets tend to be large, training is computationally expensive and time-consuming, so it is typically performed offline. After learning, the model will be static and will not adapt to new data automatically. Updating the model with new information requires retraining the model entirely. Once the model has been trained, it is launched into production where it cannot be updated. Training a model can take many hours, so many models in production settings are updated cyclically on a periodic schedule when the data distribution is stable. For example, a retail analytics team could retrain their demand forecast model every Sunday using the previous week's sales data to predict next week's demand. If a system needs to be constantly updated to rapidly changing data, such as in stock price ennuste, a more adaptable solution such as online learning is necessary.

See also: batch, model, dataset, online learning.

Bayes estimator Consider a todennäköisyysmalli with a joint probability distribution $p(\mathbf{x}, y)$ over the features \mathbf{x} and the 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 riski $\mathbb{E}\{L((\mathbf{x}, y), h)\}$ is the minimi achievable riski [57]. Note that whether a hypothesis qualifies as a Bayes estimator depends on the underlying probability distribution and the choice for the loss function $L(\cdot, \cdot)$.

See also: todennäköisyysmalli, hypothesis, riski.

Bayes risk Consider a todennäköisyysmalli for a koneoppiminen application where each data point is interpreted as a RV. The todennäköisyysmalli includes a probability distribution for the features \mathbf{x} and label y of a data point. The Bayes riski is the minimi possible riski that can be achieved by any hypothesis $h : \mathcal{X} \rightarrow \mathcal{Y}$. Any hypothesis that achieves the Bayes riski is referred to as a Bayes estimator [57].

See also: todennäköisyysmalli, riski, Bayes estimator.

binary classification Binary luokittelu is luokittelu with two labels. The labels are usually defined as $\{-1, 1\}$ or $\{0, 1\}$.

See also: luokittelu, label, data point, feature.

binary cross-entropy (BCE) The BCE loss is the special case of cross-entropy loss for a binary luokittelu problem.

See also: cross-entropy, luokittelu.

binääritappio The 0/1 loss $L^{(0/1)}((\mathbf{x}, y), h)$ measures the quality of a luokittin $h(\mathbf{x})$ that delivers a ennuste \hat{y} (e.g., via thresholding (23)) for the label y of a data point with features \mathbf{x} . It is equal to 0 if the ennuste is correct, i.e., $L^{(0/1)}((\mathbf{x}, y), h) = 0$ when $\hat{y} = y$. It is equal to 1 if the ennuste is wrong, i.e., $L^{(0/1)}((\mathbf{x}, y), h) = 1$ when $\hat{y} \neq y$.

See also: loss, luokittin, ennuste, tarkkuus, label, data point, feature.

boosting Boosting is an iterative optimointimenetelmä to learn an accurate hypothesis kuvaus (or strong learner) by sequentially combining less accurate base learners (referred to as weak learners) [74], [75], [76], [65,

Ch. 10]. Boosting can be understood as a generalization of gradient-based methods for ERM using parametric models and smooth loss function [88]. In particular, starting from an initialization \tilde{h} , boosting methods construct a sequence of hypotheses $\tilde{h}^{(t)}$, $t = 1, \dots$, via a generalized gradienttiaskel

$$\tilde{h}^{(t)} = \tilde{h}^{(t-1)} + \eta^{(t)} \hat{h}^{(t)}.$$

Here, $\eta^{(t)}$ denotes a learning rate and $\hat{h}^{(t)}$ is provided by the t th base learner. Comparing the above update with the plain gradienttiaskel suggests that we view $\hat{h}^{(t)}$ as a (negative) generalized gradientti. Boosting methods differ in their choice of base learners for computing the generalized gradientit $\hat{h}^{(t)}$.

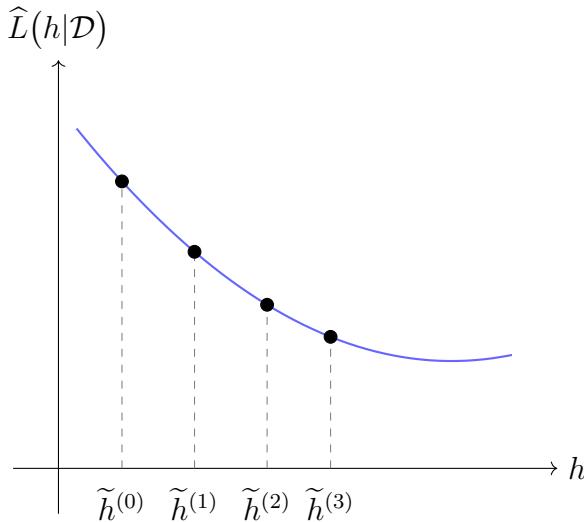


Fig. 63. Boosting methods construct a sequence of hypothesis kuvaukset via a generalized gradienttiaskel. This generalized gradienttiaskel uses the output of base learners.

See also: ensemble, adaptive boosting (AdaBoost), gradient boosting.

bootstrap aggregation See bagging.

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. An example of such clusters is shown in Fig. 64.

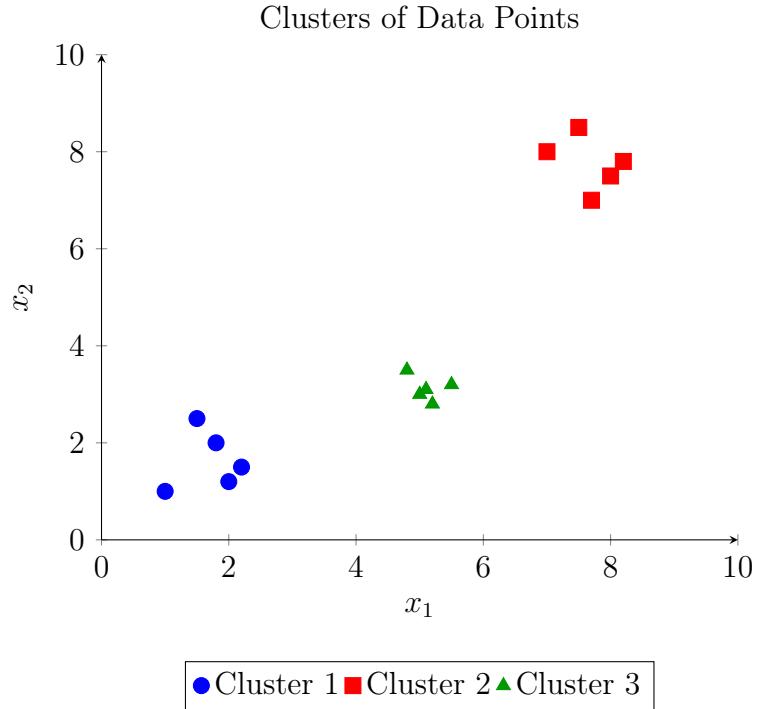


Fig. 64. Illustration of three clusters in a 2-D feature space. Each cluster groups data points that are more similar to each other than to those in other clusters, based on the Euclidean distance.

See also: data point, feature vector, Euclidean distance, feature space.

cluster centroid Clustering methods decompose a given dataset into few clusters. Different clustering methods use different representations for these clusters. If data points are characterized by numerical feature vectors $\mathbf{x} \in \mathbb{R}^d$, we can use some vector $\boldsymbol{\mu} \in \mathbb{R}^d$, referred to as cluster centroid, to represent a cluster. For example, if a cluster consists of a set of data points, we use the average of their feature vectors as a cluster centroid. However, there are also other choices for how to construct a

cluster centroid.

See also: clustering, feature vector, k -means.

clustered federated learning (CFL) CFL trains local models for the devices in a federated learning (FL) application by using a clustering assumption, i.e., the devices of an 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 assumption, FL network, cluster, graph clustering.

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 of a cluster (i.e., the cluster keskiarvo) as its representative. A popular soft clustering method based on GMM represents a cluster by a moninormaalijakauma.

See also: cluster, k -means, soft clustering, GMM.

clustering assumption The clustering assumption postulates that data

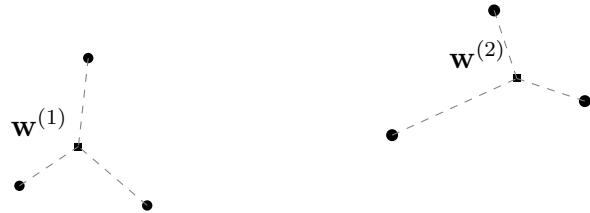


Fig. 65. For data point with numeric feature vectors, we can use the average squared Euclidean distance to the nearest cluster centroids as a measure of the clustering error.

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 [90]. We obtain different clustering methods by using different notions of similarity between data points.

See also: clustering, data point, dataset, cluster.

clustering error Consider a clustering method that decomposes a given dataset into clusters. The clustering error is a quantitative measure of the usefulness of the clusters. Different clustering methods use different choices for the clustering error. For example, the hard clustering method k -means measures the clustering error via the average squared Euclidean distance between the feature vector of a data point and the nearest cluster centroid (see Figure 65). Another construction for the clustering error can be based on a todennäköisyysmalli such as the GMM where the cluster centroids are parameters of the underlying probability distribution.

See also: GMM, k -means, todennäköisyysmalli, maximum likelihood.

concept activation vector (CAV) Consider a deep net, consisting of several hidden layers, trained to predict the label of a data point from its feature vector. One way to explain the behavior of the trained deep net is by using the activations of a hidden layer as a new feature vector \mathbf{z} . We then probe the geometry of the resulting new feature space by applying the deep net to data points that represent a specific concept \mathcal{C} . By applying the deep net also to data points that do not belong to this concept, we can train a binary linear classifier $g(\mathbf{z})$ that distinguishes between concept and non-concept data points based on the activations of the hidden layer. The resulting decision boundary is a hyperplane whose normal vector is the CAV [91] for the concept \mathcal{C} .

See also: deep net, lineaarinen malli, luotettava tekoäly, tulkittavuus, transparency.

confusion matrix Consider a finite dataset with m data points, each characterized by a feature vector \mathbf{x} and a label $y \in \mathcal{Y}$ with a finite label space $\mathcal{Y} = \{1, \dots, k\}$. For a given hypothesis h , the confusion matriisi is a $k \times k$ matriisi where each row corresponds to a specific value of the true label $y \in \mathcal{Y}$ and each column to a specific value of the ennuste $h(\mathbf{x}) \in \mathcal{Y}$. The matriisi entry in the c th row and c' th column is the number of data points with the true label $y = c$ that are predicted as $h(\mathbf{x}) = c'$. The sum of the main diagonal entries is the number of correctly classified data points, i.e., those for which $y = h(\mathbf{x})$. Summing the off-diagonal entries results in the total number of data points that are misclassified by h .

See also: label, label space, hypothesis, matriisi, luokittelu.

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

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

Here, $\|\mathbf{u}\|_p := (\sum_{j=1}^d |u_j|^p)^{1/p}$ denotes the p -normi (for $p \geq 1$). It turns out that many of the optimal vektorit $\widehat{\mathbf{w}}^{(1)}, \dots, \widehat{\mathbf{w}}^{(m)}$ coincide. A cluster then consists of those data points $r \in \{1, \dots, m\}$ with identical $\widehat{\mathbf{w}}^{(r)}$ [92], [93].

See also: dataset, convex, clustering, vektori, normi, cluster, data point.

coreset A coresset is a small subset of a larger dataset that approximates certain properties of the original dataset [94]. The construction of a coresset typically involves selecting representative data points and assigning them weights to reflect their importance in the original dataset (Fig. 66).

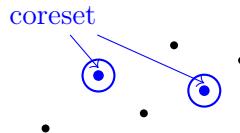


Fig. 66. A coresset (highlighted in blue) is a small subset of a larger dataset.

Coresets are particularly useful for koneoppiminen applications (such as clustering) involving large datasets, as they allow for efficient computation while preserving the essential characteristics of the data [95].

See also: dataset, data point, clustering.

covariance The covariance between two real-valued RVs x and y , defined on a common probability space, measures their linear dependence. It is defined as

$$\text{cov}(x, y) = \mathbb{E}\{(x - \mathbb{E}\{x\})(y - \mathbb{E}\{y\})\}.$$

A positive covariance indicates that x and y tend to increase together, while a negative covariance suggests that one tends to increase as the other decreases. If $\text{cov}(x, y) = 0$, the RVs are said to be uncorrelated, though not necessarily statistically independent. See Fig. 67 for visual illustrations.

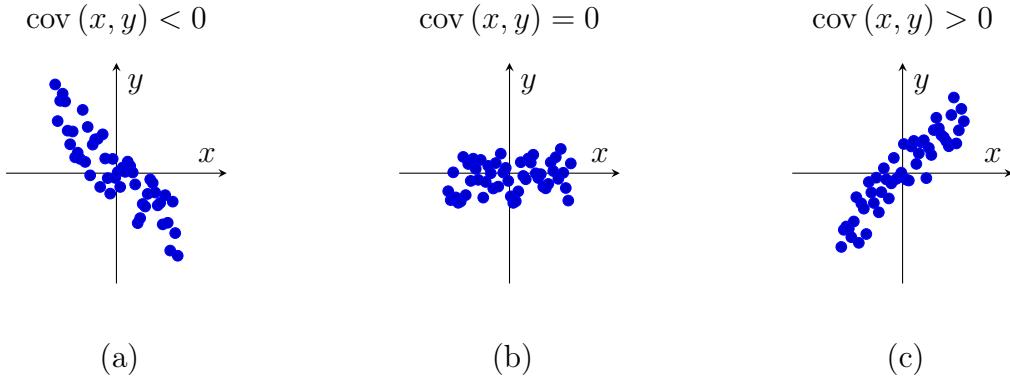


Fig. 67. Scatterplots illustrating realizations from three different todennäköisyysmallit for two RVs with different covariance values. (a) Negative. (b) Zero. (c) Positive.

See also: todennäköisyysmalli, expectation.

cross-entropy Consider a multi-class luokittelu problem with a feature space \mathcal{X} and a finite label space $\mathcal{Y} = \{1, \dots, k\}$. A data point with a feature vector \mathbf{x} is represented as a pmf $\mathbf{y} = (y_1, \dots, y_k)^T$ over \mathcal{Y} , where y_c

denotes the todennäköisyys that a randomly chosen data point with feature vector \mathbf{x} has label c . A hypothesis $h(\mathbf{x})$ outputs a predicted pmf $\hat{\mathbf{y}} = (\hat{y}_1, \dots, \hat{y}_k)^T$. The associated cross-entropy loss is [31]

$$L((\mathbf{x}, \hat{\mathbf{y}}), h) := - \sum_{c=1}^k y_c \log \hat{y}_c.$$

The cross-entropy loss quantifies the dissimilarity between the true pmf \mathbf{y} and the predicted pmf $\hat{\mathbf{y}}$. It is also a measure of the expected number of bits required to encode labels drawn from the true pmf \mathbf{y} when using a coding scheme optimized for the predicted pmf $\hat{\mathbf{y}}$ [31].

Note that, for binary luokittelu (with $k = 2$), the cross-entropy loss reduces to the logistic loss when employing a parametric model with model parameters \mathbf{w} such that $\hat{y}_2/\hat{y}_1 = \exp(\mathbf{w}^T \mathbf{x})$. Moreover, the representation (22) of logistic loss requires encoding the label space $\{1, 2\}$ using the values -1 and 1 .

See also: luokittelu, pmf, logistic loss.

data In the context of koneoppiminen, the term data is often used as a synonym for dataset [69], [70]. The ISO/IEC 2382:2015 standard defines data as a "reinterpretable representation of information in a formalized manner suitable for communication, interpretation, or processing" [96].

See also: dataset, data point, sample.

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. 68). Data augmentation can be an efficient form of regularization.

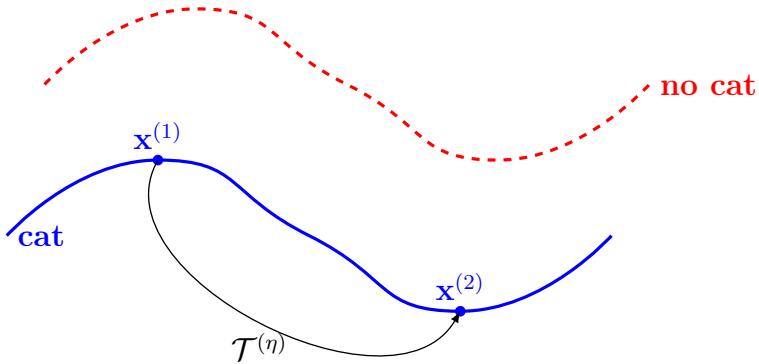


Fig. 68. Data augmentation exploits intrinsic symmetries of data points in some feature space \mathcal{X} . We can represent a symmetry by an operator $\mathcal{T}^{(\eta)} : \mathcal{X} \rightarrow \mathcal{X}$, parameterized by some number $\eta \in \mathbb{R}$. For example, $\mathcal{T}^{(\eta)}$ might represent the effect of rotating a cat image by η degrees. A data point with feature vector $\mathbf{x}^{(2)} = \mathcal{T}^{(\eta)}(\mathbf{x}^{(1)})$ must have the same label $y^{(2)} = y^{(1)}$ as a data point with feature vector $\mathbf{x}^{(1)}$.

See also: data, data point, label, feature vector, stacking, regularization, feature space.

data imputation See missing data.

data matrix The term data matriisi is sometimes used as a synonym for the feature matrix $\mathbf{X} \in \mathbb{R}^{m \times d}$ of a dataset containing m data points,

each characterized by a feature vector $\mathbf{x} \in \mathbb{R}^d$ [97]. In particular, when no label information is available, the term data matriisi highlights that the feature matrix fully characterizes the dataset.

See also: feature matrix, dataset, data point, feature vector.

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 [98, Article 5(1)(c)], [99].

See also: data.

data normalization Data normalization refers to transformations applied to the feature vectors of data points to improve the koneoppiminen method's laskennallinen ominaisuus or laskennallinen ominaisuus. For example, in linear regression with gradient-based methods using a fixed learning rate, convergence depends on controlling the normi of feature vectors in the training set. A common approach is to normalize feature vectors such that their normi does not exceed one [8, Ch. 5].

See also: gradient-based method, learning rate, feature map.

data parallelism Data parallelism refers to a widely used class of distributed optimointimenetelmät for training an koneoppiminen model. Here, each participating device stores a full replica of the model parameters but processes a disjoint subset of the global dataset [100]. Compared to model parallelism, which distributes the model parameters across devices, data

parallelism distributes the computational workload associated with large datasets. It is especially effective when the model fits into the memory of a single device, but the dataset or training complexity would be prohibitive without parallel processing.

See also: model parallelism, horizontal federated learning (HFL).

data point A data point is any object that conveys information [31]. Examples include students, radio signals, trees, images, RVs, real numbers, or proteins. We describe data points of the same type by two categories of properties. The first category includes features that are measurable or computable properties of a data point. These attributes can be automatically extracted or computed using sensors, computers, or other data collection systems. For a data point that represents a patient, one feature could be the body weight. The second category includes labels that are higher level facts (or quantities of interest)—that is, facts that typically require human expertise or domain knowledge to determine rather than being directly measurable—associated with the data point. For a data point that represents a patient, a cancer diagnosis provided by a physician would serve as the label. Fig. 69 depicts an image as an example of a data point along with its features and labels. Importantly, what constitutes a feature or a label is not inherent to the data point itself—it is a design choice that depends on the specific application.



A single data point.

Features:

- x_1, \dots, x_d : Color intensities of all image pixels.
- x_{d+1} : Time-stamp of the image capture.
- x_{d+2} : Spatial location of the image capture.

Labels:

- y_1 : Number of cows depicted.
- y_2 : Number of wolves depicted.
- y_3 : Condition of the pasture (e.g., healthy, overgrazed).

Fig. 69. Illustration of a data point consisting of an image. We can use different properties of the image as features and higher level facts about the image as labels.

The distinction between features and labels is not always clear-cut. A property that is considered a label in one setting (e.g., a cancer diagnosis) may be treated as a feature in another setting—particularly if reliable automation (e.g., via image analysis) allows it to be computed without human intervention. Koneoppiminen broadly aims to predict the label of a data point based on its features.

See also: data, feature, label, dataset.

data poisoning Data poisoning refers to the intentional manipulation (or fabrication) of data points to maliciously steer the training of an koneoppiminen model [101], [102]. Data poisoning hyökkäykset take various forms, including backdoor and denial-of-service attackt. A backdoor hyökkäys implants triggers into training data, so that the trained model behaves normally for typical data points but misclassifies a data point with a feature vector that contains a trigger pattern. A denial-of-service attack degrades the trained model's overall performance by injecting mislabeled or adversarial examples to prevent effective learning. Data poisoning is particularly harmful in decentralized or distributed koneoppiminen settings (such as FL), where training data cannot be centrally verified.

See also: hyökkäys, backdoor, denial-of-service attack, luotettava tekoäly.

dataset A dataset is a set of distinct data points. Strictly speaking, a dataset is an unordered collection of data points that does not contain any repetitions. However, in koneoppiminen literature, the term dataset is

often used as a synonym for sample, i.e., a sequence (or finite list) of data points that may contain repetitions. *koneoppiminen* methods use datasets for model training and validation. The notion of a dataset is broad, i.e., data points may represent concrete physical entities (such as humans or animals) or abstract objects (such as numbers). For illustration, Fig. 70 depicts a dataset whose data points are cows.



Fig. 70. A cow herd somewhere in the Alps.

Quite often, an *koneoppiminen* engineer does not have direct access to the underlying dataset. For instance, accessing the dataset in Fig. 70 would require visiting the cow herd. In practice, we work with a more convenient representation (or approximation) of the dataset. Various mathematical models have been developed for this purpose [103], [104], [105], [106]. One of the most widely used is the relational model, which organizes data as a table (or relation) [103], [107]. A table consists of rows and columns, where each row corresponds to a single data point, and each column represents a specific attribute of a data point. *koneoppiminen* methods typically interpret these attributes as features or as a label of a data point. As an illustration, Table I shows a relational representation of the dataset from Fig. 70. In the relational

model, the order of rows is immaterial, and each attribute (i.e., column) is associated with a domain that specifies the set of admissible values. In koneoppiminen applications, these attribute domains correspond to the feature space and the label space.

TABLE I
A RELATION (OR TABLE) THAT REPRESENTS THE DATASET IN FIG. 70

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

While the relational model is useful for the study of many koneoppiminen applications, it may be insufficient regarding the requirements for luotettava tekoäly. Modern approaches like datasheets for datasets provide more comprehensive documentation, including details about the data collection process, intended use, and other contextual information [108].

See also: data point, sample, data, feature, feature space, label space.

decision boundary Consider a hypothesis kuvaus 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 vektorit $\mathbf{x} \in \mathbb{R}^d$ that lie between different decision regions. More precisely, a vektori \mathbf{x} belongs to the decision boundary if and only if each naapurusto $\{\mathbf{x}' : \|\mathbf{x} - \mathbf{x}'\| \leq \varepsilon\}$, for any $\varepsilon > 0$, contains at least two vektorit with different funkto values.

See also: hypothesis, kuvaus, feature vector, vektori, decision region, naapurusto, funktio.

decision region Consider a hypothesis kuvaus h that delivers values from a finite set \mathcal{Y} . For each label value (i.e., 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 .

See also: hypothesis, kuvaus, label, feature.

decision tree A decision tree is a flowchart-like representation of a hypothesis kuvaus h . More formally, a decision tree is a directed verkko 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 child 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 child nodes itself, 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 without any children. See Fig. 71 for visual illustrations.

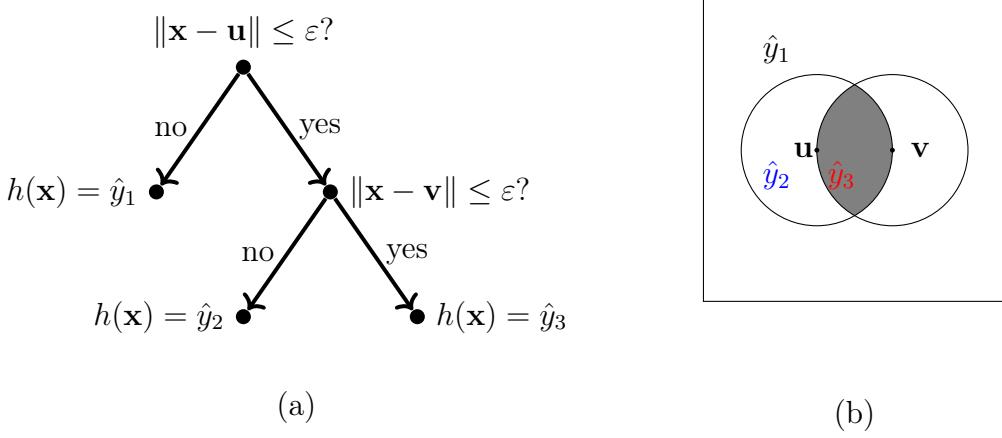


Fig. 71. (a) A decision tree is a flowchart-like representation of a piecewise constant hypothesis $h : \mathcal{X} \rightarrow \mathbb{R}$. Each piece is a decision region $\mathcal{R}_{\hat{y}} := \{\mathbf{x} \in \mathcal{X} : h(\mathbf{x}) = \hat{y}\}$. The depicted decision tree can be applied to numeric feature vectors, i.e., $\mathcal{X} \subseteq \mathbb{R}^d$. It is parameterized by the threshold $\varepsilon > 0$ and the vector $\mathbf{u}, \mathbf{v} \in \mathbb{R}^d$. (b) 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.

See also: decision region.

deep net A deep net is an ANN with a (relatively) large number of hidden layers. Deep learning is an umbrella term for komeoppiminen methods that use a deep net as their model [81].

See also: ANN, layer, model, large language model (LLM).

degree of belonging Degree of belonging is a number that indicates the extent to which a data point belongs to a cluster [8, Ch. 8]. The degree of belonging can be interpreted as a soft cluster assignment. Soft clustering

methods can encode the degree of belonging with 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.

See also: data point, cluster, soft clustering, hard clustering.

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

See also: hyökkäys, data poisoning, model, data point.

denoising autoencoder A denoising autoenkoodaaja extends the basic autoenkoodaaja by perturbing (e.g., by adding noise to) the input vector (or feature vector) before feeding it into the autoenkoodaaja itself [81, Sec. 14.2.2]. Once trained, we can use a denoising autoenkoodaaja to denoise a corrupted representation of a data point (see Fig. 72).



Fig. 72. A denoising autoenkoodaaja reads in (a) a corrupted (i.e., noisy) representation of a data point and computes a ennuste for (b) the clean representation.

See also: autoenkoodaaja, diffusion method.

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, DBSCAN also 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 (i.e., path) of nearby 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.

See also: clustering, k -means, GMM, cluster, verkko.

design matrix The term design matriisi is a synonym for the feature matrix, particularly used in statistics [70], [97]. It collects the feature vectors of the data points in a dataset that is used for model training or validation.

See also: feature matrix, feature vectors, data point, dataset.

device A physical system that can store and process data. In the context of koneoppiminen, the term typically refers to a computer capable of reading data points from different sources and using them to train an koneoppiminen model [109].

See also: data, koneoppiminen, data point, model.

diagnosis Consider an ERM-based method that resulted in a trained model (or learned hypothesis) $\hat{h} \in \mathcal{H}$. We can diagnose the method by comparing the training error E_t with the validation error E_v incurred by \hat{h} on

the training set and the validation set.

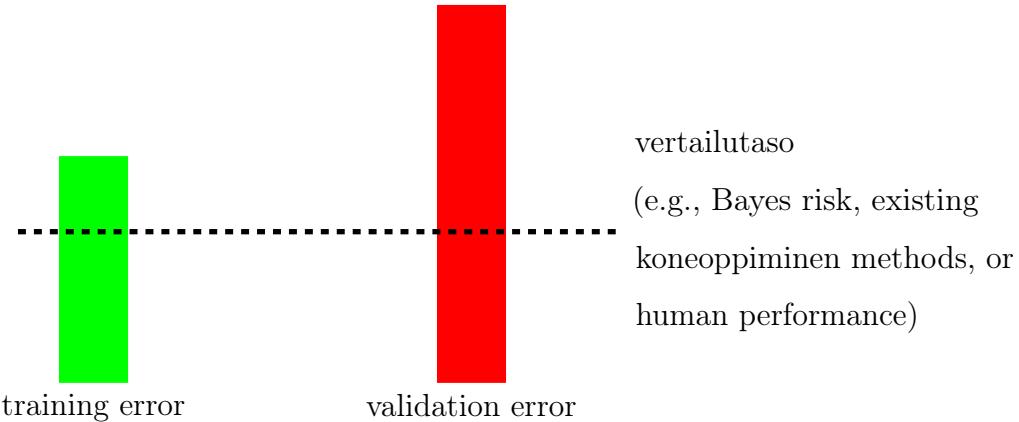


Fig. 73. We can diagnose an ERM-based koneoppiminen method by comparing its training error with its validation error. Ideally, both are on the same level as a vertailutaso.

See also: vertailutaso, validation, k -fold cross-validation (k -fold CV), generalization.

differentiaalinen yksityisyys Consider some koneoppiminen 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 ennusteet for specific data points. DP is a precise measure of privacy leakage incurred by revealing the output. Roughly speaking, an koneoppiminen method is differentially private if the probability distribution of the output $\mathcal{A}(\mathcal{D})$ remains largely unchanged when the sensitive attribute of one data point in the training set is changed.

Note that DP builds on a todennäköisyysmalli for an koneoppiminen 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 (i.e., adding noise) to the output of the koneoppiminen method.

See also: output, privacy leakage, sensitive attribute, yksityisyyskyökkäys, privacy funnel.

diffusion method A diffusion method is a generative koneoppiminen method that trains a model to approximate the reversal of a gradual stokastinen corruption process [110], [111]. Diffusion methods train models using a combination of a forward (diffusion) process and a learned reverse kuvaus. In the forward process, random noise is incrementally added to clean representations of a data point, such as an image, an audio recording, or other types of data. Similar to a denoising autoencoder, the trained model can be applied to a noisy representation of a data point. The resulting ennuste is a denoised representation. What sets diffusion methods apart from generic denoising autoencoders is the gradual nature of the corruption process used for model training.

See also: denoising autoencoder, training.

early stopping Consider an ERM-based method that uses an iterative optimointimenetelmä (such as GD) to learn model parameters by minimizing the empiirinen riski on a given training set. Early stopping refers to terminating the iterations even if they still substantially decrease the empiirinen riski on the training set. Instead of monitoring the objective

function (which is the empiirinen riski on the training set), early stopping monitors the validation error incurred by the model parameters in each iteration. Early stopping can be interpreted as an implementation of regularization via model pruning. Indeed, terminating an iterative optimointimenetelmä after a small number of iterations restricts the set of model parameters that can be reached from the initialization (see Fig. 74).

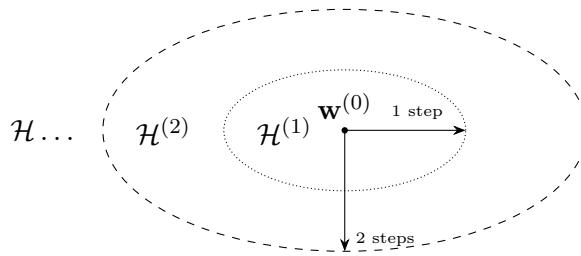


Fig. 74. A gradient-based method for ERM using a hypothesis space \mathcal{H} defines a nested sequence of effective hypothesis spaces $\mathcal{H}^{(1)} \subseteq \mathcal{H}^{(2)} \subseteq \dots \subseteq \mathcal{H}$. The effective hypothesis space $\mathcal{H}^{(t)}$ is determined by all model parameters that can be reached from the initialization $\mathbf{w}^{(0)}$ within t gradienttiaskeleet.

See also: regularization, gradient-based method, ylisovittaminen.

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

See also: FL network.

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 lineaarikuvaus or the weights and harha terms of an ANN.

See also: hypothesis space, model parameter, ANN.

eigenvector An eigenvector of a matriisi $\mathbf{A} \in \mathbb{R}^{d \times d}$ is a nonzero vektori $\mathbf{x} \in \mathbb{R}^d \setminus \{\mathbf{0}\}$ such that $\mathbf{Ax} = \lambda\mathbf{x}$ with some eigenvalue λ .

See also: matriisi, vektori, eigenvalue.

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

See also: riski, hypothesis, dataset, loss, data point.

empirical risk minimization (ERM) ERM is the optimointitehtävä of selecting a hypothesis $\hat{h} \in \mathcal{H}$ that minimizes the average loss (or empiirinen riski) on a training set \mathcal{D} . The hypothesis is chosen from a hypothesis space (or model) \mathcal{H} . The dataset \mathcal{D} is referred to as training set. A plethora of ERM-based koneoppiminen methods is obtained for different design choices for the dataset, model, and loss [8, Ch. 3]. Fig. 75 illustrates ERM for a lineaarinen malli and data points that are characterized by a single feature x and a label y . The hypothesis h is a lineaarikuvaus that predicts the label of a data point as a linear funktio of its feature x , i.e., $h(x) = w_1x + w_0$, where w_1 and w_0 are the model parameters of the hypothesis h . The ERM problem is to find the model parameters w_1 and w_0 that minimize the average loss (or empiirinen riski) incurred by the hypothesis h on the training set \mathcal{D} .

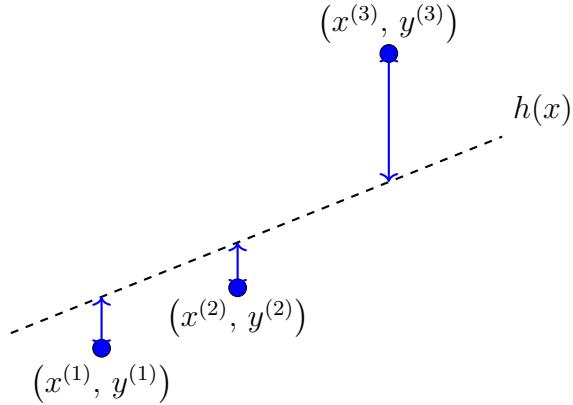


Fig. 75. ERM learns a hypothesis $h \in \mathcal{H}$ out of a model \mathcal{H} by minimizing the average loss (or empiirinen riski) $1/m \sum_{r=1}^m L((\mathbf{x}^{(r)}, y^{(r)}), h)$ incurred on a training set \mathcal{D} .

See also: optimointitehtävä, empiirinen riski, training set, loss, optimointimenetelmä.

encoder See autoenkoodaja.

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

See also: koneoppiminen, hypothesis, kuvaus, feature, data point, label.

ensemble An ensemble method combines multiple koneoppiminen methods, each of those referred to as a base learner, to improve overall performance. The base learners can be ERM-based using different choices for the loss,

model, and training set. By aggregating the ennusteet of base learners, ensemble methods can often achieve better performance than any single base learner. The aggregation can amount to averaging the ennusteet of base learners (in regressio) or using a majority vote (for luokittelu methods).

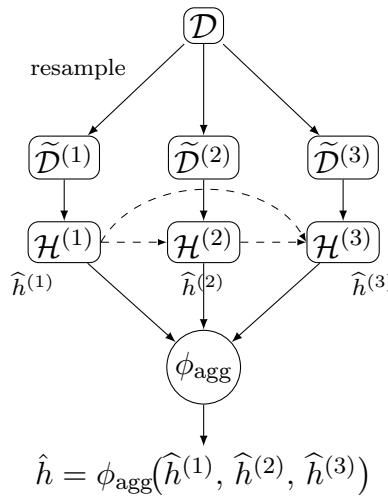


Fig. 76. A generic ensemble with three base learners, each using ERM to learn $\mathcal{H}^{(j)} \in \mathcal{H}^{(j)}$ based on the training set $\tilde{\mathcal{D}}^{(j)}$. A base learner might also use the output of other base learners. The final hypothesis \hat{h} is obtained by aggregating the hypotheses generated by the base learners.

Different ensemble methods use different constructions for the base learners. For example, bagging methods (such as a random forest) use random sampling to construct slightly different training sets for each base learner. On the other hand, boosting methods run the base learners sequentially, i.e., each base learner tries to correct the ennuste errors of

the previous ones. A third family of ensemble methods is stacking, where base learners are trained on the same training set but with different models.

See also: bagging, boosting, stacking.

epoch An epoch represents one complete pass of the entire training set through some learning algoritmi. It refers to the point at which a model has processed every data point in the training set once. Training a model usually requires multiple epochs, since each iteration allows the model to refine the parameters and improve ennusteet. The number of epochs is something predefined by the user, and thus a hyperparameter, which plays a crucial role in determining how the model will generalize to unseen data. Too few epochs will result in alisovittaminen, while too many epochs can result in ylisovittaminen.

See also: training set, algoritmi, model, data point, parameter, ennuste, alisovittaminen, ylisovittaminen.

epävarmuus In the context of koneoppiminen, uncertainty refers to the presence of multiple plausible outcomes or selityst based on available data. For example, the ennuste $\hat{h}(\mathbf{x})$ produced by a trained koneoppiminen model \hat{h} often reflects a range of possible values for the true label of a given data point. The broader this range, the greater the associated uncertainty. Todennäköisyys theory allows us to represent, quantify, and reason about uncertainty in a mathematically rigorous manner.

See also: todennäköisyysmalli, riski, entropia, varianssi.

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 a koneoppiminen method that learns a hypothesis \hat{h} , e.g., using ERM, is defined as $\hat{h}(\mathbf{x}) - \bar{h}(\mathbf{x})$, for some feature vector. For a parametric hypothesis space, which consists of hypothesis kuvauskset determined by model parameters \mathbf{w} , we can define the estimation error as $\Delta\mathbf{w} = \hat{\mathbf{w}} - \bar{\mathbf{w}}$ [65], [114].

See also: data point, feature vector, label, hypothesis, koneoppiminen, ERM, hypothesis space, kuvaus, model parameter.

Euclidean distance The term Euclidean distance is used as a synonym for the Euclidean norm.

See also: vektori, Euclidean space.

Euclidean norm The Euclidean normi $\|\mathbf{w}\|_2$ of a vektori

$$\mathbf{w} = (w_1, \dots, w_d) \in \mathbb{R}^d$$

is defined as

$$\|\mathbf{w}\|_2 := \sqrt{\sum_{j=1}^d w_j^2}.$$

The Euclidean normi is distinct among all normit on \mathbb{R}^d in the sense that it is induced by the inner product $\mathbf{w}^T \mathbf{v}$ [1], [23], [115]. In other words, $\|\mathbf{w}\|_2 = \sqrt{\mathbf{w}^T \mathbf{w}}$.

See also: normi, vektori, Euclidean space.

Euclidean space The Euclidean space \mathbb{R}^d of dimension $d \in \mathbb{N}$ consists of vektorit $\mathbf{x} = (x_1, \dots, x_d)$, with d real-valued entries $x_1, \dots, x_d \in \mathbb{R}$. Such a 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 vektorit $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^d$ [2].

See also: vektori.

expectation Consider a numeric feature vector $\mathbf{x} \in \mathbb{R}^d$ that we interpret as the realization of an RV with probability distribution $p(\mathbf{x})$. The expectation of \mathbf{x} is defined as the integral $\mathbb{E}\{\mathbf{x}\} := \int \mathbf{x} p(\mathbf{x})$. Note that the expectation is only defined if this integral exists, i.e., if the RV is integrable [2], [6], [56]. Fig. 77 illustrates the expectation of a scalar discrete RV x that takes on values from a finite set only.

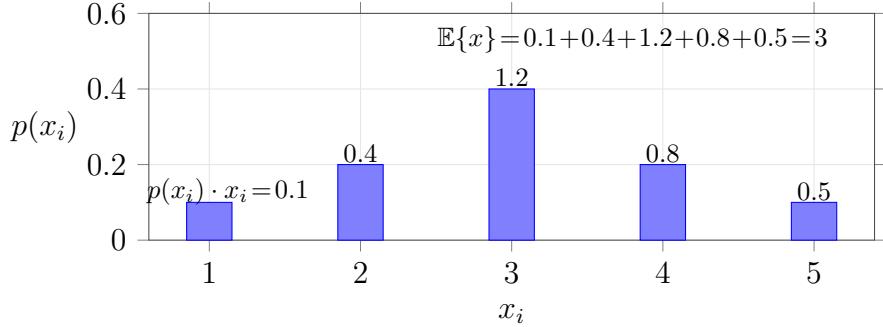


Fig. 77. The expectation of a discrete RV x is obtained by summing its possible values x_i , weighted by the corresponding todennäköisyys $p(x_i) = \mathbb{P}(x = x_i)$.

See also: feature vector, realization, RV, probability distribution, todennäköisyys.

expert koneoppiminen aims to learn a hypothesis h that accurately predicts

the label of a data point based on its features. We measure the ennuste 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 i.i.d. assumption and by using the Bayes risk as the vertailutaso for the (average) loss of a hypothesis. An alternative approach to obtaining a vertailutaso is to use the hypothesis h' learned by an existing koneoppiminen method. We refer to this hypothesis h' as an expert [116]. Regret minimization methods learn a hypothesis that incurs a loss comparable to the best expert [116], [117].

See also: loss function, vertailutaso, regret.

explainable empirical risk minimization (EERM) EERM 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 kuaukset that are intrinsically explainable for a specific user. This user is characterized by their ennusteet provided for the data points in a training set [119].

See also: SRM, regularization, ERM, training set.

explainable machine learning (XML) XML methods aim to complement each ennuste with an selitys of how the ennuste has been obtained. The construction of an explicit selitys might not be necessary if the koneoppiminen method uses a sufficiently simple (or interpretable) model [122].

See also: ennuste, selitys, koneoppiminen, model.

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 (RGB) intensities of its pixels as features. Another example is shown in Fig. 78, where the signal samples of a finite-duration audio signal are used as its features.

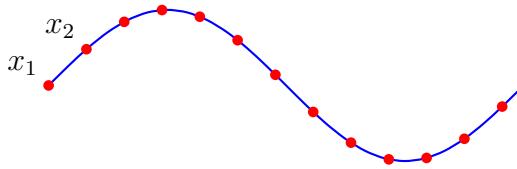


Fig. 78. An audio signal (blue waveform) and its discretized signal samples (red dots) that can be used as its features x_1, \dots, x_d .

Domain-specific synonyms for the term feature are "covariate," "explanatory variable," "independent variable," "input (variable)," "predictor (variable)," or "regressor" [69], [125], [126].

See also: data point.

feature map A feature kuvaus refers to a funktion

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

that transforms a feature vector $\mathbf{x} \in \mathcal{X}$ of a data point into a new feature vector $\mathbf{x}' \in \mathcal{X}'$, where \mathcal{X}' is typically different from \mathcal{X} . The transformed representation \mathbf{x}' is often more useful than the original \mathbf{x} . For instance, the geometry of data points may become more linear in \mathcal{X}' , allowing the application of a lineaarinen malli to \mathbf{x}' . This idea is central to the design of kernel methods [62]. Other benefits of using a feature kuvaus include reducing ylisovittaminen and improving tulkittavuus [127]. A

common use case is data visualization, where a feature kuvaus with two output dimensions allows the representation of data points in a 2-D scatterplot. Some koneoppiminen methods employ trainable feature kuvaaukset, whose parameters are learned from data. An example is the use of hidden layers in a deep net, which act as successive feature kuvaaukset [128]. A principled way to train a feature kuvaus is through ERM, using a loss function that measures reconstruction quality, e.g., $L = \|\mathbf{x} - r(\mathbf{x}')\|^2$, where $r(\cdot)$ is a trainable kuvaus that attempts to reconstruct \mathbf{x} from the transformed feature vector \mathbf{x}' .

See also: feature, kuvaus, kernel method, piirreoptimointi, PCA.

feature matrix Consider a dataset \mathcal{D} with m data points with feature vectors $\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)} \in \mathbb{R}^d$. It is convenient to collect the individual feature vectors into the feature matriisi

$$\mathbf{X} := (\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)})^T = \begin{pmatrix} x_1^{(1)} & x_2^{(1)} & \cdots & x_d^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \cdots & x_d^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{(m)} & x_2^{(m)} & \cdots & x_d^{(m)} \end{pmatrix} \in \mathbb{R}^{m \times d}.$$

Note that the feature matrix is of size $m \times d$, i.e., it has m rows and d columns.

See also: dataset, data point, feature vector, feature, matriisi.

feature space The feature space of a given koneoppiminen application or method is constituted by all potential values that the feature vector of a data point can take on. For data points described by a fixed number d of numerical features, a common choice for the feature space

is the Euclidean space \mathbb{R}^d . However, the mere presence of d numeric features does not imply that \mathbb{R}^d is the most appropriate representation of the feature space. Indeed, the numerical features might be assigned to data points in a largely arbitrary or random manner, resulting in data points that are randomly scattered throughout \mathbb{R}^d without any meaningful geometric structure. Piirreoptimointi methods try to learn a transformation of the original (potentially non-numeric) features to ensure a more meaningful arrangement of data points in \mathbb{R}^d . Three examples of feature spaces are shown in Fig. 79.

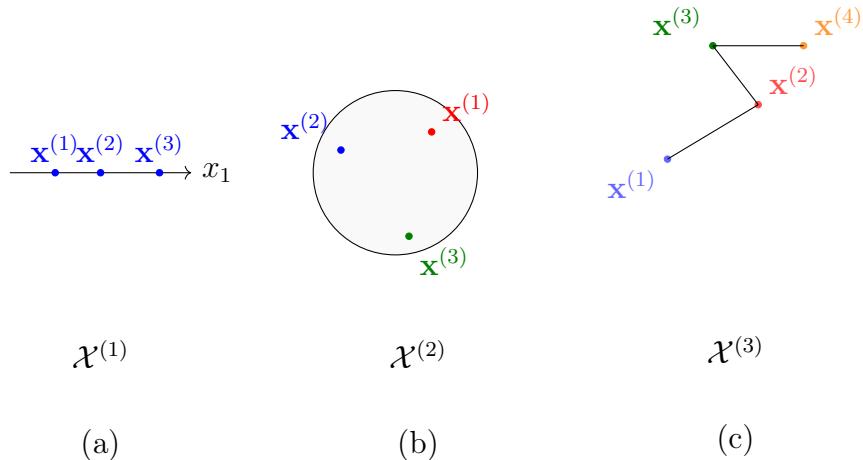


Fig. 79. Three different feature spaces. (a) A linear space $\mathcal{X}^{(1)} = \mathbb{R}$. (b) A bounded convex set $\mathcal{X}^{(2)} \subseteq \mathbb{R}^2$. (c) A discrete space $\mathcal{X}^{(3)}$ whose elements are nodes of an undirected verkko.

See also: feature vector, Euclidean space.

feature vector Feature vektori refers to a vektori $\mathbf{x} = (x_1, \dots, x_d)^T$ whose entries are individual features x_1, \dots, x_d . Many koneoppiminen methods

use feature vektorit that belong to some finite-dimensional Euclidean space \mathbb{R}^d . For some koneoppiminen methods, however, it can be more convenient to work with feature vektorit that belong to an infinite-dimensional vektoriavaruus (e.g., see kernel method).

See also: feature, vektori, koneoppiminen, Euclidean space, vektoriavaruus.

federated averaging (FedAvg) FedAvg refers to a family of iterative FL algoritmit. It uses a server-client setting and alternates between clientwise local models retraining, followed by the aggregation of updated model parameters at the server [129]. The local update at client $i = 1, \dots, n$ at time t starts from the current model parameters $\mathbf{w}^{(t)}$ provided by the server and typically amounts to executing few iterations of SGD. After completing the local updates, they are aggregated by the server (e.g., by averaging them). Fig. 80 illustrates the execution of a single iteration of FedAvg.

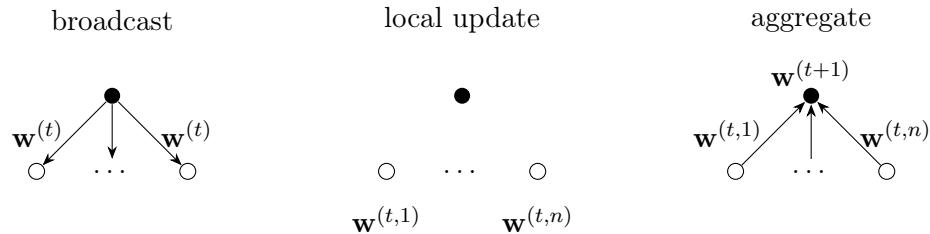


Fig. 80. Illustration of a single iteration of FedAvg, which consists of broadcasting model parameters by the server, performing local updates at clients, and aggregating the updates by the server.

See also: FL, algoritmi, local model, SGD.

federated gradient descent (FedGD) An FL hajautettu algoritmi that can be implemented as message passing across an FL network.

See also: FL, hajautettu algoritmi, FL network, gradienttiaskel, gradient-based method.

federated learning (FL) FL is an umbrella term for koneoppiminen met-hods that train models in a collaborative fashion using decentralized data and computation.

See also: koneoppiminen, model, data.

federated learning network (FL network) An FL network consists of an undirected weighted verkko \mathcal{G} . The nodes of \mathcal{G} represent devices that can access a local dataset and train a local model. The edges of \mathcal{G} represent communication links between devices as well as statistical similarities between their local datasets. A principled approach to train the local models is GTVMin. The solutions of GTVMin are local model parameters that optimally balance the loss incurred on local datasets with their poikkeama across the edges of \mathcal{G} .

See also: FL, verkko, device, GTVMin.

federated proximal (FedProx) FedProx refers to an iterative FL algorit-mi 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 [130].

See also: FL, algoritmi, local model, model parameter, FedAvg, SGD, proximal operator.

federated relaxed (FedRelax) An FL hajautettu algoritmi.

See also: FL, hajautettu algoritmi.

federated stochastic gradient descent (FedSGD) An FL hajautettu algoritmi that can be implemented as message passing across an FL network.

See also: FL, hajautettu algoritmi, FL network, gradienttiaskel, gradient-based method, SGD.

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

See also: data.

flow-based clustering Flow-based clustering groups the nodes of an undirected verkko by applying k -means clustering to nodewise feature vectors. These feature vectors are built from network flows between carefully selected sources and destination nodes [131].

See also: clustering, verkko, k -means, feature vector.

Gaussian mixture model (GMM) A GMM is a particular type of todennäköisyysmalli for data points characterized by a numeric feature vector \mathbf{x} . Within a GMM, the feature vector \mathbf{x} is drawn from a randomly selected moninormaalijakauma $p^{(c)} = \mathcal{N}(\boldsymbol{\mu}^{(c)}, \mathbf{C}^{(c)})$ with $c = I$. The index $I \in \{1, \dots, k\}$ is a RV with todennäköisyydet $\mathbb{P}(I = c) = p_c$. A GMM is parameterized, for each $c = 1, \dots, k$, by the todennäköisyys

p_c , the keskiarvo vektori $\mu^{(c)}$, and the covariance matrix $\mathbf{C}^{(c)}$.

See also: todennäköisyysmalli, moninormaalijakauma, clustering.

generalization Generalization refers to the ability of a model trained on a training set to make accurate ennusteet on new unseen data points. This is a central goal of koneoppiminen and AI, i.e., to learn patterns that extend beyond the training set. Most koneoppiminen systems use ERM to learn a hypothesis $\hat{h} \in \mathcal{H}$ by minimizing the average loss over a training set of data points $\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(m)}$, which is denoted by $\mathcal{D}^{(\text{train})}$. However, success on the training set does not guarantee success on unseen data—this discrepancy is the challenge of generalization. To study generalization mathematically, we need to formalize the notion of “unseen” data. A widely used approach is to assume a todennäköisyysmalli for data generation, such as the i.i.d. assumption. Here, we interpret data points as independent RVs with an identical probability distribution $p(\mathbf{z})$. This probability distribution, which is assumed fixed but unknown, allows us to define the riski of a trained model \hat{h} as the expected loss

$$\bar{L}(\hat{h}) = \mathbb{E}_{\mathbf{z} \sim p(\mathbf{z})} \{ L(\hat{h}, \mathbf{z}) \}.$$

The difference between riski $\bar{L}(\hat{h})$ and empiirinen riski $\widehat{L}(\hat{h}|\mathcal{D}^{(\text{train})})$ is known as the generalization gap. Tools from todennäköisyys theory, such as concentration inequalities and uniform convergence, allow us to bound this gap under certain conditions [132].

Generalization without todennäköisyys: Todennäköisyys theory is one way to study how well a model generalizes beyond the training set, but it is not the only way. Another option is to use simple deterministic

changes to the data points in the training set. The basic idea is that a good model \hat{h} should be robust, i.e., its ennuste $\hat{h}(\mathbf{x})$ should not change much if we slightly change the features \mathbf{x} of a data point \mathbf{z} . For example, an object detector trained on smartphone photos should still detect the object if a few random pixels are masked [133]. Similarly, it should deliver the same result if we rotate the object in the image [128]. See Fig. 81 for a visual illustration.

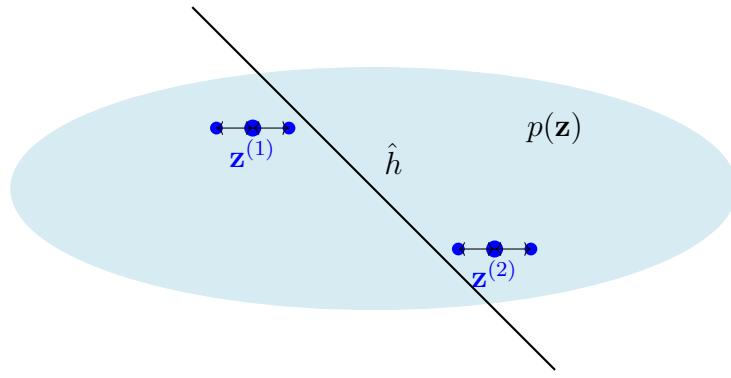


Fig. 81. Two data points $\mathbf{z}^{(1)}, \mathbf{z}^{(2)}$ that are used as a training set to learn a hypothesis \hat{h} via ERM. We can evaluate \hat{h} outside $\mathcal{D}^{(\text{train})}$ either by an i.i.d. assumption with some underlying probability distribution $p(\mathbf{z})$ or by perturbing the data points.

See also: ERM, i.i.d. assumption, ylisovittainen, validation.

generalization gap Generalization gap is the difference between the performance of a hypothesis $h \in \mathcal{H}$ on the training set $\mathcal{D}^{(\text{train})}$ and its performance on data points outside $\mathcal{D}^{(\text{train})}$. We can make this notion precise by using a todennäköisyysmalli that allows us to compute the riski (or expected loss) $\bar{L}(\hat{h})$ of a hypothesis h .

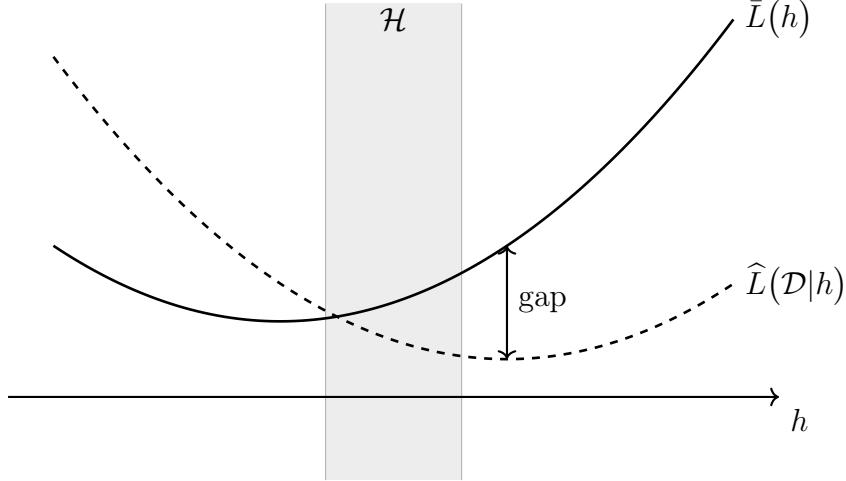


Fig. 82. The generalization gap can be defined as the difference between the riski $\bar{L}(h)$ and the average loss (or empiirinen riski) $\hat{L}(h|\mathcal{D}^{(\text{train})})$ computed on a training set.

In practice, the probability distribution underlying this expectation is unknown. Thus, we need to estimate the expectation based on observed data points. Validation techniques use different constructions of a validation set, which is different from the training set, to estimate the generalization gap.

See also: generalization, validation, ERM, loss function.

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 $i = 1, \dots, n$ of an undirected weighted verkko \mathcal{G} with edges \mathcal{E} . Given a measure $d^{(h,h')}$ for the poikkeama between hypothesis

kuvaaukset 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 $\{i, i'\} \in \mathcal{E}$.

See also: local model, model parameter, verkko, poikkeama, hypothesis, kuvaus.

generalized total variation minimization (GTVMin) GTVMin is an instance of regularized empirical risk minimization (RERM) using the generalized total variation (GTV) of local model parameters as a regularizer [134].

See also: RERM, GTV, regularizer.

geometrinen mediaani The GM of a set of input vectors $\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}$ in \mathbb{R}^d is a point $\mathbf{z} \in \mathbb{R}^d$ that minimizes the sum of distances to the vektorit [23] such that

$$\mathbf{z} \in \arg \min_{\mathbf{y} \in \mathbb{R}^d} \sum_{r=1}^m \|\mathbf{y} - \mathbf{x}^{(r)}\|_2. \quad (16)$$

Fig. 83 illustrates a fundamental property of the GM: If \mathbf{z} does not coincide with any of the input vectors, then the unit vektorit pointing from \mathbf{z} to each $\mathbf{x}^{(r)}$ must sum to zero—this is the zero-subgradient (optimality) condition for (16). It turns out that the solution to (16) cannot be arbitrarily pulled away from trustworthy input vectors as long as they are the majority [135, Th. 2.2].

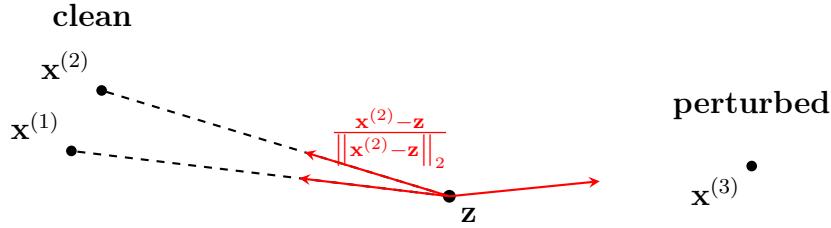


Fig. 83. Consider a solution \mathbf{z} of (16) that does not coincide with any of the input vectors. The optimality condition for (16) requires that the unit vector from \mathbf{z} to the input vectors sum to zero.

See also: vektori, subgradient.

gradient boosting Gradientti boosting is a boosting algoritmi that learns a hypothesis \tilde{h} by sequentially combining the hypotheses $\hat{h}^{(t)}$ [65, Algorithm 10.3], [88]. Similar to AdaBoost, gradientti boosting uses a generalized gradienttiaskel to combine the results of the base learners

$$\tilde{h}^{(t)} = \tilde{h}^{(t-1)} - \eta^{(t)} \hat{h}^{(t)}$$

where the generalized gradientti \hat{h}^t is constructed from the t th base learner. The difference between AdaBoost and gradientti boosting is in the construction of \hat{h}^t . While AdaBoost uses weighted ERM for this construction, gradientti boosting uses ERM on a modified training set. This modification is obtained by leaving the feature vectors untouched but replacing the labels with the partial derivative of the loss function with respect to the ennusteet of the previous base learner.

See also: boosting, AdaBoost, GD.

gradient descent (GD) GD is an iterative method for finding the minimi of a differentiable funkto $f : \mathbb{R}^d \rightarrow \mathbb{R}$. GD generates a sequence of estimates $\mathbf{w}^{(0)}, \mathbf{w}^{(1)}, \mathbf{w}^{(2)}, \dots$ that (ideally) converge to a minimi of f . At each iteration k , GD refines the current estimate $\mathbf{w}^{(k)}$ by taking a step in the direction of the steepest descent of a local linear approximation. This direction is given by the negative gradientti $\nabla f(\mathbf{w}^{(k)})$ of the funkto f at the current estimate $\mathbf{w}^{(k)}$. The resulting update rule is given by

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} - \eta \nabla f(\mathbf{w}^{(k)}) \quad (17)$$

where $\eta > 0$ is a suitably small step size. For a suitably choosen step size η , the update typically reduces the funkto value, i.e., $f(\mathbf{w}^{(k+1)}) < f(\mathbf{w}^{(k)})$. Fig. 84 illustrates a single GD step.

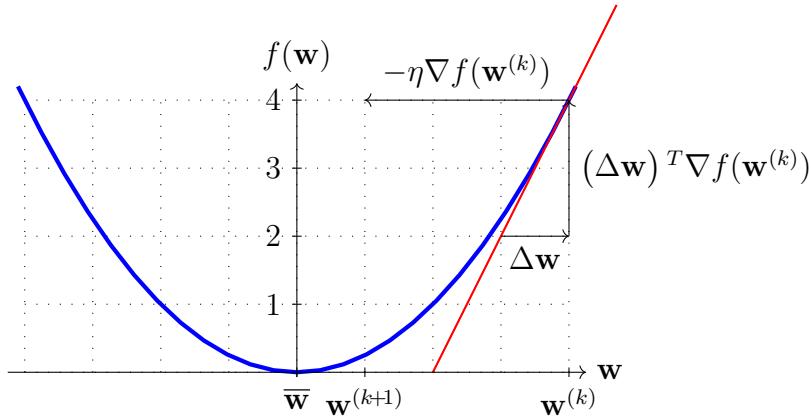


Fig. 84. A single gradienttiaskel (17) toward the minimizer $\bar{\mathbf{w}}$ of $f(\mathbf{w})$.

See also: minimi, differentiable, gradientti, step size, gradienttiaskel.

gradient-based method A gradientti-based method is an iterative technique for finding the minimi (or maksimi) of a differentiable objective

function $f(\mathbf{w})$ of the model parameters \mathbf{w} . Such a method constructs a sequence of approximations to an optimal choice for \mathbf{w} . As the name indicates, a gradientti-based method uses the gradientit of the objective function evaluated during previous iterations to construct new, (hopefully) improved model parameters. One important example of a gradientti-based method is GD.

See also: gradientti, differentiable, objective function, optimointimene-telma, GD.

graph clustering Verkko clustering aims to cluster data points that are represented as the nodes of a verkko \mathcal{G} . The edges of \mathcal{G} represent pairwise similarities between data points. We can sometimes quantify the extent of these similarities by an edge weight [131], [136].

See also: verkko, clustering, data point, edge weight.

hajautettu algoritmi A distributed algoritmi is an algoritmi designed for a special type of computer, i.e., a collection of interconnected computing devices (or nodes). These devices communicate and coordinate their local computations by exchanging messages over a network [112], [113]. Unlike a classical algoritmi, which is implemented on a single device, a distributed algoritmi is executed concurrently on multiple devices with computational capabilities. Similar to a classical algoritmi, a distributed algoritmi can be modeled as a set of potential executions. However, each execution in the distributed setting involves both local computations

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

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

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

See also: algoritmi, device, event, FL, model.

hard clustering Hard clustering refers to the task of partitioning a given set of data points into (a few) non-overlapping clusters. This requirement allows to represent a cluster by a subset of data points, i.e., precisely those belonging to the cluster. In contrast to hard clustering, soft clustering methods allow for overlapping clusters and specify, for each data point, a numeric degree of belonging to each cluster. Hard clustering is an extreme case of soft clustering where the degrees of belonging take only two values, indicating either no belonging or full belonging. For data points characterized by numeric feature vectors $\mathbf{x} \in \mathbb{R}^d$, a widely used hard clustering method is k -means. Any hard clustering method for numeric feature vectors $\mathbf{x} \in \mathbb{R}^d$ can be adapted for non-numerical data using piirreoptimointi methods. One important example of this approach is spectral clustering, where data points have a similarity structure in the

form of an undirected verkko. The nodes of this verkko represent data points while undirected (possibly weighted) edges represent similarities (and their extend) between data points. We can then use the entries of the eigenvectors of the Laplacian matrix as numeric features for each data point.

See also: clustering, data point, cluster, k -means.

harha Consider an ERM-based koneoppiminen method that learns a hypothesis $\hat{h} \in \mathcal{H}$ from a given training set. The analysis of the koneoppiminen method is often based on a todennäköisyyssmalli (such as the i.i.d. assumption) for the data generation. Here, data points and, in turn, the learned hypothesis \hat{h} are viewed as (realizations of) RVs. Any property $\theta(\hat{h})$ of \hat{h} , such as specific model parameters in a parametric model or the ennuste error $y - \hat{h}(\mathbf{x})$ for a fixed data point, then also becomes an RV. The squared bias of a numeric property $\theta(\hat{h}) \in \mathbb{R}^r$ is [38], [65]

$$B^2 := \|\mathbb{E}\{\theta(\hat{h})\} - \theta(\bar{h})\|_2^2.$$

Here, \bar{h} is a reference hypothesis, which could be defined by $\bar{h}(\mathbf{x}) = y$ for a fixed test data point with feature vector \mathbf{x} and label y .

See also: todennäköisyyssmalli, i.i.d., RV, estimation error.

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 koneoppiminen 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 todennäköisyys theory that studies the behavior of koneoppiminen methods in the high-dimensional regime [16], [137].

See also: ERM, effective dimension, ylisovittaminen, regularization.

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 kuvaus $h(\mathbf{x})$ is defined as

$$L((\mathbf{x}, y), h) := \max\{0, 1 - yh(\mathbf{x})\}. \quad (18)$$

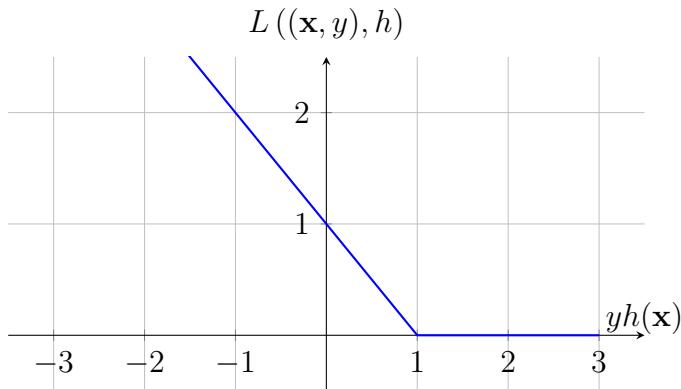


Fig. 85. The hinge loss incurred by the ennuste $h(\mathbf{x}) \in \mathbb{R}$ for a data point with label $y \in \{-1, 1\}$. A regularized variant of the hinge loss is used by the support vector machine (SVM) [138].

See also: SVM, luokittelu, luokitin.

histogrammi Consider a dataset \mathcal{D} that consists of m data points $\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(m)}$,

each of them belonging to some cell $[-U, U] \times \dots \times [-U, U] \subseteq \mathbb{R}^d$ 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. A visual example of such a histogram is provided in Fig. 86.

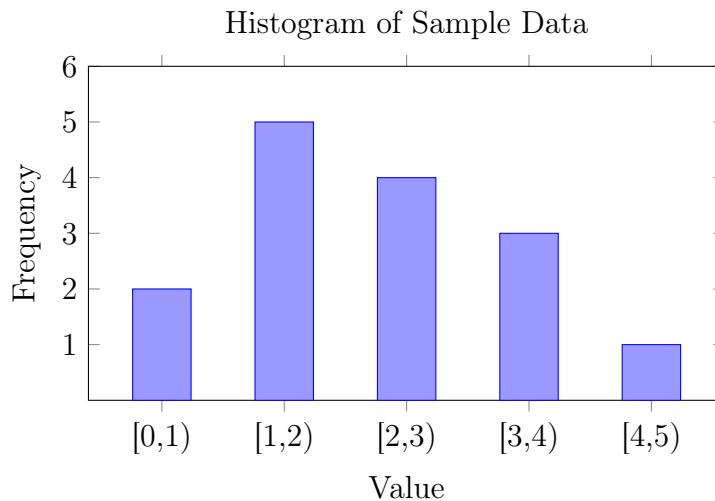


Fig. 86. A histogram consists of the fractions of data points that fall within different value ranges (i.e., bins). Each bar height shows the count of samples in the corresponding interval.

See also: dataset, data point, sample.

horizontal federated learning (HFL) HFL uses local datasets constituted by different data points but uses the same features to characterize them [139]. For example, weather forecasting uses a network of spa-

tially 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 spatiotemporal regions. Each spatiotemporal region represents an individual data point, each characterized by the same features (e.g., daily temperature or air pressure). See also: semi-supervised learning (SSL), FL, vertical federated learning (VFL).

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

See also: loss, neliövirhehäviö, absolute error loss.

Huber regression Huber regressio [140] refers to ERM-based methods that use the Huber loss as a measure of the ennuste error. Two important special cases of Huber regressio are least absolute deviation regression and linear regression. Tuning the threshold parameter of the Huber loss allows the user to trade the vakaus of the absolute error loss against the computational benefits of the smooth neliövirhehäviö.

See also: least absolute deviation regression, linear regression, absolute error loss, neliövirhehäviö.

hyperparameter A hyperparameter associated with a koneoppiminen method is a quantity that is used to select among a family of models. Typical examples include the learning rate used in a gradient-based method, the number of features used in a lineaarinen malli, or the maximum depth of a decision tree. The usefulness of a specific choice of hyperpa-

rameter can be assessed via validation. Similar to learning (or tuning) model parameters by ERM on a training set, we can learn (or tune) hyperparameters via minimizing the validation error. Thus, in a sense hyperparameters are higher-level model parameters that are learned via a higher-level form of ERM: minimize the validation error obtained for the trained model with a given hyperparameter value.

See also: model parameter, model, validation.

hypothesis A hypothesis refers to a function $h : \mathcal{X} \rightarrow \mathcal{Y}$ from the feature space \mathcal{X} to the label space \mathcal{Y} . Given a data point with features \mathbf{x} , we use a hypothesis function h to estimate (or approximate) the label y using the estimate $\hat{y} = h(\mathbf{x})$. Koneoppiminen is all about learning (or

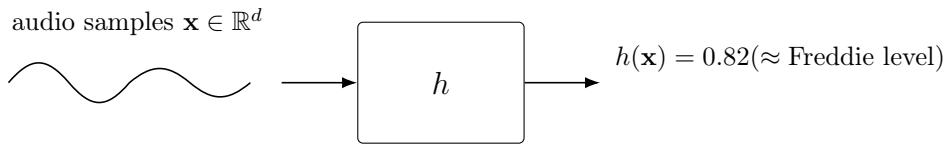


Fig. 87. A hypothesis $h : \mathcal{X} \rightarrow \mathcal{Y}$ maps the features $\mathbf{x} \in \mathcal{X}$ of a data point to a estimate $h(\mathbf{x}) \in \mathcal{Y}$ of the label. For example, the koneoppiminen application <https://freddiemeter.withyoutube.com/> uses the samples of an audio recording as features predict how closely a person's singing resembles that of Freddie Mercury.

finding) a hypothesis function h such that $y \approx h(\mathbf{x})$ for any data point (with features \mathbf{x} and label y). Practical koneoppiminen methods, limited by finite computational resources, must restrict learning to a subset of all possible hypothesis maps. This subset is called the hypothesis space or simply the model underlying the method.

See also: kuvaus, funktio, ennuste, model.

hypothesis space A hypothesis space is a mathematical model that characterizes the learning capacity of an koneoppiminen method. The goal of such a method is to learn a hypothesis kuvaus that maps features of a data point to a ennuste of its label. Given a finite amount of computational resources, a practical koneoppiminen method typically explores only a restricted set of all possible kuvaaukset from the feature space to the label space. Such a restricted set is referred to as a hypothesis space \mathcal{H} underlying the koneoppiminen method (see Fig. 88). For the analysis of a given koneoppiminen method, the choice of a hypothesis space \mathcal{H} is not unique, i.e., any superset containing all kuvaaukset the method can learn is also a valid hypothesis space.

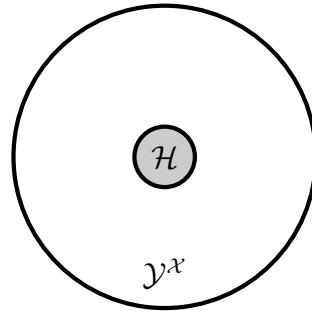


Fig. 88. The hypothesis space \mathcal{H} of an koneoppiminen method is a (typically very small) subset of the (typically very large) set $\mathcal{Y}^{\mathcal{X}}$ of all possible kuvaaukset from the feature space \mathcal{X} into the label space \mathcal{Y} .

On the other hand, from an koneoppiminen engineering perspective, the hypothesis space \mathcal{H} is a design choice for ERM-based methods. This

design choice can be guided by the available computational resources and laskennallinen ominaisuus. For instance, if efficient matriisi operations are feasible and a roughly linear relation exists between features and labels, a lineaarinen malli can be a useful choice for \mathcal{H} .

See also: hypothesis, model, kuvaus, lineaarinen malli.

hyökkäys An attack on an koneoppiminen system refers to an intentional action—either active or passive—that compromises the system’s integrity, availability, or confidentiality. Active attacks involve perturbing components such as datasets (via data poisoning) or communication links between devices within an koneoppiminen application. Passive attacks, such as yksityisyshyökkäykset, aim to infer sensitive attributes without modifying the system. Depending on their goal, we distinguish among denial-of-service attacks, backdoor attacks, and yksityisyshyökkäykset. See also: data poisoning, yksityisyshyökkäys, sensitive attribute, denial-of-service attack, backdoor.

image segmentation Image segmentation refers to the task of clustering the pixels of an image into few segments.

See also: clustering.

independent and identically distributed assumption (i.i.d. assumption)

The i.i.d. assumption is a widely used todennäköisyysmalli for the generation of data points. In particular, data points are represented as i.i.d. RVs.

See also: i.i.d., todennäköisyysmalli, data point, RV.

input vector The term input vektori is often used as a synonym for the feature vector of a data point. In settings where data points arise from a dynamical system observed over time, features are obtained from measuring input variables. These input variables are then used by koneoppiminen methods to predict the system's output (which is a label in koneoppiminen terminology).

See also: vektori, feature vector, data point, output.

iteration The elementary computational step during the execution of an algoritmi is referred to as iteration [77], [144]. For example, the elementary computational step of gradient-based methods is a gradienttiaaskel. More generally, the elementary computational step of a fixed-point iteration is the evaluation of an underlying operator \mathcal{F} , which might vary across iterations (see Fig. 89). Many important koneoppiminen algoritmit, including Lloyd's algorithm and GD, are fixed-point iterations.

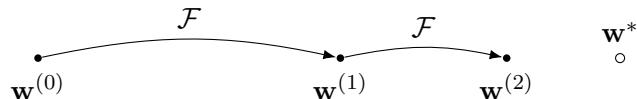


Fig. 89. A fixed-point iteration consists of the repeated application of an operator \mathcal{F} with some fixed point \mathbf{w}^* , i.e., $\mathcal{F}\mathbf{w}^* = \mathbf{w}^*$.

See also: algoritmi, gradienttiaaskel, GD.

Jacobi method The Jacobi method is an algoritmi for solving systems of linear equations (i.e., a linear system) of the form $\mathbf{Ax} = \mathbf{b}$. Here, $\mathbf{A} \in \mathbb{R}^{d \times d}$ is a square matriisi with nonzero main diagonal entries. The

method constructs a sequence $\mathbf{x}^{(0)}, \mathbf{x}^{(1)}, \dots$ by updating each entry of $\mathbf{x}^{(t)}$ according to

$$x_i^{(t+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j \neq i} a_{ij} x_j^{(t)} \right).$$

Note that all entries $x_1^{(k)}, \dots, x_d^{(k)}$ are updated simultaneously. The above iteration converges to a solution, i.e., $\lim_{t \rightarrow \infty} \mathbf{x}^{(t)} = \mathbf{x}$, under certain conditions on the matrix \mathbf{A} , e.g., being strictly diagonally dominant or symmetric positive definite [3], [32], [52]. Jacobi-type methods are appealing for large linear systems due to their parallelizable structure [113]. We can interpret the Jacobi method as a fixed-point iteration. Indeed, using the decomposition $\mathbf{A} = \mathbf{D} + \mathbf{R}$, with \mathbf{D} being the diagonal of \mathbf{A} , allows us to rewrite the linear equation $\mathbf{Ax} = \mathbf{b}$ as a fixed-point equation

$$\mathbf{x} = \underbrace{\mathbf{D}^{-1}(\mathbf{b} - \mathbf{Rx})}_{\mathcal{F}\mathbf{x}}$$

which leads to the iteration $\mathbf{x}^{(t+1)} = \mathbf{D}^{-1}(\mathbf{b} - \mathbf{Rx}^{(t)})$.

As an example, for the linear equation $\mathbf{Ax} = \mathbf{b}$, where

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

the Jacobi method updates each component of \mathbf{x} as follows:

$$\begin{aligned} x_1^{(k+1)} &= \frac{1}{a_{11}} \left(b_1 - a_{12}x_2^{(k)} - a_{13}x_3^{(k)} \right); \\ x_2^{(k+1)} &= \frac{1}{a_{22}} \left(b_2 - a_{21}x_1^{(k)} - a_{23}x_3^{(k)} \right); \\ x_3^{(k+1)} &= \frac{1}{a_{33}} \left(b_3 - a_{31}x_1^{(k)} - a_{32}x_2^{(k)} \right). \end{aligned}$$

See also: algoritmi, matriisi, fixed-point iteration, optimointimenetelmä.

k -fold cross-validation (k -fold CV) A k -fold CV is a method for evaluating the generalization gap of an ERM-based koneoppiminen method. The idea is to divide a dataset \mathcal{D} evenly into k subsets (or folds) $\mathcal{D}^{(1)}, \dots, \mathcal{D}^{(k)}$.

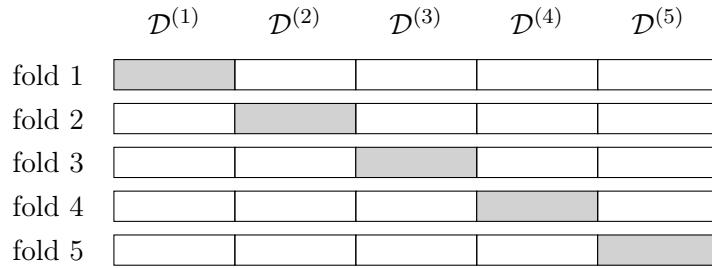


Fig. 90. In k -fold CV, the available dataset \mathcal{D} is evenly divided into k folds $\mathcal{D}^{(1)}, \dots, \mathcal{D}^{(k)}$. Each fold is used once as a validation set, while the remaining $k - 1$ folds form the training set.

For each fold $b = 1, \dots, k$, we train the model on the union of all folds except $\mathcal{D}^{(b)}$ and validate it on $\mathcal{D}^{(b)}$. The overall performance is obtained by averaging the validation results across all k folds.

See also: validation, validation error.

k -means The k -keskiarvot principle is an optimization-based approach to the clustering of data points with a numeric feature vector [8, Ch. 8]. As a hard clustering approach, k -keskiarvot partitions a dataset into k disjoint subsets (or clusters), which are indexed by $c = 1, \dots, k$. Each cluster \mathcal{C} is characterized by the average feature vector of data points

that belong to it. This average (or keskiarvo) feature vector is referred to as the cluster centroid $\mathbf{w}^{(c)}$. A visual illustration is provided in Fig. 91.

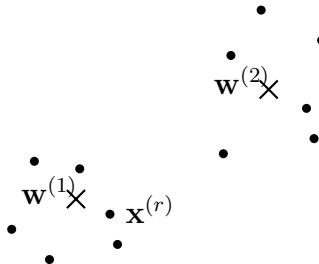


Fig. 91. A scatterplot of data points, indexed by $r = 1, \dots, m$ and characterized by feature vectors $\mathbf{x}^{(r)} \in \mathbb{R}^2$. The scatterplot also includes two cluster centroids $\mathbf{w}^{(1)}, \mathbf{w}^{(2)} \in \mathbb{R}^2$.

In general, solving the k -keskiarvot optimointitehtävä exactly is challenging (or NP-hard) [145]. However, there are simple iterative methods for finding approximately optimal cluster centroids. One such method is referred to as Lloyd's algorithm.

See also: hard clustering, cluster, Lloyd's algorithm.

k -means++ TBC.

See also: k -means.

kernel Consider a set of data points, each represented by a feature vector $\mathbf{x} \in \mathcal{X}$, where \mathcal{X} denotes the feature space. A (real-valued) kernel is a funktion $K : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ that assigns to every pair of feature vectors $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$ a real number $K(\mathbf{x}, \mathbf{x}')$. This value is typically interpreted as a similarity measure between \mathbf{x} and \mathbf{x}' . The defining property of a

kernel is that it is symmetric, i.e., $K(\mathbf{x}, \mathbf{x}') = K(\mathbf{x}', \mathbf{x})$, and that for any finite set of feature vectors $\mathbf{x}_1, \dots, \mathbf{x}_n \in \mathcal{X}$, the matrix is

$$\mathbf{K} = \begin{pmatrix} K(\mathbf{x}_1, \mathbf{x}_1) & K(\mathbf{x}_1, \mathbf{x}_2) & \dots & K(\mathbf{x}_1, \mathbf{x}_n) \\ K(\mathbf{x}_2, \mathbf{x}_1) & K(\mathbf{x}_2, \mathbf{x}_2) & \dots & K(\mathbf{x}_2, \mathbf{x}_n) \\ \vdots & \vdots & \ddots & \vdots \\ K(\mathbf{x}_n, \mathbf{x}_1) & K(\mathbf{x}_n, \mathbf{x}_2) & \dots & K(\mathbf{x}_n, \mathbf{x}_n) \end{pmatrix} \in \mathbb{R}^{n \times n}$$

is psd. A kernel naturally defines a transformation of a feature vector \mathbf{x} into a function $\mathbf{z} = K(\mathbf{x}, \cdot)$. The function \mathbf{z} maps an input $\mathbf{x}' \in \mathcal{X}$ to the value $K(\mathbf{x}, \mathbf{x}')$. We can view the function \mathbf{z} as a new feature vector that belongs to a feature space \mathcal{X}' that is typically different from \mathcal{X} . This new feature space \mathcal{X}' has a particular mathematical structure, i.e., it is a reproducing kernel Hilbert space (RKHS) [62], [138]. Since \mathbf{z} belongs to a RKHS, which is a vector space, we can interpret it as a generalized feature vector. Note that a finite-length feature vector $\mathbf{x} = (x_1, \dots, x_d)^T \in \mathbb{R}^d$ can be viewed as a function $\mathbf{x} : \{1, \dots, d\} \rightarrow \mathbb{R}$ that assigns a real value to each index $j \in \{1, \dots, d\}$.

See also: feature vector, feature space, Hilbert space, kernel method.

kernel method A kernel method is a homogeneous method that uses a kernel K to map the original (i.e., raw) feature vector \mathbf{x} of a data point to a new (transformed) feature vector $\mathbf{z} = K(\mathbf{x}, \cdot)$ [62], [138]. 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 methods.

mallit, even if the data points are not linearly separable in the original feature space (see Fig. 92).

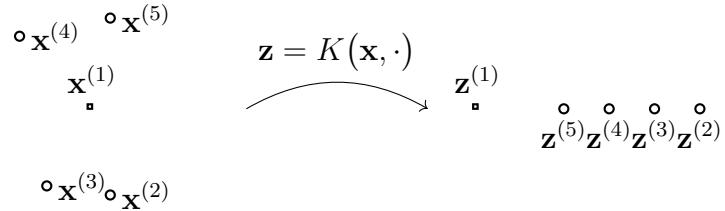


Fig. 92. Five data points characterized by feature vectors $\mathbf{x}^{(r)}$ and labels $y^{(r)} \in \{\circ, \square\}$, for $r = 1, \dots, 5$. With these feature vectors, there is no way to separate the two classes by a straight line (representing the decision boundary of a linear classifier). In contrast, the transformed feature vectors $\mathbf{z}^{(r)} = K(\mathbf{x}^{(r)}, \cdot)$ allow us to separate the data points using a linear classifier.

See also: kernel, feature vector, feature space, linear classifier.

koneoppiminen ML methods aim to learn (or find) a useful hypothesis kuvaus $\hat{h} \in \mathcal{H}$ out of a model \mathcal{H} . The learned \hat{h} is used to compute a ennuste $\hat{y} = \hat{h}(\mathbf{x})$ for the label y of a data point. The learning process is guided by a quantitative measure of the loss incurred when the ennusteet obtained from the learned hypothesis differ from the actual label y . Different ML methods use different design choices for this quantitative measure (or loss function) as well as different choices for the model and the data points (i.e., their features and labels) [8, Ch. 3].

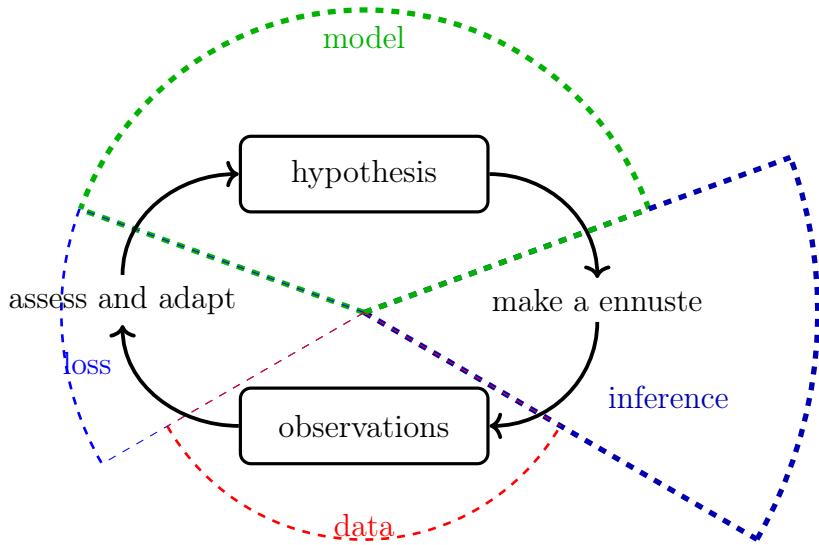


Fig. 93. ML learns a hypothesis out of a model (or hypothesis space) by trying to minimize the loss incurred by the ennusteet for the labels of data points. The ennusteet are computed solely from the features of the data points.

Another distinction between ML methods is how they access data points during learning. For example, some methods have access to a complete dataset during training, which allows them to use ERM [8], [154]. In contrast, online learning methods access data sequentially and, in turn, update the learned hypothesis whenever a new data point arrives [116], [117], [155].

See also: model, data, loss, ERM, online learning.

Kronecker product The Kronecker product of two matriisit $\mathbf{A} \in \mathbb{R}^{m \times n}$ and

B ∈ ℝ^{p×q} is a block matriisi denoted by **A** ⊗ **B** and defined as [3], [29]

$$\mathbf{A} \otimes \mathbf{B} = \begin{pmatrix} a_{11}\mathbf{B} & \cdots & a_{1n}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{m1}\mathbf{B} & \cdots & a_{mn}\mathbf{B} \end{pmatrix} \in \mathbb{R}^{mp \times nq}.$$

The Kronecker product is a special case of the tensor product for matriisit and is widely used in multivariate statistics, linear algebra, and structured koneoppiminen models. It satisfies the identity $(\mathbf{A} \otimes \mathbf{B})(\mathbf{x} \otimes \mathbf{y}) = (\mathbf{Ax}) \otimes (\mathbf{By})$ for vektorit **x** and **y** of compatible dimensions.
See also: matriisi, koneoppiminen, model, vektori.

Kullback–Leibler divergence (KL divergence) The KL divergence is a quantitative measure of how different one probability distribution is from another [31].

See also: probability distribution.

label A label is 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. Synonyms for label, commonly used in specific domains, include "response variable," "output variable," and "target" [69], [125], [126].

See also: data point, label space.

label space In an koneoppiminen application, each data point is described by a set of features together with an associated label. The set of all admissible label values is called the label space, denoted by \mathcal{Y} . Importantly, \mathcal{Y} may include values that no observed data point has as its label

value. To a large extent, the choice of \mathcal{Y} is up to the engineer and depends on the problem formulation. Fig. 94 shows some examples of label spaces that are commonly used in applications.

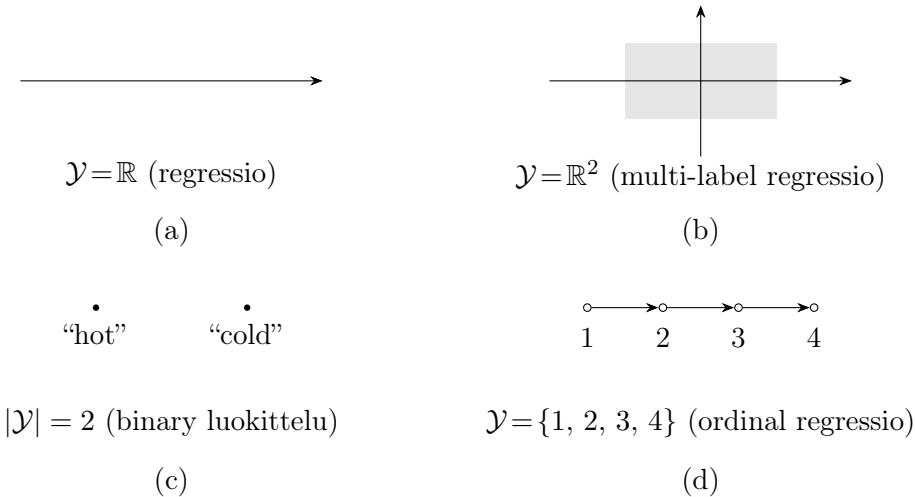


Fig. 94. Examples of label spaces and the corresponding types of regression. (a) Regressio. (b) Multi-label regressio. (c) Binary luokittelu. (d) Ordinal regressio.

The choice of the label space \mathcal{Y} determines the type of regression methods appropriate for the application at hand. Regression methods use the $\mathcal{Y} = \mathbb{R}$, while binary classification methods use a label space \mathcal{Y} that consists of two different elements, i.e., $|\mathcal{Y}| = 2$. Ordinal regression methods use a finite, ordered set of label values, e.g., $\mathcal{Y} = \{1, 2, 3, 4\}$ with the natural ordering $1 < 2 < 3 < 4$.

See also: data point, label, regression, classification.

label vector Given a dataset of m labeled data pointt

$$(\mathbf{x}^{(1)}, y^{(1)}), \dots, (\mathbf{x}^{(m)}, y^{(m)})$$

it is convenient to collect the corresponding labels into a single label vektori $\mathbf{y} := (y_1, \dots, y_m)^T$ [38], [65].

See also: dataset, labeled data point, label, data point.

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

See also: data point, label.

Laplacian matrix The structure of a verkko \mathcal{G} , with nodes $i = 1, \dots, n$, can be analyzed using the properties of special matriisit that are associated with \mathcal{G} . One such matriisi is the verkko Laplacian matriisi $\mathbf{L}^{(\mathcal{G})} \in \mathbb{R}^{n \times n}$, which is defined for an undirected and weighted verkko [136], [146]. It is defined elementwise as (see Fig. 95)

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

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

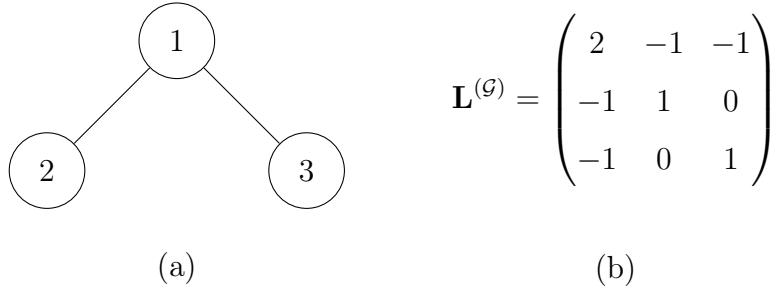


Fig. 95. (a) Some undirected verkko \mathcal{G} with three nodes $i = 1, 2, 3$. (b) The Laplacian matriisi $\mathbf{L}^{(G)} \in \mathbb{R}^{3 \times 3}$ of \mathcal{G} .

See also: verkko, matriisi, edge weight.

large language model (LLM) An LLM is an umbrella term for koneoppiminen methods that use high-dimensional koneoppiminen models (with billions of model parameters) trained on large collections of text data. LLMs are used to analyze or generate sequences of tokens that constitute text data. Many current LLMs use some variant of a transformer that is trained via self-supervised learning, i.e., the training is based on the task of predicting a few words that are intentionally removed from a large text corpus. Thus, we can construct labeled data points simply by selecting some words from a given 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 LLMs.

See also: token, transformer, NLP.

laskennalliset ominaisuudet By computational aspects of an koneoppimi-

nen method, we mainly refer to the computational resources required for its implementation. For example, if an koneoppiminen 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 (i.e., a gradienttiaskel); and 2) how many iterations are needed to obtain useful model parameters. One important example of an iterative optimization technique is GD.

See also: koneoppiminen, ERM, gradienttiaskel, model parameter, GD.

layer A deep net is an ANN that consists of consecutive layers, indexed by $\ell = 1, 2, \dots, L$. The ℓ -th layer consists of artificial neurons $a_1^{(\ell)}, \dots, a_{d^{(\ell)}}^{(\ell)}$ with the layer width $d^{(\ell)}$. Each of these artificial neurons evaluates an aktivointifunktio for a weighted sum of the outputs (or activations) of the previous layer $\ell - 1$. The input to layer $\ell = 1$ is formed from weighted sums of the features of the data point for which the deep net computes a ennuste. The outputs of the neurons in layer ℓ are then, in turn, used to form the inputs for the neurons in the next layer. The final (output) layer consists of a single neuron whose output is used as the ennuste delivered by the deep net.

See also: deep net, ANN.

learning rate Consider an iterative koneoppiminen method for finding or learning a useful hypothesis $h \in \mathcal{H}$. Such an iterative method repeats similar computational (update) steps that adjust or modify the current hypothesis to obtain an improved hypothesis. A 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. Consider, for example, the gradienttiaskel [8, Ch. 5]

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \nabla f(\mathbf{w}^{(t)}) \quad (19)$$

of a gradient-based method for ERM, where the objective function $f(\mathbf{w})$ is the empiirinen riski incurred by $h(\mathbf{w})$ on a training set. Given the current model parameters $\mathbf{w}^{(t)}$ at iteration t , the gradienttiaskel produces updated model parameters $\mathbf{w}^{(t+1)}$ by moving in the opposite direction of the gradientti $\nabla f(\mathbf{w}^{(t)})$.

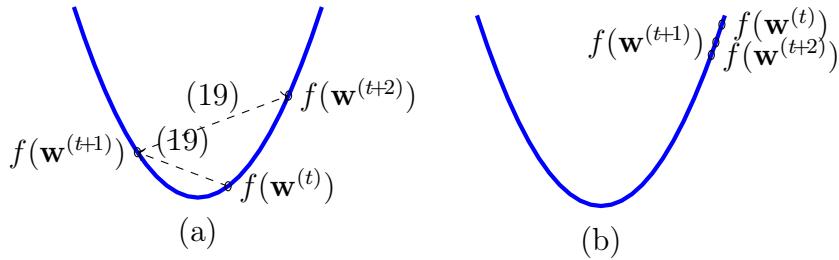


Fig. 96. Effect of using an inadequate learning rate η in the gradienttiaskel (19). (a) If η is too large, the gradienttiaskeleet can “overshoot” such that the iterates $\mathbf{w}^{(t)}$ diverge away from the optimum, i.e., $f(\mathbf{w}^{(t+1)}) > f(\mathbf{w}^{(t)})$. (b) If η is too small, the gradienttiaskeleet make too little progress towards the optimum within the available number of iterations (due to limited computational budget).

See also: koneoppiminen, hypothesis, parameter, GD, SGD, projected gradient descent (projected GD), step size.

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.

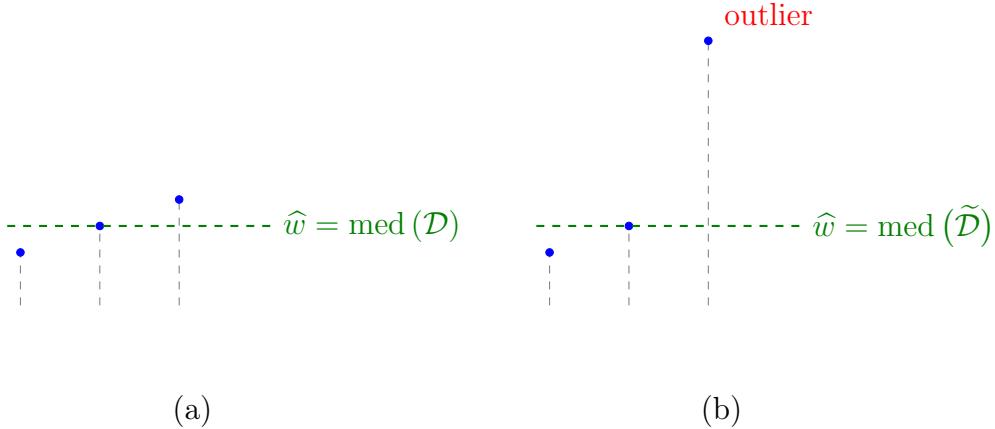


Fig. 97. For the simple parametric model $h^{(w)}(\mathbf{x}) = w$, ERM with absolute error loss amounts to computing the mediaani. (a) Original dataset \mathcal{D} . (b) Noisy dataset $\tilde{\mathcal{D}}$ including an outlier.

For the parametric model $h^{(w)}(\mathbf{x}) = w$, ERM with absolute error loss is solved by the mediaani. Using neliövirhehäviö instead for the same parametric model, makes ERM computing the keskiarvo.

See also: ERM, absolute error loss, Huber regression.

least absolute shrinkage and selection operator (Lasso) The Lasso [150] is an instance of SRM. It learns the weights \mathbf{w} of a lineaarikuvaus $h(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$ from a training set. Lasso is obtained from linear regression by adding the scaled ℓ_1 -normi $\alpha \|\mathbf{w}\|_1$ to the average neliövirhehäviö incurred on the training set.

See also: SRM, weights, lineaarikuvaus, training set, linear regression,

normi, neliövirhehäviö.

least squares Least squares refers to ERM-based methods that use the average neliövirhehäviö

$$\frac{1}{m} \sum_{r=1}^m (y^{(r)} - h(\mathbf{x}^{(r)}))^2$$

on a training set $\mathcal{D} = \{ (\mathbf{x}^{(1)}, y^{(1)}) , \dots , (\mathbf{x}^{(m)}, y^{(m)}) \}$ to measure the quality of a hypothesis kuvaus $h \in \mathcal{H}$. We obtain different least squares methods by using different models in ERM. For example, the least squares variant of a lineaarinen malli is a least squares method that uses a lineaarinen malli.

See also: ERM, neliövirhehäviö, lineaarinen malli, linear regression.

leave-one-out (LOO) cross-validation A special case of the k -fold CV where the validation set is size one, i.e. the validation set is only a single data point.

See also: k -fold CV, validation, validation error.

lineaарinen malli Consider an koneoppiminen application involving data points, each represented by a numeric feature vector $\mathbf{x} \in \mathbb{R}^d$. A linear model defines a hypothesis space consisting of all real-valued lineaarikuvauskset from \mathbb{R}^d to \mathbb{R} such that

$$\mathcal{H}^{(d)} := \{ h : \mathbb{R}^d \rightarrow \mathbb{R} \mid h(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} \text{ for some } \mathbf{w} \in \mathbb{R}^d \} .$$

Each value of d defines a different hypothesis space, corresponding to the number of features used to compute the ennuste $h(\mathbf{x})$. The choice of d is often guided not only by laskennallinen ominaisuus (e.g., fewer

features reduce computation) and laskennallinen ominaisuus (e.g., more features typically reduce harha and riski), but also by tulkittavuus. A linear model using a small number of well-chosen features is generally considered more interpretable [122], [127]. The linear model is attractive because it can typically be trained using scalable convex optimointimenetelmät [61], [65]. Moreover, linear models often permit rigorous statistical analysis, including fundamental limits on the minimi achievable riski [16]. They are also useful for analyzing more complex nonlinear models such as ANNs. For instance, a deep net can be viewed as the composition of a feature map—implemented by the input and hidden layers—and a linear model in the output layer. Similarly, a decision tree can be interpreted as applying a one-hot-encoded feature map based on decision regions, followed by a linear model that assigns a ennuste to each region. More generally, any trained model $\hat{h} \in \mathcal{H}$ that is differentiable at some \mathbf{x}' can be locally approximated by a lineaarikuvaus $g(\mathbf{x})$. Fig. 98 illustrates such a local linear approximation, defined by the gradientti $\nabla \hat{h}(\mathbf{x}')$. Note that the gradientti is only defined where \hat{h} is differentiable. To ensure vakaus in the context of luotettava tekoäly, one may prefer models whose associated kuvaus \hat{h} is Lipschitz jatkuva. A classic result in mathematical analysis—Rademacher’s Theorem—states that if \hat{h} is Lipschitz jatkuva with some constant L over an open set $\Omega \subseteq \mathbb{R}^d$, then \hat{h} is differentiable almost everywhere in Ω [152, Th. 3.1].

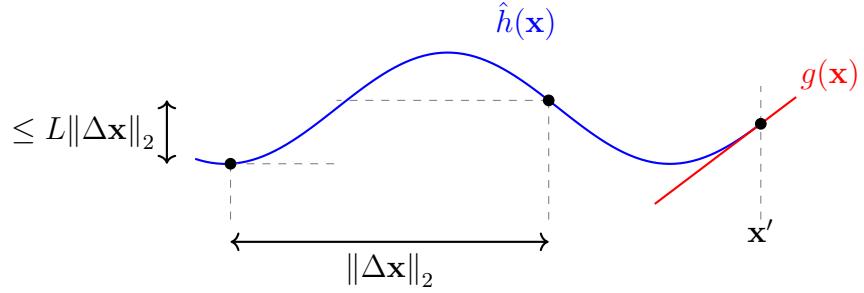


Fig. 98. A trained model $\hat{h}(\mathbf{x})$ that is differentiable at a point \mathbf{x}' can be locally approximated by a lineaarikuvaus $g \in \mathcal{H}^{(d)}$. This local approximation is determined by the gradientti $\nabla\hat{h}(\mathbf{x}')$.

See also: model, hypothesis space, lineaarikuvaus, tulkittavuus, LIME.

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 luokitin is characterized by having decision regions that are separated by hyperplanes in \mathbb{R}^d [8, Ch. 2].

See also: data point, feature, label, label space, luokitin, decision region.

linear discriminant analysis (LDA) LDA is a classical piirreoptimointi method [38], [151]. In the context of binary luokittelu problems, LDA seeks a linear feature map $\phi^{(\mathbf{w})} : \mathbb{R}^d \rightarrow \mathbb{R} : \mathbf{x} \mapsto \mathbf{w}^T \mathbf{x}$ such that the new feature $\phi^{(\mathbf{w})}(\mathbf{x})$ optimally allows us to predict the label of a data point. See also: piirreoptimointi, ulottuvuuksien vähentäminen, self-supervised learning.

linear least squares Linear least squares refers to the variant of linear regression that uses the neliövirhehäviö to measure the quality of a

linear hypothesis kuvaus. Conversely, it can also be viewed as the variant of least squares that restricts the hypothesis to a lineaarinen malli. In particular, linear least squares learns the parameters \mathbf{w} of a linear hypothesis $h^{(\mathbf{w})}(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$ by solving

$$\min_{\mathbf{w} \in \mathbb{R}^d} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2. \quad (20)$$

Here, the feature matrix is $\mathbf{X} = (\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)})$ and the label vector is $\mathbf{y} = (y^{(1)}, \dots, y^{(m)})^T$. Both are constructed from the training set

$$\mathcal{D} = \{(\mathbf{x}^{(1)}, y^{(1)}), \dots, (\mathbf{x}^{(m)}, y^{(m)})\}.$$

The optimointitehtävä in (20) admits a clear geometric interpretation, i.e., we seek the vektori $\mathbf{X}\hat{\mathbf{w}}$ in the column space of \mathbf{X} that is closest to the label vector \mathbf{y} (see Fig. 99) [23, Ch. 8]. A necessary and sufficient condition for $\hat{\mathbf{w}}$ to minimize (20) is the normal equations

$$\mathbf{X}^T \mathbf{X} \hat{\mathbf{w}} = \mathbf{X}^T \mathbf{y}.$$

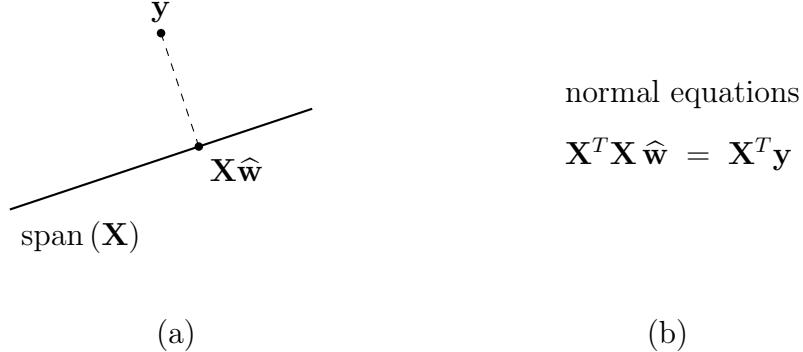


Fig. 99. Linear least squares has both geometric and algebraic interpretations.
 (a) Geometrically, it finds the orthogonal projection of the label vector \mathbf{y} onto the column space of the feature matrix \mathbf{X} [23, Ch. 8]. (b) Algebraically, it solves a linear system known as normal equations.

See also: least squares, linear regression, neliövirhehäviö, lineaarinen malli, ERM.

linear regression Linear regression methods learn a linear hypothesis $h^{(w)}$ which is used to predict the numeric label $y \in \mathbb{R}$ of a data point based on its numeric feature vectors $\mathbf{x} = (x_1, \dots, x_d)^T \in \mathbb{R}^d$. The least-squares variant of linear regression measures the quality of a linear hypothesis $h^{(w)}$ via the average squared error incurred on a training set

$$(\mathbf{x}^{(1)}, y^{(1)}), \dots, (\mathbf{x}^{(m)}, y^{(m)})$$

As an instance of ERM, linear (least-squares) regression learns the model parameters \mathbf{w} by solving the optimization problem

$$\min_{\mathbf{w} \in \mathbb{R}^d} \frac{1}{m} \sum_{r=1}^m (y^{(r)} - \mathbf{w}^T \mathbf{x}^{(r)})^2.$$

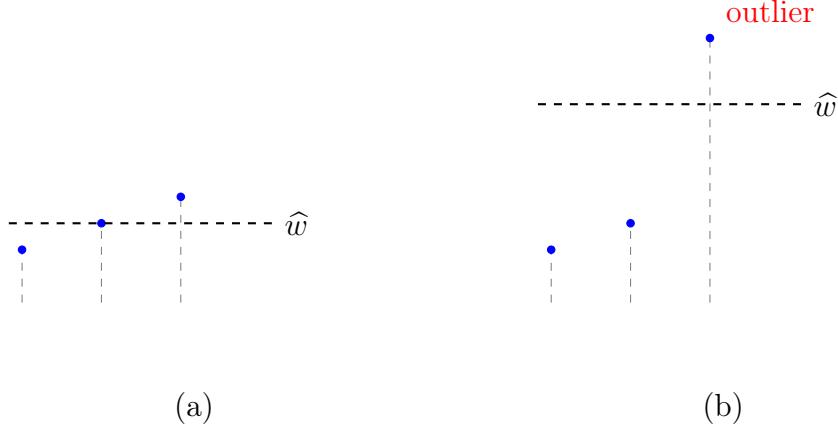


Fig. 100. For a lineaarinen malli with $d = 1$ and using the trivial feature $x = 1$ for any data point, linear regression reduces to computing the average $\hat{w} = (1/m) \sum_{r=1}^m y^{(r)}$. (a) A clean training set and resulting parameter (given by the average). (b) A perturbed dataset (including an outlier) and the resulting parameter.

We can rewrite the above optimointitehtävä more compactly using the feature matrix $\mathbf{X} = (\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)})^T \in \mathbb{R}^{m \times d}$ and the label vektori $\mathbf{y} = (y^{(1)}, \dots, y^{(m)})^T \in \mathbb{R}^m$. This allows to rewrite the above optimointitehtävä as

$$\min_{\mathbf{w} \in \mathbb{R}^d} \frac{1}{m} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2.$$

By the zero-gradient condition, a necessary and sufficient condition for a vector $\hat{\mathbf{w}}$ to be a solution to the above optimointitehtävä is the linear system of equations [32]

$$\mathbf{X}^T \mathbf{X} \hat{\mathbf{w}} = \mathbf{X}^T \mathbf{y}. \quad (21)$$

Instead of solving (21) directly (via computing the käänteismatriisi or

pseudokäänteisluku), many koneoppiminen mehtods use variants of GD to construct a sequence $\mathbf{w}^{(0)}, \mathbf{w}^{(1)}, \dots$, of increasingly accurate approximations of a solution $\hat{\mathbf{w}}$ to (21). These gradient-based methods can be interpreted as a fixed-point iteration for the following re-formulation of (21),

$$(\mathbf{I} - \mathbf{A}\mathbf{X}^T\mathbf{X})\hat{\mathbf{w}} + \mathbf{X}^T\mathbf{y} = \hat{\mathbf{w}}, \text{ with some invertible matriisi } \mathbf{A}.$$

This equation is solved by a vektori $\hat{\mathbf{w}}$ if and only if this vektori also solves (21). The optimality condition (21) is also useful for the study of the stability of linear regressio. Ideally, we would like the solutions of (21) to be insensitive to small perturbations of the training set. We can capture these perturbations via a perturbed feature matrix $\tilde{\mathbf{X}} = \mathbf{X} + \Delta\mathbf{X}$ and perturbed label vektori $\tilde{\mathbf{y}} = \mathbf{y} + \Delta\mathbf{y}$. Here, $\Delta\mathbf{X}$ and $\Delta\mathbf{y}$ represent small perturbations to the feature vectors and labels of the data points in the original training set. Matriisi perturbation theory allows to evaluate how much the solutions of the perturbed linear regressio problem [3, Sec. 2.6]

$$\tilde{\mathbf{X}}^T\tilde{\mathbf{X}}\tilde{\mathbf{w}} = \tilde{\mathbf{X}}^T\tilde{\mathbf{y}}$$

deviate from the solutions $\tilde{\mathbf{w}}$ of the original linear regressio problem.

See also: regressio, lineaarinen malli, ERM.

Lloyd's algorithm Lloyd's algoritmi [153] is an iterative optimointimene-telma for finding cluster centroids that are approximately optimal for the k -means objective function. Lloyd's algoritmi alternates between: 1) updating the cluster assignment of each data point based on the nearest

current cluster centroid; and 2) recalculating the cluster centroids given the updated cluster assignments [153].

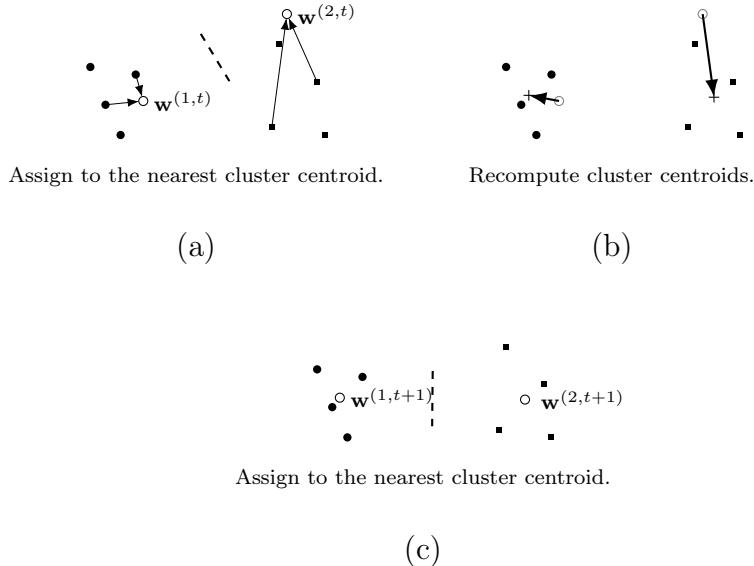


Fig. 101. Lloyd's algorithm alternates between (a) and (c) assigning data points to the nearest cluster centroid and, in turn, (b) recomputing the cluster centroids based on the new cluster assignments.

See also: cluster centroid, k -means, clustering.

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 characterized by features and labels. In contrast to a single dataset used in basic k-means methods, a local dataset is also related to other local datasets via different notions of similarity. These similarities might arise from shared or communication

infrastructure and are encoded in the edges of an FL network.

See also: dataset, data point, feature, label, koneoppiminen, todennäköisyysmalli, FL network.

local interpretable model-agnostic explanations (LIME) Consider a trained model (or learned hypothesis) $\hat{h} \in \mathcal{H}$, which maps the feature vector of a data point to the ennuste $\hat{y} = \hat{h}$. LIME is a technique for explaining the behavior of \hat{h} , locally around a data point with feature vector $\mathbf{x}^{(0)}$ [127]. The selitys is given in the form of a local approximation $g \in \mathcal{H}'$ of \hat{h} (see Fig. 102). This approximation can be obtained by an instance of ERM with a carefully designed training set. In particular, the training set consists of data points with feature vectors centered around $\mathbf{x}^{(0)}$ and the (pseudo-)label $\hat{h}(\mathbf{x})$. 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 locally approximate a deep net. Another widely used choice for \mathcal{H}' is the lineaarinen malli.

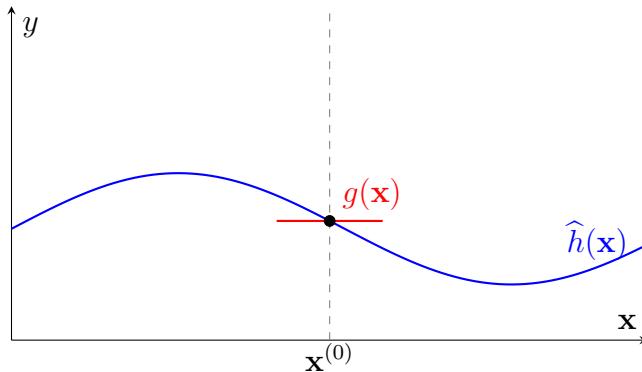


Fig. 102. To explain a trained model $\hat{h} \in \mathcal{H}$, around a given feature vector $\mathbf{x}^{(0)}$, we can use a local approximation $g \in \mathcal{H}'$.

See also: model, selitys, ERM, training set, label, decision tree, deep net, lineaarinen malli.

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

See also: device, FL network, model, hypothesis space.

logistic loss Consider a data point characterized by 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 ennuuste is defined as [38]

$$L((\mathbf{x}, y), h) := \log(1 + \exp(-yh(\mathbf{x}))). \quad (22)$$

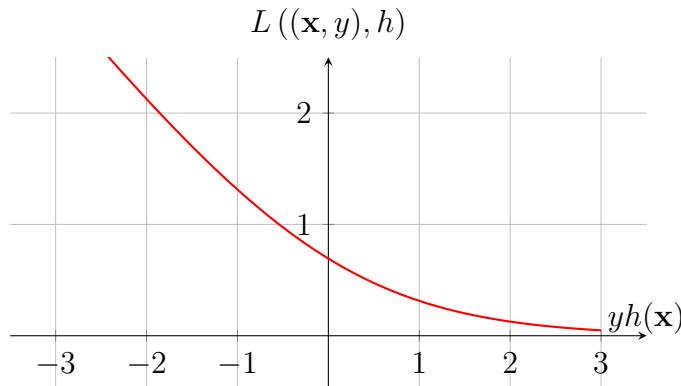


Fig. 103. The logistic loss incurred by the ennuuste $h(\mathbf{x}) \in \mathbb{R}$ for a data point with label $y \in \{-1, 1\}$.

Note that the expression (22) for the logistic loss applies only for the label space $\mathcal{Y} = \{-1, 1\}$ and when using the thresholding rule (23).

See also: loss, luokittelu, luokitin, lineaarinen malli.

logistic regression Logistic regressio learns a linear hypothesis kuvaus (or luokitin) $h(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$ to predict a binary label y based on the numeric feature vector \mathbf{x} of a data point [38], [65]. The quality of a linear hypothesis kuvaus is measured by the average logistic loss on some labeled data points (i.e., the training set).

See also: regressio, hypothesis, kuvaus, luokitin, label, feature vector, data point, logistic loss, labeled data point, training set.

loss koneoppiminen methods use a loss function $L(\mathbf{z}, h)$ 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 $L(\mathbf{z}, h)$, for a data point \mathbf{z} and hypothesis h .

See also: loss function, empiirinen riski.

loss function A loss funktio is a kuvaus

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

It assigns a nonnegative real number (i.e., the loss) $L((\mathbf{x}, y), h)$ 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 $L((\mathbf{x}, y), h)$ quantifies the discrepancy between the true label y and the ennuste $h(\mathbf{x})$. Lower (closer to zero) values $L((\mathbf{x}, y), h)$ indicate a smaller discrepancy between ennuste $h(\mathbf{x})$

and label y . Fig. 104 depicts a loss function for a given data point, with features \mathbf{x} and label y , as a function of the hypothesis $h \in \mathcal{H}$.

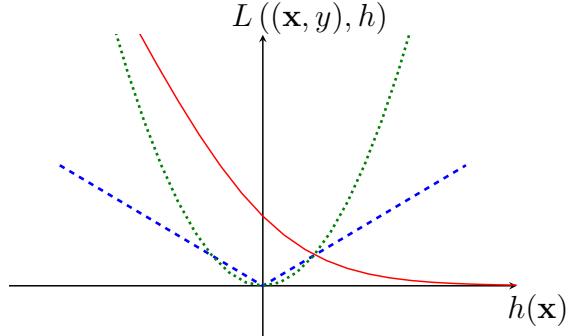


Fig. 104. Some loss function $L((\mathbf{x}, y), h)$ for a fixed data point, with feature vector \mathbf{x} and label y , and a varying hypothesis h . Some optimization methods try to find (or learn) a hypothesis that incurs minimal loss.

See also: loss, label, feature vector, ERM.

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

$$\hat{y} = 1 \text{ for } h(\mathbf{x}) \geq 0 \text{ and } \hat{y} = -1 \text{ otherwise.} \quad (23)$$

We can characterize a classifier by its decision regions \mathcal{R}_a , for every

possible label value $a \in \mathcal{Y}$.

See also: hypothesis, luokittelu, decision region.

luokittelu 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 $y \in \{1, \dots, 19\}$, and represents the category to which the corresponding data point belongs.

See also: label, data point, feature.

luotettava teköäly Besides the laskennalliset ominaisuudet and tilastollinen ominaisuus, a third main design aspect of koneoppiminen methods is their trustworthiness [187]. The European Union (EU) has put forward seven key requirements (KRs) for trustworthy AI (which typically build on koneoppiminen methods) [188]:

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

See also: laskennalliset ominaisuudet, tilastollinen ominaisuus, koneoppiminen, AI, vakaus, data, transparency.

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

See also: metric, naapuri.

machine unlearning Consider an koneoppiminen method that learns a hypothesis \hat{h} via ERM on a training set \mathcal{D} . The learned hypothesis can reveal information about \mathcal{D} , which is exploited by yksityisyysyökkäykset such as model inversion. Machine unlearning refers to techniques that modify \hat{h} , so that it is harder to infer properties of individual data points in \mathcal{D} [156]. Machine unlearning helps to meet legal requirements for yksityisyysden suoja in AI systems [98].

See also: model inversion, yksityisyysden suoja, Yleinen tietosuoja-asetus (GDPR).

maximum likelihood Consider data points $\mathcal{D} = \{\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(m)}\}$ that are interpreted as the realizations of i.i.d. RVs with a common probability distribution $\mathbb{P}^{(\mathbf{w})}$, which depends on the model parameters $\mathbf{w} \in \mathcal{W} \subseteq \mathbb{R}^n$. Maksimi likelihood methods learn model parameters \mathbf{w} by maximizing the probability (density) $\mathbb{P}^{(\mathbf{w})}(\mathcal{D}) = \prod_{r=1}^m \mathbb{P}(\mathbf{z}^{(r)}; \mathbf{w})$ of the observed dataset. Thus, the maksimi likelihood estimator is a solution to the optimointitehtävä $\max_{\mathbf{w} \in \mathcal{W}} \mathbb{P}(\mathcal{D}; \mathbf{w})$.

See also: probability distribution, optimointitehtävä, todennäköisyyssmalli.

mean absolute error (MAE) The MAE of a hypothesis is the average absolute error loss computed over a given dataset. In theoretical analyses, MAE also denotes the expected absolute error loss, i.e., the corresponding riski.

See also: absolute error loss, riski.

mean squared error (MSE) The MSE of a hypothesis is the average neliövirhehäviö computed over a given dataset. In theoretical analyses, MSE also denotes the expected neliövirhehäviö, i.e., the corresponding riski.

See also: neliövirhehäviö, riski.

mean squared estimation error (MSEE) Consider an koneoppiminen method that learns model parameters $\hat{\mathbf{w}}$ based on some dataset \mathcal{D} . If we interpret the data points in \mathcal{D} as i.i.d. realizations of a RV \mathbf{z} , we define the estimation error $\Delta\mathbf{w} := \hat{\mathbf{w}} - \bar{\mathbf{w}}$. Here, $\bar{\mathbf{w}}$ denotes the true model parameters of the probability distribution of \mathbf{z} . The MSEE is defined as the expectation $\mathbb{E}\{\|\Delta\mathbf{w}\|^2\}$ of the squared Euclidean norm of the estimation error [57], [114].

See also: RV, estimation error, todennäköisyysmalli, neliövirhehäviö.

membership inference attack Consider an koneoppiminen method that learns a hypothesis via ERM on a training set. Membership inference hyökkäys is a form of yksityisyshyökkäys where an adversary tries to determine whether a particular data point was part of the training set. The attacker typically queries \hat{h} with candidate feature vectors $\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(B)}$, and infers the membership status of a given data point

based on the ennusteet $\hat{h}(\mathbf{x}^{(1)}), \dots, \hat{h}(\mathbf{x}^{(B)})$ [157].

See also: hyökkäys, yksityisyyshyökkäys.

metric A metric is a quantitative measure used to compare objects. In mathematics, a metric measures the distance between two points in a space and must follow specific rules, i.e., the distance is always nonnegative, zero only if the points are the same, symmetric, and it satisfies the triangle inequality [2]. In the context of koneoppiminen, the term metric refers to a quantitative measure of how well a model performs (somewhat similar to a loss function). Examples include tarkkuus, precision, and the average binääritappio on a test set [81], [38]. The term loss function is typically used in the context of model training, while the term metric is used in the context of model validation.

See also: loss, validation, tarkkuus.

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 to estimate these missing values [158]. We can interpret data imputation as an koneoppiminen problem where the label of a data point is the value of the corrupted feature.

See also: feature, label.

model The study and design of koneoppiminen methods is often based on a mathematical model [159]. Maybe the most widely used example of a mathematical model for koneoppiminen is a hypothesis space. A hypothesis space consists of hypothesis kuvaukset that are used by an

koneoppiminen method to predict labels from the features of data points. Another important type of mathematical model is a todennäköisyysmalli, which consists of probability distributions that describe how data points are generated. Unless stated otherwise, we use the term model to refer specifically to the hypothesis space underlying an koneoppiminen method. We illustrate one example of a hypothesis space and a todennäköisyysmalli in Fig. 105.

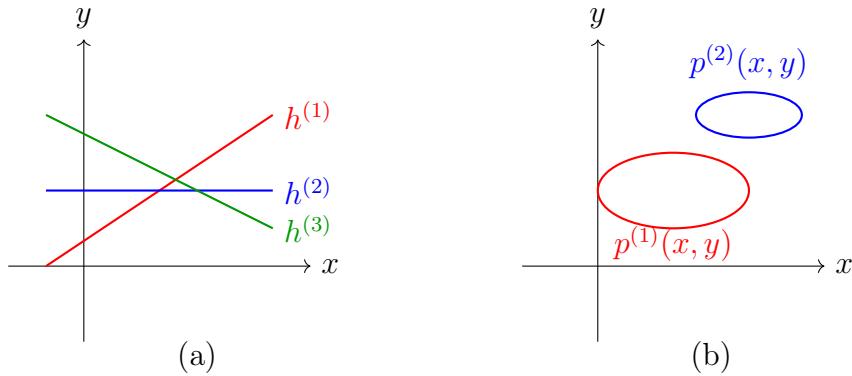


Fig. 105. Two types of mathematical models used in koneoppiminen. (a) A hypothesis space consisting of three lineaarikuvauskset. (b) A todennäköisyysmalli consisting of probability distribution over the plane spanned by the feature and label values of a data point.

See also: hypothesis space, todennäköisyysmalli, probability distribution.

model inversion A model inversion is a form of yksityisyyskyökkäys on an koneoppiminen system. An adversary seeks to infer sensitive attributes of individual data points by exploiting partial access to a trained model $\hat{h} \in \mathcal{H}$. This access typically consists of querying the model for ennusteet $\hat{h}(\mathbf{x})$

using carefully chosen inputs. Basic model inversion techniques have been demonstrated in the context of facial image luokittelu, where images are reconstructed using the (gradientti of) model outputs combined with auxiliary information such as a person's name [160] (see Fig. 106).

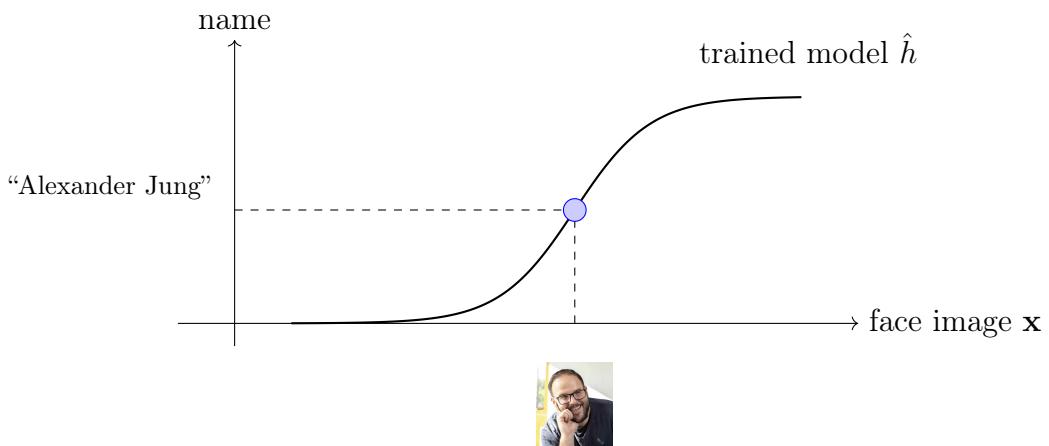


Fig. 106. Model inversion techniques implemented in the context of facial image classification.

See also: model, yksityisyshyökkäys, koneoppiminen, sensitive attribute, data point, ennuste, luokittelu, gradientti, luotettava tekoäly, yksityisyyyden suoja.

model parallelism Model parallelism refers to a particular class of distributed optimointimenetelmät used to train an koneoppiminen model. Here, different devices store and process disjoint subsets of the model parameters. This approach contrasts with data parallelism, where each device maintains a full replica of the model parameters while processing disjoint subsets of the global dataset. Model parallelism allows us to

train an koneoppiminen model whose parameters cannot fit into the memory of a single device. One key application of model parallelism is the training of extremely large ANNs, such as transformer models with billions of model parameters.

See also: VFL.

model parameter The elements of a parametric model are specified by quantities that are referred to as model parameters. In the context of koneoppiminen, a parametric model consists of hypothesis maps that are specified by a list of model parameters w_1, w_2, \dots, w_d . It is often convenient to stack these model parameters into a vektori $\mathbf{w} = (w_1, \dots, w_d)^T \in \mathbb{R}^d$.

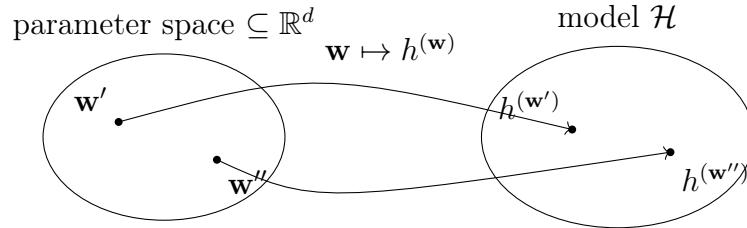


Fig. 107. The model parameters \mathbf{w} select a well-defined hypothesis $h^{(\mathbf{w})}$ out of the model \mathcal{H} .

We can think of model parameters as an identifier for a hypothesis kuvaus, similar to how a social security number identifies a person.

See also: model, parameter, hypothesis, kuvaus.

model selection In koneoppiminen, 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 [8, Ch. 6].

See also: *koneoppiminen*, *model*, *training*, *validation error*.

monitehtäväoppiminen Multitask learning aims to leverage relations between different oppimistehtävät. Consider two oppimistehtävät 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 may 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.

See also: *oppimistehtävä*, *dataset*, *deep net*, *weights*, *layer*.

multi-label classification Multi-label luokittelu 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.

See also: *label*, *luokittelu*, *data point*.

mutual information (MI) The MI $I(\mathbf{x}; y)$ between two RVs \mathbf{x} , y defined on the same probability space is given by [31]

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

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 ennuste could be obtained by a hypothesis learned by an ERM-based *koneoppiminen* method.

See also: RV, probability space, ennuste, hypothesis, ERM, koneoppiminen.

naapuri A neighbor of a node $i \in \mathcal{V}$ within an undirected graph is a node $i' \in \mathcal{V} \setminus \{i\}$ that is kytetty verkko via an edge to node i .

See also: undirected graph, kytetty verkko.

naapurusto Consider some metric space \mathcal{X} with metric $d : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}_+$.

The neighborhood of a point $\mathbf{x} \in \mathcal{X}$ is the set of other points having a sufficiently small distance to \mathbf{x} . For example, the ϵ -neighborhood of \mathbf{x} is defined as

$$\{\mathbf{x}' \in \mathcal{X} : d(\mathbf{x}, \mathbf{x}') \leq \epsilon\}.$$

If \mathcal{X} is an undirected graph, which is a special case of a metric space, the neighborhood of a node $i \in \mathcal{V}$ is the set of its naapurit.

See also: metric, naapuri.

natural language processing (NLP) NLP studies koneoppiminen methods for the analysis and generation of human language. Typical NLP tasks include text luokittelu, machine translation, sentiment analysis, and question answering. Modern NLP systems represent language as sequences of tokens and train models that capture contextual dependencies, such as attention-based methods.

See also: token, attention.

neliövirhehäviö The squared error loss measures the ennuste 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

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

See also: loss, ennuste, hypothesis, label, feature, data point.

nested cross-validation Nested cross-validation is a method of extending the k -fold CV from training and validation sets to also cover the test set. Instead of simply choosing a single test set from the data, two loops are run: the outer and the inner loop. The outer loop uses k -fold CV to separate a test set from the data and the inner loop again uses k -fold CV to separate the rest of the data into a training and validation sets. Doing this decreases the varianssi and harha in the results.

See also: k -fold CV, leave-one-out (LOO) cross-validation, validation, validation error.

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

See also: data, local dataset, verkko, FL, device.

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 [161].

See also: FL network, model parameter, GTV.

networked federated learning (NFL) NFL 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.

See also: model, local dataset, FL.

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} .

See also: model, FL network, local model, hypothesis space.

normal equations The optimality condition for the model parameters in linear least squares are often referred to as normal equations.

See also: model parameter, linear least squares.

objective function An objective funkatio is a kuvaus that assigns a numeric objective value $f(\mathbf{w})$ to each choice \mathbf{w} of some variable that we want to optimize (see Fig. 108). In the context of koneoppiminen, the optimization variable could be the model parameters of a hypothesis $h^{(\mathbf{w})}$. Common objective funktiot include the riski (i.e., expected loss) or the empiirinen riski (i.e., average loss over a training set). koneoppiminen methods apply optimization techniques, such as gradient-based methods, to find the choice \mathbf{w} with the optimal value (e.g., the minimi or the maksimi) of the objective funkio.

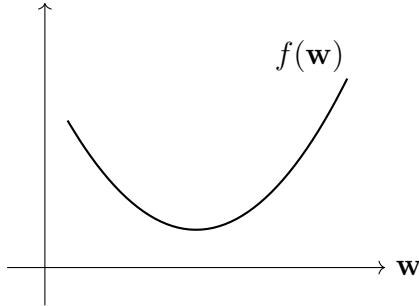


Fig. 108. An objective function maps each possible value \mathbf{w} of an optimization variable, such as the model parameters of a machine learning model, to a value $f(\mathbf{w})$ that measures the usefulness of \mathbf{w} .

See also: loss, empiirinen riski, ERM, optimointitehtävä.

online gradient descent (online GD) Consider an machine learning 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 $t = 1, 2, \dots$. Let us interpret the (generation of) data points $\mathbf{z}^{(t)}$ as i.i.d. RVs with a common probability distribution $\mathbb{P}(\mathbf{z})$. The riski $\mathbb{E}\{L(\mathbf{z}, \mathbf{w})\}$ of a hypothesis $h^{(\mathbf{w})}$ can then (under mild conditions) be obtained as the limit

$$\lim_{T \rightarrow \infty} (1/T) \sum_{t=1}^T L(\mathbf{z}^{(t)}, \mathbf{w}).$$

We might use this limit as the objective function for learning the model parameters \mathbf{w} . Unfortunately, the above limit can only be evaluated if we wait infinitely long in order to collect all data points. However, many machine learning 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 $L(\mathbf{z}^{(t)}, \mathbf{w})$ to the riski. As its name suggests, online GD updates $\mathbf{w}^{(t)}$ via a (projected) gradienttiaskel such that

$$\mathbf{w}^{(t+1)} := P_{\mathcal{W}}(\mathbf{w}^{(t)} - \eta_t \nabla_{\mathbf{w}} L(\mathbf{z}^{(t)}, \mathbf{w})). \quad (24)$$

Note that (24) is a gradienttiaskel for the current component $L(\mathbf{z}^{(t)}, \cdot)$ of the riski. The update (24) ignores all previous components $L(\mathbf{z}^{(t')}, \cdot)$, for $t' < t$. It might therefore happen that, compared with $\mathbf{w}^{(t)}$, the updated model parameters $\mathbf{w}^{(t+1)}$ increase the retrospective average loss $\sum_{t'=1}^{t-1} L(\mathbf{z}^{(t')}, \cdot)$. 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 $\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(T)}$ are at least as good as those delivered by any other learning method [117], [162].

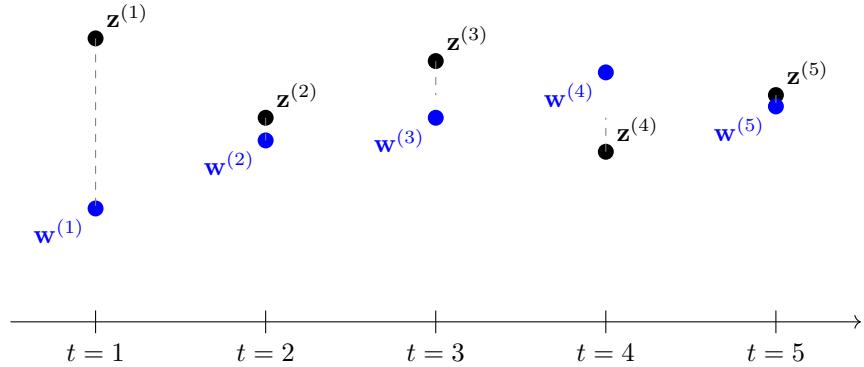


Fig. 109. An instance of online GD that updates the model parameters $\mathbf{w}^{(t)}$ using the data point $\mathbf{z}^{(t)} = x^{(t)}$ arriving at time t . This instance uses the neliövirhehäviö $L(\mathbf{z}^{(t)}, w) = (x^{(t)} - w)^2$.

See also: objective function, GD, gradienttiaskel, online learning.

online learning Some koneoppiminen methods are designed to process data in a sequential manner, updating their model parameters one at a time, as new data points become available. A typical example is time-series data, such as daily minimi and maksimi temperatures recorded by an FMI weather station. These values form a chronological sequence of observations. During each time step t , online learning methods update (or refine) the current hypothesis $h^{(t)}$ (or model parameters $\mathbf{w}^{(t)}$) based on the newly observed data point $\mathbf{z}^{(t)}$.

See also: online gradient descent (online GD), online-algoritmi.

online-algoritmi An online algoritmi processes input data incrementally, receiving data points sequentially and making decisions or producing outputs (or decisions) immediately without having access to the entire

input in advance [116], [117]. Unlike an offline algoritmi, which has the entire input available from the start, an online algoritmi must handle epävarmuus about future inputs and cannot revise past decisions. Similar to an offline algoritmi, we represent an online algoritmi formally as a collection of possible executions. However, the execution sequence for an online algoritmi has a distinct structure as follows:

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

Each execution begins from an initial state (i.e., in_1) and proceeds through alternating computational steps, outputs (or decisions), and inputs. Specifically, at step t , the algoritmi performs a computational step s_t , generates an output out_t , and then subsequently receives the next input (data point) in_{t+1} . A notable example of an online algoritmi in koneoppiminen is online GD, which incrementally updates model parameters as new data points arrive.

See also: algoritmi, data, data point, epävarmuus, koneoppiminen, online GD, model parameter, online learning.

oppimistehtävä Consider a dataset \mathcal{D} consisting of multiple data points $\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(m)}$. For example, \mathcal{D} can be a collection of images in an image database. A learning task is defined by specifying those properties (or attributes) of a data point that are used as its features and labels. Given a choice of model \mathcal{H} and loss function, a learning task leads to an instance of ERM and can thus be represented by the associated objective function $\widehat{L}(h|\mathcal{D})$ for $h \in \mathcal{H}$. Importantly, multiple distinct learning tasks can be constructed from the same dataset by selecting

different sets of features and labels (see Fig. 110).



An image showing cows grazing in the Austrian countryside.

Task 1 (regressio):

Features are the RGB values of all image pixels, and the label is the number of cows depicted.

Task 2 (luokittelu):

Features include the average green intensity of the image, and the label indicates whether cows should be moved to another location (i.e., yes/no).

Fig. 110. Two learning tasks constructed from a single image dataset. These tasks differ in feature selection and choice of label (i.e., the objective), but are both derived from the same dataset.

Different learning tasks arising from the same underlying dataset are often coupled. For example, when a todennäköisyysmalli is used to generate data points, statistical dependencies among different labels

induce dependencies among the corresponding learning tasks. In general, solving learning tasks jointly, e.g., using monitehtäväoppiminen methods, tends to be more effective than solving them independently (thereby ignoring dependencies among learning tasks) [147], [148], [149].

See also: monitehtäväoppiminen, label space.

optimism in the face of uncertainty koneoppiminen 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 riski of a hypothesis. Using a todennäköisyysmalli, one can construct a confidence interval $[l^{(\mathbf{w})}, u^{(\mathbf{w})}]$ 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 $\bar{f}(\mathbf{w}) \in [l^{(\mathbf{w})}, u^{(\mathbf{w})}]$ with a sufficiently high todennäköisyys. An optimist chooses the model parameters according to the most favorable—yet still plausible—value $\tilde{f}(\mathbf{w}) := l^{(\mathbf{w})}$ of the performance criterion (see Fig. 111). Two examples of koneoppiminen methods that use such an optimistic construction of an objective function are SRM [132, Ch. 11] and upper confidence bound (UCB) methods for sequential decision making [163, Sec. 2.2].

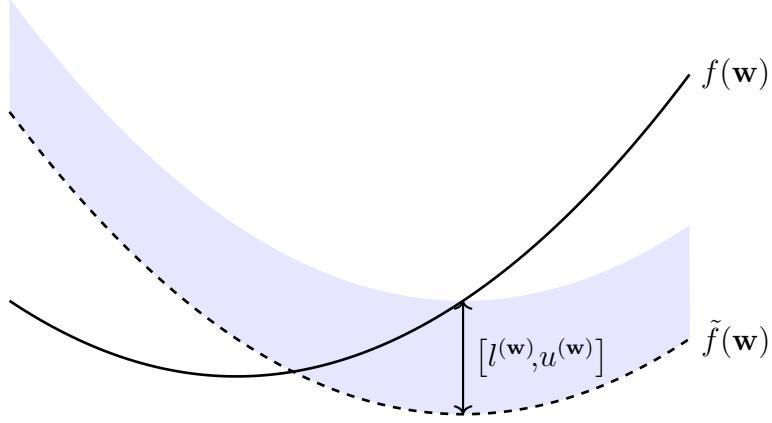


Fig. 111. koneoppiminen methods learn model parameters \mathbf{w} by using some estimate of $f(\mathbf{w})$ for the ultimate performance criterion $\bar{f}(\mathbf{w})$. Using a todennäköisyyssmalli, one can use $f(\mathbf{w})$ to construct confidence intervals $[l^{(\mathbf{w})}, u^{(\mathbf{w})}]$, which contain $\bar{f}(\mathbf{w})$ with high probability. The best plausible performance measure for a specific choice \mathbf{w} of model parameters is $\tilde{f}(\mathbf{w}) := l^{(\mathbf{w})}$.

See also: objective function, optimointimenetelmä, gradient-based method, UCB.

outlier Many koneoppiminen 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) [71]. However, in some applications, the data consist of a majority of regular data points that conform with the i.i.d. assumption as well as a small number of data points that have fundamentally different statistical properties compared with 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 of this deviation. Statistical learning theory studies fundamental limits on the ability to mitigate outliers reliably [164], [165].

See also: vakaus, stability, Huber regression, todennäköisyyssmalli.

output The term output is sometimes used as a synonym for the label of a data point [39].

See also: label, data point.

output vector The term output vektori is used as a synonym for the label vector of a dataset [39].

See also: output, label vector, dataset.

parameter The parameter of an koneoppiminen model is a tunable (i.e., learnable or adjustable) quantity that allows us to choose between different hypothesis kuvauskset. For example, the lineaarinen malli $\mathcal{H} := \{h^{(\mathbf{w})} : h^{(\mathbf{w})}(x) = w_1x + w_2\}$ consists of all hypothesis kuvauskset $h^{(\mathbf{w})}(x) = w_1x + w_2$ with a particular choice for the parameters $\mathbf{w} = (w_1, w_2)^T \in \mathbb{R}^2$. Another example of a model parameter is the weights assigned to a connection between two neurons of an ANN.

See also: koneoppiminen, model, hypothesis, kuvaus, lineaarinen malli, weights, ANN.

parameter space The parameter space \mathcal{W} of an koneoppiminen model \mathcal{H} is the set of all feasible choices for the model parameters (see Fig.

112). Many important koneoppiminen methods use a model that is parameterized by vektorit of the Euclidean space \mathbb{R}^d . Two widely used examples of parameterized models are lineaariset mallit and deep nets. The parameter space is then often a subset $\mathcal{W} \subseteq \mathbb{R}^d$, e.g., all vektorit $\mathbf{w} \in \mathbb{R}^d$ with a normi smaller than one.

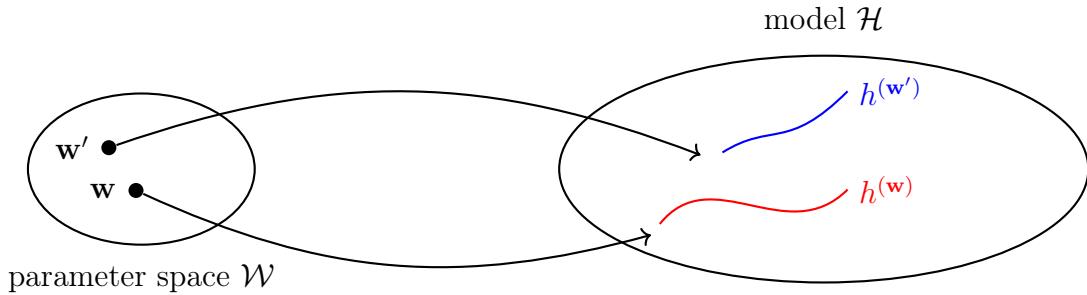


Fig. 112. The parameter space \mathcal{W} of an koneoppiminen model \mathcal{H} consists of all feasible choices for the model parameters. Each choice \mathbf{w} for the model parameters selects a hypothesis kuvaus $h^{(\mathbf{w})} \in \mathcal{H}$.

See also: parameter, model, model parameter.

parametric model A parametric model is a mathematical model characterized by a finite set of variable quantities called parameters. An important example is the todennäköisyysmalli consisting, for a given dimension d , of all moninormaalijakaumat (on sample space \mathbb{R}^d) with some kesiarvo vektori $\boldsymbol{\mu} \in \mathbb{R}^d$ and covariance matrix $\mathbf{C} \in \mathbb{R}^{d \times d}$. In the context of koneoppiminen, a parametric model defines a hypothesis space \mathcal{H} parameterized by a finite number of model parameters. Each hypothesis $h \in \mathcal{H}$ is uniquely identified by a list of model parameters

w_1, w_2, \dots, w_d (see Fig. 112). We can stack these parameters into a vektori $\mathbf{w} \in \mathbb{R}^d$. Two widely used examples of parametric models are the lineaarinen malli and the ANN. The corresponding parameter space is typically a subset of \mathbb{R}^d .

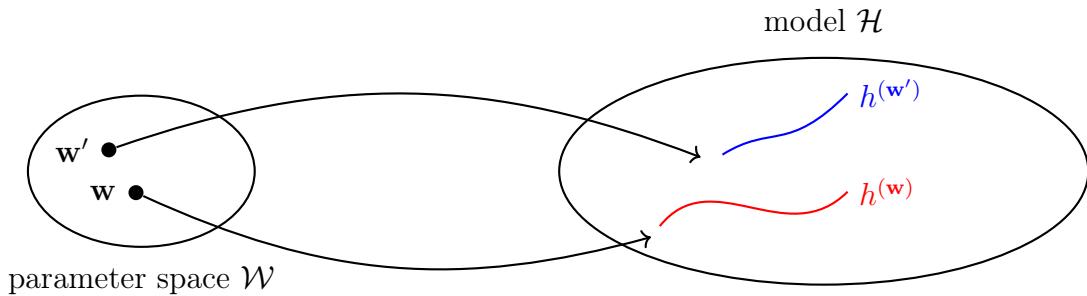


Fig. 113. The parameter space \mathcal{W} of an koneoppiminen model \mathcal{H} consists of all feasible choices for the model parameters. Each choice \mathbf{w} for the model parameters selects a hypothesis kuvaus $h^{(\mathbf{w})} \in \mathcal{H}$.

See also: parameter space, model, model parameter.

penalty term Consider an ERM-based koneoppiminen method that learns model parameters \mathbf{w} by minimizing the average loss (or empiirinen riski) $\widehat{L}(\mathbf{w}|\mathcal{D})$ on a training set \mathcal{D} . To avoid ylisovittaminen, and control the generalization gap, it is common to augment the objective function $\widehat{L}(\mathbf{w}|\mathcal{D})$ with a penalty term $\alpha \mathcal{R}\{\mathbf{w}\}$. We refer to the resulting modified ERM as RERM.

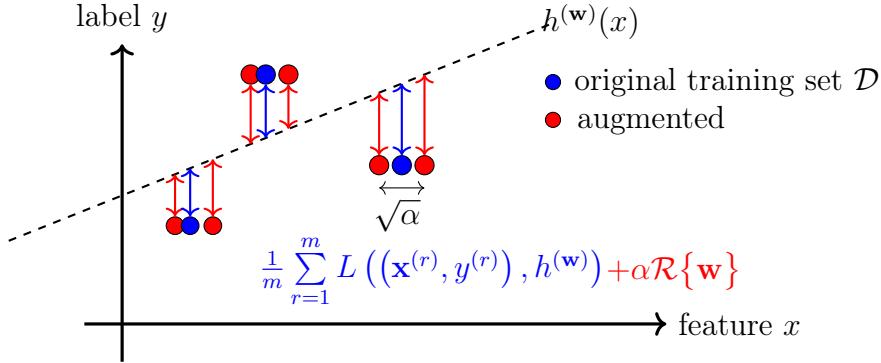


Fig. 114. Adding a penalty term $\alpha\mathcal{R}\{\mathbf{w}\}$ to the objective function in ERM is equivalent to including perturbations of the training set \mathcal{D} during training. The regularization parameter α controls the extend of the perturbations.

The penalty term depends only on the model parameters but not on the data points in the training set. For some combinations of model and loss function, the penalty term can be obtained as the average loss incurred on (an infinite number of) perturbed copies of the training set. In other words, adding a penalty term in ERM can be viewed as a form of data augmentation.

See also: ylisovittainen, RERM, regularization, data augmentation.

perceptron algorithm The perceptron algoritmi is one of the oldest koneop-piminen algoritmit; it was developed by Frank Rosenblatt in 1957 [166]. The perceptron algoritmi works on data with numeric features $\mathbf{x}_r \in \mathbb{R}^d$ and binary labels $y_r \in \{-1, 1\}$ and returns a linear classifier. It is guaranteed to find such a linear classifier if the classes are linearly separable. As the algoritmi is trying to find a hyperplane $g(\mathbf{x}) := \mathbf{w}^\top \mathbf{x}$ separating

the classes, i.e., $\text{sgn}(\mathbf{w}^\top \mathbf{x}_r) = y_r \forall r \in \{1, \dots, m\}$, at each iteration, a sample r' that is wrongly classified, i.e., $\text{sgn}(\mathbf{w}^\top \mathbf{x}'_{r'}) \neq y_{r'}$. Then, the weights \mathbf{w} are updated by $\mathbf{w} \leftarrow \mathbf{w} + y_{r'} \mathbf{x}_{r'}$. While this does not guarantee that the sample is now correctly classified, the error margin has decreased.

See also: binary classification, linear classifier, optimointitehtävä.

perplexity The perplexity of an RV x is defined as e^{H_x} .

See also: entropia.

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

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

that reads in the 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 kuvauskset Φ , and for a quantitative measure of the usefulness of a specific $\Phi \in \mathcal{H}$. For example, PCA uses $\mathcal{X} := \mathbb{R}^d$, $\mathcal{X}' := \mathbb{R}^{d'}$ with $d' < d$, and a hypothesis space

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

PCA measures the usefulness of a specific kuvaus $\Phi(\mathbf{x}) = \mathbf{F}\mathbf{x}$ by the minimi linear reconstruction error incurred on a dataset such that

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

See also: feature, feature space, hypothesis space, PCA.

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

See also: FL, FL network, local model.

polynomial regression Polynomial regressio is an instance of ERM that learns a polynomial hypothesis kuvaus to predict a numeric label based on the numeric features of a data point. For data points characterized by a single numeric feature, polynomial regressio 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 kuvaus is measured using the average neliövirhehäviö incurred on a set of labeled data points (which we refer to as the training set).

See also: regressio, ERM, neliövirhehäviö.

precision Precision is a metric commonly used in binary classification. It measures the proportion of true positives (correctly predicted positive data points) in all the positively predicted data points. I.e., it is the tarkkuus of the positively predicted data points. Precision can also be extended to multiclass luokittelu.

Precision is rarely used in isolation, as it encourages selecting only the most likely data points. Still, unlike tarkkuus, precision remains a valid metric even with imbalanced datasets.

See also: confusion matrix, metric, recall, luokittelu.

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

See also: hypothesis, kuvaus, data point, feature, ennuste, label.

principal component analysis (PCA) Consider a dataset

$$\mathcal{D} = \{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}\}$$

constituted by data points characterized by feature vectors $\mathbf{x}^{(r)} \in \mathbb{R}^d$, for $r = 1, \dots, m$. PCA determines, for a given number $d' < d$ a linear feature map

$$\Phi^{(\mathbf{W})} : \mathbb{R}^d \rightarrow \mathbb{R}^{d'} : \mathbf{x} \mapsto \mathbf{W}\mathbf{x}$$

such that the new feature vectors $\mathbf{z}^{(r)} = \mathbf{W}\mathbf{x}^{(r)}$ allow us to reconstruct the original features with minimi linear reconstruction error [8, 38, 65]

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

We can view PCA as a form of ERM using the loss function $L(\mathbf{x}, \mathbf{W}) = \|\mathbf{x} - \widehat{\mathbf{R}}\mathbf{W}\mathbf{x}\|_2^2$ with a reconstruction matriisi $\widehat{\mathbf{R}}$ that achieves the above minimum reconstruction error. It turns out that this ERM problem can be solved by a matriisi $\mathbf{W} = (\mathbf{u}^{(1)}, \dots, \mathbf{u}^{(d')})^T$ whose rows are given by d' eigenvectors corresponding to the d' largest eigenvalues of the matriisi

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

Note that $\widehat{\mathbf{Q}}$ coincides with the sample covariance matrix of \mathcal{D} if its sample mean vanishes. The psd matriisi $\widehat{\mathbf{Q}}$ allows for an EVD of the

form [29, 67]

$$\widehat{\mathbf{Q}} = \sum_{j=1}^d \lambda_j \mathbf{u}^{(d)} (\mathbf{u}^{(d)})^T = \begin{pmatrix} \mathbf{u}^{(1)} & \dots & \mathbf{u}^{(d)} \end{pmatrix} \begin{pmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_d \end{pmatrix} \begin{pmatrix} (\mathbf{u}^{(1)})^T \\ \vdots \\ (\mathbf{u}^{(d)})^T \end{pmatrix}.$$

This decomposition consists of decreasing nonnegative eigenvalues $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_d \geq 0$ and corresponding eigenvectors $\mathbf{u}^{(1)}, \dots, \mathbf{u}^{(d)}$ that form an orthonormal basis of \mathbb{R}^d .

See also: feature map, piirreoptimointi, ulottuvuuksien vähentäminen, EVD.

privacy funnel The privacy funnel is a method for learning a feature map that provides privacy-friendly features for a data point [167].

See also: data point, feature, GDPR, differentiaalinen yksityisyys.

privacy leakage Consider an koneoppiminen application that processes a dataset \mathcal{D} and delivers some output, such as the ennusteet obtained for new data points. Privacy leakage arises if the output carries information about a private (or sensitive) feature of a data point of \mathcal{D} (such as a human). Based on a todennäköisyyssmalli for the data generation, we can measure the privacy leakage via the MI between the output and the sensitive feature. Another quantitative measure of privacy leakage is differentiaalinen yksityisyys. The relations between different measures of privacy leakage have been studied in the literature (see [168]).

See also: MI, differentiaalinen yksityisyys, yksityisyshyökkäys, GDPR.

Q-learning Q-learning is a popular vahvistusoppiminen algoritmi that learns an optimal policy by estimating the optimal action-value function (or

Q-function) [169].

See also: vahvistusoppiminen, fixed-point iteration.

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

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

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

See also: funktio, matriisi, vektori.

Rademacher complexity Similar to the VC dimension, the Rademacher complexity [170] is a quantitative measure of the size of a hypothesis space \mathcal{H} . It is based on the empirical Rademacher complexity, which is defined for a given dataset \mathcal{D} as

$$\mathcal{R}_{\mathcal{D}}(\mathcal{H}) = \mathbb{E}_{\varepsilon_1, \dots, \varepsilon_m} \sup_{h \in \mathcal{H}} \frac{1}{m} \sum_{r=1}^m \varepsilon_r h(\mathbf{x}^{(r)}).$$

Here, the expectation is taken with respect to the RVs $\varepsilon_1, \dots, \varepsilon_m$, which are i.i.d. and take values in $\{-1, +1\}$ with equal todennäköisyys $1/2$. The Rademacher complexity of \mathcal{H} is then defined as the expectation of the empirical Rademacher complexity of a random dataset $\mathcal{D} = \{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}\}$ that consists of m i.i.d. RVs $\mathbf{x}^{(r)} \in \mathcal{X}$, for $r = 1, \dots, m$.

See also: VC dimension, hypothesis space, generalization, koneoppiminen, effective dimension.

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.

See also: decision tree, dataset.

random projection A random projektio uses a random matriisi $\mathbf{A} \in \mathbb{R}^{d' \times d}$, with $d' < d$, to map a feature vector $\mathbf{x} \in \mathbb{R}^d$ to a shorter feature vector $\mathbf{Ax} \in \mathbb{R}^{d'}$. It is a basic method for piirreoptimointi and ulottuvuuksien vähentäminen. The projektio matriisi \mathbf{A} is typically generated entrywise by i.i.d. RVs with a common probability distribution \mathbb{P} . For a broad class of such probability distributions, a random projektio approximately preserves pairwise Euclidean distances between feature vectors of a given finite dataset. The celebrated Johnson–Lindenstrauss lemma (JL lemma) guarantees the existence of such a distance-preserving ulottuvuuksien vähentäminen kuvaus but does not itself involve randomness. Random projektiot provide a probabilistic construction that realizes this guarantee with high todennäköisyyss. Roughly speaking, for many relevant applications, random projektiot preserve the most relevant information contained in the original (typically very long) feature vector. Fig. 115 illustrates this behavior for an RGB image. The left panel shows the original image. The middle panel shows a masked image where a randomly selected five percent of the original pixels are kept, and the remaining pixels are set to a fixed light-gray color. The right panel shows the result of a simple reconstruction based on repeated averaging of nearby retained pixels.

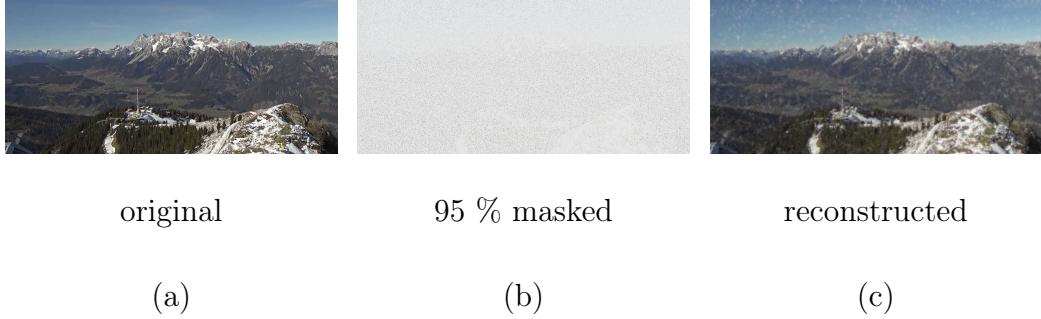


Fig. 115. Illustration of a random projection in the form of removing (or masking) all image pixels, except those in a small random subset. (a) The left panel shows the original RGB image. (b) The middle panel shows a version with only a random five percent subset of pixels retained. (c) The right panel shows a simple convolution-based reconstruction that diffuses information from the known pixels into masked regions.

See also: piirreoptimointi, ulottuvuuksien vähentäminen, JL lemma.

Python demo: click me

realization Consider a RV \mathbf{x} that maps each outcome $\omega \in \Omega$ of a probability space to an element a of a measurable space \mathcal{N} [2], [6], [56]. A realization of \mathbf{x} is any element $\mathbf{a} \in \mathcal{N}$ such that there exists an element $\omega' \in \Omega$ with $\mathbf{x}(\omega') = \mathbf{a}$.

See also: RV, outcome, probability space, measurable.

recall Recall is a metric commonly used in binary classification. It is the ratio of true positives (correctly predicted positive data points) to all actual positives (data points with positive true label). I.e., it tells the proportion of true positives correctly labelled. Recall can also be

extended to multiclass luokittelu.

Recall is not usually used alone, as perfect recall can be achieved by labelling all points as positive. Still, it is robust against imbalanced datasets unlike, e.g., tarkkuus.

See also: confusion matrix, metric, precision, luokittelu.

receiver operating characteristic (ROC) Consider a label space $\mathcal{Y} = \{-1, 1\}$ and a luokitin that uses a real-valued hypothesis $h(\mathbf{x})$. For a given threshold $\eta \in \mathbb{R}$, the ultimate ennuste is $\hat{y} = 1$ if $h(\mathbf{x}) \geq \eta$ and $\hat{y} = -1$ otherwise. On a test set $\mathcal{D}^{(\text{test})}$, we compute, for each value of η , the following two quantities: 1) true positive rate $\text{TPR}^{(\eta)}$; and 2) false positive rate $\text{FPR}^{(\eta)}$. The ROC curve is the funktio [171]

$$\text{ROC} : \mathbb{R} \rightarrow \mathbb{R}^2 : \eta \mapsto (\text{TPR}^{(\eta)}, \text{FPR}^{(\eta)}).$$

One important characteristic of the ROC curve is the area under the curve (AUC).

See also: luokitin, AUC.

rectified linear unit (ReLU) The ReLU is a popular choice for the aktivoointifunktio 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.

See also: aktivoointifunktio, ANN.

recurrent neural network (RNN) An RNN is a specific type of ANN that is designed for processing data that consist of a sequence of tokens. An RNN maintains an internal hidden state that is updated recurrently as new tokens are processed. This recurrent dependence allows information

to propagate across time steps, making RNNs suitable for tasks such as speech recognition, language modeling, or time series ennuste. However, their inherently sequential computation limits parallelization and is challenging for gradient-based methods. Variants like the long short-term memory (LSTM) and gated recurrent unit (GRU) mitigate these problems.

See also: ANN, token.

regressio Regression problems revolve around the ennuste of a numeric label solely from the features of a data point [8, Ch. 2].

See also: ennuste, label, feature, data point.

regularization A key challenge of modern koneoppiminen 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 ylisovittaminen, i.e., 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 ylisovittaminen, i.e., to ensure that the learned hypothesis does not perform 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 higher the expected loss (or riski) is compared with 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 a RV to the feature vector of a data point.

Fig. 116 illustrates the above three routes to regularization. These routes are closely related and sometimes fully equivalent. Data augmentation using normaalijakautuneet satunnaismuuttujat 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.

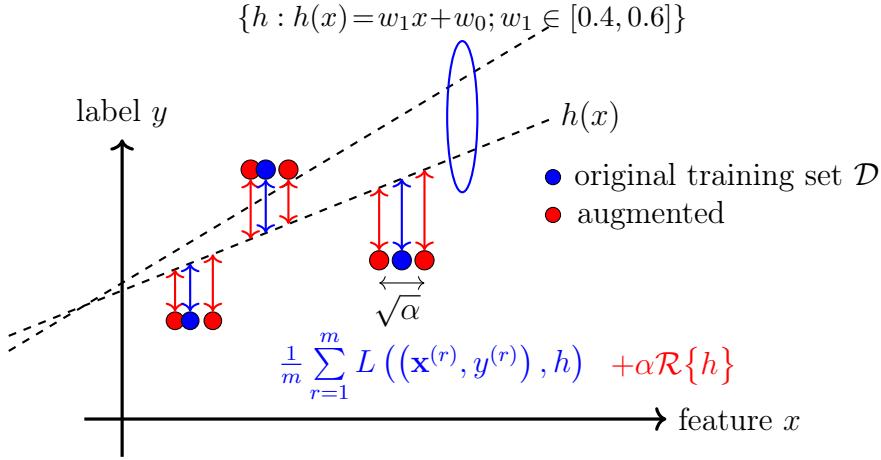


Fig. 116. Three approaches to regularization: 1) data augmentation; 2) loss penalization; and 3) model pruning (via constraints on model parameters).

See also: ylisovittainen, data augmentation, validation, ridge regression, Lasso, model selection.

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

$$\hat{h} \in \arg \min_{h \in \mathcal{H}} \widehat{L}(h|\mathcal{D}) + \alpha \mathcal{R}\{h\}. \quad (25)$$

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

$\mathcal{R}\{h\}$. The component $\alpha\mathcal{R}\{h\}$ in the objective function of (25) 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 lineaarinen malli trained using neliövirhehäviö and the regularizer $\mathcal{R}\{h\} = \|\mathbf{w}\|_2^2$. In this setting, $\alpha\mathcal{R}\{h\}$ corresponds to the expected increase in loss caused by adding normaalijakautuneet satunnaismuuttujat to the feature vectors in the training set [8, Ch. 3]. A principled construction for the regularizer $\mathcal{R}\{h\}$ arises from approximate upper bounds on the generalization error. The resulting RERM instance is known as SRM [172, Sec. 7.2].

See also: ERM, regularization, loss, SRM.

regularized loss minimization (RLM) See RERM.

regularizer A regularizer assigns each hypothesis h from a hypothesis space \mathcal{H} a quantitative measure $\mathcal{R}\{h\}$ conveying to what extent its ennusste errors might differ on data points on and outside a training set. Ridge regression uses the regularizer $\mathcal{R}\{h\} := \|\mathbf{w}\|_2^2$ for linear hypothesis kuvauskset $h^{(\mathbf{w})}(\mathbf{x}) := \mathbf{w}^T \mathbf{x}$ [8, Ch. 3]. Lasso uses the regularizer $\mathcal{R}\{h\} := \|\mathbf{w}\|_1$ for linear hypothesis kuvauskset $h^{(\mathbf{w})}(\mathbf{x}) := \mathbf{w}^T \mathbf{x}$ [8, Ch. 3].

See also: ridge regression, Lasso, loss, objective function.

response The term response is sometimes used as a synonym for the label of a data point [65].

See also: label, data point, target.

response vector The term response vektori is used as a synonym for the label vector of a dataset [65].

See also: response, label vector, dataset, target vector.

reward A reward refers to some observed (or measured) quantity that allows us to estimate the loss incurred by the ennuste (or decision) of a hypothesis $h(\mathbf{x})$. For example, in an koneoppiminen 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 toward an obstacle. We define a low reward for the steering direction $h(\mathbf{x})$ if the vehicle moves dangerously toward an obstacle.

See also: loss, monikäytinen rosvo, vahvistusoppiminen.

ridge regression Consider a regressio problem where the goal is to learn a hypothesis $h^{(\mathbf{w})}$ for predicting the numeric label of a data point based on its feature vector. Ridge regressio learns the parameters \mathbf{w} by minimizing the penalized average neliövirhehäviö. The average neliövirhehäviö is measured on a set of labeled data points (i.e., the training set)

$$(\mathbf{x}^{(1)}, y^{(1)}) , \dots , (\mathbf{x}^{(m)}, y^{(m)}) .$$

The penalty term is the scaled squared Euclidean norm $\alpha \|\mathbf{w}\|_2^2$ with a regularization parameter $\alpha > 0$. The purpose of the penalty term is regularization, i.e., to prevent ylisovittaminen in the high-dimensional regime, where the number of features d exceeds the number of data points m in the training set. For the training of a lineaarinen malli,

adding $\alpha\|\mathbf{w}\|_2^2$ to the average neliövirhehäviö is equivalent to computing the average neliövirhehäviö on an augmented training set.

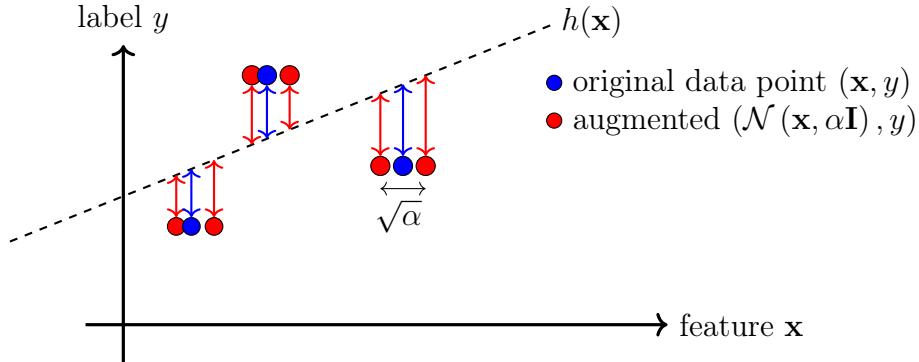


Fig. 117. For a lineaarinen malli, adding the penalty term $\alpha\|\mathbf{w}\|_2^2$ to the objective function in ERM is equivalent to ERM on an augmented dataset

This augmented training set is obtained by replacing each data point $(\mathbf{x}^{(r)}, y^{(r)})$ in the original training set by the realization of infinitely many i.i.d. RVs whose probability distribution is centered around $(\mathbf{x}^{(r)}, y^{(r)})$. See also: regressio, regularization, kuvaus, data augmentation.

risk stratification Risk stratification assigns data points to risk groups (or strata) by quantizing the ennuste $\hat{h}(\mathbf{x})$ obtained from a trained risk prognosis model. Typical choices for these strata include low, intermediate, and high risk [173], [174].

See also: stratification, ennuste, clustering.

riski 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 ennuste using a loss function $L((\mathbf{x}, y), h)$. If we interpret data points as the

realizations of i.i.d. RVs, the $L((\mathbf{x}, y), h)$ also becomes the realization of an RV. The i.i.d. assumption allows us to define the risk of a hypothesis as the expected loss $\mathbb{E}\{L((\mathbf{x}, y), h)\}$. Note that the risk of h depends on both the specific choice for the loss function and the probability distribution of the data points.

See also: i.i.d. RV, i.i.d. assumption, loss, probability distribution.

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

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

See also: sample, keskiarvo, dataset, feature vector.

sample size The number of individual data points contained in a sample or dataset. Consider a ERM-based method that uses a training set \mathcal{D} with sample size m and a model \mathcal{H} with effective dimension $d_{\text{eff}}(\mathcal{H})$. If the training set can be well-approximated by the i.i.d. assumption, then the ratio between m and $d_{\text{eff}}(\mathcal{H})$ can be a useful indicator for the occurrence of ylisovittaminen [8, Ch. 6].

See also: data point, dataset.

sample space A sample space is the set of all possible outcomes of a satunnaiskoe [6], [7], [25], [26].

See also: probability space.

sample weighting Consider an ERM-based method that learns a hypothesis by minimizing the average loss on a training set. In its basic form, ERM

treats all data points equally important. However, in some applications, it can be useful to put different emphasis on the errors obtained for different data points. For example, if a data point is considered an outlier we should reduce its influence on the learned hypothesis. We can implement this idea by assigning a nonnegative weight $q^{(r)}$ to each data point $(\mathbf{x}^{(r)}, y^{(r)})$ in the training set. This results in the weighted ERM principle

$$\min_{h \in \mathcal{H}} \sum_{r=1}^m q^{(r)} L((\mathbf{x}^{(r)}, y^{(r)}), h). \quad (26)$$

Fig. 118 illustrates the concept of a training set of three data points that contribute unequally to the empirical risk.

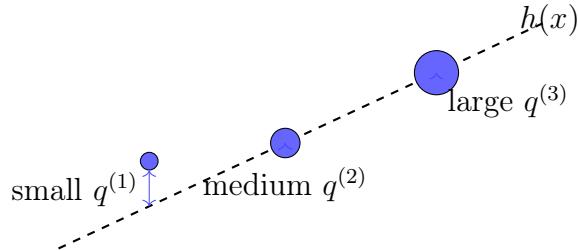


Fig. 118. Sample weighting assigns each data point of a training set a weight $q^{(r)}$. Assigning a small weight (such as $q^{(1)}$ in this example) to a data point decreases its influence on the hypothesis learned via solving (26).

See also: ERM, outlier, AdaBoost.

satunnaisalgoritmi A stokastinen algoritmi uses a random mechanism during its execution. For example, SGD uses a randomly selected subset of data points to compute an approximation for the gradientti of an

objective function. We can represent a stokastinen algoritmi by a satunnaisprosessit, i.e., the possible execution sequence is the possible outcomes of a satunnaiskoe [7], [181], [182].

See also: stokastinen, algoritmi, SGD, data point, gradientti, objective function, satunnaisprosessi, satunnaiskoe, optimointimenetelmä, gradient-based method.

scatterplot A visualization technique that depicts data points using markers in a 2-D plane. Fig. 119 depicts an example of a scatterplot.

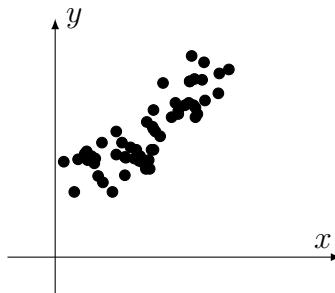


Fig. 119. A scatterplot with circle markers, where the data points represent daily weather conditions in Finland. Each data point is characterized by its minimi daytime temperature x as the feature and its maksimi daytime temperature y as the label. The temperatures have been measured at the FMI weather station Helsinki Kaisaniemi during 1 September 2024—28 October 2024.

A scatterplot can enable the visual inspection of data points that are naturally represented by feature vectors in high-dimensional spaces.

See also: data point, minimi, feature, maksimi, label, FMI, feature vector, ulottuvuuksien vähentäminen.

self-supervised learning Self-supervised learning uses some of the features

of a data point as its label. For example, if a data point consists of a sentence within a text document, we can use the last word of the sentence as the label that is to be predicted from all the previous words, which form the features of the data point. A main application of self-supervised learning is in NLP for the training of LLMs from large collections of text data.

See also: feature, label, LLM.

selitettävyys We define the (subjective) explainability of an *koneoppiminen* method as the level of simulability [118] of the *ennusteet* delivered by an *koneoppiminen* system to a human user. Quantitative measures of the (subjective) explainability of a trained model can be constructed by comparing its *ennusteet* with the *ennusteet* provided by a user on a test set [118], [119]. Alternatively, we can use *todennäköisyysmallit* for data and measure the explainability of a trained *koneoppiminen* model via the conditional (or differential) entropia of its *ennusteet*, given the user's *ennusteet* [120], [121].

See also: luotettava tekoäly, regularization.

selitys One approach to enhance the transparency of an *koneoppiminen* method for its human user is to provide an explanation alongside the *ennusteet* delivered by the method. Explanations can take different forms. For instance, they may consist of human-readable text or quantitative indicators, such as feature importance scores for the individual features of a given data point [123]. Alternatively, explanations can be visual—for example, intensity kuvaukset that highlight image regions that drive

the ennuste [124]. Fig. 120 illustrates two types of explanations. The first is a local linear approximation $g(\mathbf{x})$ of a nonlinear trained model $\hat{h}(\mathbf{x})$ around a specific feature vector \mathbf{x}' , as used in the method LIME. The second form of explanation depicted in the figure is a sparse set of ennusteet $\hat{h}(\mathbf{x}^{(1)}), \hat{h}(\mathbf{x}^{(2)}), \hat{h}(\mathbf{x}^{(3)})$ at selected feature vectors, offering concrete reference points for the user.

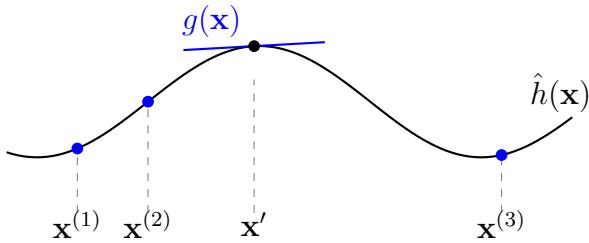


Fig. 120. A trained model $\hat{h}(\mathbf{x})$ can be explained locally at some point \mathbf{x}' by a linear approximation $g(\mathbf{x})$. For a differentiable $\hat{h}(\mathbf{x})$, this approximation is determined by the gradientti $\nabla\hat{h}(\mathbf{x}')$. Another form of explanation could be the funkto values $\hat{h}(\mathbf{x}^{(r)})$ for $r = 1, 2, 3$.

See also: koneoppiminen, ennuste, feature, data point, luokittelu.

semi-supervised learning (SSL) SSL methods use unlabeled data points to support the learning of a hypothesis from labeled data points [90]. This approach is particularly useful for koneoppiminen applications that offer a large number of unlabeled data points, but only a limited number of labeled data points.

See also: data point, hypothesis, labeled data point, koneoppiminen.

sensitive attribute koneoppiminen revolves around learning a hypothesis

kuvaus 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 *koneoppiminen* 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.

See also: *koneoppiminen*, hypothesis, kuvaus, label, data point, feature.

similarity graph Some *koneoppiminen* applications generate data points that are related by a domain-specific notion of similarity. These similarities can be represented conveniently using a similarity verkko $\mathcal{G} = (\mathcal{V} := \{1, \dots, m\}, \mathcal{E})$. 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.

See also: *koneoppiminen*, data point, verkko.

skip connection Consider a deep net with neurons that are organized in consecutive layers. A skip connection links the output of a neuron in some layer to the input of a neuron in a non-consecutive layer [82].

See also: deep net, DAG.

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 for data points characterized by numerical feature vectors is via a todennäköisyysmalli such as the GMM. The conditional

todennäköisyys of a data point belonging to a specific mixture component is then a natural choice for the degree of belonging. GMM soft clustering methods can be applied to non-numeric data by using piirreoptimointi methods to provide numerical features (such as in spectral clustering).

See also: clustering, cluster, degree of belonging, GMM, spectral clustering.

soft label Consider a luokittelu problem where data points are characterized by features \mathbf{x} and labels from a finite label space $\mathcal{Y} = \{1, \dots, k\}$. Some koneoppiminen applications involve data points that have almost identical features but different labels. In such cases, instead of assigning to each data point a single label $y \in \mathcal{Y}$, it can be more useful to assign an entire probability distribution over the label space. This probability distribution can be represented as a pmf $\mathbf{y} = (y_1, \dots, y_k)^T \in \Delta^k$. We can view the entries y_c , for $c = 1, \dots, k$, as soft labels of a data point. Mathematically, the soft label y_c is the todennäköisyys that a randomly chosen data point with feature vector \mathbf{x} has label c .

See also: luokittelu, pmf, cross-entropy.

spectral clustering Spectral clustering is a particular instance of graph clustering, i.e., it clusters data points represented as the nodes $i = 1, \dots, n$ of a verkko \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) $i = 1, \dots, n$. We can feed these feature vectors into Euclidean space-based clustering methods, such as k -means

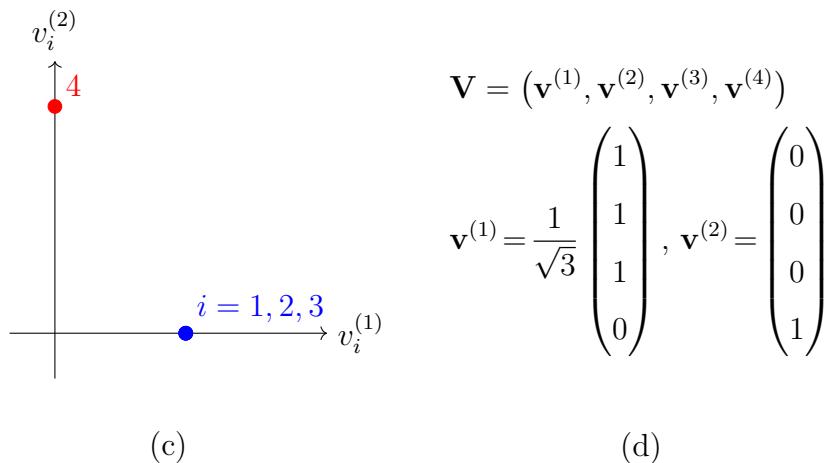
or soft clustering via GMM. Roughly speaking, the feature vectors of nodes belonging to a well-connected subset (or cluster) of nodes in \mathcal{G} are located nearby in the Euclidean space \mathbb{R}^d (see Fig. 121).

$$i = 1$$

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

(a)

(b)



(c)

(d)

Fig. 121. (a) An undirected verkko \mathcal{G} with four nodes $i = 1, 2, 3, 4$, each representing a data point. (b) The Laplacian matrix $\mathbf{L}^{(\mathcal{G})} \in \mathbb{R}^{4 \times 4}$ and its EVD. (c) A scatterplot of data points using the feature vectors $\mathbf{x}^{(i)} = (v_i^{(1)}, v_i^{(2)})^T$. (d) Two eigenvectors $\mathbf{v}^{(1)}, \mathbf{v}^{(2)} \in \mathbb{R}^d$ corresponding to the eigenvalue $\lambda = 0$ of the Laplacian matrix $\mathbf{L}^{(\mathcal{G})}$.

See also: clustering, graph clustering, Laplacian matrix, eigenvalue.

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) [175]. Fig. 122 depicts a time signal along with its spectrogram.

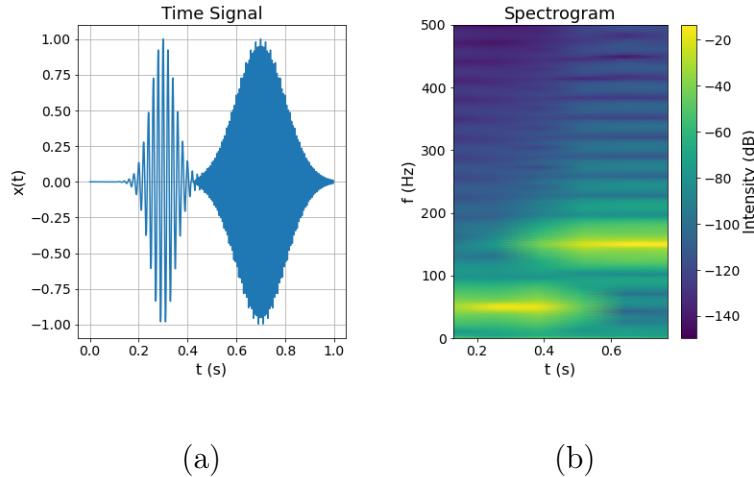


Fig. 122. (a) A time signal consisting of two modulated Gaussian pulses. (b) An intensity plot of the spectrogram.

The intensity plot of its spectrogram can serve as an image of a signal. A simple recipe for audio signal luokittelu is to feed this signal image into deep nets originally developed for image luokittelu and object detection [176]. It is worth noting that, beyond the spectrogram, several alternative representations exist for the time-frequency distribution of

signal energy [177], [178].

See also: luokittelu, deep net.

stability Mathematically, an *koneoppiminen* method is a *kuvaus* \mathcal{A} from a given dataset \mathcal{D} to an output $\mathcal{A}(\mathcal{D})$. As a case in point, consider an ERM-based *koneoppiminen* method that maps a training set \mathcal{D} to the learned model parameters $\mathcal{A}(\mathcal{D}) = \hat{\mathbf{w}}$, which achieve the minimi average loss on the training set. Instead of the learned model parameters, the output $\mathcal{A}(\mathcal{D})$ could also be the ennusteet obtained from the trained model. Stability refers to the desirable property of \mathcal{A} that small changes in the input dataset \mathcal{D} result in small changes in the output $\mathcal{A}(\mathcal{D})$. The notion of stability is intimately related to the notion of generalization. In particular, there are formal notions of stability that allow us to bound the generalization gap (see [132, Ch. 13]). To build intuition, consider the three datasets depicted in Fig. 123, 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 *koneoppiminen* 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.

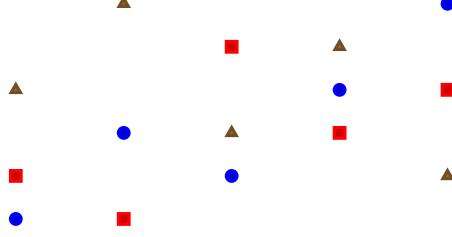


Fig. 123. Three datasets $\mathcal{D}^{(*)}$, $\mathcal{D}^{(\square)}$, and $\mathcal{D}^{(\triangle)}$, each sampled independently from the same data-generating probability distribution. A stable koneoppi-minen method should return similar outputs when trained on any of these datasets.

See also: generalization, vakaus.

stacking Stacking is one of the main types of ensemble methods. In stacking, a finite number M of base learners are trained on the same dataset but with different models $\mathcal{H}^{(j)}$ or loss functiontont $L^{(j)}$, for $j = 1, \dots, M$ [65, Ch. 8.8], [179], [180]. The j th base learner delivers a learned hypothesis $\hat{h}^{(j)} \in \mathcal{H}^{(j)}$. The final ennuste for a data point is obtained by aggregating the ennusteet of the base learners via an aggregation rule ϕ_{agg} , such as majority voting for luokittelu or averaging for regressio. We can interpret stacking as a form of piirreoptimointi, where each base learner extracts a new feature. The aggregation rule can be obtained by another instance of ERM that learns a hypothesis kuvaus $\phi_{\text{agg}} \in \mathcal{H}$ from a meta-model \mathcal{H} . The hypothesis ϕ_{agg} is applied to the transformed feature vector

$$\Phi(\mathbf{x}) = (\hat{h}^{(1)}(\mathbf{x}), \dots, \hat{h}^{(M)}(\mathbf{x}))^T.$$

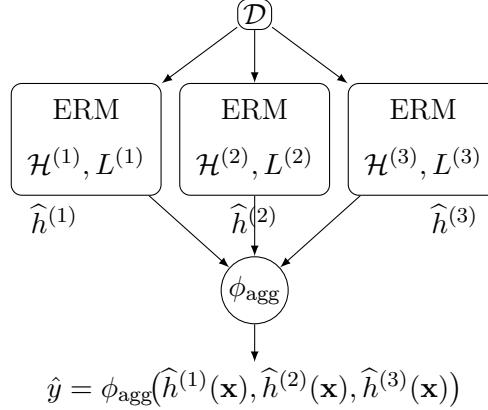


Fig. 124. Three base learners using ERM with different models and loss functiont to obtain learned hypotheses $\hat{h}^{(1)}, \hat{h}^{(2)}, \hat{h}^{(3)}$. For a data point with feature vector \mathbf{x} , each base learner delivers a ennuste $\hat{h}^{(j)}(\mathbf{x})$, for $j = 1, 2, 3$. These ennusteet are then used as new features for an aggregation rule ϕ_{agg} that delivers the overall ennuste \hat{y} . The aggregation rule can be obtained by training a meta-model \mathcal{H} .

See also: ensemble, bagging.

step size See learning rate.

stochastic block model (SBM) The SBM [183] is a probabilistic generati-
ve model for an undirected verkko $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with a given set of nodes
 \mathcal{V} [184]. In its most basic variant, the SBM generates a verkko by first
randomly assigning each node $i \in \mathcal{V}$ to a cluster index $c_i \in \{1, \dots, k\}$.
A pair of different nodes in the verkko is connected by an edge with to-
dennäköisyys $p_{i,i'}$ that depends solely on the labels $c_i, c_{i'}$. The presence
of edges between different pairs of nodes is statistically independent.

See also: model, verkko, cluster, todennäköisyys, label.

stochastic gradient descent (SGD) SGD is obtained from GD by replacing the gradientti of the objective function with a stokastinen approximation. A main application of SGD is to train a parameterized 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 globally). To evaluate the gradientti of the empiirinen riski (as a funktio of the model parameters \mathbf{w}), we need to compute a sum $\sum_{r=1}^m \nabla_{\mathbf{w}} L(\mathbf{z}^{(r)}, \mathbf{w})$ over all data points in the training set. We obtain a stokastinen approximation to the gradientti by replacing the sum $\sum_{r=1}^m \nabla_{\mathbf{w}} L(\mathbf{z}^{(r)}, \mathbf{w})$ with a sum $\sum_{r \in \mathcal{B}} \nabla_{\mathbf{w}} L(\mathbf{z}^{(r)}, \mathbf{w})$ over a randomly chosen subset $\mathcal{B} \subseteq \{1, \dots, m\}$ (see Fig. 125). We often refer to these randomly chosen data points as a batch. The batch size $|\mathcal{B}|$ is an important parameter of SGD. SGD with $|\mathcal{B}| > 1$ is referred to as mini-batch SGD [155].

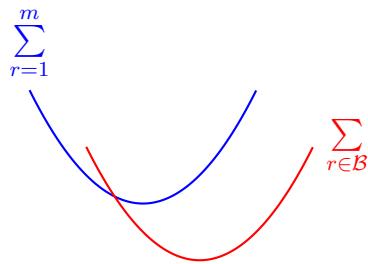


Fig. 125. SGD for ERM approximates the gradientti by replacing the sum $\sum_{r=1}^m \nabla_{\mathbf{w}} L(\mathbf{z}^{(r)}, \mathbf{w})$ over all data points in the training set (indexed by $r = 1, \dots, m$) with a sum over a randomly chosen subset $\mathcal{B} \subseteq \{1, \dots, m\}$.

See also: GD, gradientti, objective function, stokastinen, model, ERM,

training set, data point, empiirinen riski, funktio, model parameter, batch, parameter.

stopping criterion Many koneoppiminen methods use iterative algoritmit that construct a sequence of model parameters in order to minimize the training error. For example, gradient-based methods iteratively update the parameters of a parametric model, such as a lineaarinen malli or a deep net. Given a finite amount of computational resources, we need to stop updating the parameters after a finite number of iterations. A stopping criterion is any well-defined condition for deciding when to stop updating.

See also: algoritmi, gradient-based method.

stratification The process of splitting a dataset into subsets, so called strata,

according to some key attribute is called stratification [69], [70], [185].

The goal is to ensure that an koneoppiminen method performs well for each stratum defined by these attributes. For example, in a medical dataset, we may want to stratify a patient dataset by age groups to ensure that an koneoppiminen model performs well across all age groups.

When splitting a dataset into a training set and a validation set, stratification ensures that both sets have similar distributions of the key attribute. Without stratification, using a small validation set may underrepresent or even completely miss data points with a rare attribute, leading to misleading performance estimates. See Fig. 126 for a visual illustration.

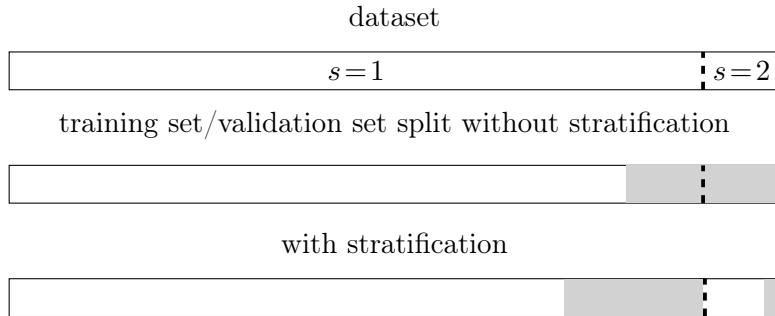


Fig. 126. Stratification ensures that both the training set and the validation set (shaded grey) have similar distributions of a binary key attribute s . In other words, with stratification, both strata (i.e., $s = 1$ and $s = 2$ based on the key attribute) allocate the same validation proportion—20% of their own width.

See also: stratum, validation, k -fold CV.

stratum A stratum is a subset of data points that all share a common property (which could be a feature or a label). For example, in a weather dataset, all measurements from the same FMI weather station form one stratum.

Example (CSV snippet):

```
time, station, value, unit
2023-06-01 12:00, Helsinki, 18.2, degree Celsius
2023-06-01 13:00, Helsinki, 18.5, degree Celsius
2023-06-01 14:00, Helsinki, 19.0, degree Celsius
2023-06-01 12:00, Oulu, 12.1, degree Celsius
2023-06-01 13:00, Oulu, 12.4, degree Celsius
```

2023-06-01 14:00, Oulu, 12.7, degree Celsius
2023-06-01 12:00, Tampere, 15.3, degree Celsius
2023-06-01 13:00, Tampere, 15.6, degree Celsius
2023-06-01 14:00, Tampere, 16.0, degree Celsius

Here, the rows for each station (i.e., Helsinki, Oulu, Tampere) represent different strata.

See also: data point, dataset, stratification.

structural risk minimization (SRM) SRM is an instance of RERM, with

which the model \mathcal{H} can be expressed as a countable union of submodels such that $\mathcal{H} = \bigcup_{n=1}^{\infty} \mathcal{H}^{(n)}$. 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 [132, Sec. 7.2].

See also: RERM, model, generalization, ERM, regularizer, riski.

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

See also: subgradient, generalization, GD, funkto, gradientti, objective

function, empiirinen riski, model parameter, hypothesis.

support vector machine (SVM) The SVM is a binary luokittelu method that learns a linear hypothesis kuvaus. Thus, like linear regression and logistic regression, it is also an instance of ERM for the lineaarinen malli. However, the SVM uses a different loss function from the one used in those methods. As illustrated in Fig. 127, it aims to maximally separate data points from the two different classes in the feature space (i.e., maksimi margin principle). Maximizing this separation is equivalent to minimizing a regularized variant of the hinge loss (18) [38], [138], [186].

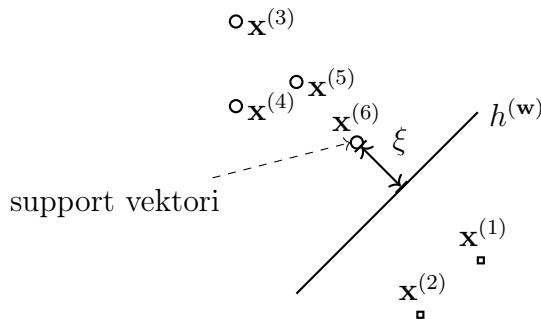


Fig. 127. The SVM learns a hypothesis (or luokitin) $h^{(\mathbf{w})}$ with minimal average soft-margin hinge loss. Minimizing this loss is equivalent to maximizing the margin ξ between the decision boundary of $h^{(\mathbf{w})}$ and each class of the training set.

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

See also: luokittelu, lineaarinen malli, luokitin, hinge loss.

target The term target is sometimes used as a synonym for the label of a data point [38], [81].

See also: label, data point.

target vector The term target vektori is used as a synonym for the label vector of a dataset [38], [81].

See also: target, label vector, dataset.

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

See also: binääritappio, loss, metric.

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

See also: data point, model, ERM, validation set.

tilastolliset ominaisuudet By statistical aspects of an koneoppiminen method, we refer to (properties of) the probability distribution of its output under a todennäköisyysmalli for the data fed into the method.

See also: koneoppiminen, probability distribution, todennäköisyysmalli, data.

todennäköisyys We assign a probability value, typically chosen in the interval $[0, 1]$, to each event that can occur in a satunnaiskoe [6], [7], [56], [20].

See also: event, satunnaiskoe.

token A token is a basic unit of information obtained by splitting a sequence of symbols, such as a text string, into smaller parts. In NLP, tokens often correspond to words, subwords, or characters that form the features of a data point. Tokenization transforms raw text (e.g., “The cat sleeps”) into a sequence of tokens (e.g., [“The”, “cat”, “sleeps”]), which can then be mapped to numerical feature vectors.

See also: sequence, feature vector.

total variation See GTV.

training In the context of koneoppiminen, training refers to the process of learning a useful hypothesis \hat{h} out of a model \mathcal{H} . The training of a model \mathcal{H} is guided by the loss incurred on a set of data points, which serve as the training set. For parametric models, where each hypothesis $h^{(\mathbf{w})}$ is characterized by a specific choice for the model parameters, training amounts to finding an optimal choice for the model parameters \mathbf{w} . A widely-used approach to training is ERM, which learns a hypothesis by minimizing the average loss incurred on a training set. One of the main challenges in koneoppiminen is to control the poikkeama between the loss incurred on the training set and the loss incurred on other (unseen) data points.

See also: model, loss, ERM.

training error The average loss of a hypothesis when predicting the labels of the data points in a training set. We sometimes also refer to training error as the minimal average loss that is achieved by a solution of ERM. See also: loss, hypothesis, label, data point, training set, ERM.

training set A training set is a dataset \mathcal{D} that 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 koneoppiminen method and informs how to improve the validation error (e.g., using a different hypothesis space or collecting more data points) [8, Sec. 6.6].

See also: training, dataset, data point, ERM, hypothesis, loss, training error, validation error, koneoppiminen, hypothesis space.

transfer learning Transfer learning aims at leveraging information obtained while solving an existing oppimistehtävä to solve another oppimistehtävä. See also: oppimistehtävä, monitehtäväoppiminen

transformer In the context of koneoppiminen, the term transformer refers to an ANN that uses some form of attention mechanism to capture dependencies among tokens [84]. The attention mechanism is what sets transformers apart from previous models used for sequential data such as recurrent neural networks (RNNs). A transformer ANN often combines several attention layers via more traditional layer architectures.

See also: attention, NLP.

tulkittavuus An koneoppiminen method is interpretable for a human user if

they can comprehend the decision process of the method. One approach to develop a precise definition of interpretability is via the concept of simulatability, i.e., the ability of a human to mentally simulate the model behavior [118], [121], [141], [142], [143]. The idea is as follows: If a human user understands an *koneoppiminen* method, then they should be able to anticipate its ennustheet on a test set. We illustrate such a test set in Fig. 128, which also depicts two learned hypotheses \hat{h} and \hat{h}' . The *koneoppiminen* method producing the hypothesis \hat{h} is interpretable to a human user familiar with the concept of a lineaarikuvaus. Since \hat{h} corresponds to a lineaarikuvaus, the user can anticipate the ennustheet of \hat{h} on the test set. In contrast, the *koneoppiminen* method delivering \hat{h}' is not interpretable, because its behavior is no longer aligned with the user's expectations.

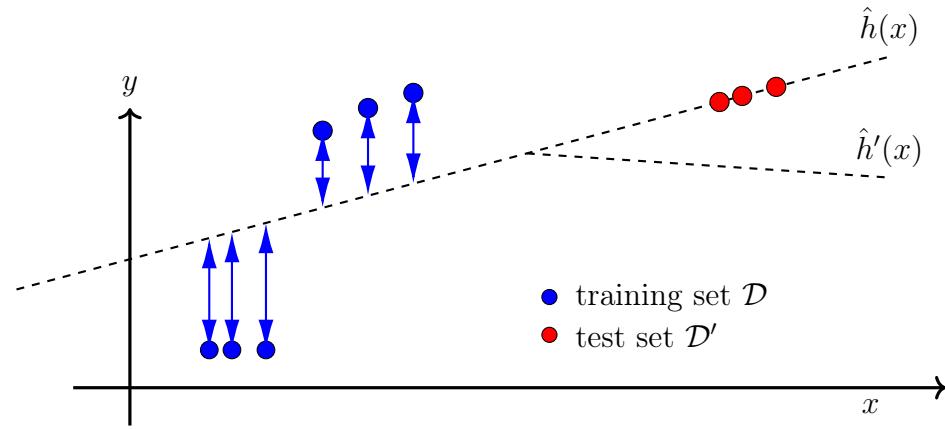


Fig. 128. We can assess the interpretability of trained *koneoppiminen* models \hat{h} and \hat{h}' by comparing their ennustheet to pseudo-labels generated by a human user for \mathcal{D}' .

The notion of interpretability is closely related to the notion of selitettävyys, as both aim to make koneoppiminen methods more understandable for humans. In the context of Fig. 128, interpretability of an koneoppiminen method \hat{h} requires that the human user can anticipate its ennusteet on an arbitrary test set. This contrasts with selitettävyys, where the user is supported by external selitteet—such as saliency kuvaaukset or reference examples from the training set—to understand the ennusteet of \hat{h} on a specific test set \mathcal{D}' .

See also: selitettävyys, luotettava tekoäly, regularization, LIME.

ulottuvuuksien vähentäminen Dimensionality reduction refers to methods that learn a transformation $h : \mathbb{R}^d \rightarrow \mathbb{R}^{d'}$ of a (typically large) set of raw features x_1, \dots, x_d into a smaller set of informative features $z_1, \dots, z_{d'}$. Using a smaller set of features is beneficial in several ways:

- Statistical benefit: It typically reduces the riski of ylisovittaminen, as reducing the number of features often reduces the effective dimension of a model.
- Computational benefit: Using fewer features means less computation for the training of koneoppiminen models. As a case in point, linear regression methods need to invert a matriisi whose size is determined by the number of features.
- Visualization: Dimensionality reduction is also instrumental for data visualization. For example, we can learn a transformation that delivers two features z_1, z_2 , which we can use, in turn, as the coordinates of a scatterplot. Fig. 129 depicts the scatterplot

of handwritten digits that are placed using transformed features. Here, the data points are naturally represented by a large number of greyscale values (one value for each pixel).

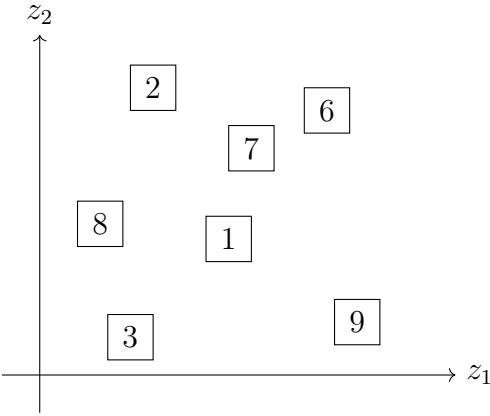


Fig. 129. Example of dimensionality reduction: High-dimensional image data (e.g., high-resolution images of handwritten digits) embedded into 2-D using learned features (z_1, z_2) and visualized in a scatterplot.

See also: [ylisovittaminen](#), [autoenkoodaaja](#), [random projection](#), [PCA](#), [JL lemma](#).

upper confidence bound (UCB) Consider an [koneoppiminen](#) application that requires selecting, at each time step t , an action a_t from a finite set of alternatives \mathcal{A} . The utility of selecting action a_t is quantified by a numeric reward signal $r^{(a_t)}$. A widely used [todennäköisyysmalli](#) for this type of sequential decision-making problem is the [stokastinen monikäätinen rosvo](#) setting [163]. In this model, the reward $r^{(a)}$ is viewed as the realization of a RV with unknown [keskiarvo](#) $\mu^{(a)}$. Ideally, we would

always choose the action with the largest expected reward $\mu^{(a)}$, but these keskiarvot are unknown and must be estimated from observed data. Simply choosing the action with the largest estimate $\hat{\mu}^{(a)}$ can lead to suboptimal outcomes due to estimation epävarmuus. The UCB strategy addresses this by selecting actions not only based on their estimated keskiarvot but also by incorporating a term that reflects the epävarmuus in these estimates—favoring actions with a high-potential reward and high epävarmuus. Theoretical guarantees for the performance of UCB strategies, including logarithmic regret bounds, are established in [163]. See also: optimism in the face of uncertainty, reward, monikäinen rosvo, epävarmuus, regret.

uusio-otanta For the analysis of koneoppiminen methods, it is often useful to interpret a given dataset $\mathcal{D} = \{\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(m)}\}$ as a collection of (realizations of) i.i.d. RVs with common probability distribution \mathbb{P} . In practice, the probability distribution \mathbb{P} is unknown and must be estimated from \mathcal{D} . The idea of the bootstrap method is to use the empirical distribution $\mathbb{P}^{(\mathcal{D})}$ of \mathcal{D} as an estimator for \mathbb{P} [65],

$$(1/m)|r : \mathbf{z}^{(r)} \in \mathcal{A}| \approx \mathbb{P}(\mathcal{A}).$$

Repeatedly sampling from the empirical distribution, which is equivalent to sampling with replacement from \mathcal{D} [89], results in new datasets $\mathcal{D}^{(1)}, \dots, \mathcal{D}^{(B)}$, each containing m data points. We then use each of those datasets for model training (e.g., via ERM), resulting in the learned hypotheses $\hat{h}^{(1)}, \dots, \hat{h}^{(B)}$. We can use these learned hypotheses to estimate important characteristics of an koneoppiminen method such

as harha, varianssi, or generalization gap [65].

See also: i.i.d., RV, probability distribution, histogrammi.

vakaus Robustness is a key requirement for luotettava tekoäly. It refers to the property of an koneoppiminen system to maintain acceptable performance even when subjected to different forms of perturbations. These perturbations may affect the features of a data point in order to manipulate the ennuste delivered by a trained koneoppiminen model. Robustness also includes the stability of ERM-based methods against perturbations of the training set. Such perturbations can occur within data poisoning hyökkäykset.

See also: luotettava tekoäly, stability, data poisoning, hyökkäys.

validation Consider a hypothesis \hat{h} that has been learned via some koneoppiminen method, e.g., by solving ERM on a training set \mathcal{D} . Validation refers to the process 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} . This set of data points is called the validation set. The average loss of \hat{h} on the validation set is referred to as the validation error. An example of validation is shown in Fig. 130.

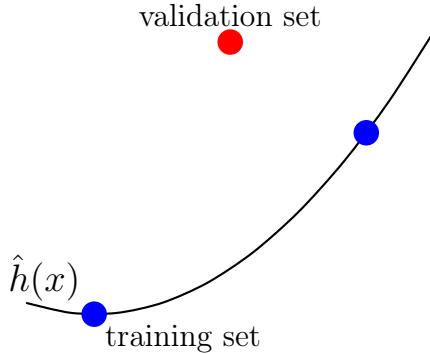


Fig. 130. Illustration of validation. The blue points represent the data points in the training set, while the red point represents a data point in the validation set. The hypothesis \hat{h} (black curve) fits the data points in the training set perfectly, but incurs a large loss on the data point in the validation set.

See also: training set, validation set, validation error, ylisovittaminen, generalization, k -fold CV, LOO CV.

validation error Consider a hypothesis \hat{h} that is obtained by some koneoppiminen 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.

See also: hypothesis, koneoppiminen, ERM, training set, loss, validation set, validation.

validation set A set of data points used to estimate the riski of a hypothesis \hat{h} that has been learned by some koneoppiminen 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 koneoppiminen method

(see [8, Sec. 6.6]). The comparison between training error and validation error can inform directions for the improvement of the koneoppiminen method (such as using a different hypothesis space).

See also: data point, riski, hypothesis, koneoppiminen, ERM, loss, validation, validation error, training error, hypothesis space, k -fold CV, LOO CV.

Vapnik–Chervonenkis dimension (VC dimension) The statistical properties of an ERM-based method depend critically on the expressive capacity of its hypothesis space (or model) \mathcal{H} . A standard measure of this capacity is the VC dimension $\text{VCdim}(\mathcal{H})$ [189]. Formally, it is the largest integer m such that there exists a dataset $\mathcal{D} = \{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}\} \subseteq \mathcal{X}$ that can be perfectly classified (or shattered) by some $h \in \mathcal{H}$. Formally, this means that for every one of the 2^m possible assignments of binary labels to each feature vector in \mathcal{D} , there exists some hypothesis $h \in \mathcal{H}$ that realizes this labeling. Intuitively, the VC dimension quantifies how well \mathcal{H} can fit arbitrary label assignments, and thus captures its approximate power. It plays a central role in deriving bounds on the generalization gap. Fig. 131 illustrates the definition of the VC dimension for a lineaarinen malli $\mathcal{H}^{(2)}$ with $d = 2$ features. Fig. 131(a) and 131(b) show the same set of three noncollinear feature vectors under two different binary labelings. In both cases, a separating hyperplane exists that realizes the labeling. Since this holds for all $2^3 = 8$ possible binary labelings of the three feature vectors, the set is shattered. Fig. 131(c) depicts four feature vectors with a specific labeling. No linear separator can correctly classify all data points in this case. Thus, $\text{VCdim}(\mathcal{H}^{(2)}) = 3$.

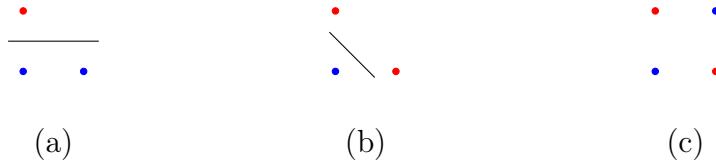


Fig. 131. Illustration of the VC dimension for a lineaarinen malli $\mathcal{H}^{(2)}$ that is used to learn a linear classifier in the feature space \mathbb{R}^2 .

More generally, for a lineaarinen malli $\mathcal{H}^{(d)}$, the VC dimension equals $d+1$. In other words, for lineaariset mallit, the VC dimension essentially matches the dimension of the underlying parameter space \mathbb{R}^d . For more complex hypothesis spaces, such as decision trees or ANNs, the relation between VC dimension and the dimension of the feature space is far less direct. In these cases, alternative complexity measures, such as the Rademacher complexity, can be more useful for analyzing ERM-based methods.

See also: hypothesis space, Rademacher complexity, generalization, koneoppiminen, effective dimension.

vertailutaso Consider some koneoppiminen 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 such that 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 [86]. Another source for a baseline is an existing, but for some reason unsuitable, *koneoppiminen* method. For example, the existing *koneoppiminen* method might be computationally too expensive for the intended *koneoppiminen* application. Nevertheless, its test set error can still serve as a baseline. Another, somewhat more principled, approach to constructing a baseline is via a *todennäköisyysmalli*. In many cases, given a *todennäköisyysmalli* $p(\mathbf{x}, y)$, we can precisely determine the minimi achievable riski among any hypotheses (not even required to belong to the hypothesis space \mathcal{H}) [57].

This minimi achievable riski (referred to as the Bayes risk) is the riski 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 \mathbb{P} [57, Ch. 4]. However, computing the Bayes estimator and Bayes risk presents two main challenges. First, the probability distribution \mathbb{P} is unknown and must be estimated from observed data. Second, even if \mathbb{P} were known, computing the Bayes risk exactly may be computationally infeasible [87]. A widely used *todennäköisyysmalli* is the moninormaalijakauma $(\mathbf{x}, y) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ for data points characterized by numeric features and labels. Here, for the *neliövirhehäviö*, the Bayes estimator is given by the posterior keskiarvo $\mu_{y|\mathbf{x}}$ of the label y , given

the features \mathbf{x} [18], [57]. The corresponding Bayes risk is given by the posterior varianssi $\sigma_{y|\mathbf{x}}^2$ (see Fig. 132).

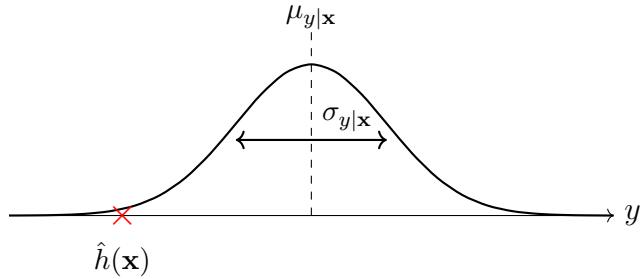


Fig. 132. If the features and the label of a data point are drawn from a moninormaalijakauma, we can achieve the minimi riski (under neliövirhehäviö) by using the Bayes estimator $\mu_{y|\mathbf{x}}$ to predict the label y of a data point with features \mathbf{x} . The corresponding minimi riski is given by the posterior varianssi $\sigma_{y|\mathbf{x}}^2$. We can use this quantity as a baseline for the average loss of a trained model \hat{h} .

See also: Bayes risk, Bayes estimator.

vertical federated learning (VFL) In VFL, different devices have access to different features of the same set of data points [190]. Formally, the underlying global dataset is

$$\mathcal{D}^{(\text{global})} := \{(\mathbf{x}^{(1)}, y^{(1)}), \dots, (\mathbf{x}^{(m)}, y^{(m)})\}.$$

We denote by $\mathbf{x}^{(r)} = (x_1^{(r)}, \dots, x_{d'}^{(r)})^T$, for $r = 1, \dots, m$, the complete feature vectors for the data points. Each device $i \in \mathcal{V}$ observes only a subset $\mathcal{F}^{(i)} \subseteq \{1, \dots, d'\}$ of features, resulting in a local dataset $\mathcal{D}^{(i)}$

with feature vectors

$$\mathbf{x}^{(i,r)} = (x_{j_1}^{(r)}, \dots, x_{j_d}^{(r)})^T.$$

Some of the devices may also have access to the labels $y^{(r)}$, for $r = 1, \dots, m$, of the global dataset (see Fig. 133).

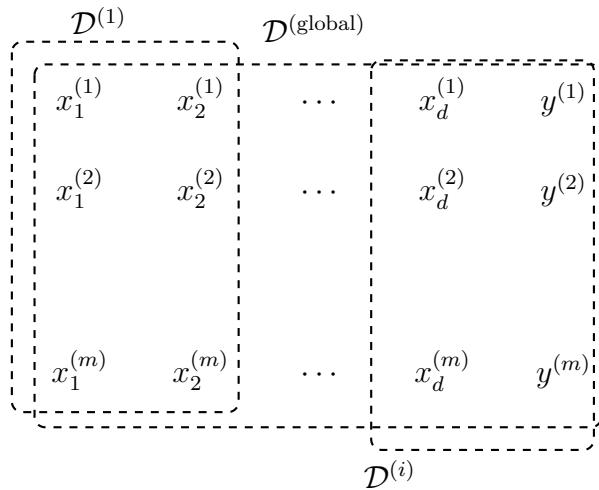


Fig. 133. VFL uses local datasets that are derived from the data points of a common global dataset. The local datasets differ in the choice of features used to characterize the data points.

One potential application of VFL is to enable collaboration between different healthcare providers. Each provider collects distinct types of measurements—such as blood values, electrocardiography, and lung X-rays—for the same patients. Another application is a national social insurance system, where health records, financial indicators, consumer behavior, and mobility data are collected by different institutions. VFL

enables joint learning across these parties while allowing well-defined levels of yksityisyyden suoja. We can view VFL as a specific form of model parallelism.

See also: yksityisyyden suoja, FL, model parallelism.

weight Consider a parameterized 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 lineaarinen malli uses weights $\mathbf{w} = (w_1, \dots, w_d)^T$ 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 (see Fig. 134).

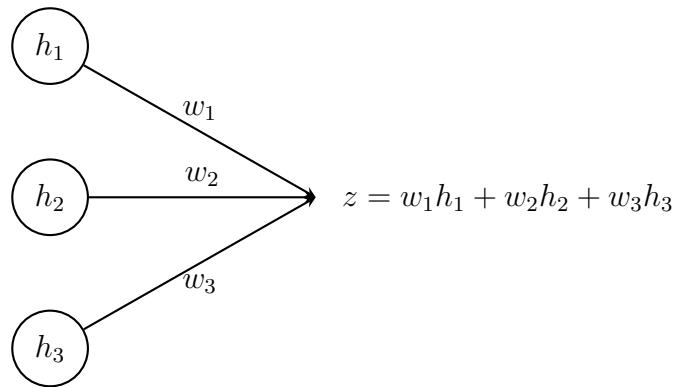


Fig. 134. A section of an ANN that contains a hidden layer with outputs (or activations) h_1, h_2 , and h_3 . These outputs are combined linearly to compute z , which can be used either as output of the ANN or as input to another layer.

See also: hypothesis space, model parameters, feature, lineaarinen malli, ANN, layer, activation.

weighted least squares Weighted least squares refers to ERM-based methods that use the weighted average neliövirhehäviö

$$\frac{1}{m} \sum_{r=1}^m q^{(r)} (y^{(r)} - h(\mathbf{x}^{(r)}))^2$$

on a training set $\mathcal{D} = \{ (\mathbf{x}^{(1)}, y^{(1)}) , \dots , (\mathbf{x}^{(m)}, y^{(m)}) \}$ to measure the quality of a hypothesis kuvaus $h \in \mathcal{H}$. The weights $q^{(1)}, \dots, q^{(m)} \in \mathbb{R}_+$ allow us to emphasize or de-emphasize the contribution of individual data points in the training set. Ideally, we assign a small weight $q^{(r)}$ to the r th data point if it is an outlier (see Fig. 135). We obtain different weighted least squares methods by using different models in ERM.

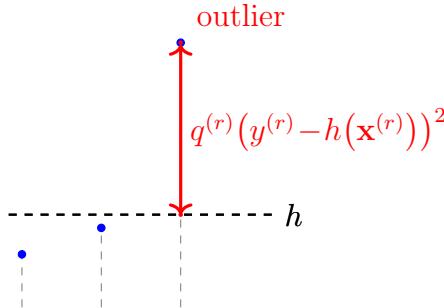


Fig. 135. Weighted least squares can be used to mitigate the effect of outlier data points in a training set.

See also: ERM, neliövirhehäviö, linear regression, lineaarinen malli.

yksityisyyden suoja Consider some koneoppiminen 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 $\hat{\mathbf{w}}$ or the ennuste $\hat{h}(\mathbf{x})$ obtained for a specific data point with features \mathbf{x} . Many important koneoppiminen

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 kuvaus $\mathcal{A}(\mathcal{D})$. 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. See also: koneoppiminen, dataset, model parameter, ennuste, data point, feature, label, sensitive attribute, kuvaus.

yksityisyyshyökkäys A privacy hyökkäys on an koneoppiminen system aims to infer sensitive attributes of individuals by exploiting partial access to a trained koneoppiminen model. One form of a privacy hyökkäys is model inversion.

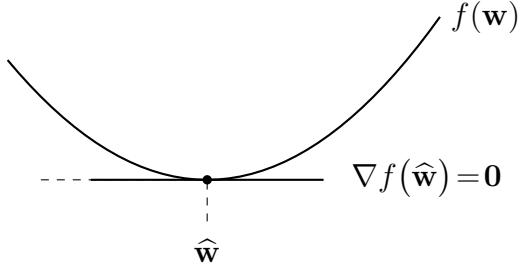
See also: hyökkäys, sensitive attribute, model inversion, luotettava tekologia, GDPR.

ylisovittaminen Consider an koneoppiminen method that uses ERM to learn a hypothesis with the minimi empiirinen riski on a given training set. Such a method is overfitting the training set if it learns a hypothesis with a empiirinen riski on the training set that is significantly smaller than the average loss outside the training set. In other words, if a koneoppiminen method overfits it has a large generalization gap.

See also: ERM, generalization, validation, generalization gap.

zero-gradient condition Consider the unconstrained optimointitehtävä

$\min_{\mathbf{w} \in \mathbb{R}^d} f(\mathbf{w})$ with a smooth and convex objective function $f(\mathbf{w})$. A



necessary and sufficient condition for a vektori $\hat{\mathbf{w}} \in \mathbb{R}^d$ to solve this problem is that the gradientti $\nabla f(\hat{\mathbf{w}})$ is the zero vektori $\nabla f(\hat{\mathbf{w}}) = \mathbf{0}$. In other words [23, p. 140],

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

By defining the gradientti operator $\nabla f : \mathbb{R}^d \rightarrow \mathbb{R}^d$, we can rewrite the zero-gradient condition as a fixed-point equation

$$(\mathcal{I} - \alpha \nabla f) \hat{\mathbf{w}} = \hat{\mathbf{w}}.$$

Here, \mathcal{I} denotes the identity operator (i.e., $\mathcal{I}(\mathbf{w}) = \mathbf{w}$) and α is an arbitrary positive number.

See also: optimointitehtävä, smooth, convex, objective function, vektori, gradientti.

Reinforcement Learning

action An action refers to a decision taken by an AI system at a given time step t that influences the observed reward signal. The actions are elements of an action space \mathcal{A} and are typically denoted by $a_t \in \mathcal{A}$. The action a_t is selected based on the feature vector $\mathbf{x}^{(t)}$ (that collects all available observations) and the current hypothesis $h^{(t)}$. RL uses online learning methods to learn a hypothesis $h^{(t)}$ that predicts an (nearly) optimal action. The usefulness of the ennuste a_t is evaluated indirectly through the resulting reward signal $r^{(t)}$. In the special case of a monikätinen rosvo, the set of possible actions is finite and each action corresponds to selecting one arm. In more general RL settings, the action space may be continuous.

See also: vahvistusoppiminen, reward, loss function, monikätinen rosvo, hypothesis.

action space See action.

Markov decision process (MDP) An MDP is a mathematical structure for the study of vahvistusoppiminen. Formally, an MDP is a satunnaisprosessi that is defined by a specific choice for

- a state space \mathcal{S} ;
- an action space \mathcal{A} ;
- a transition funktion $\mathbb{P}(s' | s, a)$ specifying the probability distribution over the next state $s' \in \mathcal{S}$, given the current state $s \in \mathcal{S}$ and action $a \in \mathcal{A}$;

- a reward funktion $r(s, a) \in \mathbb{R}$ that assigns a numerical reward to each state-action pair (s, a) .

These components define the probability distribution of a sequence

$$s_1, a_1, r_1, s_2, a_2, r_2, \dots, s_t, a_t, r_t$$

of RVs. The defining property of an MDP is the Markov property. That is, at time instant t , the conditional probability distribution of the next state s_{t+1} and reward r_t depends on the past only via the current state s_t and action a_t .

See also: vahvistusoppiminen, satunnaisprosessi, funktion, probability distribution, reward, ennuste.

monikätinens rosvo An MAB problem is a precise formulation of a sequential decision-making task under epävarmuus. At each discrete time step t , a learner selects one of several possible actions—called arms—from a finite set \mathcal{A} . Pulling arm a at time t yields a reward $r^{(a,t)}$ that is drawn from an unknown probability distribution $\mathbb{P}(r^{(a,t)})$. We obtain different classes of MAB problems by placing different restrictions on this probability distribution. In the simplest setting, the probability distribution $\mathbb{P}(r^{(a,t)})$ does not depend on t . Given an MAB problem, the goal is to construct koneoppiminen methods that maximize the cumulative reward over time by strategically balancing exploration (i.e., gathering information about uncertain arms) and exploitation (i.e., selecting arms known to perform well). MAB problems form an important special case of vahvistusoppiminen problems [163], [169].

See also: reward, regret.

regret The regret of a hypothesis h relative to another hypothesis h' , which serves as a vertailutaso, is the difference between the loss incurred by h and the loss incurred by h' [116]. The vertailutaso hypothesis h' is also referred to as an expert.

See also: vertailutaso, loss, expert.

vahvistusoppiminen RL refers to an online learning setting where we can only evaluate the usefulness of a single hypothesis (i.e., a choice of model parameters) at each time step t . In particular, RL methods apply the current hypothesis $h^{(t)}$ to the feature vector $\mathbf{x}^{(t)}$ of the newly received data point. The usefulness of the resulting ennuste $h^{(t)}(\mathbf{x}^{(t)})$ is quantified by a reward signal $r^{(t)}$ (see Fig. 136).

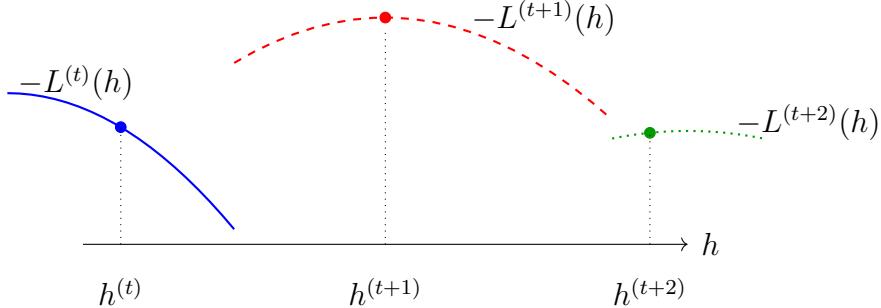


Fig. 136. Three consecutive time steps $t, t + 1, t + 2$ with corresponding loss functiont $L^{(t)}, L^{(t+1)}, L^{(t+2)}$. During time step t , an RL method can evaluate the loss function only for one specific hypothesis $h^{(t)}$, resulting in the reward signal $r^{(t)} = -L^{(t)}(h^{(t)})$.

In general, the reward depends also on the previous ennusteet $h^{(t')}\left(\mathbf{x}^{(t')}\right)$

for $t' < t$. The goal of RL is to learn $h^{(t)}$, for each time step t , such that the (possibly discounted) cumulative reward is maximized [8], [169].

See also: reward, loss function, koneoppiminen.

Machine Learning Systems

ML system An koneoppiminen system consists of computational devices that can gather and store data, execute algoritmit, and exchange information via communication networks. Examples of the exchanged information include data or updates of model parameters.

Machine Learning Regulation

artificial intelligence system (AI system) The EU Artificial Intelligence

Act (AI Act) [191] defines an AI system as a machine-based system that is designed to operate with varying levels of autonomy and that may exhibit adaptiveness (e.g., model retraining) after deployment. AI systems compute ennusteet that can influence environments or decisions [192]. In line with this definition, regulatory obligations and risk classifications apply at the level of the AI system rather than at the level of individual models or algoritmit. The system-level view emphasizes that properties such as vakaus, fairness, and transparency emerge from the interaction of models, data, and operational context, rather than from isolated components.

See also: AI, vakaus, transparency.

automated decision-making Koneoppiminen applications that use ennus-

teet delivered by a trained model directly (without human involvement) to make decisions affecting individuals. Under the GDPR, individuals have the right not to be subject to decisions based solely on automated processing, when these decisions produce legal or similarly significant effects, unless appropriate safeguards (e.g., human oversight, contestability, or explicit consent) are implemented.

deepfake Deepfakes are synthetic media generated or substantially modified

by an artificial intelligence system (AI system) such that it falsely appears to depict a real person, object, or event. Deepfakes are typically produced using generative methods, trained to imitate visual, audio,

or audiovisual characteristics of real data. From a system perspective, deepfakes are characterized by a deliberate mismatch between the observable content and its true origin, which can lead to deception, misinformation, or manipulation. CITE relevant legal sources.

See also: AI system.

high-risk artificial intelligence system (high-risk AI system) A subset of AI systems is classified as high-risk due to its potential to significantly impact safety, fundamental rights, or critical societal functions. High-risk AI systems are subject to stringent regulatory requirements under the EU AI Act, including conformity assessments, risk management, transparency obligations, and post-market monitoring [192]. Examples of high-risk AI systems include those used in critical infrastructure, education, employment, law enforcement, and biometric identification.

See also: AI system, transparency.

personal data Personal data are any information relating to an identified or identifiable natural person (i.e., the data subject). A natural person is identifiable if they can be identified, directly or indirectly, in particular by reference to an identifier such as a name, an identification number, location data, an online identifier, or one or more factors specific to the physical, physiological, genetic, mental, economic, cultural, or social identity of that person [98]. In koneoppiminen systems, personal data may occur in training data, model inputs, intermediate representations (e.g., feature vectors or embeddings), or model outputs, provided that the information relates to an identifiable natural person. The EU AI

Act does not introduce a separate definition of personal data; instead, whenever an AI system processes personal data, the GDPR definition and obligations apply in full.

See also: data, AI system, GDPR.

profiling Profiling aims at identifying patterns and make inferences about individuals based on their data. Profiling techniques use koneoppiminen methods to predict individuals' performance at work, economic situation, health or personal preferences. Profiling is instrumental in targeted advertising, credit scoring, fraud detection, and personalized services. The GDPR imposes strict requirements on organizations that engage in profiling activities to ensure that individuals' rights are protected [98].

See also: data, GDPR.

transparency Transparency is a fundamental requirement for luotettava tekööly [193]. In the context of koneoppiminen methods, transparency is often used interchangeably with selitettävyyys [120], [194]. However, in the broader scope of AI systems, transparency extends beyond selitettävyyys 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 ennusteet 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 koneoppiminen methods inherently offer transparency. For example, lo-

gistic regression provides a quantitative measure of luokittelu reliability through the value $|h(\mathbf{x})|$. Decision trees are another example, as they allow human-readable decision rules [122]. 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 [108] and AI system cards [195] help practitioners understand the intended use cases and limitations of an AI system [196].

See also: luotettava tekoäly, selitettävyys.

Yleinen tietosuoja-asetus (GDPR) The GDPR was enacted by the EU, effective from 25 May 2018 [98]. It safeguards the privacy and data rights of individuals in the EU. The GDPR has significant implications for how data are collected, stored, and used in koneoppiminen applications. Key provisions include the following:

- Data minimization principle: koneoppiminen systems should only use the necessary amount of personal data for their purpose.
- Transparency and selitettävyys: koneoppiminen 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.

See also: data, koneoppiminen, data minimization principle, transparency, selitettävyys.

Hakemisto

- σ -algebra, 106
- σ -field, 106
- k -fold cross-validation (k -fold CV), 187
- k -means, 187
- k -means++, 188
- absolute error loss, 119
- action, 283
- action space, 283
- activation, 120
- adaptive boosting (AdaBoost), 120
- aktivointifunktio, 121
- algebraic connectivity, 22
- algoritmi, 121
- alisovittaminen, 122
- application programming interface (API), 123
- area under the curve (AUC), 124
- artificial intelligence (AI), 124
- artificial intelligence system (AI system), 288
- artificial neural network (ANN), 125
- attention, 126
- autoenkoodaaja, 127
- backdoor, 128
- backpropagation, 128
- bagging, 130
- Banach's fixed-point theorem, 22
- base learner, 131
- basis, 23
- batch, 131
- batch learning, 132
- Bayes estimator, 132
- Bayes risk, 133
- binary cross-entropy (BCE), 133
- binääritappio, 133
- boosting, 133
- bootstrap aggregation, 135
- Bregman divergence, 24
- Cauchy sequence, 25
- Chebyshev's inequality, 25
- Chernoff bound, 26
- cluster, 135
- cluster centroid, 136
- clustered federated learning (CFL),

137	min–max characterization), 33
clustering, 137	
clustering assumption, 137	covariance, 141
clustering error, 138	covariance matrix, 34
co-domain, 27	cross-entropy, 142
column space, 27	cumulative distribution function
concentration inequality, 27	(cdf), 34
concept activation vector (CAV), 139	data, 142
condition number, 28	data augmentation, 142
conditional expectation, 28	data imputation, 143
conditional pmf, 29	data matrix, 143
conditional probability distribution, 28	data minimization principle, 144
confusion matrix, 139	data normalization, 144
conjugate transpose, 29	data parallelism, 144
contractive operator, 29	data point, 145
convergence, 30	data poisoning, 147
convex, 31	dataset, 147
convex clustering, 140	decision boundary, 149
convex optimization problem, 32	decision region, 150
coreset, 140	decision tree, 150
countable, 32	deep net, 151
Courant–Fischer–Weyl min–max characterization (CFW)	deepfake, 288
	degree of belonging, 151
	denial-of-service attack, 152
	denoising autoencoder, 152

density-based spatial clustering of applications with noise (DBSCAN), 153

derivative, 34

design matrix, 153

determinantti, 34

device, 153

diagnosis, 153

diagonalizable, 36

differentiaalinen yksityisyys, 154

differentiable, 36

differential entropy, 36

dimension, 37

directed acyclic graph (DAG), 37

directed cycle, 38

directed graph, 39

discrete random variable (discrete RV), 39

domain, 39

early stopping, 155

edge weight, 156

effective dimension, 156

eigenvalue, 40

eigenvalue decomposition (EVD), 40

eigenvector, 157

empiirinen riski, 157

empirical distribution, 40

empirical risk minimization (ERM), 157

encoder, 158

ensemble, 158

entropia, 42

epigrafi, 42

epoch, 160

epävarmuus, 160

Erdős–Rényi -verkko (ER-verkko), 43

estimation error, 160

Euclidean distance, 161

Euclidean norm, 161

Euclidean space, 162

event, 43

expectation, 162

expectation–maximization (EM), 43

expert, 162

explainability, 250

explainable empirical risk minimization (EERM), 163

explainable machine learning
(XML), 163

feature, 163

feature map, 164

feature matrix, 165

feature space, 165

feature vector, 166

federated averaging (FedAvg), 167

federated gradient descent
(FedGD), 168

federated learning (FL), 168

federated learning network (FL
network), 168

federated proximal (FedProx), 168

federated relaxed (FedRelax), 169

federated stochastic gradient
descent (FedSGD), 169

Finnish Meteorological Institute
(FMI), 169

firmly non-expansive operator, 45

fixed point, 46

fixed-point equation, 46

fixed-point iteration, 47

flow-based clustering, 169

full-rank, 49

funktio, 49

Gaussian, 50

Gaussian mixture model (GMM),
169

Gaussin prosessi, 50

generalization, 170

generalization gap, 171

generalized total variation (GTV),
172

generalized total variation
minimization (GTVMin),
173

geometric median (GM), 173

gradient boosting, 174

gradient descent (GD), 175

gradient-based method, 175

gradientti, 51

gradienttiaskel, 52

graph clustering, 176

hajautettu algoritmi, 176

halfspace, 54

hard clustering, 177

harha, 178

Hermitian, 54

Hessian, 54

high-dimensional regime, 178
 high-risk artificial intelligence system (high-risk AI system), 289
 Hilbert space, 55
 hinge loss, 179
 histogrammi, 180
 Hoeffding's inequality, 56
 horizontal federated learning (HFL), 180
 Huber loss, 181
 Huber regression, 181
 hyperparameter, 181
 hyperplane, 57
 hypothesis, 182
 hypothesis space, 183
 hyökkäys, 184
 image segmentation, 184
 independent and identically distributed (i.i.d.), 57
 independent and identically distributed assumption (i.i.d. assumption), 184
 injective, 58
 inner product, 58
 input vector, 185
 integrable, 59
 iteration, 185
 Jacobi method, 185
 jatkuva, 59
 Johnson–Lindenstrauss lemma (JL lemma), 60
 kaksoisnormi, 61
 kernel, 188
 kernel method, 189
 keskeinen raja-arvolause, 62
 keskiarvo, 63
 koneoppiminen, 190
 konveksi optimointi, 64
 Kronecker product, 191
 Kullback–Leibler divergence (KL divergence), 192
 kuvaus, 66
 kytketty verkko, 66
 käanteismatriisi, 66
 label, 192
 label space, 192
 label vector, 194
 labeled data point, 194

Lagrange dual problem, 67
Lagrangian, 68
Laplacian matrix, 194
large language model (LLM), 195
laskennalliset ominaisuudet, 195
law of large numbers, 68
layer, 196
learning rate, 196
least absolute deviation regression, 197
least absolute shrinkage and selection operator (Lasso), 198
least squares, 199
leave-one-out cross-validation (LOO CV), 199
Lebesgue integral, 68
lineaarikuvaus, 69
lineaarinan malli, 199
linear classifier, 201
linear discriminant analysis (LDA), 201
linear least squares, 201
linear regression, 203
linearly independent, 70
Lloyd's algorithm, 205
local dataset, 206
local interpretable model-agnostic explanations (LIME), 207
local model, 208
logistic loss, 208
logistic regression, 209
loss, 209
loss function, 209
luokitin, 210
luokittelu, 211
luotettava teköäly, 211
lähinaapurimenetelmä, 212
machine unlearning, 212
majorize-minimize (MM), 70
maksimi, 71
Markov chain, 71
Markov decision process (MDP), 283
Markov property, 72
Markov's inequality, 72
matriisi, 73
maximum likelihood, 212
mean absolute error (MAE), 213
mean squared error (MSE), 213

mean squared estimation error
(MSEE), 213

measurable, 75

measure, 76

measure space, 76

mediaani, 76

membership inference attack, 213

metric, 214

metric space, 78

minimi, 79

mirror descent, 79

missing data, 214

ML system, 287

model, 214

model inversion, 215

model parallelism, 216

model parameter, 217

model selection, 217

moment generating function
(MGF), 80

monikäinen rosvo, 284

moninormaalijakauma, 81

monitehtäväoppiminen, 218

multi-label classification, 218

mutual information (MI), 218

naapuri, 219

naapurusto, 219

natural language processing (NLP),
219

neliövirhehäviö, 219

nested cross-validation, 220

networked data, 220

networked exponential families
(nExpFam), 220

networked federated learning
(NFL), 221

networked model, 221

Newtonin menetelmä, 82

nolla-avaruuus, 84

non-expansive operator, 85

non-smooth, 86

normaalijakautunut
satunnaismuuttuja, 86

normal distribution, 88

normal equations, 221

normal matrix, 88

normal vector, 88

normi, 88

objective function, 221

ominaisfunktio, 88

online gradient descent (online GD), 222
online learning, 224
online-algoritmi, 224
operator, 88
oppimistehtävä, 225
optimism in the face of uncertainty, 227
optimointimenetelmä, 88
optimointitehtävä, 89
outcome, 89
outlier, 228
output, 229
output vector, 229
parameter, 229
parameter space, 229
parametric model, 230
partial derivative, 89
penalty term, 231
perceptron algorithm, 232
perplexity, 233
personal data, 289
piirreoptointi, 233
poikkeama, 234
polynomial regression, 234
positive semi-definite (psd), 90
posterior, 90
precision, 234
prediction, 158
predictor, 235
preimage, 91
principal component analysis (PCA), 235
privacy funnel, 236
privacy leakage, 236
probabilistic principal component analysis (PPCA), 91
probability density function (pdf), 91
probability distribution, 91
probability mass function (pmf), 92
probability simplex, 93
probability space, 93
profiling, 290
projected gradient descent (projected GD), 94
projektio, 95
proximable, 95
proximal operator, 95

- pseudokäänteisluku, 96
- Q-learning, 236
- quadratic function, 237
- Rényi divergence, 100
- Rademacher complexity, 237
- random forest, 237
- random geometric graph (RGG), 97
- random projection, 238
- random variable (RV), 98
- rank, 99
- rank-deficient, 99
- realization, 239
- recall, 239
- receiver operating characteristic (ROC), 240
- rectified linear unit (ReLU), 240
- recurrent neural network (RNN), 240
- regressio, 241
- regret, 285
- regularization, 241
- regularized empirical risk minimization (RERM), 243
- regularized loss minimization (RLM), 244
- regularizer, 244
- response, 244
- response vector, 245
- reward, 245
- ridge regression, 245
- risk stratification, 246
- riski, 246
- sample, 100
- sample covariance matrix, 101
- sample mean, 247
- sample size, 247
- sample space, 247
- sample weighting, 247
- satunnaisalgoritmi, 248
- satunnaiskoe, 102
- satunnaisprosessi, 104
- scatterplot, 249
- Schur decomposition, 104
- self-supervised learning, 249
- selitys, 250
- semi-supervised learning (SSL), 251
- sensitive attribute, 251

sequence, 105
similarity graph, 252
simple function, 106
singular value decomposition (SVD), 106
skip connection, 252
smooth, 107
soft clustering, 252
soft label, 253
solmun aste, 108
spectral clustering, 253
spectral decomposition, 108
spectrogram, 256
stability, 257
stacking, 258
standard deviation, 109
standard normal random vector, 109
state space, 109
step size, 259
stochastic block model (SBM), 259
stochastic gradient descent (SGD), 260
stokastinen, 110
stopping criterion, 261
stratification, 261
stratum, 262
strictly convex, 110
strongly convex, 110
structural risk minimization (SRM), 263
subgradient, 110
subgradient descent, 263
subspace, 111
support vector machine (SVM), 264
supremum (or least upper bound), 111
symmetric matrix, 111
tall matrix, 111
target, 265
target vector, 265
tarkkuus, 265
test set, 265
tilastolliset ominaisuudet, 265
todennäköisyys, 266
todennäköisyysmalli, 112
token, 266
total variation, 266
trace, 112

- training, 266
training error, 267
training set, 267
transfer learning, 267
transformer, 267
transparency, 290
transpose, 113
tulkittavuus, 267
typical set, 113

ulottuvuuksien vähentäminen, 269
undirected graph, 113
unitary matrix, 113
upper confidence bound (UCB),
 270
uusio-otanta, 271

vahvistusoppiminen, 285
vakaus, 272
validation, 272
validation error, 273
validation set, 273

Vapnik–Chervonenkis dimension
 (VC dimension), 274
varianssi, 114
vector space, 115
vektori, 114
verkko, 116
vertailutaso, 275
vertical federated learning (VFL),
 277

weight, 279
weighted graph, 117
weighted least squares, 280
wide matrix, 117

yksityisyadden suoja, 280
yksityisyshyökkäys, 281
Yleinen tietosuoja-asetus (GDPR),
 291
ylisovittaminen, 281

zero-gradient condition, 281

Viitteet

- [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] A. Klenke, *Probability Theory: A Comprehensive Course*, 3rd ed. Cham, Switzerland: Springer Nature, 2020.
- [6] P. Billingsley, *Probability and Measure*, 3rd ed. New York, NY, USA: Wiley, 1995.
- [7] D. P. Bertsekas and J. N. Tsitsiklis, *Introduction to Probability*, 2nd ed. Belmont, MA, USA: Athena Scientific, 2008.
- [8] A. Jung, *Machine Learning: The Basics*. Singapore, Singapore: Springer Nature, 2022.
- [9] F. R. K. Chung, *Spectral Graph Theory*. Providence, RI, USA: American Mathematical Society, 1997.

- [10] D. A. Spielman, *Spectral and Algebraic Graph Theory (Incomplete Draft)*. Ebook, 2025, Accessed: December 9, 2025. [Online]. Available: <http://cs-www.cs.yale.edu/homes/spielman/sagt>
- [11] C. T. Kelley, *Iterative Methods for Linear and Nonlinear Equations*. Society for Industrial and Applied Mathematics, 1995. [Online]. Available: <https://pubs.siam.org/doi/abs/10.1137/1.9781611970944>
- [12] A. W. van der Vaart, *Asymptotic Statistics*. Cambridge, U.K.: Cambridge Univ. Press, 1998.
- [13] G. B. Folland, *Real Analysis: Modern Techniques and Their Applications*, 2nd ed. New York, NY, USA: Wiley, 1999.
- [14] W. Feller, *An Introduction to Probability Theory and Its Applications*, vol. 1, 3rd ed. New York, NY, USA: Wiley, 1967.
- [15] R. Vershynin, *High-dimensional probability: An introduction with applications in data science*. Cambridge, U.K.: Cambridge Univ. Press, 2018.
- [16] M. J. Wainwright, *High-Dimensional Statistics: A Non-Asymptotic Viewpoint*. Cambridge, U.K.: Cambridge Univ. Press, 2019.
- [17] R. Durrett, *Probability: Theory and Examples*, 4th ed. Cambridge, U.K.: Cambridge Univ. Press, 2010.
- [18] R. M. Gray, *Probability, Random Processes, and Ergodic Properties*, 2nd ed. New York, NY, USA: Springer Science+Business Media, 2009.
- [19] S. Ross, *A First Course in Probability*, 9th ed. Boston, MA, USA: Pearson Education, 2014.

- [20] O. Kallenberg, *Foundations of Modern Probability*. New York, NY, USA: Springer-Verlag, 1997.
- [21] H. H. Bauschke and P. L. Combettes, *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. New York, NY, USA: Springer Science+Business Media, 2011.
- [22] N. Parikh and S. Boyd, “Proximal algorithms,” *Found. Trends Optim.*, vol. 1, no. 3, pp. 127–239, Jan. 2014, doi: 10.1561/2400000003.
- [23] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
- [24] P. R. Halmos, *Naive Set Theory*. New York, NY, USA: Springer-Verlag, 1974.
- [25] R. B. Ash, *Probability and Measure Theory*, 2nd ed. San Diego, CA, USA: Academic, 2000.
- [26] A. Papoulis and S. U. Pillai, *Probability, Random Variables, and Stochastic Processes*, 4th ed. New York, NY, USA: McGraw-Hill Higher Education, 2002.
- [27] H. J. Dirschmid, *Tensors and Fields*, (in German). Vienna, Austria: Springer-Verlag, 1996.
- [28] G. Strang, *Computational Science and Engineering*. Wellesley, MA, USA: Wellesley-Cambridge Press, 2007.
- [29] R. A. Horn and C. R. Johnson, *Matrix Analysis*, 2nd ed. New York, NY, USA: Cambridge Univ. Press, 2013.

- [30] S. Axler, *Linear Algebra Done Right*, 3rd ed. Cham, Switzerland: Springer Nature, 2015.
- [31] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, 2nd ed. Hoboken, NJ, USA: Wiley, 2006.
- [32] G. Strang, *Introduction to Linear Algebra*, 5th ed. Wellesley, MA, USA: Wellesley-Cambridge Press, 2016.
- [33] M. E. J. Newman, *Networks: An Introduction*. New York, NY, USA: Oxford Univ. Press, 2010.
- [34] D. P. Bertsekas, A. Nedic, and A. E. Ozdaglar, *Convex Analysis and Optimization*. Belmont, MA, USA: Athena Scientific, 2003.
- [35] P. Erdős and A. Rényi, “On random graphs I,” *Publ. Math. Debrecen*, vol. 6, no. 3–4, pp. 290–297, 1959, doi: 10.5486/PMD.1959.6.3-4.12.
- [36] E. N. Gilbert, “Random graphs,” *Ann. Math. Statist.*, vol. 30, no. 4, pp. 1141–1144, Dec. 1959.
- [37] A. P. Dempster, N. M. Laird, and D. B. Rubin, “Maximum likelihood from incomplete data via the em algorithm,” *Journal of the royal statistical society: series B (methodological)*, vol. 39, no. 1, pp. 1–22, 1977.
- [38] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York, NY, USA: Springer Science+Business Media, 2006.
- [39] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. Cambridge, MA, USA: MIT Press, 2012.

- [40] A. Chambolle and T. Pock, “An introduction to continuous optimization for imaging,” *Acta Numer.*, vol. 25, pp. 161–319, 2016. [Online]. Available: <http://dx.doi.org/10.1017/S096249291600009X>
- [41] 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.
- [42] V. I. Istrătescu, *Fixed Point Theory: An Introduction*. Dordrecht, The Netherlands: D. Reidel, 1981.
- [43] A. Lapidoth, *A Foundation in Digital Communication*. Cambridge, U.K.: Cambridge Univ. Press, 2009.
- [44] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. Cambridge, MA, USA: MIT Press, 2006.
- [45] Y. Nesterov, *Introductory Lectures on Convex Optimization: A Basic Course*. Boston, MA, USA: Kluwer Academic, 2004.
- [46] R. T. Rockafellar, *Network Flows and Monotropic Optimization*. Belmont, MA, USA: Athena Scientific, 1998.
- [47] R. J. Wilson, *Introduction to Graph Theory*, 5th ed. Prentice Hall, 2010.
- [48] W. Hoeffding, “Probability inequalities for sums of bounded random variables,” *J. Amer. Statistical Assoc.*, vol. 58, no. 301, pp. 13–30, 1963, doi: 10.1080/01621459.1963.10500830.
- [49] M. Burr, S. Gao, and F. Knoll, “Optimal bounds for Johnson-Lindenstrauss transformations,” *J. Mach. Learn. Res.*, vol. 19, no. 73,

pp. 1–22, Oct. 2018. [Online]. Available: <http://jmlr.org/papers/v19/18-264.html>

- [50] W. B. Johnson and J. Lindenstrauss, “Extensions of Lipschitz mappings into a Hilbert space,” *Contemporary mathematics*, vol. 26, pp. 189–206, 1984.
- [51] S. Dasgupta and A. Gupta, “An elementary proof of a theorem of Johnson and Lindenstrauss,” *Random Struct. Algorithms*, vol. 22, no. 1, pp. 60–65, Nov. 2003, doi: 10.1002/rsa.10073.
- [52] R. A. Horn and C. R. Johnson, *Topics in Matrix Analysis*. Cambridge, U.K.: Cambridge Univ. Press, 1991.
- [53] K. Lange, *MM Optimization Algorithms*. Philadelphia, PA, USA: SIAM, 2016.
- [54] D. R. Hunter and K. Lange, “A tutorial on MM algorithms,” *Amer. Statistician*, vol. 58, no. 1, pp. 30–37, 2004, doi: 10.1198/0003130042836.
- [55] J. R. Norris, *Markov Chains*, ser. Cambridge Series on Statistical and Probabilistic Mathematics. Cambridge: Cambridge University Press, 1997, no. 2, first published 1997; first paperback edition 1998.
- [56] P. R. Halmos, *Measure Theory*. New York, NY, USA: Springer-Verlag, 1974.
- [57] E. L. Lehmann and G. Casella, *Theory of Point Estimation*, 2nd ed. New York, NY, USA: Springer-Verlag, 1998.

- [58] S. Bubeck, “Convex optimization: Algorithms and complexity,” *Found. Trends Mach. Learn.*, vol. 8, no. 3–4, pp. 231–357, Nov. 2015, doi: 10.1561/2200000050.
- [59] A. Lapidoth, *A Foundation in Digital Communication*, 2nd ed. Cambridge, U.K.: Cambridge Univ. Press, 2017.
- [60] N. Dunford and J. T. Schwartz, *Linear Operators, Part I: General Theory*. New York, NY, USA: Wiley, 1988.
- [61] D. P. Bertsekas, *Nonlinear Programming*, 2nd ed. Belmont, MA, USA: Athena Scientific, 1999.
- [62] B. Schölkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA, USA: MIT Press, 2002.
- [63] M. E. Tipping and C. M. Bishop, “Probabilistic principal component analysis,” *J. Roy. Statist. Soc.: Ser. B (Statist. Methodology)*, vol. 61, no. 3, pp. 611–622, 1999, doi: 10.1111/1467-9868.00196.
- [64] 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.
- [65] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd ed. New York, NY, USA: Springer Science+Business Media, 2009.

- [66] A. Ben-Israel and T. N. E. Greville, *Generalized Inverses: Theory and Applications*, 2nd ed. New York, NY, USA: Springer-Verlag, 2003.
- [67] C. D. Meyer, *Matrix Analysis and Applied Linear Algebra*. Philadelphia, PA, USA: SIAM, 2000.
- [68] 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.
- [69] B. S. Everitt and A. Skrondal, *The Cambridge Dictionary of Statistics*, 4th ed. Cambridge, U.K.: Cambridge Univ. Press, 2010.
- [70] G. Upton and I. Cook, *A Dictionary of Statistics*, 3rd ed. Oxford, U.K.: Oxford Univ. Press, 2014.
- [71] P. J. Brockwell and R. A. Davis, *Time Series: Theory and Methods*, 2nd ed. New York, NY, USA: Springer-Verlag, 1991.
- [72] D. P. Bertsekas, *Convex Optimization Algorithms*. Belmont, MA, USA: Athena Scientific, 2015.
- [73] S. Sra, S. Nowozin, and S. J. Wright, Eds., *Optimization for Machine Learning*. Cambridge, MA, USA: MIT Press, 2012.
- [74] G. Ridgeway, D. Madigan, and T. S. Richardson, “Boosting methodology for regression problems,” in *Proc. 7th Int. Workshop Artif. Intell. Statist.*, D. Heckerman and J. Whittaker, Eds., vol. R2, Jan. 1999. [Online]. Available: <https://proceedings.mlr.press/r2/ridgeway99a.html>

- [75] R. E. Schapire, “A brief introduction to boosting,” in *Proc. 16th Int. Joint Conf. Artif. Intell.*, vol. 2, 1999, pp. 1401–1406. [Online]. Available: <https://www.ijcai.org/Proceedings/99-2/Papers/103.pdf>
- [76] H. Drucker, “Improving regressors using boosting techniques,” in *Proc. 14th Int. Conf. Mach. Learn.*, D. H. Fisher, Ed., Jul. 1997, pp. 107–115.
- [77] 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>
- [78] M. Sipser, *Introduction to the Theory of Computation*, 3rd ed. Andover, U.K.: Cengage Learning, 2013.
- [79] L. Richardson and M. Amundsen, *RESTful Web APIs*. Sebastopol, CA, USA: O'Reilly Media, 2013.
- [80] J. A. Hanley and B. J. McNeil, “The meaning and use of the area under a receiver operating characteristic (ROC) curve.” *Radiol.*, vol. 143, no. 1, pp. 29–36, Apr. 1982, doi: 10.1148/radiology.143.1.7063747.
- [81] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.
- [82] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

- [83] D. M. Blei, A. Y. Ng, and M. I. Jordan, “Latent dirichlet allocation,” *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, Jan. 2003.
- [84] 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/3f5ee243547dee91fb053c1c4a845aa-Abstract.html
- [85] L. Breiman, “Bagging predictors,” *Mach. Learn.*, vol. 24, no. 2, pp. 123–140, Aug. 1996, doi: 10.1007/BF00058655.
- [86] 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.
- [87] 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.
- [88] J. H. Friedman, “Greedy function approximation: A gradient boosting machine,” *Ann. Statist.*, vol. 29, no. 5, pp. 1189–1232, Oct. 2001, doi: 10.1214/aos/1013203451.
- [89] B. Efron and R. J. Tibshirani, *An introduction to the Bootstrap*. New York, NY, USA: Chapman & Hall, 1993.
- [90] O. Chapelle, B. Schölkopf, and A. Zien, Eds., *Semi-Supervised Learning*. Cambridge, MA, USA: MIT Press, 2006.

- [91] B. Kim, M. Wattenberg, J. Gilmer, C. Cai, J. Wexler, F. Viegas *et al.*, “Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav),” in *International conference on machine learning*. PMLR, 2018, pp. 2668–2677.
- [92] 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>
- [93] 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.
- [94] C. Chai, J. Wang, N. Tang, Y. Yuan, J. Liu, Y. Deng, and G. Wang, “Efficient coreset selection with cluster-based methods,” in *Proc. 29th ACM SIGKDD Conf. Knowl. Discovery Data Mining*, Aug. 2023, pp. 167–178, doi: 10.1145/3580305.3599326.
- [95] M. F. Balcan, S. Ehrlich, and Y. Liang, “Distributed k-means and k-median clustering on general topologies,” in *Adv. Neural Inf. Process. Syst.*, C. J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds., vol. 26, 2013, pp. 1995–2003. [Online]. Available: https://papers.nips.cc/paper_files/paper/2013/hash/7f975a56c761db6506eca0b37ce6ec87-Abstract.html
- [96] International Organization for Standardization and International Electrotechnical Commission, “Information technology — Vocabulary,”

ISO/IEC 2382:2015, 2015, Accessed: September 21, 2025. [Online]. Available: <https://www.iso.org/standard/63598.html>

- [97] B. S. Everitt, *The Cambridge Dictionary of Statistics*, 2nd ed. Cambridge, U.K.: Cambridge Univ. Press, 2002.
- [98] European Parliament and Council of the 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),” Official Journal of the European Union, L 119/1, May 4, 2016, Accessed: July, 2025. [Online]. Available: <https://eur-lex.europa.eu/eli/reg/2016/679/oj>
- [99] European Parliament and Council of the 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),” Official Journal of the European Union, L 295/39, Nov. 21, 2018, Accessed: December 1, 2025. [Online]. Available: <https://eur-lex.europa.eu/eli/reg/2018/1725/oj>
- [100] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, “Distributed optimization and statistical learning via the alternating direction method

of multipliers,” *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, Jul. 2011, doi: 10.1561/2200000016.

- [101] 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.
- [102] 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.
- [103] 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/>
- [104] S. Abiteboul, R. Hull, and V. Vianu, *Foundations of Databases*. Reading, MA, USA: Addison-Wesley, 1995.
- [105] S. Hoberman, *Data Modeling Made Simple: A Practical Guide for Business and IT Professionals*, 2nd ed. Basking Ridge, NJ, USA: Technics Publications, 2009.
- [106] R. Ramakrishnan and J. Gehrke, *Database Management Systems*, 3rd ed. New York, NY, USA: McGraw-Hill, 2002.
- [107] 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.

- [108] T. Gebru et al., “Datasheets for datasets,” *Commun. ACM*, vol. 64, no. 12, pp. 86–92, Nov. 2021, doi: 10.1145/3458723.
- [109] D. A. Patterson and J. L. Hennessy, *Computer Organization and Design: The Hardware/Software Interface*, 5th ed. San Francisco, CA, USA: Morgan Kaufmann, 2013.
- [110] C. M. Bishop and H. Bishop, *Deep Learning: Foundations and Concepts*. Cham, Switzerland: Springer Nature, 2024.
- [111] S. J. D. Prince, *Understanding Deep Learning*. Cambridge, MA, USA: MIT Press, 2023.
- [112] G. Tel, *Introduction to Distributed Algorithms*, 2nd ed. Cambridge, U.K.: Cambridge Univ. Press, 2000.
- [113] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Belmont, MA, USA: Athena Scientific, 2015.
- [114] S. M. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1993.
- [115] P. R. Halmos, *Finite-Dimensional Vector Spaces*. New York, NY, USA: Springer-Verlag, 1974.
- [116] N. Cesa-Bianchi and G. Lugosi, *Prediction, Learning, and Games*. New York, NY, USA: Cambridge Univ. Press, 2006.
- [117] E. Hazan, “Introduction to online convex optimization,” *Found. Trends Optim.*, vol. 2, no. 3–4, pp. 157–325, Aug. 2016, doi: 10.1561/2400000013.

- [118] 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
- [119] 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.
- [120] 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.
- [121] 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>
- [122] 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.

- [123] C. Molnar, *Interpretable Machine Learning: A Guide for Making Black Box Models Explainable*, 3rd ed. Ebook, 2025, Accessed: August 1, 2025. [Online]. Available: <https://christophm.github.io/interpretable-ml-book/>
- [124] 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.*, 2017, pp. 618–626, doi: 10.1109/ICCV.2017.74.
- [125] D. N. Gujarati and D. C. Porter, *Basic Econometrics*, 5th ed. New York, NY, USA: McGraw-Hill/Irwin, 2009.
- [126] Y. Dodge, Ed., *The Oxford Dictionary of Statistical Terms*. New York, NY, USA: Oxford Univ. Press, 2003.
- [127] 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.
- [128] S. Mallat, “Understanding deep convolutional networks,” *Philos. Trans. Roy. Soc. A*, vol. 374, no. 2065, Apr. 2016, Art. no. 20150203, doi: 10.1098/rsta.2015.0203.
- [129] 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>

- [130] 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
- [131] 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., 2021, pp. 1292–1296, doi: 10.1109/IEEECONF53345.2021.9723162.
- [132] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning: From Theory to Algorithms*. New York, NY, USA: Cambridge Univ. Press, 2014.
- [133] 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.
- [134] 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.
- [135] H. P. Lopuhaä and P. J. Rousseeuw, “Breakdown points of affine equivariant estimators of multivariate location and covariance matrices,” *Ann. Statist.*, vol. 19, no. 1, pp. 229–248, Mar. 1991, doi: 10.1214/aos/1176347978.

- [136] 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.
- [137] P. Bühlmann and S. van de Geer, *Statistics for High-Dimensional Data: Methods, Theory and Applications*. Berlin, Germany: Springer-Verlag, 2011.
- [138] 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.
- [139] 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.
- [140] P. J. Huber, “Robust estimation of a location parameter,” in *Breakthroughs in statistics: Methodology and distribution*. Springer, 1992, pp. 492–518.
- [141] F. Doshi-Velez and B. Kim, “Towards a rigorous science of interpretable machine learning,” arXiv preprint arXiv:1702.08608, Mar. 2017. [Online]. Available: <https://arxiv.org/abs/1702.08608>
- [142] P. Hase and M. Bansal, “Evaluating explainable AI: Which algorithmic explanations help users predict model behavior?” in *Proc. 58th Annu. Meeting Assoc. Comput. Linguistics*, D. Jurafsky, J. Chai, N. Schluter, and J. Tetreault, Eds., Jul. 2020, pp. 5540–5552. [Online]. Available: <https://aclanthology.org/2020.acl-main.491>

- [143] Z. C. Lipton, “The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery,” *Queue*, vol. 16, no. 3, pp. 31–57, Jun. 2018, doi: 10.1145/3236386.3241340.
- [144] J. Kleinberg and E. Tardos, *Algorithm Design*. Boston, MA, USA: Addison-Wesley, 2006.
- [145] M. Mahajan, P. Nimbhorkar, and K. Varadarajan, “The planar k-means problem is NP-hard,” in *WALCOM: Algorithms and Computation*, S. Das and R. Uehara, Eds. Berlin, Heidelberg, Germany: Springer-Verlag, 2009, pp. 274–285.
- [146] 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
- [147] R. Caruana, “Multitask learning,” *Mach. Learn.*, vol. 28, pp. 41–75, Jul. 1997, doi: 10.1023/A:1007379606734.
- [148] 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.
- [149] 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.

- [150] R. Tibshirani, “Regression shrinkage and selection via the lasso,” *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.
- [151] R. A. Fisher, “The use of multiple measurements in taxonomic problems,” *Ann. Eugenics*, vol. 7, no. 2, pp. 179–188, Sep. 1936, doi: 10.1111/j.1469-1809.1936.tb02137.x.
- [152] J. Heinonen, “Lectures on lipschitz analysis,” Dept. Math. Statist., Univ. Jyväskylä, Jyväskylä, Finland, Rep. 100, 2005. [Online]. Available: <http://www.math.jyu.fi/research/reports/rep100.pdf>
- [153] S. Lloyd, “Least squares quantization in PCM,” *IEEE Trans. Inf. Theory*, vol. 28, no. 2, pp. 129–137, Mar. 1982.
- [154] T. Hastie, R. Tibshirani, and M. Wainwright, *Statistical Learning with Sparsity: The Lasso and Generalizations*. Boca Raton, FL, USA: CRC Press, 2015.
- [155] 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.
- [156] Y. Cao and J. Yang, “Towards making systems forget with machine unlearning,” in *2015 IEEE Symp. Secur. Privacy*, 2015, pp. 463–480, doi: 10.1109/SP.2015.35.
- [157] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, “Membership inference attacks against machine learning models,” in *2017 IEEE Symp. Secur. Privacy*, 2017, pp. 3–18, doi: 10.1109/SP.2017.41.

- [158] 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.
- [159] E. A. Bender, *An Introduction to Mathematical Modeling*. New York, NY, USA: Wiley, 1978.
- [160] M. Fredrikson, S. Jha, and T. Ristenpart, “Model inversion attacks that exploit confidence information and basic countermeasures,” in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2015, pp. 1322–1333, doi: 10.1145/2810103.2813677.
- [161] 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.
- [162] 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>
- [163] S. Bubeck and N. Cesa-Bianchi, “Regret analysis of stochastic and non-stochastic multi-armed bandit problems,” *Found. Trends Mach. Learn.*, vol. 5, no. 1, pp. 1–122, Dec. 2012, doi: 10.1561/2200000024.
- [164] 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.

- [165] 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.
- [166] F. Rosenblatt, “The perceptron: A probabilistic model for information storage and organization in the brain,” *Psychol. Rev.*, vol. 65, no. 6, pp. 386–408, Nov. 1958, doi: 10.1037/h0042519.
- [167] A. Makhdoumi, S. Salamatian, N. Fawaz, and M. Médard, “From the information bottleneck to the privacy funnel,” in *2014 IEEE Inf. Theory Workshop*, 2014, pp. 501–505, doi: 10.1109/ITW.2014.6970882.
- [168] 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.
- [169] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA, USA: MIT Press, 2018.
- [170] P. L. Bartlett and S. Mendelson, “Rademacher and gaussian complexities: Risk bounds and structural results,” *J. Mach. Learn. Res.*, vol. 3, no. Nov., pp. 463–482, 2002. [Online]. Available: <https://www.jmlr.org/papers/v3/bartlett02a.html>
- [171] T. Fawcett, “An introduction to ROC analysis,” *Pattern Recognit. Lett.*, vol. 27, no. 8, pp. 861–874, Jun. 2006, doi: 10.1016/j.patrec.2005.10.010.
- [172] S. Shalev-Shwartz and A. Tewari, “Stochastic methods for ℓ_1 regularized loss minimization,” in *Proc. 26th Annu. Int. Conf. Mach. Learn.*, L. Bottou and M. Littman, Eds., Jun. 2009, pp. 929–936.

- [173] G. S. Collins and K. G. M. Moons, “Comparing risk prediction models.” *BMJ*, vol. 344, May 2012, Art. no. e3186, doi: 10.1136/bmj.e3186.
- [174] E. W. Steyerberg, *Clinical Prediction Models: A Practical Approach to Development, Validation, and Updating*, 2nd ed. Cham, Switzerland: Springer Nature, 2019.
- [175] L. Cohen, *Time-Frequency Analysis*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1995.
- [176] 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.
- [177] B. Boashash, Ed., *Time Frequency Signal Analysis and Processing: A Comprehensive Reference*. Oxford, U.K.: Elsevier, 2003.
- [178] S. Mallat, *A Wavelet Tour of Signal Processing: The Sparse Way*, 3rd ed. Burlington, MA, USA: Academic, 2009.
- [179] D. H. Wolpert, “Stacked generalization,” *Neural Netw.*, vol. 5, no. 2, pp. 241–259, 1992, doi: 10.1016/S0893-6080(05)80023-1.
- [180] Z.-H. Zhou, *Ensemble Methods: Foundations and Algorithms*. New York, NY, USA: Chapman & Hall/CRC Press, 2012.
- [181] R. Motwani and P. Raghavan, *Randomized Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 1995.

- [182] R. G. Gallager, *Stochastic Processes: Theory for Applications*. New York, NY, USA: Cambridge Univ. Press, 2013.
- [183] P. W. Holland, K. B. Laskey, and S. Leinhardt, “Stochastic blockmodels: First steps,” *Social networks*, vol. 5, no. 2, pp. 109–137, 1983.
- [184] 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>
- [185] Organisation for Economic Co-operation and Development (OECD), *OECD Glossary of Statistical Terms*. Paris, France: OECD Publishing, 2008. [Online]. Available: <https://doi.org/10.1787/9789264055087-en>
- [186] 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.
- [187] 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>
- [188] 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>
- [189] V. N. Vapnik and A. Y. Chervonenkis, “On the uniform convergence of relative frequencies of events to their probabilities,” in *Measures of*

Complexity: Festschrift for Alexey Chervonenkis. Cham, Switzerland: Springer, 2015, ch. 3, pp. 11–30.

- [190] Q. Yang, Y. Liu, Y. Cheng, Y. Kang, T. Chen, and H. Yu, “Vertical federated learning,” in *Federated Learning*. Cham, Switzerland: Springer Nature, 2020, ch. 5, pp. 69–81.
- [191] European Commission, “Proposal for a regulation of the European Parliament and of the Council laying down harmonised rules on artificial intelligence (Artificial Intelligence Act) and amending certain Union legislative acts,” COM/2021/206 final, Apr. 21, 2021, Accessed: December 16, 2025. [Online]. Available: <https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=celex:52021PC0206>
- [192] European Parliament and Council of the European Union, “Regulation (EU) 2024/1689 of the European Parliament and of the Council of 13 June 2024 laying down harmonised rules on artificial intelligence and amending Regulations (EC) No 300/2008, (EU) No 167/2013, (EU) No 168/2013, (EU) 2018/858, (EU) 2018/1139 and (EU) 2019/2144 and Directives 2014/90/EU, (EU) 2016/797 and (EU) 2020/1828 (Artificial Intelligence Act) (Text with EEA relevance),” Official Journal of the European Union, L series, Jul. 12, 2024, Accessed: July 2025. [Online]. Available: <https://eur-lex.europa.eu/eli/reg/2024/1689/oj/eng>
- [193] 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>

- [194] C. Gallese, “The AI Act proposal: A new right to technical interpretability?” *SSRN Electron. J.*, Feb. 2023. [Online]. Available: <https://ssrn.com/abstract=4398206>
- [195] M. Mitchell et al., “Model cards for model reporting,” in *Proc. Conf. Fairness, Accountability, Transparency*, 2019, pp. 220–229, doi: 10.1145/3287560.3287596.
- [196] 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.*, 2017, pp. 197–201, doi: 10.1109/IHTC.2017.8058187.