Lectures Notes

# CS-E4740 Federated Learning

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

January 18, 2024

**Abstract**

This course discusses theory and algorithms for federated learning (FL) from collections of local datasets. These FL methods exploit similarities between local datasets to train local (or personalized) models collaboratively. Two core mathematical objects of this course are empirical graphs and generalized total variation minimization (GTVMin). We use empirical graphs to store and process local datasets and parameters of local models. GTVMin formulates FL as an instance of regularized empirical risk minimization. As the regularizer, we use quantitative measures for the variation of local models across the edges of the empirical graph. We can obtain practical FL systems by applying distributed optimization methods to solve GTVMin.

*AJ is currently Associate Professor for Machine Learning at Aalto University (Finland).
This work has been partially funded by the Academy of Finland (decision numbers 331197, 331197) and the European Union (grant number 952410).

# Contents

# Lists of Symbols

## Sets and Functions

| | |
|---|---|
| $a \in \mathcal{A}$ | This statement indicates that the object $a$ is an element of the set $\mathcal{A}$. |
| $a := b$ | This statement defines $a$ to be shorthand for $b$. |
| $\lvert\mathcal{A}\rvert$ | The cardinality (number of elements) of a finite set $\mathcal{A}$. |
| $\mathcal{A} \subseteq \mathcal{B}$ | $\mathcal{A}$ is a subset of $\mathcal{B}$. |
| $\mathcal{A} \subset \mathcal{B}$ | $\mathcal{A}$ is a strict subset of $\mathcal{B}$. |
| $\mathbb{N}$ | The set of natural numbers $1, 2, \ldots$. |
| $\mathbb{R}$ | The set of real numbers $x$ [1]. |
| $\mathbb{R}_+$ | The set of non-negative real numbers $x \geq 0$. |
| $\mathbb{R}_{++}$ | The set of positive real numbers $x > 0$. |
| $h(\cdot): \mathcal{A} \to \mathcal{B} : a \mapsto h(a)$ | A function (map) that accepts any element $a \in \mathcal{A}$ from a set $\mathcal{A}$ as input and delivers a well-defined element $h(a) \in \mathcal{B}$ of a set $\mathcal{B}$. The set $\mathcal{A}$ is the domain of the function $h$ and the set $\mathcal{B}$ is the codomain of $h$. ML aims at finding (or learning) a function $h$ ("hypothesis") that reads in the features $\mathbf{x}$ of a data point and delivers a prediction $h(\mathbf{x})$ for its label $y$. |

| | |
|---|---|
| $\{0, 1\}$ | The binary set that consists of the two real numbers 0 and 1. |
| $[0, 1]$ | The closed interval of real numbers $x$ with $0 \leq x \leq 1$. |
| $\operatorname{argmin} f(\mathbf{w})$ | The set of minimizers for a real-valued function $f(\mathbf{w})$. |
| $\log a$ | The logarithm of the positive number $a \in \mathbb{R}_{++}$. |

# Matrices and Vectors

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

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

## Probability Theory

| | |
|---|---|
| $\mathbb{E}_p\{f(\mathbf{z})\}$ | The expectation of a function $f(\mathbf{z})$ of a RV $\mathbf{z}$ whose probability distribution is $p(\mathbf{z})$. If the probability distribution is clear from context we just write $\mathbb{E}\{f(\mathbf{z})\}$. |
| $p(\mathbf{x}, y)$ | A (joint) probability distribution of a RV whose realizations are data points with features $\mathbf{x}$ and label $y$. |
| $p(\mathbf{x}\|y)$ | A conditional probability distribution of a RV $\mathbf{x}$ given the value of another RV $y$ [3, Sec. 3.5]. |
| $p(\mathbf{x}; \mathbf{w})$ | A parametrized probability distribution of a RV $\mathbf{x}$. The probability distribution depends on a parameter vector $\mathbf{w}$. For example, $p(\mathbf{x}; \mathbf{w})$ could be a multivariate normal distribution with the parameter vector $\mathbf{w}$ given by the entries of te mean vector $\mathbb{E}\{\mathbf{x}\}$ and the covariance matrix $\mathbb{E}\Big\{\big(\mathbf{x}-\mathbb{E}\{\mathbf{x}\}\big)\big(\mathbf{x}-\mathbb{E}\{\mathbf{x}\}\big)^T\Big\}$. |
| $\mathcal{N}(\mu, \sigma^2)$ | The probability distribution of a scalar normal ("Gaussian") RV $x \in \mathbb{R}$ with mean (or expectation) $\mu = \mathbb{E}\{x\}$ and variance $\sigma^2 = \mathbb{E}\big\{(x-\mu)^2\big\}$. |
| $\mathcal{N}(\boldsymbol{\mu}, \mathbf{C})$ | The multivariate normal distribution of a vector-valued Gaussian RV $\mathbf{x} \in \mathbb{R}^d$ with mean (or expectation) $\boldsymbol{\mu} = \mathbb{E}\{\mathbf{x}\}$ and covariance matrix $\mathbf{C} = \mathbb{E}\big\{\big(\mathbf{x}-\boldsymbol{\mu}\big)\big(\mathbf{x}-\boldsymbol{\mu}\big)^T\big\}$. |

## Machine Learning

| | |
|---|---|
| $r$ | An index $r = 1, 2, \ldots,$ that enumerates data points. |
| $m$ | The number of data points in (the size of) a dataset. |
| $\mathcal{D}$ | A dataset $\mathcal{D} = \{\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(m)}\}$ is a list of individual data points $\mathbf{z}^{(r)}$, for $r = 1, \ldots, m$. |
| $d$ | Number of features that characterize a data point. |
| $x_j$ | The $j$th feature of a data point. The first feature of a given data point is denoted $x_1$, the second feature $x_2$ and so on. |
| $\mathbf{x}$ | The feature vector $\mathbf{x} = \big(x_1, \ldots, x_d\big)^T$ of a data point whose entries are the individual features of a data point. |
| $\mathcal{X}$ | The feature space $\mathcal{X}$ is the set of all possible values that the features $\mathbf{x}$ of a data point can take on. |
| $\mathbf{z}$ | Beside the symbol $\mathbf{x}$, we sometimes use $\mathbf{z}$ as another symbol to denote a vector whose entries are features of a data point. We need two different symbols to distinguish between "raw" or "original" and learnt features [4, Ch. 9]. |
| $\mathbf{x}^{(r)}$ | The feature vector of the $r$th data point within a dataset. |
| $x_j^{(r)}$ | The $j$th feature of the $r$th data point within a dataset. |
| $\mathcal{B}$ | A mini-batch (subset) of randomly chosen data points. |
| $B$ | The size of (the number of data points in) a mini-batch. |

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

| | |
|---|---|
| $B^2$ | The squared bias of a learnt hypothesis $\hat{h}$ delivered by a ML algorithm that is fed with data points which are modelled as realizations of RVs. If data is modelled as realizations of RVs, also the delivered hypothesis $\hat{h}$ is the realization of a RV. |
| $V$ | The variance of the (parameters of the) hypothesis delivered by a ML algorithm. If the input data for this algorithm is interpreted as realizations of RVs, so is the delivered hypothesis a realization of a RV. |
| $L\left((\mathbf{x}, y), h\right)$ | The loss incurred by predicting the label $y$ of a data point using the prediction $\hat{y} = h(\mathbf{x})$. The prediction $\hat{y}$ is obtained from evaluating the hypothesis $h \in \mathcal{H}$ for the feature vector $\mathbf{x}$ of the data point. |
| $E_v$ | The validation error of a hypothesis $h$, which is its average loss incurred over a validation set. |
| $\widehat{L}\big(h\|\mathcal{D}\big)$ | The empirical risk or average loss incurred by the predictions of hypothesis $h$ for the data points in the dataset $\mathcal{D}$. |
| $E_t$ | The training error of a hypothesis $h$, which is its average loss incurred over a training set. |
| $t$ | A discrete-time index $t = 0, 1, \ldots$ used to enumerate a sequence to sequential events ("time instants"). |
| $t$ | An index that enumerates learning tasks within a multi-task learning problem. |

| | |
|---|---|
| $\lambda$ | A regularization parameter that controls the amount of regularization. |
| $\lambda_j\big(\mathbf{Q}\big)$ | The $j$th eigenvalue (sorted either ascending or descending) of a psd matrix $\mathbf{Q}$. We also use the shorthand $\lambda_j$ if the corresponding matrix is clear from context. |
| $\sigma(\cdot)$ | The activation function used by an artificial neuron within an artificial neural network (ANN). |
| $\mathcal{R}_{\hat{y}}$ | A decision region within a feature space. |
| $\mathbf{w}$ | A parameter vector $\mathbf{w} = \big(w_1, \ldots, w_d\big)^T$ whose entries are parameters of a model. These parameters could be feature weights in linear maps, the weights in ANNs or the thresholds used for splits in decision trees. |
| $h^{(\mathbf{w})}(\cdot)$ | A hypothesis map that involves tunable model parameters $w_1, \ldots, w_d$, stacked into the vector $\mathbf{w} = \big(w_1, \ldots, w_d\big)^T$. |
| $\nabla f(\mathbf{w})$ | The gradient of a differentiable real-valued function $f : \mathbb{R}^d \to \mathbb{R}$ is the vector $\nabla f(\mathbf{w}) = \big(\frac{\partial f}{\partial w_1}, \ldots, \frac{\partial f}{\partial w_d}\big)^T \in \mathbb{R}^d$ [5, Ch. 9]. |
| $\phi(\cdot)$ | A feature map $\phi : \mathcal{X} \to \mathcal{X}' : \mathbf{x} \mapsto \mathbf{x}' := \phi\big(\mathbf{x}\big) \in \mathcal{X}'$. |

## Federated Learning

| | |
|---|---|
| $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ | Empirical graph whose nodes $i \in \mathcal{V}$ carry local datasets and local models. |
| $i \in \mathcal{V}$ | A node in the empirical graph that represents a local dataset and a corresponding local model. It might also be useful to think of node $i$ as a small computer that can collect data and execute computations to train ML models. |
| $\mathcal{D}^{(i)}$ | The local dataset $\mathcal{D}^{(i)}$ at node $i \in \mathcal{V}$ of an empirical graph. |
| $m_i$ | The number of data points (sample size) contained in the local dataset $\mathcal{D}^{(i)}$ at node $i \in \mathcal{V}$. |
| $\mathcal{N}^{(i)}$ | The neighbourhood of the node $i$ in an empirical graph. |
| $\mathbf{x}^{(i,r)}$ | The features of the $r$-th data point in the local dataset $\mathcal{D}^{(i)}$. |
| $y^{(i,r)}$ | The label of the $r$-th data point in the local dataset $\mathcal{D}^{(i)}$. |
| $L^{(\mathrm{d})}\big(\mathbf{x}, h\big(\mathbf{x}\big), h'\big(\mathbf{x}\big)\big)$ | The loss incurred by a "external" hypothesis $h'$ on a data point with features $\mathbf{x}$ and predicted label $h\big(\mathbf{x}\big)$ that is obtained from some local hypothesis. |

# 1 Lecture - "Welcome and Intro"

Welcome to the course CS-E4740 Federated Learning. This course can be completed fully remote. Any on-site event will be recorded and made available to students via this YouTube channel. The basic variant (5 credits) of this course consists of lectures (schedule here) and corresponding coding assignments (schedule here). We test your completion of the coding assignments via quizzes (implemented on the MyCourses page). You can upgrade the course to an extended variant (10 credits) by completing a student project (see Section 1.8).

## 1.1 Learning Goals

This lecture offers

- introduction of course topic and positioning in wider curricula
- discussion of learning goals, assignments and student project
- overview of course schedule

## 1.2 Introduction

Smartphones, wearables or IoT devices generate decentralized collections of local datasets [6–10]. An application-specific network structure relates these local datasets . For example, the high-precision management of pandemics uses contact networks to relate local datasets generated by patients. Network medicine relates data about diseases via co-morbidity networks [11]. Social

science uses notions of acquaintance to relate data collected from be-friended individuals [12].

Federated learning (FL) is an umbrella term for distributed optimization techniques to train machine learning (ML) models from decentralized collections of local datasets [13–17]. These methods carry out computations, such as gradient steps (see Lecture 4), for ML model training at the location of data generation. This design philosophy is different from a naive application of ML techniques, which is first to collect all local datasets at a single location (computer). We can then feed this pooled data into a conventional ML method like linear regression.

The distributed training of ML models, at locations close to the actual data generation, can be beneficial for several reasons [18]:

- **Privacy.** FL methods are appealing for applications involving sensitive data (such as healthcare) as they do not require the exchange of raw data but only model (parameter) updates [15, 16]. By exchanging only model updates, FL methods are considered privacy-friendly in the sense of not l eaking (too much) sensitive information that is contained in the local datasets (see Lecture 9).

- **Robustness.** By relying on decentralized data and computation, FL methods offer robustness (to some extent) against hardware failures (such as "stragglers") and data poisonings (see Lecture 10).

- **Parallel Computing.** Many ML systems are based on mobile networks, consisting of humans equipped with smartphones. We can interpret a mobile network as a parallel computer which is constituted by smart-

phones that can communicate via radio links. This parallel computer allows to speed up computational tasks such as the computation of gradients required to train ML models (see Lecture 4).

- **Trading Computation against Communication.** Consider a FL application where local datasets are generated by low-complexity devices at remote locations that cannot be easily accessed. The cost of communicating raw local datasets to some central unit (which then trains a single global ML model) might be much higher than the computational cost incurred by using the low-complexity devices to (partially) train ML models [19].

- **Personalization.** FL can be used to train personalized ML models for collections of local datasets, which might be generated by smartphones (and their users) [20]. A key challenge for ensuring personalization is the heterogeneity of local datasets [21, 22]. Indeed, the statistical properties of different local datasets might vary significantly such they cannot be well modelled as independent and identically distributed (i.i.d.). Each local dataset induces a separate learning task that consists of learning useful parameter values for a local model. This course discusses FL methods to train personalized models via combining the information carried in decentralized and heterogeneous data (see Lecture 6).

## 1.3 Prerequisites

The main mathematical structure used to study and design FL algorithms is the Euclidean space $\mathbb{R}^d$. We therefore expect some familiarity with the

algebraic and geometric structure of $\mathbb{R}^d$. By algebraic structure, we mean the (real) vector space obtained from the elements ("vectors") in $\mathbb{R}^d$ along with the usual definitions of vector addition and multiplication by scalars in $\mathbb{R}$ [23, 24]. We will make heavy use of concepts from linear algebra to represent and manipulate data and ML models.

The metric structure of $\mathbb{R}^d$ will be used to study the (convergence) behaviour of FL algorithms. In particular, we will study FL algorithms that are obtained as fixed-point iterations of some non-linear operator on $\mathbb{R}^d$ which depends on the data (distribution) and ML models used within a FL system. A prime example for such a non-linear operator is the gradient step of gradient-based methods (see Lecture 4). The computational properties (such as convergence speed) of these FL algorithms can then be characterized via the contraction properties of the underlying operator [25].

A main tool for the design the FL algorithms are variants of gradient descent (GD). These gradient-based methods are based on approximating a differentiable function $f(\mathbf{x})$ locally by a linear function given by the gradient $\nabla f(\mathbf{x})$. We therefore expect some familiarity with multivariable calculus [5].

## 1.4 Related Courses

In what follows we briefly explain how this course CS-E4740 relates to selected courses at Aalto University.

- **CS-EJ3211 - Machine Learning with Python.** Teaches the application of basic ML methods using the Python package (library) `scikit-learn` [26]. CS-E4740 couples a network of basic ML methods using regularization techniques to obtain tailored (personalized) ML

models for local datasets. This coupling is required to adaptive pool local datasets obtain a sufficiently large training set for the personalized ML model.

- **CS-E4510 - Distributed Algorithms.** Teaches basic mathematical tools for the study and design of distributed algorithms that are implemented via distributed systems (computers) [27]. FL is enabled by distributed algorithms to train ML models from decentralized data (see Lecture 5).

- **CS-C3240 - Machine Learning (spring 2022 edition).** Teaches basic theory of ML models and methods [4]. CS-E4740 combines the components of basic ML methods, such as data representation and models, with network models. In particular, instead of a single dataset and a single model (such as a decision tree), we will study networks of local datasets and local models.

- **ABL-E2606 - Data Protection.** This course discusses important legal constraints ("laws"), including the European general data protection regulation (GDPR), for the use of data and, in turn, for the design of trustworthy FL methods.

- **MS-C2105 - Introduction to Optimization.** This course teaches basic optimisation theory and how to model applications as (linear, integer, and non-linear) optimization problems. CS-E4740 uses optimization theory and methods to formulate FL problems (see Lecture 3) and design FL methods (see Lecture 5).

- **ELEC-E5424 - Convex Optimization.** This course teaches advanced optimisation theory for the important class of convex optimization problems [28]. Convex optimization theory and methods can be used for the study and design of FL algorithms.

## 1.5 Main Goal of the Course

The overarching goal of the course is to demonstrate how to apply concepts from graph theory and mathematical optimization to analyze and design FL algorithms. Students will learn to formulate a given FL application as an optimization problem over an undirected empirical graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ whose nodes $i \in \mathcal{V}$ represent individual local datasets. We refer to this graph as the empirical graph of a collection of local datasets (see Lecture 3).

This course uses only undirected empirical graphs with a finite number $n$ of nodes, which we identify with the first $n$ positive integers:

$$\mathcal{V} := \{1, \ldots, n\}.$$

An edge $\{i, i'\} \in \mathcal{E}$ in the empirical graph $\mathcal{G}$ connects two different local datasets if they have similar statistical properties. We quantify the amount of similarity by the positive edge weight $A_{i,i'} > 0$.

We can formalize a FL application as an optimization problem associated with an empirical graph,

$$\min_{\mathbf{w}^{(i)}} \sum_{i \in \mathcal{V}} L_i \left( \mathbf{w}^{(i)} \right) + \lambda \sum_{\{i,i'\} \in \mathcal{E}} A_{i,i'} d^{(\mathbf{w}^{(i)}, \mathbf{w}^{(i')})}. \tag{1}$$

We refer to this problem as GTV minimization (GTVMin) and devote much of the course to the discussion of its computational and statistical properties.

The optimization variables $\mathbf{w}^{(i)}$ in (1) are local model parameters at the nodes $i \in \mathcal{V}$ of an empirical graph. The objective function in (1) consists of two components: The first component is a sum over all nodes of the loss values $L_i\left(\mathbf{w}^{(i)}\right)$ incurred by local model parameters at each node $i$. The second component is the sum of local model parameters variations across the edges $\{i, i'\}$ of the empirical graph.

## 1.6 Outline of the Course

Our course is roughly divided into three parts:

- **Part I: ML Refresher.** Lecture 2 introduces data, models and loss functions as three main components of ML. This lecture also explains how these components are combined within empirical risk minimization (ERM). We also discuss how regularization of ERM can be achieved via manipulating its three main components. We then explain when and how to solve regularized ERM via simple GD methods in Lecture 4. Overall, this part serves two main purposes: (i) to briefly recap basic concepts of ML in a simple centralized setting and (ii) highlight ML techniques (such as regularization) that are particularly relevant for the design and analysis of FL methods.

- **Part II: FL Theory and Methods.** Lecture 3 introduces the empirical graph as our main mathematical structure for representing collections of local datasets and corresponding tailored models. The undirected and weighted edges of the empirical graph represent statistical similarities between local datasets. Lecture 3 also formulates FL as an instance of

regularized empirical risk minimization (RERM) which we refer to as GTVMin. GTVMin uses the variation of personalized model parameters across edges in the empirical graph as regularizer. We will see that GTVMin couples the training of tailored (or "personalized") ML models such that well-connected nodes (clusters) in the empirical graph will obtain similar trained models. Lecture 4 discusses variations of gradient descent as our main algorithmic toolbox for solving GTVMin. Lecture 5 shows how FL algorithms can be obtained in a principled fashion by applying optimization methods, such as gradient-based methods, to GTVMin. We will obtain FL algorithms that can be implemented as iterative message passing methods for the distributed training of tailored ("personalized") models. Lecture 6 derives some main flavours of FL as special cases of GTVMin. The usefulness of GTVMin crucially depends on the choice for the weighted edges in the empirical graph. Lecture 7 discusses graph learning methods that determine a useful empirical graph via different notions of statistical similarity between local datasets.

- **Part III: Trustworthy AI.** Lecture 8 enumerates seven key requirements for trustworthy artificial intelligence (AI) that have been put forward by the European Union. These key requirements include the protection of privacy as well as robustness against (intentional) perturbations of data or computation. We then discuss how FL algorithms can ensure privacy protection in Lecture 9. Lecture 10 discusses how to evaluate and ensure robustness of FL methods against intentional perturbations (poisoning) of local dataset.

## 1.7 Assignments

The course will consist of assignments, each covering the topics of a corresponding lecture. Each assignment requires you to implement the concepts discussed in the corresponding lecture using Python. After solving the assignment, you can answer MyCourses quizzes.

## 1.8 Student Project

You can extend the basic variant (which is worth 5 credits) to 10 credits by completing a student project and peer review. This project requires you to formulate an application of your choice as a FL problem using the concepts from this course. You then have to solve this FL problem using the FL algorithms taught in this course. The main deliverable will be a project report which must follow the structure indicated in the template. You will then peer-review the reports of your fellow students by answering a detailed questionnaire.

## 1.9 Schedule

The course lectures are held on Mo. and Wed. at 16.15, during 28-Feb-2024 until 30-Apr-3024. You can find the detailed schedule and lecture halls following this link. As the course can be completed fully remote, we will record each lecture and add the recording to the YouTube playlist here in a timely fashion.

After each lecture, we will release the corresponding assignment at this site. You will have then at least one week to work on the assignment before

we open the corresponding quiz on the MyCourses page of the course (click me).

## 1.10 Ground Rules

Note that as a student following this course, you must act according to the Code of Conduct of Aalto University. In particular, the main ground rules for this course are:

- **BE HONEST.** This course includes many tasks that require independent work, including the coding assignments, the working on student projects and the peer review of student projects. You must not use other's work inappropriately. For example, it is not allowed to copy other's solutions to coding assignments. We will randomly choose students who have to explain their solutions (and corresponding answers to quiz questions).

- **BE RESPECTFUL.** My personal wish is that this course provides a safe space for an enjoyable learning experience. Any form of disrespectful behaviour, including any course-related communication platforms, will be sanctioned rigorously (including reporting to university authorities).

# 2 Lecture - "ML Basics"

This lecture covers basic ML techniques that are crucial for our study of FL. This lecture is signifcantly more extensive content-wise compared to the following lectures. However, it should be relatively easy to follow as it mainly refreshes pre-requisite knowledge.

## 2.1 Learning Goals

After this lecture, you should

- be familiar with the concept of data points (their features and labels), model and loss function,
- be familiar with ERM as a design principle for ML systems,
- know why and how validation is performed,
- know three different ways to regularize a ML method.

## 2.2 Three Components and a Design Principle

Machine Learning (ML) revolves around learning a hypothesis map $h$ out of a hypothesis space $\mathcal{H}$ that allows to accurately predict the label of a data point solely from its features. One of the most crucial steps in applying ML methods to a given application domain is the definition or choice of what precisely a data point is. Coming up with a good choice or definition of data points is not trivial as it influences the overall performance of a ML method in many different ways.

During this course we will focus mainly on one specific choice for the data points. In particular, we will consider data points that represent the daily weather condition around a weather station of the Finnish Meteorological Institute (FMI). We denote a specific data point by $\mathbf{z}$. It is characterized by the following features:

- name of the FMI weather station, e.g., "TurkuRajakari"
- latitude $\mathsf{lat}$ and longitude $\mathsf{lon}$ of the weather station, e.g., $\mathsf{lat} := 60.37788$, $\mathsf{lon} := 22.0964$,
- date of the day in format DDMMYYYY, e.g., 01022022
- minimum daytime temperature.

It is convenient to stack the features into a feature vector $\mathbf{x}$. The label $y \in \mathbb{R}$ of such a data point is the maximum daytime temperature.

We predict the label by the function value hypothesis $h(\mathbf{x})$. The prediction will typically be not perfect, i.e., $h(\mathbf{x}) \neq y$. We measure the prediction error by a loss function such as the squared error loss $L\left(\mathbf{z}, h\right) := \big(y - h(\mathbf{x})\big)^2$. It seems natural to choose (or learn) a hypothesis that incurs minimum average loss (or empirical risk) on a given set of data points $\mathcal{D} := \big\{\big(\mathbf{x}^{(1)}, y^{(1)}\big), \ldots, \big(\mathbf{x}^{m}, y^{(m)}\big)\big\}$. This is known as ERM,

$$\hat{h} \in \operatorname*{argmin}_{h \in \mathcal{H}} (1/m) \sum_{r=1}^{m} \big(y^{(r)} - h\big(\mathbf{x}^{(r)}\big)\big)^2 \tag{2}$$

As our notation indicates (using the symbol "$\in$" instead of ":="), there might be several different solutions to the optimization problem (2). Unless specified otherwise, $\hat{h}$ can be used to denote any hypothesis in $\mathcal{H}$ that has minimum average loss over $\mathcal{D}$.

A large class of ML methods use a paramterized model $\mathcal{H}$ with each hypothesis $h \in \mathcal{H}$ specified by a parameter vector $\mathbf{w} \in \mathbb{R}^d$. The prime example for a parametrized model is the linear model: $h(\mathbf{x}) := \mathbf{w}^T\mathbf{x}$ [4, Sec. 3.1]. Linear regression learns the parameters of a linear model by minimizing the average squared error loss. For linear regression, the ERM is equivalent to an optimization over the parameter space $\mathbb{R}^d$,

$$\widehat{\mathbf{w}}^{(\mathrm{LR})} \in \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^d} \underbrace{(1/m)\sum_{r=1}^{m}\left(y^{(r)} - \mathbf{w}^T\mathbf{x}^{(r)}\right)^2}_{:=f(\mathbf{w})}. \tag{3}$$

Note that (3) amounts to finding the minimum of a smooth and convex function

$$f(\mathbf{w}) = (1/m)\left[\mathbf{w}^T\mathbf{X}^T\mathbf{X}\mathbf{w} - 2\mathbf{y}^T\mathbf{X}\mathbf{w} + \mathbf{y}^T\mathbf{y}\right] \tag{4}$$

$$\text{with the feature matrix } \mathbf{X} := \left(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)}\right)^T \tag{5}$$

$$\text{and the label vector } \mathbf{y} := \left(y^{(1)}, \ldots, y^{(m)}\right)^T \text{ of the training set } \mathcal{D}. \tag{6}$$

Inserting (4) into (3) allows to formulate linear regression as

$$\widehat{\mathbf{w}}^{(\mathrm{LR})} \in \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^d} \mathbf{w}^T\mathbf{Q}\mathbf{w} + \mathbf{w}^T\mathbf{q} \tag{7}$$

$$\text{with } \mathbf{Q} := (1/m)\mathbf{X}^T\mathbf{X}, \mathbf{q} := -(2/m)\mathbf{X}^T\mathbf{y}.$$

To train a ML model $\mathcal{H}$ means to solve ERM (2) (or (3) for linear regression); the dataset $\mathcal{D}$ is therefore referred to as a training set. The trained model results in the learnt hypothesis $\hat{h}$. We obtain practical ML methods by applying optimiziation algorithms to solve (2). Two key questions arise:

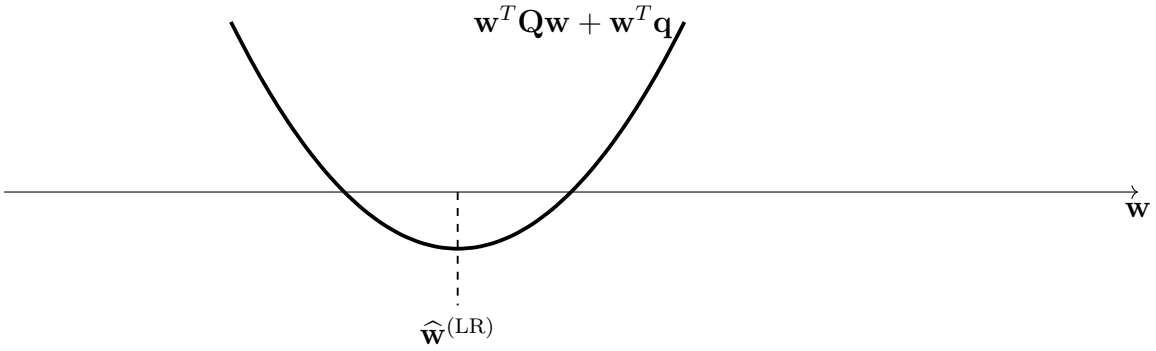


Figure 1: ERM (2) for linear regression amounts to minimizing a convex quadratic function $\mathbf{w}^T\mathbf{Q}\mathbf{w} + \mathbf{w}^T\mathbf{q}$.

- **computational aspects** How much computation is need to solve (2) ?
- **statistical aspects** How useful is the solution $\hat{h}$ to (2) in practice, i.e., how accurate is the prediction $\hat{h}(\mathbf{x})$ for the label $y$ of an **arbitrary** data point with features $\mathbf{x}$?

## 2.3 Computational Aspects of ERM

ML methods use optimization algorithms to solve (2) in order to learn a hypothesis $\hat{h}$. Within this course, we use optimization algorithms that are iterative methods: Starting from an initial choice $h^{(0)}$, they construct a sequence

$$h^{(0)}, h^{(1)}, h^{(2)}, \ldots,$$

which are hopefully increasingly accurate approximations to a solution $\hat{h}$ of (2). The computational complexity of such a ML method can be measured by the number of iterations required to guarantee some prescribed level of approximation.

When using a parameterized model and a smooth loss function, we can solve (3) by (variants of) gradient step descent: Starting from some initial parameters $\mathbf{w}^{(0)}$, we iterate the gradient step:

$$\begin{aligned}\mathbf{w}^{(k)} &:= \mathbf{w}^{(k-1)} - \alpha \nabla f\big(\mathbf{w}^{(k-1)}\big) \\ &= \mathbf{w}^{(k-1)} + (2\alpha/m) \sum_{r=1}^{m} \mathbf{x}^{(r)} \big(y^{(r)} - \big(\mathbf{w}^{(k-1)}\big)^T \mathbf{x}^{(r)}\big). \end{aligned} \tag{8}$$

How much computation do we need for one iteration of (8)? How many iterations do we need ? We will try to answer the latter question in Lecture 4. The first question can be answered more easily. Indeed, a naive evaluation of (8) requires around $m$ arithmetic operations (addition, multiplication).

It is instructive to consider the special case of a linear model which does not use any feature, i.e., $h(\mathbf{x}) = w$. For this extreme case, the ERM (3) has a simple closed-form solution:

$$\widehat{w} = (1/m) \sum_{r=1}^{m} x^{(r)}. \tag{9}$$

Thus, for this special case of the linear model, solving (9) amounts to summing $m$ numbers $x^{(1)}, \ldots, x^{(m)}$. It seems reasonable to assume that the amount of computation required for computing (9) is proportional to $m$.

## 2.4 Statistical Aspects of ERM

We have formulated the training of a linear model on a given training set as ERM (3). But how useful is its solution $\widehat{\mathbf{w}}$ for predicting the labels of data points outside the training set? Consider applying the learnt hypothesis $h^{(\widehat{\mathbf{w}})}$ to an arbitrary data point with label $y$ and features $\mathbf{x}$ that is not contained

in the training set. What can we say about the resulting prediction error $y - h^{(\widehat{\mathbf{w}})}(\mathbf{x})$ in general? In other words, how well does $h^{(\widehat{\mathbf{w}})}$ generalize beyond the training set.

Maybe the most widely used approach to study generalization of ML methods is via a probabilistic perspective. Here, we interpret each data point as a realization of an i.i.d. RV with probability distribution $p(\mathbf{x}, y)$. Under this i.i.d. assumption, we can evaluate the overall performance of a hypothesis $h \in \mathcal{H}$ via the expected loss (or risk)

$$\mathbb{E}\{L\left((\mathbf{x}, y), h\right)\}. \tag{10}$$

One example for a probability distribution $p(\mathbf{x}, y)$ is obtained via relating the label $y$ with the features $\mathbf{x}$ of a data point as

$$y = \overline{\mathbf{w}}^T \mathbf{x} + \varepsilon \text{ with } \mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \varepsilon \sim \mathcal{N}(0, \sigma^2). \tag{11}$$

A simple calculation reveals the expected squared error loss of a given linear hypothesis $h(\mathbf{x}) = \mathbf{x}^T \widehat{\mathbf{w}}$ as

$$\mathbb{E}\{(y - h(\mathbf{x}))^2\} = \|\overline{\mathbf{w}} - \widehat{\mathbf{w}}\|^2 + \sigma^2. \tag{12}$$

The component $\sigma^2$ can be interpreted as intrinsic noise level of the label $y$. We cannot hope to find a hypothesis with expected loss smaller than this level. The first component of the RHS in (12) is the estimation error $\|\overline{\mathbf{w}} - \widehat{\mathbf{w}}\|^2$ of a ML method that reads in the training set and delivers an estimate $\widehat{\mathbf{w}}$ (e.g., via (3)) for the parameters of a linear hypothesis.

We next study the estimation error $\overline{\mathbf{w}} - \widehat{\mathbf{w}}$ incurred by the specific estimate $\widehat{\mathbf{w}} = \widehat{\mathbf{w}}^{(\mathrm{LR})}$ (7) delivered by linear regression methods. To this end, we first

use the probabilistic model (11) to decompose the label vector $\mathbf{y}$ in (6) as

$$\mathbf{y} = \mathbf{X}\overline{\mathbf{w}} + \mathbf{n} \text{ , with } \mathbf{n} := \big(\varepsilon^{(1)}, \ldots, \varepsilon^{(m)}\big)^T. \tag{13}$$

Inserting (13) into (7) yields

$$\widehat{\mathbf{w}}^{(\mathrm{LR})} \in \operatorname*{argmin}_{\mathbf{w} \in \mathbb{R}^d} \mathbf{w}^T\mathbf{Q}\mathbf{w} + \mathbf{w}^T\mathbf{q}' + \mathbf{w}^T\mathbf{e} \tag{14}$$

$$\text{with } \mathbf{Q} := (1/m)\mathbf{X}^T\mathbf{X}, \mathbf{q}' := -(2/m)\mathbf{X}^T\mathbf{X}\overline{\mathbf{w}}, \text{ and } \mathbf{e} := -(2/m)\mathbf{X}^T\mathbf{n} \tag{15}$$

We illustrate the objective function of (14) in Figure 2. This function can be interpreted as a perturbation of the convex quadratic function $\mathbf{w}^T\mathbf{Q}\mathbf{w} + \mathbf{w}^T\mathbf{q}'$ which is minimized at $\mathbf{w} = \overline{\mathbf{w}}$. In general, the minimizer $\widehat{\mathbf{w}}^{(\mathrm{LR})}$ delivered by linear regression is different from $\overline{\mathbf{w}}$ due the perturbation term $\mathbf{w}^T\mathbf{e}$ in (14).

Let us assume in what follows that the matrix $\mathbf{Q} = (1/m)\mathbf{X}^T\mathbf{X}$ is invertible.[1] It is then not too difficult to verify the following upper bound

$$\big\|\widehat{\mathbf{w}}^{(\mathrm{LR})} - \overline{\mathbf{w}}\big\|_2 \leq \|\mathbf{e}\|_2 \,/\lambda_{\min}\big(\mathbf{Q}\big). \tag{16}$$

Here, $\lambda_{\min}\big(\mathbf{Q}\big)$ denotes the smallest eigenvalue of the matrix $\mathbf{Q} = (1/m)\mathbf{X}^T\mathbf{X} \in \mathbb{R}^{d\times d}$. Note that the matrix $\mathbf{Q}$ is psd and therefore its eigenvalues are all real-valued and non-negative [24]. Moreover, since we assume $\mathbf{Q}$ is invertible, they are strictly positive and, in turn, $\lambda_{\min}\big(\mathbf{Q}\big) > 0$.

## 2.5 Validation and Diagnosis of ML

The above analysis of the generalization error started from postulating a probabilistic model for the generation of data points. However, this probabilistic model might be wrong and the bound (16) does not apply. Thus, we

[1] Can you think of sufficient conditions on the feature matrix of the training set that ensure $\mathbf{Q} = (1/m)\mathbf{X}^T\mathbf{X}$ is invertible?

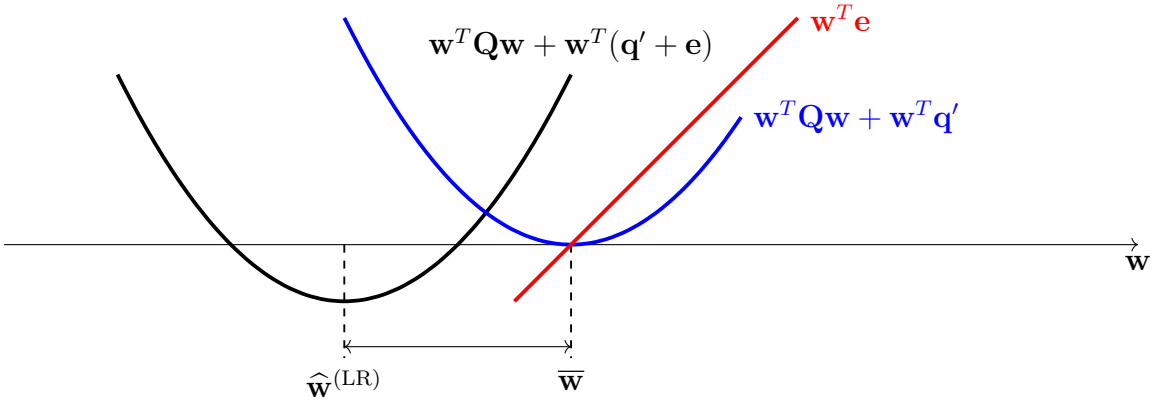


Figure 2: The estimation error of linear regression is determined by the effect a linear perturbation term $\mathbf{w}^T\mathbf{e}$ on the minimizer of a convex quadratic function.

might want to use a more data-driven approach for assesing the usefulness of a trained model.

Validation methods try to find out if a learnt hypothesis $\hat{h}$, does well outside the training set. In its most basic form, model validation amounts to computing the average loss of a learnt hypothesis $\hat{h}$ on some data points that have not been included in the training set. We refer to these data points as the validation set.

The most basic workflow of ML model training and validation can be summarized as follows:

1. gather a dataset and choose a model $\mathcal{H}$

2. split dataset into a training set $\mathcal{D}^{(\mathrm{train})}$ and a validation set $\mathcal{D}^{(\mathrm{val})}$

3. learn a hypothesis via solving ERM

$$\widehat{h} \in \operatorname*{argmin}_{h \in \mathcal{H}} \sum_{(\mathbf{x},y) \in \mathcal{D}^{(\text{train})}} L\left((\mathbf{x}, y), h\right) \tag{17}$$

4. compute resulting training error

$$E_t := (1/|\mathcal{D}^{(\text{train})}|) \sum_{(\mathbf{x},y) \in \mathcal{D}^{(\text{train})}} L\left((\mathbf{x}, y), \widehat{h}\right)$$

5. compute validation error

$$E_v := (1/|\mathcal{D}^{(\text{val})}|) \sum_{(\mathbf{x},y) \in \mathcal{D}^{(\text{val})}} L\left((\mathbf{x}, y), \widehat{h}\right)$$

We can diagnose a ERM based ML method by comparing its training error with its validation error. This diagnosis is further enabled if we know a baseline $E^{(\text{ref})}$ . One important source for a baseline $E^{(\text{ref})}$ are probabilistic models for the data points (see Section 2.4).

Given a probabilistic model $p(\mathbf{x}, y)$, we can compute the minimum achievable risk (10). Indeed, the minimum achievable risk is precisely the expected loss of the Bayes estimator $\widehat{h}(\mathbf{x})$ of the label $y$, given the features $\mathbf{x}$ of a data point. The Bayes estimator $\widehat{h}(\mathbf{x})$ is fully determined by the probability distribution $p(\mathbf{x}, y)$ [29, Chapter 4].

A further potential source for a baseline $E^{(\text{ref})}$ is an existing, but for some reason unsuitable, ML method. This existing ML method might be computationally too expensive to be used for the ML application at end. However, we might still use its statistical properties as a benchmark.

We can also use the performance of human experts as a baseline. If we want to develop a ML method that detects certain type of skin cancers from

images of the skin, a benchmark might be the current classification accuracy achieved by experienced dermatologists [30].

We can diagnose a ML method by comparing the training error $E_t$ with the validation error $E_v$ and (if available) the benchmark $E^{(\mathrm{ref})}$.

- $E_t \approx E_v \approx E^{(\mathrm{ref})}$: The training error is on the same level as the validation error error and the baseline. There is not much to improve here since the validation error is already close to the baseline. Moreover, the training error is not much smaller than the validation error which indicates that there is no overfitting.

- $E_v \gg E_t$: The validation error is significantly larger than the training error. This is an indicator for overfitting which can be addressed either by reducing the effective dimension of the hypothesis space or by increasing the size of the training set. We can reduce the effective dimension of the hypothesis space by using fewer features, a smaller maximum depth of decision trees or fewer layers in an ANN. An alternative to this discrete model selection, we can also reduce the effective dimension of a hypothesis space via regularization techniques.

- $E_t \approx E_v \gg E^{(\mathrm{ref})}$: The training error is on the same level as the validation error and both are significantly larger than the baseline. Since the training error is not much smaller than the validation error, the learnt hypothesis seems to not overfit the training set. However, the training error achieved by the learnt hypothesis is significantly larger than the baseline. There can be several reasons for this to happen. First, it might be that the hypothesis space is too small, i.e., it does not

include a hypothesis that provides a good approximation for the relation between features and label of a data point. One remedy to this situation is to use a larger hypothesis space, e.g., by including more features in a linear model, using higher polynomial degrees in polynomial regression, using deeper decision trees or ANNs (deep ANN (deep net)s). Second, besides the model being too small, another reason for a large training error could be that the optimization algorithm used to solve ERM (17) is not working properly (see Lecture 4).

- $E_t \gg E_v$: The training error is significantly larger than the validation error. The idea of ERM (17) is to approximate the risk (10) of a hypothesis by its average loss on a training set $\mathcal{D} = \{(\mathbf{x}^{(r)}, y^{(r)})\}_{r=1}^{m}$. The mathematical underpinning for this approximation is the law of large numbers which characterizes the average of (realizations of) i.i.d. RVs. The accuracy of this approximation depends on the validity of two conditions: First, the data points used for computing the average loss "should behave" like realizations of i.i.d. RVs with a common probability distribution. Second, the number of data points used for computing the average loss must be sufficiently large.

  Whenever the data points behave different than the the realizations of i.i.d. RVs or if the size of the training set or validation set is too small, the interpretation (and comparison) of the training error and the validation error of a learnt hypothesis becomes more difficult. As an extreme case, the validation set might consist of data points for which every hypothesis incurs small average loss. Here, we might try to increase the size of the validation set by collecting more labeled

data points or by using data augmentation (see Section 2.6). If the size of training set and validation set are large but we still obtain $E_t \gg E_v$, one should verify if data points in these sets conform to the i.i.d. assumption. There are principled statistical test for the validity of the i.i.d. assumption for a given dataset (see [31] and references therein).

## 2.6 Regularization

Consider a ERM-based ML method using a hypothesis space $\mathcal{H}$ and dataset $\mathcal{D}$ (we assume all data points are used for training). A key parameter for such a ML methos is the ratio $d_{\text{eff}}(\mathcal{H})/|\mathcal{D}|$ between the model size $d_{\text{eff}}(\mathcal{H})$ and the number $|\mathcal{D}|$ of data points. The tendency of the ML method to overfit increases with the ratio $d_{\text{eff}}(\mathcal{H})/|\mathcal{D}|$.

Regularization techniques aim at reducing the ratio $d_{\text{eff}}(\mathcal{H})/|\mathcal{D}|$ via three (largely equivalent) routes:

- collect more data points, possibly via data augmentation (see Fig. 3),
- add a penalty term $\lambda \mathcal{R}\big\{h\big\}$ to the average loss in ERM (2) (see Fig. 3 )
- shrink the hypothesis space, e.g., by adding constraints on the model parameters such as $\|\mathbf{w}\|_2 \leq 10$.

[4, Ch. 7] discusses the equivalence between these three perspectives on regularization in somewhat more detail.

One important example for regularization via adding a penalty term to the average loss is ridge regression. In particuar, ridge regression uses the regularizer $\mathcal{R}\big\{h\big\} := \|\mathbf{w}\|_2^2$ for a linear hypothesis $h(\mathbf{x}) := \mathbf{w}^T\mathbf{x}$. Thus, ridge

regression learns the weights of a linear hypothesis via solving

$$\widehat{\mathbf{w}}^{(\text{ridge})} \in \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^d} \left[ (1/m) \sum_{r=1}^{m} \left( y^{(r)} - \mathbf{w}^T \mathbf{x}^{(r)} \right)^2 + \lambda \left\| \mathbf{w} \right\|_2^2 \right]. \tag{18}$$

The objective function in (18) is also obtained if we replace each data point $(\mathbf{x}, y) \in \mathcal{D}$ by a sufficient large number of i.i.d. realizations of

$$(\mathbf{x} + \mathbf{n}, y) \text{ with } \mathbf{n} \sim \mathcal{N}(\mathbf{0}, \lambda \mathbf{I}). \tag{19}$$

Thus, ridge regression (18) is equivalent to linear regression applied to an augmented variant $\mathcal{D}'$ of $\mathcal{D}$. The augmentation $\mathcal{D}'$ is obtained by replacing each data point $(\mathbf{x}, y) \in \mathcal{D}$ with a sufficiently large number of noisy copies. Each copy is obtained by adding a i.i.d. realization $\mathbf{n}$ of a zero-mean Gaussian noise with covariance matrix $\lambda \mathbf{I}$ to the features $\mathbf{x}$ (see (19)). The label of each copy is equal to $y$, i.e., the label is not perturbed.

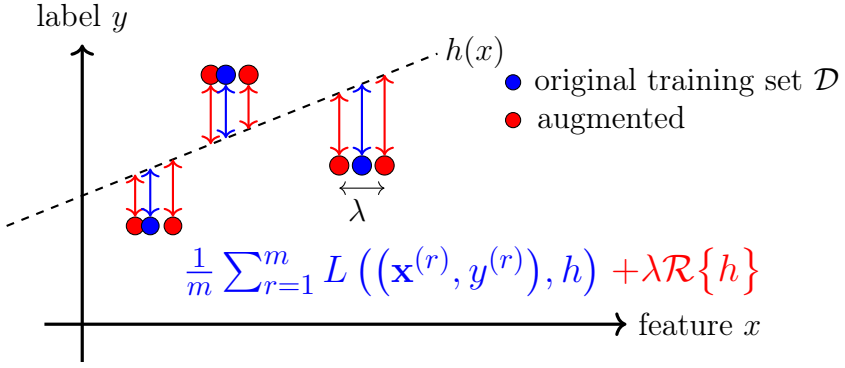


Figure 3: Equivalence between data augmentation and loss penalization.

To study the computational aspects of ridge regression, let us rewrite (18)

as

$$\widehat{\mathbf{w}}^{(\text{ridge})} \in \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^d} \mathbf{w}^T\mathbf{Q}\mathbf{w} + \mathbf{w}^T\mathbf{q}$$
$$\text{with } \mathbf{Q} := (1/m)\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I},\ \mathbf{q} := (-2/m)\mathbf{X}^T\mathbf{y}. \tag{20}$$

Thus, like linear regression (7), also ridge regression amounts to minimizing a convex quadratic function. A main difference between linear regression (7) and ridge regression (for $\lambda > 0$) is that the matrix $\mathbf{Q}$ in (20) is guaranteed to be invertible for any training set $\mathcal{D}$. In contrast, the matrix $\mathbf{Q}$ in (7) for linear regression might be singular for some training sets.[2]

## 2.7 Assignment

The coding assignment revolves around weather data collected by the FMI and stored in a simple csv file. This file contains temperature measurements at different locations in Finland. Each data point is characterized by

- the coordinates (latitude and longitude) of the location,
- a time stamp that indicates when the temperature has been measured,
- the temperature measurement itself.

[2]Consider the extreme case where all features of any data point are zero.

# 3 Lecture - "FL Design Principle"

Lecture 2 reviewed ML methods that use numeric arrays to store data and model parameters. We have also discussed ERM as a design principle or practical ML systems. This lecture will extend these concepts to FL applications. Section 3.2 introduces empirical graphs to store collections of local datasets and corresponding parameters of local models. Section 3.3 presents our main design principle for FL systems: We use the variation of local model parameters across the edges of an empirical graph for the coupling (or regularization) of the individual local models.

## 3.1 Learning Goals

After this lecture, you should

- be familiar with the concept of an empirical graph,
- know how connectivity is related to spectrum of Laplacian matrix,
- know some measures for the variation of local models,
- be familiar with the concept of GTVMin.

## 3.2 Empirical Graphs and Their Laplacian

Consider a FL application that invovles a collection of local datasets $\mathcal{D}^{(1)}, \ldots, \mathcal{D}^{(2)}$. Our goal is to train a personalized model $\mathcal{H}^{(i)}$ for each local dataset $\mathcal{D}^{(i)}$, with $i = 1, \ldots, n$. We represent such a collection of local datasets and (personal) local models, along with their relations, by an empirical graph. Figure 4 depicts an example for an empirical graph.

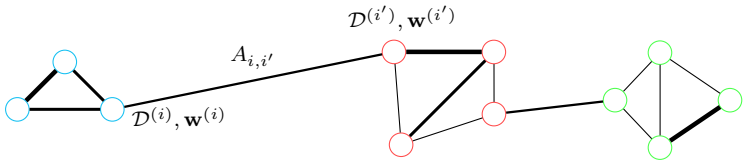


Figure 4: Example of an empirical graph whose nodes $i \in \mathcal{V}$ carry local datasets $\mathcal{D}^{(i)}$ and local models that are parametrized by local model parameters $\mathbf{w}^{(i)}$.

An empirical graph is an undirected weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ whose nodes $\mathcal{V} := \{1, \ldots, n\}$ represent local datasets $\mathcal{D}^{(i)}$, for $i \in \mathcal{V}$. Each node $i \in \mathcal{V}$ of the empirical graph $\mathcal{G}$ carries a separate local dataset $\mathcal{D}^{(i)}$.

To build intuition, think of a local dataset $\mathcal{D}^{(i)}$ as a labelled dataset

$$\mathcal{D}^{(i)} := \left\{ \left(\mathbf{x}^{(i,1)}, y^{(i,1)}\right), \ldots, \left(\mathbf{x}^{(i,m_i)}, y^{(i,m_i)}\right) \right\}. \tag{21}$$

Here, $\mathbf{x}^{(i,r)}$ and $y^{(i,r)}$ denote, respectively, the features and the label of the $r$th data point in the local dataset $\mathcal{D}^{(i)}$. Note that the size $m_i$ of the local dataset might vary between different nodes $i \in \mathcal{V}$.

It is convenient to collect the feature vectors $\mathbf{x}^{(i,r)}$ and labels $y^{(i,r)}$ into a feature matrix $\mathbf{X}^{(i)}$ and label vector $\mathbf{y}^{(i)}$, respectively,

$$\mathbf{X}^{(i)} := \left(\mathbf{x}^{(i,1)}, \ldots, \mathbf{x}^{(i,m_i)}\right)^T, \text{ and } \mathbf{y} := \left(y^{(1)}, \ldots, y^{(m_i)}\right)^T. \tag{22}$$

The local dataset $\mathcal{D}^{(i)}$ can then be represented compactly by the matrix $\mathbf{X}^{(i)} \in \mathbb{R}^{m_i \times d}$ and the vector $\mathbf{y}^{(i)} \in \mathbb{R}^{m_i}$.

Besides its local dataset $\mathcal{D}^{(i)}$, each node $i \in \mathcal{G}$ also carries a local model $\mathcal{H}^{(i)}$. Whitin this course, we focus on local models that are parametrized by local model parameters $\mathbf{w}^{(i)} \in \mathbb{R}^d$, for $i = 1, \ldots, n$. The usefulness of a

specific choice for the local model parameter $\mathbf{w}^{(i)}$ is measured by a local loss function $L_i\left(\mathbf{w}^{(i)}\right)$, for $i = 1, \ldots, n$.

An undirected edge $\{i, i'\} \in \mathcal{E}$ between two different nodes $i, i' \in \mathcal{V}$ couples the training of the corresponding local models $\mathcal{H}^{(i)}, \mathcal{H}^{(i')}$. We quantify the strength of this coupling by a positive edge weight $A_{i,i'} > 0$. The coupling will be implemented by penalizing the discrepancy between local model parameters $\mathbf{w}^{(i)}$ and $\mathbf{w}^{(i')}$ (see Section 3.3). Our main choice for measuring this discrepancy will be the squared Euclidean distance $\left\|\mathbf{w}^{(i)} - \mathbf{w}^{(i')}\right\|_2^2$.

We can characterize the connectivity of an empirical graph via the eigenvalues and eigenvectors of its Laplacian matrix $\mathbf{L} \in \mathbb{R}^{n \times n}$. The Laplacian matrix is defined element-wise as

$$L_{i,i'} := \begin{cases} -A_{i,i'} & \text{for } i \neq i', \{i, i'\} \in \mathcal{E} \\ \sum_{i'' \neq i} A_{i,i''} & \text{for } i = i' \\ 0 & \text{else.} \end{cases} \tag{23}$$

The Laplacian matrix is psd which follows from the identity

$$\mathbf{w}^T \mathbf{L} \mathbf{w} = \sum_{\{i,i'\} \in \mathcal{E}} A_{i,i'} \left\|\mathbf{w}^{(i)} - \mathbf{w}^{(i')}\right\|_2^2$$
$$\text{for any } \mathbf{w} := \left(\left(\mathbf{w}^{(1)}\right)^T, \ldots, \left(\mathbf{w}^{(n)}\right)^T\right)^T. \tag{24}$$

Since the matrix $\mathbf{L}$ is psd, all its eigenvalues are real-valued and non-negative. We denote its increasingly ordered eigenvalues by

$$0 \leq \lambda_1 \leq \lambda_2 \ldots \leq \lambda_n. \tag{25}$$

Acccording to (24), we can measure the total variation of local model parameters by stacking them into a single vector $\mathbf{w} \in \mathbb{R}^{nd}$ and computing the quadratic form $\mathbf{w}^T \mathbf{L} \mathbf{w}$.

One immediate consequence of (24) is that any collection of identical local model parameters, $\mathbf{w}^{(i)} = \mathbf{w}^{(i')}$ results in an eigenvector

$$\mathbf{c} = \left( \left(\mathbf{w}^{(1)}\right)^T, \ldots, \left(\mathbf{w}^{(n)}\right)^T \right)^T. \tag{26}$$

with eigenvalue $\lambda = 0$. Thus, the eigenvalue $\lambda = 0$ conincides with the smallest eigenvalue $\lambda_1$ (see (25)).

The second eigenvalue $\lambda_2$ of the Laplacian matrix provides a great deal of information about the connectivity structure of $\mathcal{G}$. Consider the case $\lambda_2 = 0$, i.e., besides $\mathbf{c}$ there is another eigenvector with zero eigenvalue. Then, the graph $\mathcal{G}$ contains two subsets (components) of nodes that do not have any edge between them.

On the other hand, if $\lambda_2 > 0$ then $\mathcal{G}$ is connected. Moreover, the larger the value of $\lambda_2$, the stronger the connectivity between the nodes in $\mathcal{G}$. We can make this vague statement more precise via the identity (24).

The total variation on the RHS (24) is a measure for the connectivity. Indeed, if we assume that the local model parameters $\mathbf{w}^{(i)}$ are different, then adding an edge will increase the total variation. We can lower bound this total variation as

$$\sum_{\{i,i'\} \in \mathcal{E}} A_{i,i'} \left\| \mathbf{w}^{(i)} - \mathbf{w}^{(i')} \right\|_2^2 \geq \lambda_2 \sum_{i=1}^{n} \left\| \mathbf{w}^{(i)} - \mathbf{m} \right\|_2^2. \tag{27}$$

Here, $\mathbf{m} = (1/n) \sum_{i=1}^{n} \mathbf{w}^{(i)}$ is the average of all local model parameters. The quantity $\sum_{i=1}^{n} \left\| \mathbf{w}^{(i)} - \mathbf{m} \right\|_2^2$ has a geometric interpretation: It is the squared Euclidean norm of the projection of the stacked local model parameters $\mathbf{w} := \left( \left(\mathbf{w}^{(1)}\right)^T, \ldots, \left(\mathbf{w}^{(n)}\right)^T \right)^T$ on the subspace

$$\left\{ \left(\mathbf{a}^T, \ldots, \mathbf{a}^T\right)^T, \text{ for some } \mathbf{a} \in \mathbb{R}^d \right\} \subseteq \mathbb{R}^{dn}. \tag{28}$$


Figure 5: Left: Some empirical graph $\mathcal{G}$ consisting of $n = 4$ nodes. Right: Equivalent fully connected empirical graph $\mathcal{G}'$ with the same nodes and non-zero edge weights $\mathbf{A}' = \mathbf{A}$.

It might be convenient to replace a given empirical graph $\mathcal{G}$ with an equivalent fully connected empirical graph $\mathcal{G}'$ (see Figure 5). The graph $\mathcal{G}'$ has an edge between each pair of different nodes $i, i'$,

$$\mathcal{E}' = \big\{\{i, i'\} \text{ , with some } i, i' \in \mathcal{V}, i \neq i'\big\}.$$

The edge weights are chosen $A'_{i,i'} = A_{i,i'}$ for any edge $\{i, i'\} \in \mathcal{E}$ and $A'_{i,i'} = 0$ otherwise.

Note that the undirected edges $\mathcal{E}$ of an empirical graph encode a symmetric notion of similarity between local datasets: If the local dataset $\mathcal{D}^{(i)}$ at node $i$ is similar to the local dataset $\mathcal{D}^{(i')}$ at node $i'$, i.e., $\{i, i'\} \in \mathcal{E}$, then also the local dataset $\mathcal{D}^{(i')}$ is similar to the local dataset $\mathcal{D}^{(i)}$.

## 3.3 Generalized Total Variation Minimization

Consider data with empirical graph $\mathcal{G}$ whose nodes $i \in \mathcal{V}$ carry local datasets $\mathcal{D}^{(i)}$ and local model parametrized by the vector $\mathbf{w}^{(i)}$. To learn these parameter vectors, we try to minimize their local loss and at the same time enforce a small total variation. The optimal balance are the solutions of the following

opimization problem, which we refer to as generalized total variation (GTV) minimization,

$$\big\{\widehat{\mathbf{w}}^{(i)}\big\}_{i=1}^{n} \in \operatorname*{argmin}_{\{\mathbf{w}^{(i)}\}} \sum_{i\in\mathcal{V}} L_i\left(\mathbf{w}^{(i)}\right) + \lambda \sum_{i,i'\in\mathcal{V}} A_{i,i'} \left\|\mathbf{w}^{(i)} - \mathbf{w}^{(i')}\right\|_2^2 \quad \text{(GTVMin)}. \tag{29}$$

Note that GTVMin is an instance of RERM: The regularizer is the total variation of local model parameters over weighted edges $A_{i,i'}$ of the empirical graph. Clearly, the empirical graph is an important design choice for GTVMin-based methods. This choice can be guided by computational aspects and statistical aspects of GTVMin-based FL systems.

Some application domains allow to leverage domain expertise to guess a useful choice for the empirical graph. If local datasets are generated at different geographic locations, we might use nearest neighbor graphs based on geodesic distances between data generators (e.g., FMI weather stations). Lecture 7 will also discuss graph learning methods that determine edge weights $A_{i,i'}$ in a fully data-driven fashion.

Let us now consider the special case of GTVMin with local models being a linear model. For each node $i \in \mathcal{V}$ of the empirical graph, we want to learn the weights $\mathbf{w}^{(i)}$ of a linear hypothesis $h^{(i)}(\mathbf{x}) := \left(\mathbf{w}^{(i)}\right)^T \mathbf{x}$. We measure the quality of the weigths via the average squared error loss

$$\begin{aligned} L_i\left(\mathbf{w}^{(i)}\right) &:= (1/m_i) \sum_{r=1}^{m_i} \left(y^{(i,r)} - \left(\mathbf{w}^{(i)}\right)^T \mathbf{x}^{(i,r)}\right)^2 \\ &\overset{(22)}{=} (1/m_i) \left\|\mathbf{y}^{(i)} - \mathbf{X}^{(i)}\mathbf{w}^{(i)}\right\|_2^2. \end{aligned} \tag{30}$$

Inserting (30) into (29), yields the following instance of GTVMin to train

local linear models,

$$\big\{\widehat{\mathbf{w}}^{(i)}\big\}_{i=1}^{n} \in \operatorname*{argmin}_{\{\mathbf{w}^{(i)}\}} \sum_{i\in\mathcal{V}} (1/m_i) \left\|\mathbf{y}^{(i)} - \mathbf{X}^{(i)}\mathbf{w}^{(i)}\right\|_2^2 + \lambda \sum_{i,i'\in\mathcal{V}} A_{i,i'} \left\|\mathbf{w}^{(i)} - \mathbf{w}^{(i')}\right\|_2^2. \tag{31}$$

The identity (24) allows us to rewrite (31) as

$$\big\{\widehat{\mathbf{w}}^{(i)}\big\}_{i=1}^{n} \in \operatorname*{argmin}_{\mathbf{w}=\mathrm{stack}\left\{\widehat{\mathbf{w}}^{(i)}\right\}_{i=1}^{n}} \sum_{i\in\mathcal{V}} (1/m_i) \left\|\mathbf{y}^{(i)} - \mathbf{X}^{(i)}\mathbf{w}^{(i)}\right\|_2^2 + \lambda \mathbf{w}^T \mathbf{L}\mathbf{w}. \tag{32}$$

### 3.3.1 Computational Aspects of GTVMin

Lecture 5 will apply optimization methods to solve GTVMin, resulting in practical FL algorithms. Different instances of GTVMin favor different classes of optimization methods. Using a differentiable loss function allows to apply gradient-based methods (see Lecture 4) to solve GTVMin. Another important class of loss functions is characterized by allowing to efficiently compute the proximity operator

$$\mathbf{prox}_{L,\rho}(\mathbf{w}) := \operatorname*{argmin}_{\mathbf{x}'} L(\mathbf{w}') + (\rho/2)\left\|\mathbf{w} - \mathbf{w}'\right\|_2^2 \text{ for some } \rho > 0.$$

Some authors refer to functions $L$ for which $\mathbf{prox}_{L,\rho}(\mathbf{w})$ can be computed easily as *simple* or proximable [32]. GTVMin with proximable loss functions can be solved quite efficiently via proximal algorithms [33].

Besides influencing the choice of optimization method, the design choices underlying GTVMin also determine the amount of computation needed by a given optimization method. For example, using an empirical graph with a small number of edges "sparse graphs") typically results in a smaller computational complexity. The amount of computation required by GTVMin-

based FL algorithms is typically proportional to the number of edges in the empirical graph (see Lecture 5).

### 3.3.2 Statistical Aspects of GTVMin

The empirical graph should contain sufficient number of edges between nodes that carry statistically similar local datasets. This allows regularization techniques to adaptively pool local datasets into clusters of (approximately) homogeneous data (see Section 6.3).

Statistically, we want the loss function to favour local model parameters that result in a robust and accurate trained local model, with model parameters $\widehat{\mathbf{w}}^{(i)}$, for each node $i \in \mathcal{V}$.

## 3.4 Assignment

# 4 Lecture - "Gradient Methods"

Lecture 2 introduced ERM as a central design principle for ML methods. Similarly, Lecture 3 introduced GTVMin as a central design principle for FL methods. ERM and GTVMin are optimization problems whose computational and statistical properties depend on the underlying choices for data, model and loss function.

Many popular design chocies for GTVMin result in the problem of minimizing a smooth objective function over a convex subset of the parameter space $\mathbb{R}^d$. Gradient-based methods are widely used to train ML models. These methods share a core idea: approximate the objective function locally using its gradient at the current choice for the model parameters. Lecture 5 discusses FL algorithms obtained from direct application of gradient-based methods to solving GTVMin.

## 4.1 Learning Goals

After this lecture, you should

- be able to analyze the effect of taking a gradient step for a smooth and strongly convex objective function
- understand the crucial role of the step size or learning rate
- know some stopping criterion
- be able to analyze the effect of perturbations in the gradient step
- know how to use projected GD to cope with constrains on model parameters

## 4.2 The Basic Idea of the Gradient Step

gradient-based methods are interative algorithms for finding the minimimum of a differentiable $f(\mathbf{w})$. One example for such a function is objective function (3) of linear regression. A gradient step updates a current choice for $\mathbf{w}^{(\mathrm{curr})}$ along the opposite direction of the gradient $\nabla f(\mathbf{w})$ at the current choice,

$$\mathbf{w}^{(\mathrm{new})} := \mathbf{w}^{(\mathrm{curr})} - \alpha \nabla f\big(\mathbf{w}^{(\mathrm{curr})}\big). \tag{33}$$

The gradient step (33) involves the factor $\alpha$ which is referred to as step-size or learning rate.

The usefulness of gradient-based methods depends crucially on the difficulty of evaluating the gradient. Evaluating the gradient of a given function has been made convenient by modern software libraries (such as PyTorch) that provide quite efficient methods for computing the gradient (autograd/backprop/...). However, besides the actual compuation of the gradient, it might be challegning to gather the required data points which define the objective function (empirical risk).

Algorithm 1 summarizes the most basic instance of gradient-based methods.

## 4.3 Hyperparameter of gradient-based methods

Note that Algorithm 1, as most other gradient-based methods, involve at least two hyper-parameters: (i) the learning rate $\alpha$ used for the gradient step and (ii) a stopping criterion this is used to decide when to stop repeating the gradient step.

**Algorithm 1** A blueprint for gradient-based methods

**Input:** function $f(\mathbf{w})$ ; learning rate $\alpha > 0$; some stopping criterion;

**Initialize:** set $\mathbf{w}^{(0)} := \mathbf{0}$; set iteration counter $r := 0$

1: **repeat**

2: $r := r + 1$ (increase iteration counter)

3: $\mathbf{w}^{(r)} := \mathbf{w}^{(r-1)} - \alpha \nabla f\big(\mathbf{w}^{(r-1)}\big)$ (do a gradient step (33))

4: **until** stopping criterion is met

**Output:** $\widehat{\mathbf{w}} := \mathbf{w}^{(r)}$ (hopefully $f\big(\widehat{\mathbf{w}}\big) \approx \min_{\mathbf{w}} f(\mathbf{w})$)

## 4.4 Perturbed Gradient Step

## 4.5 Constraints

## 4.6 Assignment

# 5 Lecture - "FL Algorithms"

This lecture applies some of the gradient-based methods from Lecture 4 to solve GTVMin from Lecture 3. It turns out, that the resulting FL algorithms can be implemented by message passing over the edges of the empirical graph.

## 5.1 Learning Goals

After this lecture, you should

- be able to derive the gradient for GTVMin with local linear models.
- be able to implement the gradient step for GTVMin as message passing

## 5.2 Gradient Step for GTVMin

## 5.3 Message Passing Implementation

## 5.4 Assignment

# 6 Lecture - "FL Main Flavors"

Lecture 3 discussed GTVMin as a main design principle for FL algorithms that have been obtained in Lecture 5 by applying some of the gradient-based methods from Lecture 4. This lecture discusses some important special cases of GTVMin that are obtained for specific choies for the underyling empirical graph.

## 6.1 Learning Goals

After this lecture, you should know about the following main flavours of FL:

- centralized FL
- clustered FL
- horizontal FL
- vertical FL

## 6.2 Centralized FL

## 6.3 Clustered FL

Many applications generate local datasets which do not carry sufficient statistical power to guide learning of model parameters $\mathbf{w}^{(i)}$ (see Section **??**). As a case in point, consider a local dataset $\mathcal{D}^{(i)}$ of the form (21), with feature vectors $\mathbf{x}^{(r)} \in \mathbb{R}^d$ with $m_i \ll d$. We would like to learn the parameter vector $\mathbf{w}^{(i)}$ of a linear hypothesis $h(\mathbf{x}) = \mathbf{x}^T \mathbf{w}^{(i)}$.

## 6.4 Horizontal FL

## 6.5 Vertical FL

## 6.6 Assignment

# 7 Lecture - "Graph Learning"

Lecture 3 discussed GTVMin as a main design principle for FL algorithms. The computational and statistical properties of these algorithms crucially depend on the choice for the empirical graph. In some applications, domain expertise can guide the choice for the empirical graph. However, it might be useful to learn the empirical graph in a data-driven fashion. This lecture discusses some of these graph learning techniques.

## 7.1 Learning Goals

After this lecture, you should

- understand the role of empirical graphs as a crucial design choice for GTVMin-based methods (computational aspects, statistical aspects)
- know some quantitative measures for the similarity between local datasets
- be able to learn a graph form given pairwise similarities and structural contraints (e.g., bounded node degree)

## 7.2 Measuring (Dis-)Similarity Between Datasets

The above informal notion of similarity between local datasets. can be made precise via a probabilistic model. Here, we interpret the local dataset $\mathcal{D}^{(i)}$ as realizations of RVs with some parametrized probability distribution $p^{(i)}\big(\mathcal{D}^{(i)}; \mathbf{w}^{(i)}\big)$.

The discrepancy (or lack of similarity) between local datasets $\mathcal{D}^{(i)}$ and $\mathcal{D}^{(i')}$ could then be defined via the Euclidean distance

$$d^{(i,i')} := \left\| \mathbf{w}^{(i)} - \mathbf{w}^{(i')} \right\|_2 ,$$

between the parameters of the probability distributions.

If local datasets consist of a single numeric measurement $y^{(i)}$, we can use the discrepancy measure $d^{(i,i')} := \left| y^{(i)} - y^{(i')} \right|$ [34].

## 7.3 Graph Learning Methods

Assume we have constructed a useful measure $d^{(i,i')} \in \mathbb{R}_+$ for the discrepancy between local datasets $\mathcal{D}^{(i)}, \mathcal{D}^{(i')}$. We can then formulate the problem of learning the edge weights $A_{i,i'} \in \mathbb{R}_+$ as the optimization problem

$$\min_{A_{i,i'}} A_{i,i'} d^{(i,i')}. \tag{34}$$

Unfortunately, the formulation (34) is not useful as it can be solved by the trivial choice $A_{i,i'}$. We therefore need to enforce the presence of some edges (with positive weight) by adding constraints to (34). For example, we might require

$$A_{i,i} = 0 \ , \ \sum_{i' \neq i} A_{i,i'} = d_{\max} \text{ for all } i \in \mathcal{V}, \ A_{i,i'} \in [0,1] \text{ for all } i, i' \in \mathcal{V}. \tag{35}$$

The constraints (35) require that each node $i$ is connected with other nodes using total edge weight $\sum_{i' \neq i} A_{i,i'} = d_{\max}$. We can intepret the parameter $d_{\max}$ as an effective node degree.

We combine the constraints (35) with (34) to obtain the following graph

learning principle,

$$\begin{aligned} \widehat{A}_{i,i'} &\in \underset{A_{i,i'}}{\operatorname{argmin}} A_{i,i'} d^{(i,i')} \\ A_{i,i'} &\in [0,1] \text{ for all } i, i' \in \mathcal{V}, \\ A_{i,i} &= 0 \text{ for all } i \in \mathcal{V}, \\ \sum_{i' \neq i} A_{i,i'} &= d_{\max} \text{ for all } i \in \mathcal{V}. \end{aligned} \tag{36}$$

## 7.4 Assignment

# 8 Lecture - "Trustworthy FL"

This lecture discusses some key requirements for trustworthy AI that have been put forward by the European Union. We will also see how these requirements might guide the design choices for GTVMin. Our focus will be on the four design criteria: robustness, privacy protection and explainablity. This lecture discusses the robustness and explainability of basic linear regression that we encountered in Lecture 2. We will see that regularization techniques allow to navigate robustness-explainability-accuracy trade-offs.

## 8.1 Learning Goals

After this lecture, you should

- know some key requirements for trustworthy AI
- be familiar with quantiative measures of robustness and explainability
- have some intuition about how robustness, privacy and transparancy guides design choices for local models, loss functions and empirical graph in GTVMin

We can use the upper bound (16) to study the effect of perturbing features and labels of data points.

# 9 Lecture - "Privacy-Protection in FL"

FL is inherently based on sharing information. Without any information sharing between the owners (or generators) of local datasets, FL is not possible.

## 9.1 Learning Goals

Afer this lecture, you should

- be aware about threats to privacy and the need protect it
- know some quantiative measures for privacy leakage
- understand the effect of GTVMin design choices on privacy
- be able to implement FL algorithms with privacy guarantees

## 9.2 Assignment

# 10 Lecture - "Data and Model Poisoning in FL"

This lecture discusses the robustness of FL systems against data poisoning which are a specific type of cyber attackes.

## 10.1 Learning Goals

After this lecture, you should

1. be aware of threats posed by data and model poisoning
2. understand how GTVMin design choices result in more or less safe FL algorithms

## 10.2 Data Poisoning

## 10.3 Model Poisoning

There is a trade-off between privacy protection and robustness against model poisoning attacks. From [35]: *Without secure aggregation, privacy is lost, but the aggregator may attempt to filter out "anomalous" contributions. Since the weights of a model created using*

## 10.4 Assignment

# Glossary

**$k$-means** The $k$-means algorithm is a hard clustering method which assigns each data points to precisely one out of $k$ different clusters. The method iteratively updates this assignment in order to minimize the average distance between data points in their nearest cluster mean (centre). 2

**activation function** Each artificial neuron within an ANN consists of an activation function that maps the inputs of the neuron to a single output value. In general, an activation function is a non-linear map of the weighted sum of neuron inputs (this weighted sum is the activation of the neuron). 13

**artificial intelligence** Artificial intelligence aims to develop systems that behave rational in the sense of maximizing a long-term reward. 22

**artificial neural network** An artificial neural network is a graphical (signal-flow) representation of a map from features of a data point at its input to a predicted label at its output. 1, 5, 6, 10, 11, 13, 18, 24

**baseline** A reference value or benchmark for the average loss incurred by a hypothesis when applied to the data points generated in a specific ML application. Such a reference value might be obtained from human performance (e.g., error rate of dermatologists diagnosing cancer from visual inspection of skin areas) or other ML methods ("competitors") 9, 10

**Bayes estimator** A hypothesis $h$ whose Bayes risk is minimal [29]. 2, 9

**Bayes risk** We use the term Bayes risk as a synonym for the risk or expected loss of a hypothesis. Some authors reserve the term Bayes risk for the risk of a hypothesis that achieves minimum risk, such a hypothesis being referred to as a Bayes estimator [29]. 1

**bias** Consider some unknown quantity $\bar{w}$, e.g., the true weight in a linear model $y = \bar{w}x + e$ relating feature and label of a data point. We might use an ML method (e.g., based on ERM) to compute an estimate $\hat{w}$ for the $\bar{w}$ based on a set of data points that are realizations of RVs. The (squared) bias incurred by the estimate $\hat{w}$ is typically defined as $B^2 := \big(\mathbb{E}\{\hat{w}\} - \bar{w}\big)^2$. We extend this definition to vector-valued quantities using the squared Euclidean norm $B^2 := \big\|\mathbb{E}\{\widehat{\mathbf{w}}\} - \overline{\mathbf{w}}\big\|_2^2$. 12

**classification** Classification is the task of determining a discrete-valued label $y$ of a data point based solely on its features $\mathbf{x}$. The label $y$ belongs to a finite set, such as $y \in \{-1, 1\}$, or $y \in \{1, \ldots, 19\}$ and represents a category to which the corresponding data point belongs to. 11

**clustering** Clustering methods decompose a given set of data points into few subsets, which are referred to as clusters. Each cluster consists of data points that are more similar to each other than to data points outside the cluster. Different clustering methods use different measures for the similarity between data points and different representation of clusters. The clustering method $k$-means uses the average feature vector ("cluster means") of a cluster as its representative. A popular soft-clustering method based on Gaussian mixture model (GMM) represents a cluster by a multivariate normal distribution. 1

**computational aspects** By computational aspects of a ML method, we mainly refer to the computational resources required for its implementation. For example, if a ML method uses iterative optimization technniques to solve ERM, then its computational aspects include (i) how many arithmetic operations are needed to implement a single iteration (gradient step) and (ii) how many iterations are needed to obtain useful model parameters. One important example for an iterative optimization technique is GD. 1, 4, 6, 13, 15

**convex** A set $\mathcal{C} \subseteq \mathbb{R}^d$ is convex if it contains the line segment between any two points of that set. We define a function as convex if its epigraph is a convex set [28]. 1, 3, 7, 8, 14, 20

**covariance matrix** The covariance matrix of a RV $\mathbf{x} \in \mathbb{R}^d$ is defined as $\mathbb{E}\Big\{\big(\mathbf{x} - \mathbb{E}\{\mathbf{x}\}\big)\big(\mathbf{x} - \mathbb{E}\{\mathbf{x}\}\big)^T\Big\}$. 9, 13, 17

**data** A (indexed) set of data points. 1, 16, 18, 19, 21

**data augmentation** Data augmentation methods add synthetic data points to an existing set of data points. These synthetic data points might be obtained by perturbations (adding noise) or transformations (rotations of images) of the original data points. 12, 13

**data point** A data point is any object that conveys information [36]. Data points might be students, radio signals, trees, forests, images, RVs, real numbers or proteins. We characterize data points using two types of properties. One type of property is referred to as a feature. Features are properties of a data point that can be measured or computed in an

automated fashion. Another type of property is referred to as labels. The label of a data point represents some higher-level fact (or quantity of interest). In contrast to features, determining the label of a data point typically requires human experts (domain experts). Roughly speaking, ML aims at predicting the label of a data point based solely on its features. 1–16, 18, 19, 21–23, 25, 26

**data poisoning** FL methods allow to leverage the information contained in local datasets generated by other parties to improve the training of a tailored model. Depending on how much we trust the other parties, FL can be compromised by data poisoning. Data poisoning refers to the intentional manipulation (or fabrication) of local datasets to steer the training of a specific local model [37, 38]. 16

**dataset** With a slight abuse of notation we use the terms "dataset" or "set of data points" to refer to an indexed list of data points $\mathbf{z}^{(1)}, \mathbf{z}^{(2)}, \ldots$. Thus, there is a first data point $\mathbf{z}^{(1)}$, a second data point $\mathbf{z}^{(2)}$ and so on. Strictly speaking a dataset is a list and not a set [39]. By using indexed lists of data points we avoid some of the challenges arising in concept of an abstract set. 2, 3, 6, 8, 10, 12, 14, 15, 19, 21

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

values. 11

**decision region** Consider a hypothesis map $h$ that delivers values from a finite set $\mathcal{Y}$. We refer to the set of features $\mathbf{x} \in \mathcal{X}$ that result in the same output $h(\mathbf{x}) = a$ as a decision region of the hypothesis $h$. 4, 13, 14

**decision tree** A decision tree is a flow-chart like representation of a hypothesis map $h$. More formally, a decision tree is a directed graph which reads in the feature vector $\mathbf{x}$ of a data point at its root node. The root node then forwards the data point to one of its children nodes based on some elementary test on the features $\mathbf{x}$. If the receiving children node is not a leaf node, i.e., it has itself children nodes, it represents another test. Based on the test result, the data point is further pushed to one of its neighbours. This testing and forwarding of the data point is repeated until the data point ends up in a leaf node (having no children nodes). The leaf nodes represent sets (decision regions) constituted by feature vectors $\mathbf{x}$ that are mapped to the same function value $h(\mathbf{x})$. 10, 11, 13, 19

**deep net** We refer to an ANN with a (relatively) large number of hidden layers as a deep ANN or "deep net". Deep nets are used to represent the hypothesis spaces of deep learning methods [40]. 11

**differentiable** A function $f : \mathbb{R}^d \to \mathbb{R}$ is differentiable if it has a gradient $\nabla f(\mathbf{x})$ everywhere (for every $\mathbf{x} \in \mathbb{R}^d$) [5]. 2, 7, 9, 10, 13, 23, 24

**discrepancy** Consider a FL application with networked data represented by an empirical graph. FL methods use a discrepancy measure to compare

hypothesis maps from local models at nodes $i, i'$ connected by an edge in the empirical graph. 2

**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 number of "independent" tunable parameters of the model. These parameters might be the coefficients used in a linear map or the weights and bias terms of an ANN. 10, 11

**eigenvalue** We refer to a number $\lambda \in \mathbb{R}$ as eigenvalue of a square matrix $\mathbf{A} \in \mathbb{R}^{d \times d}$ if there is a non-zero vector $\mathbf{x} \in \mathbb{R}^d \setminus \{\mathbf{0}\}$ such that $\mathbf{A}\mathbf{x} = \lambda \mathbf{x}$. 3, 4, 6, 7, 13

**eigenvector** An eigenvector of a matrix $\mathbf{A}$ is a non-zero vector $\mathbf{x} \in \mathbb{R}^d \setminus \{\mathbf{0}\}$ such that $\mathbf{A}\mathbf{x} = \lambda \mathbf{x}$ with some eigenvalue $\lambda$. 3, 4

**empirical graph** Empirical graphs represent collections of local datasets and corresponding local models [41]. An empirical graph is an undirected weighted empirical graph whose nodes carry local datasets and models. FL methods learn a local hypothesis $h^{(i)}$, for each node $i \in \mathcal{V}$, such that it incurs small loss on the local datasets. 1–3, 5–8, 13–15, 20–22

**empirical risk** The empirical risk of a given hypothesis on a given set of data points is the average loss of the hypothesis computed over all data points in that set. 2, 7, 12, 17, 24

**empirical risk minimization** Empirical risk minimization is the optimization problem of finding the hypothesis with minimum average loss (or

empirical risk) on a given set of data points (the training set). Many ML methods are special cases of empirical risk. 1–3, 5, 7, 9, 11, 12, 17, 21, 25, 26

**estimation error** Consider data points with feature vectors $\mathbf{x}$ and label $y$. In some applications we can model the relation between features and label of a data point as $y = \bar{h}(\mathbf{x}) + \varepsilon$. Here we used some true hypothesis $\bar{h}$ and a noise term $\varepsilon$ which might represent modelling or labelling errors. The estimation error incurred by a ML method that learns a hypothesis $\widehat{h}$, e.g., using ERM, is defined as $\widehat{h} - \bar{h}$. For a parametrized hypothesis space, consisting of hypothesis maps that are determined by a parameter vector $\mathbf{w}$, we define the estimation error in terms of parameter vectors as $\Delta\mathbf{w} = \widehat{\mathbf{w}} - \overline{\mathbf{w}}$. first 6, 8

**Euclidean space** The Euclidean space $\mathbb{R}^d$ of dimension $d$ refers to the space of all vectors $\mathbf{x} = \big(x_1, \ldots, x_d\big)$, with real-valued entries $x_1, \ldots, x_d \in \mathbb{R}$, whose geometry is defined by the inner product $\mathbf{x}^T\mathbf{x}' = \sum_{j=1}^{d} x_j x'_j$ between any two vectors $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^d$ [5]. 8, 14, 17, 21

**feature** A feature of a data point is one of its properties that can be measured or computed in an automated fashion. For example, if a data point is a bitmap image, then we could use the red-green-blue intensities of its pixels as features. Some widely used synonyms for the term feature are "covariate","explanatory variable", "independent variable", "input (variable)", "predictor (variable)" or "regressor" [42–44]. However, this book makes consequent use of the term features for low-level properties of data points that can be measured easily. 1–6, 8–16, 18, 21–23, 25

**feature map** A map that transforms the original features of a data point into new features. The so-obtained new features might be preferable over the original features for several reasons. For example, the shape of datasets might become simpler in the new feature space, allowing to use linear models in the new features. Another reason could be that the number of new features is much smaller which is preferable in terms of avoiding overfitting. The special case of feature maps that deliver two numeric features are particulary useful for data visualization. Indeed, we can then depict data points in a scatterplot by using these two features as the coordinates of a data point. 13

**feature matrix** Consider a dataset $\mathcal{D}$ of $m$ data points that are characterized by feature vectors $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)}$. It is convenient to collect these feature vectors into a feature matrix $\mathbf{X} := \big(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)}\big)$. 2, 7

**feature space** The feature space of a given ML application or method is constituted by all potential values that the feature vector of a data point can take on. Within this book the most frequently used choice for the feature space is the Euclidean space $\mathbb{R}^d$ with dimension $d$ being the number of individual features of a data point. 10, 11, 13

**federated learning (FL)** Federated learning is an umbrella term for ML methods that train models in a collaborative fashion using decentralized data and computation. 1, 4–8, 16–23

**Finnish Meteorological Institute** The Finnish Meteorological Institute is a government agency responsible for gathering and reporting weather data in Finland. 2, 6, 14

**Gaussian mixture model** Gaussian mixture models (GMM) are a family of probabilistic models for data points characterized by a numeric feature vector $\mathbf{x}$. A GMM interprets $\mathbf{x}$ as being drawn from one out of $k$ different multivariate normal distributions $p^{(c)} = \mathcal{N}\left(\boldsymbol{\mu}^{(c)}, \mathbf{C}^{(c)}\right)$, indexed by $c = 1, \ldots, k$. The probability that $\mathbf{x}$ is drawn from the $c$-th multivariate normal distribution is denoted $p_c$. Thus, a GMM is parametrized by the probability $p_c$, the mean vector $\boldsymbol{\mu}^{(c)}$ and covariance matrix $\boldsymbol{\Sigma}^{(c)}$ for each $c = 1, \ldots, k$. 2

**General Data Protection Regulation** The General Data Protection Regulation (GDPR) is a law that has been passed by the European Union (EU) and put into effect on May 25, 2018 `https://gdpr.eu/tag/gdpr/`. The GDPR imposes obligations onto organizations anywhere, so long as they target, collect or in any other way process data related to people (i.e., personal data) in the EU. 19

**generalized total variation** Generalized total variation measures the changes of vector-valued node attributes of a graph. 6, 10

**gradient** For a real-valued function $f : \mathbb{R}^d \to \mathbb{R} : \mathbf{w} \mapsto f(\mathbf{w})$, a vector $\mathbf{g}$ such that $\lim_{\mathbf{w} \to \mathbf{w}'} \frac{f(\mathbf{w}) - \left(f(\mathbf{w}') + \mathbf{g}^T(\mathbf{w} - \mathbf{w}')\right)}{\|\mathbf{w} - \mathbf{w}'\|} = 0$ is referred to as the gradient of $f$ at $\mathbf{w}'$. If such a vector exists it is denoted $\nabla f(\mathbf{w}')$ or $\nabla f(\mathbf{w})\big|_{\mathbf{w}'}$ [5]. 1, 2, 5, 10, 13, 17, 18, 23

**gradient descent (GD)** Gradient descent is an iterative method for finding the minimum of a differentiable function $f(\mathbf{w})$. 1, 3, 13, 14, 18, 20, 21

**gradient step** Given a differentiable real-valued function $f(\mathbf{w})$ and a vector

$\mathbf{w}'$, the gradient step updates $\mathbf{w}'$ by adding the scaled negative gradient $\nabla f(\mathbf{w}')$, $\mathbf{w}' \mapsto \mathbf{w}' - \alpha \nabla f(\mathbf{w}')$. 1–3, 5, 16, 18, 20

**gradient-based method** Gradient-based methods are iterative algorithms for finding the minimum (or maximum) of a differentiable objective function of the model parameters. These algorithms construct a sequence of approximations to an optimal choice for model parameters that results in a minimum objective function value. As their name indicates, gradient-based methods use the gradients of the objective function evaluated during previous iterations to construct new (hopefully) improved model parameters. 1–3, 7, 11, 18, 22, 24

**graph** A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a pair that consists of a node set $\mathcal{V}$ and an edge set $\mathcal{E}$. In general, a graph is specified by a map that assigns to each edge $e \in \mathcal{E}$ a pair of nodes [45]. One important family of graphs (simple undirected graphs) is obtained by identifying each edge $e \in \mathcal{E}$ with two different nodes $\{i, i'\}$. Weighted graphs also specify numeric weights $A_e$ for each edge $e \in \mathcal{E}$. 6, 20

**GTV minimization** GTV minimization is an instance of RERM using the GTV of local model parameters as a regularizer. 1, 6, 7, 20, 22

**hinge loss** Consider a data point that is 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 specific hypothesis $h$ is defined as

$$L\left((\mathbf{x}, y), h\right) := \max\{0, 1 - yh(\mathbf{x})\}. \tag{37}$$

A regularized variant of the hinge loss is used by the support vector machine (SVM) [46] to learn a linear classifier with maximum margin between the two classes (see Figure 6). 11, 25

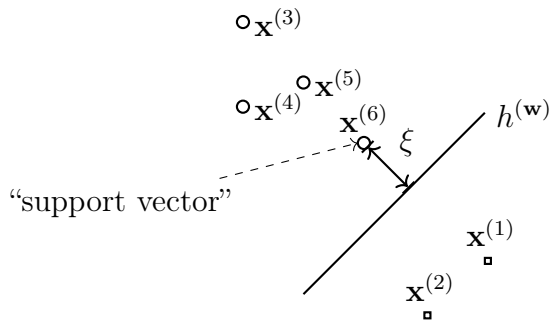


Figure 6: The SVM learns a hypothesis (or classifier) $h^{(\mathbf{w})}$ with minimum average soft-margin hinge loss. Minimizing this loss is equivalent to maximizing the margin $\xi$ between the decision boundary of $h^{(\mathbf{w})}$ and each class of the training set.

**hypothesis** A map (or function) $h : \mathcal{X} \to \mathcal{Y}$ from the feature space $\mathcal{X}$ to the label space $\mathcal{Y}$. Given a data point with features $\mathbf{x}$ we use a hypothesis map $h$ to estimate (or approximate) the label $y$ using the predicted label $\hat{y} = h(\mathbf{x})$. ML is about learning (or finding) a hypothesis map $h$ such that $y \approx h(\mathbf{x})$ for any data point. 1–18, 21–26

**hypothesis space** Every practical ML method uses a specific hypothesis space (or model) $\mathcal{H}$. The hypothesis space of a ML method is a subset of all possible maps from the feature space to label space. The design choice of the hypothesis space should take into account available computational resources and statistical aspects. If the computational

infrastructure allows for efficient matrix operations, and there is a (approximately) linear relation between features and label, a useful choice for the hypothesis space might be the linear model. 1, 5, 6, 10–12, 14–18, 22, 25, 26

**i.i.d.** It can be useful to interpret data points $\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(m)}$ as realizations of independent and identically distributed RVs with a common probability distribution. If these RVs are continuous, their joint probability density function (pdf) is $p\big(\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(m)}\big) = \prod_{r=1}^{m} p\big(\mathbf{z}^{(r)}\big)$ with $p(\mathbf{z})$ being the common marginal pdf of the underlying RVs. 6, 11–13, 17, 23

**i.i.d. assumption** The i.i.d. assumption interprets data points of a dataset as the realizations of i.i.d. RVs. 6, 12, 23

**interpretability** A ML method is interpretable for a specific user if she can well anticiapte the predictions delivered by the method. The notion of interpretability can be made precise using quantitative measures of the uncertainty about the predictions [47]. 15

**label** A higher level fact or quantity of interest associated with a data point. If a data point is an image, its label might be the fact that it shows a cat (or not). Some widely used synonyms for the term label are "response variable", "output variable" or "target" [42–44]. 1–7, 9–16, 18, 21, 23, 25

**label space** Consider a ML application that involves data points characterized by features and labels. The label space is constituted by all

potential values that the label of a data point can take on. Regression methods, aiming at predicting numeric labels, often use the label space $\mathcal{Y} = \mathbb{R}$. Binary classification methods use a label space that consists of two different elements, e.g., $\mathcal{Y} = \{-1, 1\}$, $\mathcal{Y} = \{0, 1\}$ or $\mathcal{Y} = \{$"cat image","no cat image"$\}$ 11

**Laplacian matrix** The geometry or structure of a graph $\mathcal{G}$ can be analyzed using the properties of special matrices that are associated with $\mathcal{G}$. One such matrix is the graph Laplacian matrix $\mathbf{L}$ which is defined for an undirected and weighted graph (e.g., the empirical graph of networked data) [48, 49]. 1, 3, 4

**law of large numbers** The law of large numbers refers to the convergence of the average of an increasing (large) number of i.i.d. RVs to the mean (or expectation) of their common probability distribution. Different instances of the law of large numbers are obtained using different notions of convergence. 11

**learning rate** Consider an iterative method for finding or learning a good choice for a hypothesis. Such an iterative method repeats similar computational (update) steps that adjust or modify the current choice for the hypothesis to obtain an improved hypothesis. A prime example for such an iterative learning method is GD and its variants. We refer by learning rate to any parameter of an iterative learning method that controls the extent by which the current hypothesis might be modified or improved in each iteration. A prime example for such a parameter is the step size used in GD. Some authors use the term learning rate

mostly as a synonym for the step size of (a variant of) GD 1–3, 11, 24

**learning task** A learning tasks consists of a specific choice for a collection of data points (e.g., all images stored in a particular database), their features and labels. 12, 17

**least absolute shrinkage and selection operator (Lasso)** The least absolute shrinkage and selection operator (Lasso) is an instance of structural risk minimization (SRM) for learning the weights $\mathbf{w}$ of a linear map $h(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$. The Lasso minimizes the sum consisting of an average squared error loss (as in linear regression) and the scaled $\ell_1$ norm of the weight vector $\mathbf{w}$. 22

**linear classifier** A classifier $h(\mathbf{x})$ maps the feature vector $\mathbf{x} \in \mathbb{R}^d$ of a data point to a predicted label $\hat{y} \in \mathcal{Y}$ out of a finite set of label values $\mathcal{Y}$. We can characterize such a classifier equivalently by the decision regions $\mathcal{R}_a$, for every possible label value $a \in \mathcal{Y}$. Linear classifiers are such that the boundaries between the regions $\mathcal{R}_a$ are hyperplanes in the Euclidean space $\mathbb{R}^d$. 11

**linear model** We use the term linear model in a very specific sense. In particular, a linear model is a hypothesis space which consists of all linear maps,

$$\mathcal{H}^{(d)} := \left\{ h(\mathbf{x}) = \mathbf{w}^T\mathbf{x} : \mathbf{w} \in \mathbb{R}^d \right\}. \tag{38}$$

Note that (38) defines an entire family of hypothesis spaces, which is parametrized by the number $d$ of features that are linearly combined to form the prediction $h(\mathbf{x})$. The design choice of $d$ is guided by

computational aspects (smaller $d$ means less computation), statistical aspects (increasing $d$ might reduce prediction error) and interpretability (a linear model using few carefully chosen features might be considered interpretable). 1–3, 6, 7, 11, 12

**linear regression** Linear regression aims at learning a linear hypothesis map to predict a numeric label based on numeric features of a data point. The quality of a linear hypothesis map is measured using the average squared error loss incurred on a set of labeled data points (which we refer to as training set). 1–4, 6–8, 13, 14, 16

**local dataset** The concept of a local dataset is in-between the concept of a data point and a dataset. A local dataset consists of several individual data points which are characterized by features and labels. In contrast to a single dataset used in basic ML methods, a local dataset is also related to other local datasets via different notions of similarities. These similarities might arise from probabilistic models or communication infrastructure and are encoded in the edges of an empirical graph. 1, 2, 4–6, 8, 14–22

**local model** Consider a collections of local datasets that are assigned to the nodes of an empirical graph. A local model $\mathcal{H}^{(i)}$ is a hypothesis space that is assigned to a node $i \in \mathcal{V}$. Different nodes might be assigned different hypothesis spaces, i.e., in general $\mathcal{H}^{(i)} \neq \mathcal{H}^{(i')}$ for different nodes $i, i' \in \mathcal{V}$. 1–6, 8, 14, 17, 19

**loss** With a slight abuse of language, we use the term loss either for the loss function itself or for its value for a specific pair of a data point and a

hypothesis. 2, 6, 8, 9, 11–14, 16, 17, 21–26

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

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

which assigns a pair consisting of a data point, with features $\mathbf{x}$ and label $y$, and a hypothesis $h \in \mathcal{H}$ the non-negative real number $L\left((\mathbf{x}, y), h\right)$. The loss value $L\left((\mathbf{x}, y), h\right)$ quantifies the discrepancy between the true label $y$ and the predicted label $h(\mathbf{x})$. Smaller (closer to zero) values $L\left((\mathbf{x}, y), h\right)$ mean a smaller discrepancy between predicted label and true label of a data point. Figure 7 depicts a loss function for a given data point, with features $\mathbf{x}$ and label $y$, as a function of the hypothesis $h \in \mathcal{H}$. 1–3, 5, 7, 8, 15, 16, 21–23

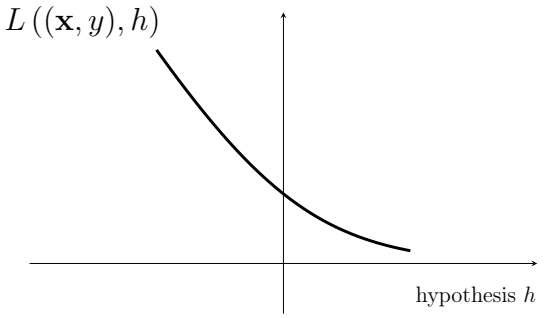


Figure 7: Some loss function $L\left((\mathbf{x}, y), h\right)$ for a fixed data point, with feature vector $\mathbf{x}$ and label $y$, and varying hypothesis $h$. ML methods try to find (learn) a hypothesis that incurs minimum loss.

**model** We use the term model as a synonym for hypothesis space 1, 3, 4, 6, 8, 10–12, 18, 21, 26

**model parameters** Model parameters are numbers that select a hypothesis map out of a hypothesis space. 1–3, 8, 10, 12, 13, 20, 21

**multivariate normal distribution** The multivariate normal distribution $\mathcal{N}(\mathbf{m}, \mathbf{C})$ is an important family ofprobability distributions for a continuous RV $\mathbf{x} \in \mathbb{R}^d$ [3, 50, 51]. This family is paramtrized by the mean $\mathbf{m}$ and covariance matrix $\mathbf{C}$ of $\mathbf{x}$. If the covariance matrix is invertible, the probability distribution of $\mathbf{x}$ is

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

2, 9

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

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

It is a measure for how well we can estimate $y$ based solely from $\mathbf{x}$. A large value of $I(\mathbf{x}; y)$ means that $y$ can be well predicted solely from $\mathbf{x}$. The prediction could be obtained by a hypothesis learnt by a ML method. 19

**overfitting** Consider a ML method that uses ERM to learn a hypothesis with minimum empirical risk on a given training set. Such a method is "overfitting" the training set if it learns hypothesis with small empirical risk on the training set but unacceptably large loss outside the training set. 8, 10

**parameters** The parameters of a ML model are tunable (learnable or adjustable) quantities that allow to choose between different hypothesis maps. For example, the linear model $\mathcal{H} := \{h : h(x) = w_1 x + w_2\}$ consists of all hypothesis maps $h(x) = w_1 x + w_2$ with a particular choice for the parameters $w_1, w_2$. Another example of parameters are the weights assigned to the connections of an ANN. 1, 5, 6

**polynomial regression** Polynomial regression aims at learning a polynomial hypothesis map to predict a numeric label based on numeric features of a data point. For data points characterized by a single numeric features, polynomial regression uses the hypothesis space $\mathcal{H}_d^{(\text{poly})} := \{h(x) = \sum_{j=0}^{d-1} x^j w_j\}$. The quality of a polynomial hypothesis map is measured using the average squared error loss incurred on a set of labeled data points (which we refer to as training set). 11

**positive semi-definite** A symmetric matrix $\mathbf{Q} = \mathbf{Q}^T \in \mathbb{R}^{d \times d}$ is referred to as positive semi-definite if $\mathbf{x}^T \mathbf{Q} \mathbf{x} \geq 0$ for every vector $\mathbf{x} \in \mathbb{R}^d$. 3, 7, 8, 13

**prediction** A prediction is an estimate or approximation for some quantity of interest. ML revolves around learning or finding a hypothesis map $h$ that reads in the features $\mathbf{x}$ of a data point and delivers a prediction $\widehat{y} := h(\mathbf{x})$ for its label $y$. 5, 6, 12, 14, 15, 17, 22

**privacy leakage** Consider a (ML or FL) system that processes a local dataset $\mathcal{D}^{(i)}$ and shares data, such as the predictions obtained for new data points, with other parties. Privacy leakage arises if the shared data carries information about a private (sensitive) feature of a data point

(which might be a human) of $\mathcal{D}^{(i)}$. The amount of privacy leakage can be measured via mutual information using a probabilistic model for the local dataset. 19

**privacy protection** Privacy protection aims at avoiding (or minimizing) the privacy leakage occuring within data processing systems (such as ML or FL methods). 1

**probabilistic model** A probabilistic model interprets data points as realizations of RVs with a joint probability distribution.This joint probability distribution typically involves parameters which have to be manually chosen (=design choice) or learnt via statistical inference methods [29]. 1, 7, 9, 15, 19, 23

**probability density function (pdf)** The probability density function (pdf) $p(x)$ of a real-valued RV $x \in \mathbb{R}$ is a particular representation of its probability distribution. If the pdf exists, it can be used to compute the probability that $x$ takes on a value from a (measurable) set $\mathcal{B} \subseteq \mathbb{R}$ via $p(x \in \mathcal{B}) = \int_{\mathcal{B}} p(x')dx'$ [3, Ch. 3]. The pdf of a vector-valued RV $\mathbf{x} \in \mathbb{R}^d$ (if it exists) allows to compute the probability that $\mathbf{x}$ falls into a (measurable) region $\mathcal{R}$ via $p(\mathbf{x} \in \mathcal{R}) = \int_{\mathcal{R}} p(\mathbf{x}')dx'_1 \ldots dx'_d$ [3, Ch. 3]. 12

**probability distribution** The data generated in some ML applications can be reasonably well modelled as realizations of a RV. The overall statistical properties (or intrinsic structure) of such data are then governed by the probability distribution of this RV. We use the term probability distribution in a highly informal manner and mean the collection of probabilities assigned to different values or value ranges

of a RV. The probability distribution of a binary RV $y \in \{0,1\}$ is fully specified by the probabilities $p(y=0)$ and $p(y=1)\big(=1\text{–}p(y=0)\big)$. The probability distribution of a real-valued RV $x \in \mathbb{R}$ might be specified by a probability density function $p(x)$ such that $p(x \in [a,b]) \approx p(a)|b-a|$. In the most general case, a probability distribution is defined by a probability measure [50, 52]. 1, 2, 6, 9, 11–13, 17, 19, 23

**projected GD** Projected GD extends basic GD for unconstrained optimization to handle constraints on the optimization variable (model parameters). A single iteration of projected GD consists of first taking a gradient step and then projecting the result back into a constrain set. 1

**proximable** A Convex function for which the proximity operator can be computed efficiently are sometimes referred to as "proximable" or "simple" [32]. 7

**proximity operator** Given a convex function and a vector $\mathbf{x}$, we define its proximity operator as

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

Convex functions for which the proximity operator can be computed efficiently are sometimes referred to as "proximable" or "simple" [32]. 7, 20

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

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

with some matrix $\mathbf{Q} \in \mathbb{R}^{d \times d}$, vector $\mathbf{q} \in \mathbb{R}^d$ and scalar $a \in \mathbb{R}$. 7, 8, 14

**random variable (RV)** A random variable is a mapping from a probability space $\mathcal{P}$ to a value space [52]. The probability space, whose elements are elementary events, is equipped with a probability measure that assigns a probability to subsets of $\mathcal{P}$. A binary random variable maps elementary events to a set containing two different values, e.g., $\{-1, 1\}$ or $\{\text{cat}, \text{no cat}\}$. A real-valued random variable maps elementary events to real numbers $\mathbb{R}$. A vector-valued random variable maps elementary events to the Euclidean space $\mathbb{R}^d$. Probability theory uses the concept of measurable spaces to rigorously define and study the properties of (large) collections of random variables [50, 52]. 1–3, 6, 9, 11–13, 17, 19–23, 26

**realization** Consider a RV $x$ which maps each element (outcome, or elementary event) $\omega \in \mathcal{P}$ of a probability space $\mathcal{P}$ to an element $a$ of a measurable space $\mathcal{N}$ [5, 52, 53]. A realization of $x$ is any element $a' \in \mathcal{N}$ such that there is an element $\omega' \in \mathcal{P}$ with $x(\omega') = a'$. 1, 11–13, 19, 23

**regression** Regresion problems revolve around the problem of predicting a numeric label solely from the features of a data point. 11

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

compared to the average loss on the training set. 1, 8, 10, 12, 13, 18, 21, 22, 24

**regularized empirical risk minimization** Synoym for SRM. 6, 10, 22

**regularizer** A regularizer assigns each hypothesis $h$ from a hypothesis space $\mathcal{H}$ a quantitative measure $\mathcal{R}\big\{h\big\}$ for how much its prediction error on a training set might differ from its prediction errors on data points outside the training set. Ridge regression uses the regularizer $\mathcal{R}\big\{h\big\} := \|\mathbf{w}\|_2^2$ for linear hypothesis maps $h^{(\mathbf{w})}(\mathbf{x}) := \mathbf{w}^T\mathbf{x}$ [4, Ch. 3]. The least absolute shrinkage and selection operator (Lasso) uses the regularizer $\mathcal{R}\big\{h\big\} := \|\mathbf{w}\|_1$ for linear hypothesis maps $h^{(\mathbf{w})}(\mathbf{x}) := \mathbf{w}^T\mathbf{x}$ [4, Ch. 3]. 6, 10, 12, 22

**ridge regression** Ridge regression learns the parameter (or weight) vector $\mathbf{w}$ of a linear hypothesis map $h^{(\mathbf{w})}(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$. The quality of a particular choice for the parameter vector $\mathbf{w}$ is measured by the sum of two components. The first components is the average squared error loss incurred by $h^{(\mathbf{w})}$ on a set of labeled data points (the training set). The second component is the scaled squared Euclidean norm $\lambda\|\mathbf{w}\|_2^2$ with a regularization parameter $\lambda > 0$. It can be shown that the effect of adding to $\lambda\|\mathbf{w}\|_2^2$ to the average squared error loss is equivalent to replacing the original data points by an ensemble of realizations of a RV centered around these data points. 12–14, 22

**risk** Consider a hypothesis $h$ that is used to predict the label $y$ of a data point based on its features $\mathbf{x}$. We measure the quality of a particular prediction using a loss function $L\left((\mathbf{x}, y), h\right)$. If we interpret data points

as the realizations of i.i.d. RVs, also the $L\left((\mathbf{x},y),h\right)$ becomes the realization of a RV. Using such an i.i.d. assumption allows to define the risk of a hypothesis as the expected loss $\mathbb{E}\big\{L\left((\mathbf{x},y),h\right)\big\}$. Note that the risk of $h$ depends on both, the specific choice for the loss function and the probability distribution of the data points. 2, 6, 9, 21

**scatterplot** A visualization technique that depicts data points by markers in a two-dimensional plane. 8

**smooth** We refer to a real-valued function as smooth if it is differentiable and its gradient is continuous [54, 55]. In particular, a differentiable function $f(\mathbf{w})$ is refered to as $\beta$-smooth if the gradient $\nabla f(\mathbf{w})$ is Lipschitz continuous with Lipschitz constant $\beta$, i.e.,

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

1, 3

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

$$L\left((\mathbf{x},y),h\right) := \big(y - \underbrace{h(\mathbf{x})}_{=\hat{y}}\big)^2. \tag{39}$$

2, 3, 6, 15, 18, 22

**statistical aspects** By statistical aspects of a ML method, we refer to (properties of) the probability distribution of its ouput given a probabilistic model for the data fed into the method. 1, 4, 6, 15

**step size** Many ML methods use iterative optimization methods (such as gradient-based methods) to construct a sequence of increasingly accurate hypothesis maps $h^{(1)}, h^{(2)}, \ldots$. The $r$th iteration of such an algorithm starts from the current hypothesis $h^{(r)}$ and tries to modify it to obtain an improved hypothesis $h^{(r+1)}$. Iterative algorithms often use a step size (hyper-) parameter. The step size controls the amount by which a single iteration can change or modify the current hypothesis. Since the overall goal of such iteration ML methods is to learn a (approximately) optimal hypothesis we refer to a step size parameter also as a learning rate. 1

**stopping criterion** Many ML methods use iterative algorithms that construct a sequence of model parameters (such as the weights of a linear map or the weights of an ANN) that (hopefully) converge to an optimal choice for the model parameters. In practice, given finite computational resources, we need to stop iterating after a finite number of times. A stopping criterion is any well-defined condition required for stopping iterating. 1–3

**strongly convex** A continuously differentiable real-valued function $f(\mathbf{x})$ is strongly convex with coefficient $\sigma$ if $f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla f(\mathbf{x})^T(\mathbf{y} - \mathbf{x}) + (\sigma/2)\left\|\mathbf{y} - \mathbf{x}\right\|_2^2$ [54], [56, Sec. B.1.1.]. 1

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

perturbations of the data points in the training set. 14, 22

**support vector machine** A binary classification method for learning a linear hypothesis map that maximally separates data points from the two different classes in the feature space ("maximum margin"). Maximizing this separation is equivalent to minimizing a regularized variant of the hinge loss (37). 11

**training error** The average loss of a hypothesis when predicting the labels of data points in a training set. We sometimes refer by training error also the minimum average loss incurred on the training set by any hypothesis out of a hypothesis space. 9–12, 25, 26

**training set** A set of data points that is used in ERM to learn a hypothesis $\hat{h}$. The average loss of $\hat{h}$ on the training set is referred to as the training error. The comparison between training error and validation error of $\hat{h}$ allows to diagnose ML methods and informs how to improve them (e.g., using a different hypothesis space or collecting more data points). 3, 5–8, 10–15, 17–19, 21, 22, 24–26

**validation** Consider a hypothesis $\hat{h}$ that has been learn via ERM on some training set $\mathcal{D}$. Validation refers to the practice of trying out a hypothesis $\hat{h}$ on a validation set that consists of data points that are not contained in the training set $\mathcal{D}$. 1, 8

**validation error** Consider a hypothesis $\widehat{h}$ which is obtained by ERM on a training set. The average loss of $\widehat{h}$ on a validation set, which is different from the training set, is referred to as the validation error. 9–12, 25, 26

**validation set** A set of data points that have not been used as training set in ERM to learn a hypothesis $\widehat{h}$. The average loss of $\widehat{h}$ on the validation set is referred to as the validation error and used to diagnose the ML method (see [4, Sec. 6.6.]). The comparison between training error and validation error can inform directions for improvements of the ML method (such as using a different hypothesis space). 8, 11, 12, 25

**variance** The variance of a real-valued RV $x$ is defined as the expectation $\mathbb{E}\big\{\big(x - \mathbb{E}\{x\}\big)^2\big\}$ of the squared difference $x$ and its expectation $\mathbb{E}\{x\}$. We extend this definition to vector-valued RVs $\mathbf{x}$ as $\mathbb{E}\big\{\big\|\mathbf{x} - \mathbb{E}\{\mathbf{x}\}\big\|_2^2\big\}$. 12

**weights** We use the term weights synonymously for a finite set of parameters within a model. For example, the linear model consists of all linear maps $h(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$ that read in a feature vector $\mathbf{x} = \big(x_1, \ldots, x_d\big)^T$ of a data point. Each specific linear map is characterized by specific choices for the parameters for weights $\mathbf{w} = \big(w_1, \ldots, w_d\big)^T$. 13

# References


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

[2] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed. Baltimore, MD: Johns Hopkins University Press, 1996.

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

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

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

[6] M. Wollschlaeger, T. Sauter, and J. Jasperneite, "The future of industrial communication: Automation networks in the era of the internet of things and industry 4.0," *IEEE Industrial Electronics Magazine*, vol. 11, no. 1, pp. 17–27, 2017.

[7] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, Jan. 2017. [Online]. Available: https://doi.org/10.1109/MC.2017.9

[8] H. Ates, A. Yetisen, F. Güder, and C. Dincer, "Wearable devices for the detection of covid-19," *Nature Electronics*, vol. 4, no. 1, pp. 13–14, 2021. [Online]. Available: https://doi.org/10.1038/s41928-020-00533-1

[9] H. Boyes, B. Hallaq, J. Cunningham, and T. Watson, "The industrial internet of things (iiot): An analysis framework," *Computers in Industry*, vol. 101, pp. 1–12, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0166361517307285

[10] S. Cui, A. Hero, Z.-Q. Luo, and J. Moura, Eds., *Big Data over Networks*. Cambridge Univ. Press, 2016.

[11] A. Barabási, N. Gulbahce, and J. Loscalzo, "Network medicine: a network-based approach to human disease," *Nature Reviews Genetics*, vol. 12, no. 56, 2011.

[12] M. E. J. Newman, *Networks: An Introduction*. Oxford Univ. Press, 2010.

[13] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, A. Singh and J. Zhu, Eds., vol. 54. Fort Lauderdale, FL, USA: PMLR, 20–22 Apr 2017, pp. 1273–1282. [Online]. Available: http://proceedings.mlr.press/v54/mcmahan17a.html

[14] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, May 2020.

[15] Y. Cheng, Y. Liu, T. Chen, and Q. Yang, "Federated learning for privacy-preserving ai," *Communications of the ACM*, vol. 63, no. 12, pp. 33–36, Dec. 2020.

[16] N. Agarwal, A. Suresh, F. Yu, S. Kumar, and H. McMahan, "cpSGD: Communication-efficient and differentially-private distributed sgd," in *Proc. Neural Inf. Proc. Syst. (NIPS)*, 2018.

[17] V. Smith, C.-K. Chiang, M. Sanjabi, and A. Talwalkar, "Federated Multi-Task Learning," in *Advances in Neural Information Processing Systems*, vol. 30, 2017. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/6211080fa89981f66b1a0c9d55c61d0f-Paper.pdf

[18] J. You, J. Wu, X. Jin, and M. Chowdhury, "Ship compute or ship data? why not both?" in *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)*. USENIX Association, April 2021, pp. 633–651. [Online]. Available: https://www.usenix.org/conference/nsdi21/presentation/you

[19] D. Tse and P. Viswanath, *Fundamentals of Wireless Communication*. Cambridge University Press, 2005.

[20] T. Yang, G. Andrew, H. Eichner, H. Sun, W. Li, N. Kong, D. Ramage, and F. Beaufays, "Applied federated learning: Improving google keyboard query suggestions," 2018. [Online]. Available: https://arxiv.org/abs/1812.02903

[21] A. Ghosh, J. Chung, D. Yin, and K. Ramchandran, "An efficient framework for clustered federated learning," in *34th Conference on Neural*

*Information Processing Systems (NeurIPS 2020)*, Vancouver, Canada, 2020.

[22] F. Sattler, K. Müller, and W. Samek, "Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[23] G. Strang, *Computational Science and Engineering*. Wellesley-Cambridge Press, MA, 2007.

[24] ——, *Introduction to Linear Algebra*, 5th ed. Wellesley-Cambridge Press, MA, 2016.

[25] H. H. Bauschke and P. L. Combettes, *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. New York: Springer, 2011.

[26] F. Pedregosa, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, no. 85, pp. 2825–2830, 2011. [Online]. Available: http://jmlr.org/papers/v12/pedregosa11a.html

[27] J. Hirvonen and J. Suomela. (2023) Distributed algorithms 2020.

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

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

[30] A. Esteva, B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau, and S. Thrun, "Dermatologist-level classification of skin cancer with deep neural networks," *Nature*, vol. 542, 2017.

[31] H. Lütkepohl, *New Introduction to Multiple Time Series Analysis*. New York: Springer, 2005.

[32] L. Condat, "A primal–dual splitting method for convex optimization involving lipschitzian, proximable and linear composite terms," *Journal of Opt. Th. and App.*, vol. 158, no. 2, pp. 460–479, Aug. 2013.

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

[34] S. Chepuri, S. Liu, G. Leus, and A. Hero, "Learning sparse graphs under smoothness prior," in *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 2017, pp. 6508–6512.

[35] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, S. Chiappa and R. Calandra, Eds., vol. 108. PMLR, 26–28 Aug 2020, pp. 2938–2948. [Online]. Available: https://proceedings.mlr.press/v108/bagdasaryan20a.html

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

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

[38] J. Zhang, B. Chen, X. Cheng, H. T. T. Binh, and S. Yu, "Poisongan: Generative poisoning attacks against federated learning in edge computing systems," *IEEE Internet of Things Journal*, vol. 8, no. 5, pp. 3310–3322, 2021.

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

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

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

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

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

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

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

[46] C. Lampert, "Kernel methods in computer vision," *Foundations and Trends in Computer Graphics and Vision*, 2009.

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

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

[49] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Adv. Neur. Inf. Proc. Syst.*, 2001.

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

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

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

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

[54] Y. Nesterov, *Introductory lectures on convex optimization*, ser. Applied Optimization. Kluwer Academic Publishers, Boston, MA, 2004, vol. 87, a basic course. [Online]. Available: http://dx.doi.org/10.1007/978-1-4419-8853-9

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

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