## Contents



## 2 Preliminaries

Preliminary definitions and results for the deep learning part are primarily drawn from the first chapter of the book by Grohs and Kutyniok (2022). Regarding the topological data analysis part, most of the content is sourced from the book by Edelsbrunner and Harer (2022) and the survey conducted by Hensel et al. (2021).

### 2.1 Notation

We denote by $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{R}$, and $\bar{\mathbb{R}}$ respectively the set of natural numbers (including zero), the ring of integers, the field of real numbers, and the extended real line with a point at infinity.

For $n \geq 2$, we abbreviate $\{1, \ldots, n\}$ as $[n]$. Given a set $S$, its power set, i.e., the set of its subsets, is denoted by $\mathcal{P}(S)$. The indicator function on a set $A$ is written as $\mathbb{1}_A$ and it is defined as $\mathbb{1}_A(x) = 1$ if $x \in A$ and $\mathbb{1}_A(x) = 0$ if $x \notin A$. The Kronecker delta $\delta_{i,j}$ is defined as $\delta_{i,j} = 1$ if $i = j$ and $\delta_{i,j} = 0$ if $i \neq j$.

The set of matrices of size $m \times n$ with coefficients in a ring $S$ (usually a field) is denoted by $M_{m,n}(S)$. The $(i, j)$ entry of a matrix $M$ is written as $M_{i,j}$. For a square matrix $M$, its trace is denoted by $\mathrm{tr}(M)$. Given a vector $v \in \mathbb{R}^d$, we denote its $i$-th component by $v_i$.

We denote by $\mathcal{M}(\mathcal{X}, \mathcal{Y})$ the set of all measurable functions from $\mathcal{X}$ to $\mathcal{Y}$, where $\mathcal{X}$ and $\mathcal{Y}$ are measure spaces. Given a function $f$, we denote its domain and image as $\mathrm{Dom}(f)$ and $\mathrm{Im}(f)$, respectively, and we denote its graph by $G(f) = \{(x, f(x)) : x \in \mathrm{Dom}(f)\}$.

The *support* of a function $f\colon X \to \mathbb{R}$, denoted $\mathrm{supp}(f)$, is the set of elements of $X$ that are not sent to zero, or most commonly the closure of this set if $X$ is a topological space. Similarly, for a probability distribution $\mathbb{P}$ associated with a random variable $X$, the support $\mathrm{supp}(\mathbb{P})$ is the smallest closed set $S$ of real numbers such that $\mathbb{P}(X \in S) = 1$. We also call support of $\mathbb{P}$ the set of elements of the sample space mapped to $\mathrm{supp}(\mathbb{P})$ by $X$. The letter $\mathbb{E}$ is used to denote expectation of a random variable.

For a graph $G$, we denote by $V(G)$ and $E(G)$ the sets of vertices and edges of $G$, respectively. By a *weighted graph* we refer to an undirected graph $G$ whose vertices and edges are weighted by functions $w_V\colon V(G) \to \mathbb{R}$ and $w_E\colon E(G) \to \mathbb{R}$, and we use the notation $(G, w_V, w_E)$. A bipartite graph is denoted by $G = (V_1, V_2, E)$, where $V_1$ and $V_2$ are disjoint sets of vertices.

### 2.2 Deep learning

Deep learning is a subfield of machine learning that deals with (deep) neural networks to solve a variety of computational tasks. Some computational tasks include, but are not limited to, classification, regression, and synthetic data generation. These three tasks are central in this survey.

Classification and regression are particular examples of prediction tasks. In prediction tasks, we have data lying on the Cartesian product of two measure spaces $\mathcal{X}$ and $\mathcal{Y}$, where elements of $\mathcal{Y}$ are seen as *labels* of elements of $\mathcal{X}$. Such data are distributed according to a distribution $\mathbb{P}_{(X,Y)}$ with marginals $\mathbb{P}_X$ and $\mathbb{P}_Y$ for the restrictions to $\mathcal{X}$ and $\mathcal{Y}$, respectively. In prediction tasks, we look for a function $f \in \mathcal{M}(\mathcal{X}, \mathcal{Y})$ that, given a value $x \in \mathcal{X}$, outputs a "good" label prediction $y \in \mathcal{Y}$ according to some quality criterion. To precisely quantify the concept of quality, a loss function

$$\mathcal{L}\colon \mathcal{M}(\mathcal{X}, \mathcal{Y}) \times \mathcal{X} \times \mathcal{Y} \longrightarrow \mathbb{R}$$

is typically employed. A loss function indicates the degree of inaccuracy in predictions relative to a data point $(x, y) \in \mathcal{X} \times \mathcal{Y}$ sampled from the data distribution —where a lower value signifies greater quality. Thus, the objective is to find a function $f$ that minimizes the expected loss $\mathbb{E}[\mathcal{L}(f, X, Y)]$, called *risk* of $f$ and denoted by $\mathcal{R}(f)$. Due to the hardness of finding such $f$ (if it exists) in the huge space $\mathcal{M}(\mathcal{X}, \mathcal{Y})$, the problem is often reduced to find a function $f$ whose risk is arbitrarily near to the infimum of risks in a smaller set of functions called *hypothesis set* and denoted by $\mathcal{F}$.

One of the main difficulties in this approach is that the data distribution $\mathbb{P}_{(X,Y)}$ is often unknown, and we only have access to a finite sample $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^m$ from it, that we

assume independent and identically distributed (i.i.d.) with distribution $\mathbb{P}^m_{(X,Y)}$. Therefore, instead of trying to minimize directly the risk $\mathcal{R}(f)$, one tries to minimize the empirical risk $\widehat{\mathcal{R}}_{\mathcal{D}_{\text{train}}}(f)$ for a subset $\mathcal{D}_{\text{train}} \subseteq \mathcal{D}$, called *training dataset*, given by

$$\widehat{\mathcal{R}}_{\mathcal{D}_{\text{train}}}(f) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}\left(f, x_i, y_i\right), \tag{1}$$

with the hope that minimizing the empirical risk also reduces the real risk. In some situations, it is convenient to minimize the empirical risk of another related loss $\mathcal{L}^{\text{surr}}$ that is not the original loss function $\mathcal{L}$ and that can depend on a surrogate function $f^{\text{surr}}$ depending on $f$, instead of using $f$ directly. This happens, for example, when we use minimization algorithms that rely on the differentiability of the loss function, yet our function $\mathcal{L}$ is not differentiable or its gradient is zero.

In deep learning, one formally sees the process of obtaining a neural network $\mathcal{N}$ given data distributed according to $\mathbb{P}_{(X,Y)}$ as a map

$$\mathcal{A}\colon \bigcup_{m\in\mathbb{N}} (\mathcal{X} \times \mathcal{Y})^m \longrightarrow \mathcal{F} \tag{2}$$

that takes as input a training (ordered) set $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^m$ and provides a neural network $\mathcal{N} \in \mathcal{F}$ that minimizes the empirical risk for the original or surrogate loss on $\mathcal{D}_{\text{train}}$. This map is called *training algorithm*. Training algorithms are generally iterative minimization methods based on gradient descent.

Regression and classification are the two modalities studied for prediction tasks in this survey. In both problems, we assume that there is an unknown function $f\colon \mathcal{X} \to \mathcal{Y}$ assigning, to each $x \in \mathcal{X}$, a *true value* $y = f(x)$, and that the data distribution $\mathbb{P}_{(X,Y)}$ has the property that $G(f) \subseteq \operatorname{supp}\left(\mathbb{P}_{(X,Y)}\right)$. In this scenario, the goal of a neural network is, for each value $x \in \mathcal{X}$, to output the corresponding value $f(x)$.

We say that a prediction task is a *regression task* if $\mathcal{X}$ is a Euclidean space $\mathbb{R}^d$ for some $d \in \mathbb{N}_{>0}$ and $\mathcal{Y} = \mathbb{R}$, and we say that a prediction task is a *classification task* if $\mathcal{X}$ is a Euclidean space $\mathbb{R}^d$ and $\mathcal{Y}$ is a finite set. In the latter case we assume, without loss of generality, that $\mathcal{Y} = [k]$ with $k \in \mathbb{N}_{\geq 2}$ and that $\operatorname{supp}\left(\mathbb{P}_Y\right) = \mathcal{Y}$. If $k = 2$, then we call it a *binary* classification task. For a broader introduction to machine learning tasks, we refer the reader to Goodfellow et al. (2016, Section 5.1.1).

Generative tasks are not as straightforward to define as prediction tasks, and formal definitions can vary depending on the problem or the solving method. In generative tasks, the purpose is to generate synthetic data that are indistinguishable from source data. We assume that source data live in an ambient space $\mathcal{X}$ and follow a distribution $\mathbb{P}_X$. Some models explicitly approximate the given probability distribution, allowing to directly sample from it, while other models learn a mechanism to generate new examples without explicitly describing the data distribution. We refer to Prince (2023, Section 1.2.1) for a detailed explanation of generative models.

In this survey we mainly consider *fully connected feedforward neural networks* (FCFNN). FCFNNs follow a basic structure in which most of the current, more complex, models, can be expressed. A FCFNN is a parameterized function $\mathcal{N}$ consisting of: