Topics in Mathematics for Deep Learning

Lecture notes, Lent Term 2024

University of Cambridge

Carlos Esteve-Yagüe and Willem Diepeveen

March 12, 2024

Lecture notes from the Part III graduate course *Topics in Mathematics for Deep Learning*, given by Carlos Esteve-Yagüe and Willem Diepeveen at the University of Cambridge during the Lent Term of the academic year 2023-2024 . Complementary material can be found in the following review papers and books:

1. Julius Berner, Philipp Gtohs, Gitta Kutyniok, Philipp Petersen, *The Modern Mathematics of Deep Learning*, 2022. Book chapter in *Theory of Deep Learning*, Cambridge University Press. arXiv version
2. Ian Goodfellow, Yoshua Bengio, Aaron Courville, *Deep Learning*, 2016, MIT Press.
3. Francis Bach, *Learning theory from first principles*, Online version, 2021.
4. S. Mukherjee, A. Hauptmann, O. Öktem, M. Pereyra, and C.-B. Schönlieb. *Learned reconstruction methods with convergence guarantees: A survey of concepts and applicattions.* IEEE Signal Processing Magazine, 40(1):164–182, 2023. Link to the journal

Disclaimer: This document is under constant redevelopment and may contain typos and errors. If you find mistakes and feel like telling me, I will be grateful and happy to hear from you. You can reach me by email at ce423@cam.ac.uk.

Contents

Lecture 1: Introduction to Deep Learning **1**
1.1 Learning theory . . . 1
1.2 Bayes risk . . . 4
1.3 Types of error . . . 5
1.3.1 Approximation error: . . . 6
1.3.2 Generalisation error . . . 6
1.3.3 Optimisation error . . . 7
1.4 Upper bound for the error . . . 7
1.5 Deep Neural Networks . . . 9

Lecture 2: Kernel methods **12**
2.1 Linear regression . . . 12
2.2 Rising the dimension of the feature space . . . 13
2.3 Reformulation of risk minimization problem . . . 15

Lecture 3: Reproducing kernel Hilbert spaces **17**
3.1 Definition of RKHS . . . 17
3.2 Connection between RKHS and NNs . . . 19

Lecture 4: Neural Tangent Kernel **22**
4.1 Kernel gradient . . . 23
4.2 Neural Tangent Kernel . . . 24

Lecture 5: Kernel Regime **27**
5.1 Deep Neural Networks and Gaussian processes . . . 27
5.2 Convergence of the Neural Tangent Kernel . . . 31

Lecture 6: The Optimisation Landscape **32**
6.1 Optimisation landscape analysis: paths and level sets . . . 32

Lecture 7: Lazy Training and Linear Convergence **36**
7.1 Control of the gradient at random initializations . . . 37
7.2 Linear convergence of randomly initialized gradient descent . . . 39

Lecture 8: Universal approximation with two-layer NNs **41**
8.1 Universal approximation with sigmoidal activation . . . 42
8.2 Quantitative estimates depending on the regularity . . . 44

Lecture 9: Variation norm and Barron space **46**
9.1 Universal approximation property in one dimension . . . 47
9.2 Variation norm and the Barron space . . . 48
9.3 Construction of $\mathcal{F}_1$ through measures . . . 49
9.4 The Hilbert space $\mathcal{F}_2$ . . . 51
9.5 Comparison between $\mathcal{F}_1$ and $\mathcal{F}_2$ . . . 53

Lecture 10: Quantitative estimates in the Barron space **54**
10.1 Variation norm in one dimension 54
10.2 Approximation error in $\mathcal{F}_1$ 55

Lecture 11: Convolutional Neural Networks **58**
11.1 Group equivariant Convolutional Neural Networks 59
11.2 Scattering Transform 61

Lecture 12: Neural Networks and its dynamics **64**
12.1 Vanishing/exploding gradient 64
12.2 Residual Neural Networks 66
12.3 Infinite-depth limit and Neural ODEs 68
12.4 Stability and dissipativity 69
12.4.1 Dissipative models and gradient flows 70
12.4.2 Hamiltonian flows 71

Lecture 13: Generative models I: Diffusion models **72**
13.1 Diffusion models 72
13.2 Denoising diffusion 73
13.3 Score-based diffusion models 74
13.4 Two probabilistic models 75
13.5 Likelihood estimates for score-based diffusion models 76

Lecture 14: Generative models II: Variational Autoencoders **78**
14.1 Problem formulation 78
14.2 Training 79
14.3 Reparametrization 80

Lecture 15: Deep Learning meets inverse problems I **81**
15.1 Model-based vs data-driven reconstruction 82
15.2 Supervised learning 83
15.3 Unsupervised learning 83

January 22 2024 Mon (12:00-13:00)

Lecture 1: Introduction to Deep Learning

Machine Learning refers to the set of techniques that allow a machine to "learn" through data, and improve the performance on one or multiple tasks. Here, the word "learn" means to build a model that, given an input, is able to return the desired output. Deep Learning arises as a sub-field of Machine Learning, and is characterized by using artificial neural networks (NNs) as building blocks to perform the learning task (i.e. the learned model consist of a NN). The word "deep" refers to the fact that the NNs may contain multiple layers, and the input information is propagated through them to generate the output. The learning part (also known as training) is typically formulated as a minimization problem, which is addressed by gradient-based optimization methods.

Depending on the type of data available, we can classify any learning problem in three categories:

- **Supervised learning:** The dataset consists of input-output pairs (labelled data) of the form $(x, y) \in \mathcal{X} \times \mathcal{Y}$. For prediction tasks, the NN is required to predict, for each input $x \in \mathcal{X}$, the corresponding (or the most likely) label $y \in \mathcal{Y}$.
- **Unsupervised learning:** The dataset consists of only inputs $x \in \mathcal{X}$. In this case, the NN is sometimes required to learn tasks such as extracting unknown features from a dataset, clustering in an unknown number of classes or reduce the dimension of a dataset.
- **Reinforcement learning:** The data is generated by the interaction of the model with an environment. Given some information about the current state of the environment, the NN is required to learn the best output (action) based on the response of the environment to the past actions.

1.1 Learning theory

Before describing the mathematical formulation of deep learning, let us formulate the general problem that we aim to address.

The goal is to approximate (learn) a measurable function

$$f : \mathcal{X} \longrightarrow \mathcal{Y}$$

where $\mathcal{X}$ and $\mathcal{Y}$ are measurable spaces. We denote by $\mathcal{M}(\mathcal{X}, \mathcal{Y})$ the space of measurable functions from $\mathcal{X}$ to $\mathcal{Y}$. We assume that the data available to carry out the learning task lives in a measurable space $\mathcal{Z}$. We call training data to any collection of elements of $\mathcal{Z}$, e.g. $\{z^{(i)}\}_{i=1}^m \in \mathcal{Z}^m$.

Note:-

In a prediction task, such as regression or classification, the space of the data is often of the form $\mathcal{Z} := \mathcal{X} \times \mathcal{Y}$, and therefore, the training data consists of a collection of input-output pairs $\{(x^{(i)}, y^{(i)})\}_{i=1}^m \in [\mathcal{X} \times \mathcal{Y}]^m$.

In order to address the above learning problem, we need to provide the three following ingredients: a hypothesis set, a loss function and a learning algorithm.

Definition 1.1: Hypothesis set

The hypothesis set $\mathcal{F} \subset \mathcal{M}(\mathcal{X}, \mathcal{Y})$ is a family of measurable functions $\mathcal{X} \to \mathcal{Y}$ among which we will choose the one that performs better for the required task.

Example. For a regression task in which $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{Y} = \mathbb{R}$, examples of hypothesis sets are:

- Linear functions: $\mathcal{F} = \{f_{a,b}(x) = a \cdot x + b \ : \quad a \in \mathbb{R}^d,\ b \in \mathbb{R}\}$
- Fourier series: $\mathcal{F} = \{f_a(x) = \sum_{n=0}^{N} a_i \cos(2nx\pi) \ : \quad a \in \mathbb{R}^{N+1}\}$
- Polynomial functions: $\mathcal{F} = \{f_a(x) = \sum_{n=0}^{N} a_i x^n \ : \quad a \in \mathbb{R}^{N+1}\}$

(in the two latter examples we assume $d = 1$)

Example. For binary classification tasks, we can consider the composition of the sign function with the functions of any of the hypothesis set in the previous example, i.e.

$$\mathcal{F}_{sgn} = \{\text{sgn}(f(x)) \ : \quad f \in \mathcal{F}\},$$

where $\mathcal{F}$ is any of the hyopthesis set in the previous example.

Definition 1.2: Loss function

The loss function, denoted by

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

is a measurable function such that, given a function $f \in \mathcal{M}(\mathcal{X}, \mathcal{Y})$ and a data point $z \in \mathcal{Z}$, measures the discrepancy between the function f and the data point.

Example. In a regression task with $\mathcal{X} = \mathbb{R}^d$, $\mathcal{Y} = \mathbb{R}$ and $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$, a typical choice for the loss function is the squared error

$$\mathcal{L}(f, (x, y)) := (f(x) - y)^2.$$

Example. In a binary classification task with $\mathcal{X} = \mathbb{R}^d$, $\mathcal{Y} = \{-1, 1\}$ and $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$, a typical choice for the loss function is the 0-1 loss

$$\mathcal{L}(f, (x, y)) := \mathbb{1}_{(-\infty, 0)}(f(x)y).$$

Other choices are:

- $\mathcal{L}(f, (x, y)) := \max\{1 - f(x)y,\ 0\}$ (leading to support vector machine)

- $\mathcal{L}(f,(x,y)) := \log\left(1+\exp -f(x)y\right)$ (leading to logistic regression)

Definition 1.3: Learning algorithm

A learning algorithm is a map

$$\mathcal{A} : \bigcup_{m\in\mathbb{N}} \mathcal{Z}^m \longrightarrow \mathcal{F}$$

that, given a training data $Z = \{z^{(i)}\}_{i=1}^m \in \mathcal{Z}^m$ for some $m \in \mathbb{N}$, returns an element in the hypothesis set $\mathcal{F}$.

Example. Given a family set $\mathcal{F}$ and a loss function $\mathcal{L}$, a popular learning algorithm is the so-called **empirical risk minimization**, which minimizes the average loss on the given training data $Z = \{z^{(i)}\}_{i=1}^m \in \mathcal{Z}^m$, i.e.

$$\mathcal{A}(Z) \in \arg\min_{f\in\mathcal{F}} \widehat{\mathcal{R}}_Z(f) := \frac{1}{m}\sum_{i=1}^m \mathcal{L}(f, z^{(i)})$$

Given a loss function, $\mathcal{L}$, the **goal** in any learning task is to choose a suitable hypothesis set $\mathcal{F}$ and a learning algorithm $\mathcal{A}$ such that, for a given sufficiently large training data $Z = \{z^{(i)}\}_{i=1}^m$, the resulting function $f_Z := \mathcal{A}(Z)$ performs well (i.e. $\mathcal{L}(f_Z, z)$ is small) not only for the in-sample data $z \in Z$, but also for the out-of-sample data $z \in \mathcal{Z} \setminus Z$. This is known in the Machine Learning literature as **generalization**.

In order to establish a connection between the performance of $f_Z = \mathcal{A}(Z)$ on the training data and the unseen data, we need to make the following assumption:

Assumption 1

We assume that the training data $Z = \{z^{(i)}\}_{i=1}^m \in \mathcal{Z}^m$ are realizations of i.i.d. random variables $Z^{(1)}, Z^{(2)}, \ldots, Z^{(m)}$.
We shall denote by $\mathbb{P}_Z(\cdot) : \mathcal{Z} \longrightarrow \mathbb{R}$ the probability density function (p.d.f.) associated to the random variable of the training data $Z^{(i)}$.

We can now define the notion of **risk** as a measure of the performance of a function $f \in \mathcal{M}(\mathcal{X}, \mathcal{Y})$ on the space of data $\mathcal{Z}$.

Definition 1.4: Risk

Let $\mathcal{L} : \mathcal{M}(\mathcal{X}, \mathcal{Y}) \times \mathcal{Z} \to \mathbb{R}$ be a loss function and $\mathbb{P}_Z(\cdot)$ be the p.d.f. of the data distribution on $\mathcal{Z}$. For a function $f \in \mathcal{M}(\mathcal{X}, \mathcal{Y})$, we define the risk as

$$\mathcal{R}(f) := \mathbb{E}[\mathcal{L}(f, z)] = \int_{\mathcal{Z}} \mathcal{L}(f, z)\mathbb{P}_Z(z)dz.$$

Defining $Z := \{Z^{(i)}\}_{i=1}^m$ as a sequence of i.i.d. random variables with p.d.f. $\mathbb{P}_Z(\cdot)$, the risk of a model $f_Z := \mathcal{A}(Z)$ is given by

$$\mathcal{R}(f_Z) = \mathbb{E}[\mathcal{L}(f_Z, z)].$$

Since the training data Z is a random variable, under the assumption that the training algorithm $\mathcal{A}$ and the loss function $\mathcal{L}$ are measurable, the quantity $\mathcal{R}(f_Z)$ is also a random variable.

Example. In a prediction task, in which $\mathcal{Z} := \mathcal{X} \times \mathcal{Y}$, the risk is typically given by

- For regression: the expected squared distance between $f(x)$ and the label y

$$\mathcal{R}(f) = \mathbb{E}\left[(f(X) - Y)^2\right] = \int_{\mathcal{X}\times\mathcal{Y}} (f(x) - y)^2 \mathbb{P}_Z(x, y)d(x, y).$$

- For binary classification: the probability of missclassification

$$\mathcal{R}(f) = \mathbb{E}\left[\mathbb{1}_{(-\infty,0)}(f(X)Y)\right] = \mathbb{P}[f(X) \neq Y].$$

1.2 Bayes risk

Considering that the training dataset $Z = \{Z_i\}_{i=1}^m$ is an i.i.d. sampling from an unknown probability distribution over $\mathcal{Z}$, the **goal** is to choose a hypothesis set $\mathcal{F}$ and a learning algorithm $\mathcal{A}$ such that the risk of the model $f_Z := \mathcal{A}(Z)$, defined as

$$\mathcal{R}(f_Z) = \mathbb{E}[\mathcal{L}(f_Z, z)]$$

is as small as possible. Here, the expectation is taken with respect to the random variable z with p.d.f. $\mathbb{P}_Z(\cdot)$ over $\mathcal{Z}$. Next we define the Bayes risk, which is the smallest risk that one can expect considering all the measurable functions. Of course, this risk is not attainable in practice since we restrict ourselves to the hypothesis set $\mathcal{F} \subset \mathcal{M}(\mathcal{X}, \mathcal{Y})$, and moreover, we don't have access to compute the risk $\mathcal{R}(f)$, but we can only compute the empirical risk $\widehat{\mathcal{R}}_Z(f)$ instead.

Definition 1.5

We define the **Bayes risk** as

$$\mathcal{R}^* := \inf_{f \in \mathcal{M}(\mathcal{X}, \mathcal{Y})} \mathcal{R}(f),$$

and any function $f^* \in \mathcal{M}(\mathcal{X}, \mathcal{Y})$ such that $\mathcal{R}(f^*) = \mathcal{R}^*$ is called a **Bayes optimal function**.

Example. Let us consider a regression task with $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ and $\mathcal{L}(f, (x, y)) = (f(x) - y)^2$. Let us denote the p.d.f. of the sampling random variable by $\mathbb{P}_Z(z) = \mathbb{P}(x, y)$, so that the risk is given by

$$\begin{aligned} \mathcal{R}(f) &= \mathbb{E}\big[(f(X) - Y)^2\big] = \int_{\mathcal{X} \times \mathcal{Y}} (f(x) - y)^2 \mathbb{P}(x, y) d(x, y) \\ &= \int_{\mathcal{X}} \left(\int_{\mathcal{Y}} (f(x) - y)^2 \mathbb{P}(y | X = x) dy \right) \mathbb{P}_{\mathcal{X}}(x) dx. \end{aligned}$$

Then, the unique Bayes optimal function is given by

$$f^*(x) = \int_{\mathcal{Y}} y \, \mathbb{P}(y | X = x) dy = \mathbb{E}[Y | X = x],$$

and the Bayes risk is then given by

$$\mathcal{R}^* = \mathbb{E}\big[(\mathbb{E}[Y | X = x] - Y)^2\big] = \mathbb{E}_X\big[\mathbb{V}[Y | X]\big],$$

where $\mathbb{E}_X$ denotes the expectation with respect to the marginal distribution ρ_X on $\mathcal{X}$.

When $\mathcal{R}^*$ is big, we say that the problem is **ill-conditioned**.

Exercise 1.1. Consider a binary classification task with $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$, with $\mathcal{Y} = \{-1, 1\}$ and loss given by $\mathcal{L}(f, (x, y)) = \mathbb{1}_{(-\infty, 0)}[f(x) y]$.

Prove that

$$f^*(x) = \operatorname{sgn}\big(\mathbb{E}[Y | X = x]\big)$$

1.3 Types of error

The choice of the hypothesis set $\mathcal{F}$ (typically a finite-dimensional subset of $\mathcal{M}(\mathcal{X}, \mathcal{Y})$), along with the fact that we cannot compute the risk $\mathcal{R}(f)$ (the data distribution in $\mathcal{Z}$ is unknown), makes impossible to construct the Bayes optimal function f^*. Taking into account that the training data $Z = \{z^{(i)}\}_{i=1}^m$ corresponds to an i.i.d. sampling of some (unknown) distribution on $\mathcal{Z}$, the challenge is to choose a hypothesis set $\mathcal{F}$ and a training algorithm $\mathcal{A}$ such that the quantity $\mathcal{R}(f_Z) - \mathcal{R}^*$, where $f_Z = \mathcal{A}(Z)$ is small with high probability.

1.3.1 Approximation error:

The first source of error comes from the choice of the hypothesis set $\mathcal{F} \subset \mathcal{M}(\mathcal{X}, \mathcal{Y})$. This one has to be sufficiently large to ensure that the target function f^* can be approximated by some function in $\mathcal{F}$.

Definition 1.6

Given a hypothesis set $\mathcal{F} \subset \mathcal{M}(\mathcal{X}, \mathcal{Y})$, let us define

$$\mathcal{R}^*_{\mathcal{F}} := \inf_{f \in \mathcal{F}} \mathcal{R}(f).$$

The approximation error is defined as

$$\varepsilon_{approx} := \mathcal{R}^*_{\mathcal{F}} - \mathcal{R}^*.$$

It is proven that certain classes of Neural Networks are universal approximators, meaning that they can approximate any continuous function in a compact set, up to an arbitrarily small error, provided the number of parameters in the NN is sufficiently big. We can also prove bounds for the approximation error, depending on the regularity of the target function f^* and the dimension of the input space. We shall review some of these results in upcoming lectures.

1.3.2 Generalisation error

As mentioned before, we do not have access to the data distribution on $\mathcal{Z}$, so we cannot compute the risk $\mathcal{R}(f)$. Instead, we approximate it, using the training data $Z = \{z_i\}_{s=1}^m \in \mathcal{Z}^m$, by the so-called **empirical risk**, defined as

$$\widehat{\mathcal{R}}_Z(f) := \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(f, z_i).$$

By minimizing the empirical risk, we ensure good performance on the in-sample data.

Generally speaking, in many regression tasks one can prove that when the number of parameters in the NN is larger than the number of samples in the training data, there exist parameters that interpolate the data, meaning that the empirical risk can be rendered to 0.

However, we aim for a model which performs well in the out-of-sample data as well. The error arising from the fact that we have only access to a finite number of observations is known as generalisation error.

Definition 1.7

Let the hypothesis set $\mathcal{F}$ and the training data $Z = \{z_i\}_{s=1}^m$ be given. We define the generalisation error as

$$\varepsilon_{gen} := \sup_{f \in \mathcal{F}} \left| \widehat{\mathcal{R}}_Z(f) - \mathcal{R}(f) \right|$$

Choosing a very large hypothesis set $\mathcal{F}$ may yield to a situation in which, for some $f \in \mathcal{F}$, the empirical risk $\widehat{\mathcal{R}}_Z(f)$ is very small while the risk $\mathcal{R}(f)$ is large. This undesirable phenomenon is known as **overfitting** and there are different techniques to overcome this:

- Reducing the hypothesis set $\mathcal{F}$, for instance, by setting randomly chosen parameters to 0 (pruning).
- Adding a regularization term to the minimisation problem

$$\widehat{\mathcal{R}}_Z(f) + \gamma \|f\|_*^2,$$

where $\gamma > 0$ is a hyperparameter and $\|\cdot\|_*$ is a semi-norm, whose choice is made based on regularity assumptions on the actual distribution of the data $\mathbb{P}_Z$, or alternatively, on the optimal Bayes function f^*.

1.3.3 Optimisation error

The last source of error comes from the fact that, in many occasions, the learning algorithm is not able to find a function in $\mathcal{F}$ that minimizes the empirical risk, i.e.

$$\widehat{f}_Z := \arg\min_{f \in \mathcal{F}} \widehat{R}_Z(f).$$

In some cases, a minimizer may not exist, and in other cases, the training algorithm may get stuck in a local minimum due to the non-convexity of the minimisation problem. However, sometimes we are not actually interested in finding a minimizer in order to prevent overfitting. Remember that many learning algorithms are based on minimizing a regularized empirical risk.

Definition 1.8

Let the hypothesis set $\mathcal{F}$, the training algorithm $\mathcal{A}$ and the training data $Z = \{z_i\}_{s=1}^m$ be given. Let us denote

$$f_Z := \mathcal{A}(Z) \qquad \text{and} \qquad \widehat{\mathcal{R}}_Z^* := \inf_{f \in \mathcal{F}} \widehat{\mathcal{R}}_Z(f).$$

We define the optimisation error as

$$\varepsilon_{opt} := \widehat{\mathcal{R}}_Z(f_Z) - \widehat{\mathcal{R}}_Z^*.$$

1.4 Upper bound for the error

Given a hypothesis set $\mathcal{F}$, a training algorithm $\mathcal{A}$ and training data $Z = \{z_i\}_{s=1}^m$, we can upper estimate $\mathcal{R}(f_Z) - \mathcal{R}^*$ in terms of the approximation, generalisation and optimisation error as follows.

Lemma 1.1. Let $\mathcal{F} \subset \mathcal{M}(\mathcal{X}, \mathcal{Y})$ be the hypothesis set and $\mathcal{A} : \mathcal{Z}^m \to \mathcal{F}$ the training algorithm. For any training data $Z = \{z_i\}_{i=1}^m \subset \mathcal{Z}^m$, let us define $f_Z := \mathcal{A}(Z)$. Then,

$$\mathcal{R}(f_Z) - \mathcal{R}^* \leq \varepsilon_{approx} + 2\varepsilon_{gen} + \varepsilon_{opt}.$$

Proof. By the definition of $\mathcal{R}_{\mathcal{F}}^*$, for any $\varepsilon > 0$, there exists $f_{\mathcal{F},\varepsilon}^* \in \mathcal{F}$ such that

$$\mathcal{R}(f_{\mathcal{F},\varepsilon}^*) < \mathcal{R}_{\mathcal{F}}^* + \varepsilon.$$

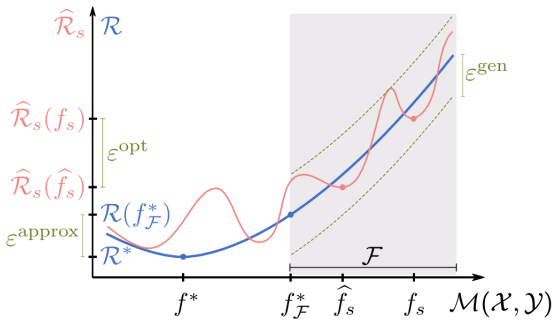

Figure 1: Illustration of the three types of error involved in a learning task.

Then we have

$$\begin{aligned}\mathcal{R}(f_Z) - \mathcal{R}^* &= \underbrace{\mathcal{R}(f_Z) - \widehat{\mathcal{R}}_Z(f_Z)}_{A_1} + \underbrace{\widehat{\mathcal{R}}_Z(f_Z) - \widehat{\mathcal{R}}_Z(f^*_{\mathcal{F},\varepsilon})}_{A_2} \\ &\quad + \underbrace{\widehat{\mathcal{R}}_Z(f^*_{\mathcal{F},\varepsilon}) - \mathcal{R}(f^*_{\mathcal{F},\varepsilon})}_{A_3} + \underbrace{\mathcal{R}(f^*_{\mathcal{F},\varepsilon}) - \mathcal{R}^*}_{A_4} \\ &= A_1 + A_2 + A_3 + A_4.\end{aligned}$$

Every term A_1, A_2, A_3, A_4 can be estimated as follows:

$$\max\{A_1, A_3\} \leq \sup_{f\in\mathcal{F}} \left|\mathcal{R}(f) - \widehat{\mathcal{R}}_Z(f)\right| = \varepsilon_{gen}.$$

Using the definition of $\widehat{\mathcal{R}}^*_Z$ we obtain

$$A_2 \leq \widehat{\mathcal{R}}(f_Z) - \widehat{\mathcal{R}}^*_Z = \varepsilon_{opt}.$$

And finally, by the choice of $f^*_{\mathcal{F},\varepsilon}$ we have

$$A_4 < \mathcal{R}^*_{\mathcal{F}} - \mathcal{R}^* + \varepsilon = \varepsilon_{approx} + \varepsilon.$$

This implies that, for all $\varepsilon > 0$, we have

$$\mathcal{R}(f_Z) - \mathcal{R}^* < \varepsilon_{approx} + 2\varepsilon_{gen} + \varepsilon_{opt} + \varepsilon.$$

The result follows by letting $\varepsilon \to 0$. □

As a consequence of the above lemma, one can provide theoretical guarantees for a training algorithm by estimating the approximation, generalization and optimization errors. That being said, proving bounds for these quantities is not a straightforward task. In general, we rely on assumptions about the unknown distribution $\mathbb{P}_Z$.

1.5 Deep Neural Networks

In Deep Learning, the hypothesis set $\mathcal{F}$ consists of artificial neural networks, that we shall denote by

$$\Phi(\cdot, \theta) : \mathcal{X} \longrightarrow \mathcal{Y} \qquad \text{for } \theta \in \mathcal{P}$$

with a given **architecture** and a **parameter space** $\mathcal{P}$. For a given task, the goal in Deep Learning is to find a suitable architecture and then to compute the parameters $\theta \in \mathcal{P}$ that perform the best (both on the training data and the unseen data).

An Artificial Neural Network can be represented as a directed acyclic graph in which the vertices (neurons) represent elementary operations (parametrizable by θ) and the edges (connections) represent compositions between the operations in the neurons.

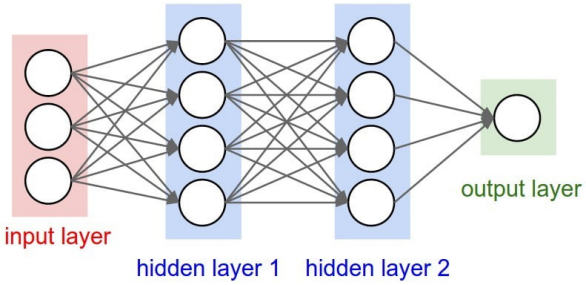

Figure 2: Image taken from [Berner et al., 2021]

The elementary operations at every neuron consist of compositions of affine transformations, followed by a non-linear function, known as the **activation function**. Let us give a more precise definition of what we mean by artificial neuron.

Definition 1.9

We define **artificial neuron** as any function of the form

$$\begin{array}{rccc} \phi(\cdot, \theta): & \mathbb{R}^{d_{in}} & \longrightarrow & \mathbb{R} \\ & x & \longmapsto & \sigma(w \cdot x + b) \end{array}$$

where $\theta = (w, b) \in \mathbb{R}^{d_{in}+1}$ are the parameters and $\sigma(\cdot)$ is a non-linear function $\mathbb{R} \to \mathbb{R}$.
Here, $d_{in} \in \mathbb{N}$ stands for the input dimension of the neuron, i.e. the number of edges arriving to the neuron.

Note:-

We observe that for a given neuron as above, the parameters $\theta = (w, b)$ define a hyperplane $\{w \cdot x + b = 0\}$ which divides the space $\mathbb{R}^{d_{in}}$ in two half-spaces. The activation function is then used as an indicator of which side of the hyperplane the input belongs to.

In view of that, a reasonable choice for the activation function would be the heavy-

side function $\sigma(t) = \mathbb{1}_{(0,\infty)}(t)$. However, for practical reasons, it is more convenient to consider continuous functions.

The name "artificial neuron" is obviously inspired by neurons in biology. However, this inspiration is not enough to justify their performance, and then their application, in machine learning problems.

In practice, we consider activation functions which are continuously increasing and differentiable almost everywhere in $\mathbb{R}$. Examples of activation functions are the following:

1. **Rectified Linear Unit (ReLU):** $\sigma(t) := \max\{0, t\}$
2. **Sigmoid functions:** Any continuously increasing function satisfying

$$\lim_{t\to-\infty} f(t) = 0 \quad \text{and} \quad \lim_{t\to+\infty} f(t) = 1,$$

   for instance the logistic function $\sigma(t) := \dfrac{1}{1+e^{-t}}$.
3. **Hyperbolic tangent:** $\sigma(t) := \dfrac{e^t - e^{-t}}{e^t + e^{-t}}$.

One of the most popular classes of neural networks is the so-called **feedforward fully connected neural necworks**, and they can be used for tasks such as regression and classification.

Definition 1.10

A feedforward fully connected neural network is determined by its architecture $a = (N, \sigma)$, where $N \in \mathbb{N}^{L+1}$, for some $L \in \mathbb{N}$, and $\sigma : \mathbb{R} \to \mathbb{R}$ is the activation function.

Here, L represents the number of layers, and determines the depth of the NN. Moreover, N_0 represents the dimension of the input, and N_L the dimension of the output.

For a given input $x \in \mathbb{R}^{N_0}$, the output is given by

$$\Phi_a(x, \theta) = x_L \in \mathbb{R}^{N_L}$$

where x_L is the L-th term in the sequence given by

$$\begin{cases} x_l = \left[\phi_l^{(1)}\left(x_{l-1}, \theta_l^{(1)}\right), \ldots, \phi_l^{(N_l)}\left(x_{l-1}, \theta_l^{(N_l)}\right)\right] \in \mathbb{R}^{N_l} \\ x_0 = x \in \mathbb{R}^{N_0} \end{cases} \tag{1.1}$$

Here, each of the functions $\phi_l^{(j)}\left(\cdot, \theta_l^{(j)}\right) : \mathbb{R}^{N_{l-1}} \to \mathbb{R}$ represents an artificial neuron as defined in Definition 1.5.
The parameters of the NN are of the form $\theta_l^{(j)} = (a_l^{(j)}, b_l^{(j)}) \in \mathbb{R}^{N_{l-1}} \times \mathbb{R}$, thus the parameter space is

$$\mathcal{P} := \bigtimes_{l=1}^{L} \left(\mathbb{R}^{N_{l-1}} \times \mathbb{R}\right)^{N_l}.$$

Note:-

We can write (1.1) in a more compact way as

$$\begin{cases} x_l = \sigma(W_l x_{l-1} + b_l) & l \in \{1, \dots L\} \\ x_0 = x \in \mathbb{R}^{N_0} \end{cases}$$

where, $W_l \in \mathbb{R}^{N_l \times N_{l-1}}$ and $b_l \in \mathbb{R}^l$ for each $l \in \{1, \dots, L\}$, and the activation function acts component wise on vectors of any size.

Note that the numper of parameters is given by

$$P(N) := \sum_{l=1}^{L} N_l(N_{l-1} + 1).$$

- It is a usual practice to omit the activation in the last layer, so that the we can write Φ_a as a linear combination of NNs with $L-1$ layers. This gives $\mathcal{F}$ the structure of a linear space.

Example. Let us give some examples of hypothesis set for some prediction tasks using fully connected NNs. Let $a = (N, \sigma)$ be an NN architecture.

- For regression task we consider

$$\mathcal{F} := \left\{ \Phi_a(\cdot, \theta) \ : \quad \theta \in \mathbb{R}^{P(N)} \right\}.$$

- For binary classification we consider $N_L = 1$ and

$$\mathcal{F}_{sgn} := \left\{ \operatorname{sgn}\left(\Phi_a(\cdot, \theta)\right) \ : \quad \theta \in \mathbb{R}^{P(N)} \right\}.$$

- For multiclass classification we consider $N_L = n_c$ the number of classes and

$$\mathcal{F}_{argmax} := \left\{ \underset{j=1,\dots,n_c}{\arg\max} \left(\Phi_a(\cdot, \theta)^{(j)} \right) \ : \quad \theta \in \mathbb{R}^{P(N)} \right\}.$$

January 24 2024 Wed (12:00-13:00)

Lecture 2: Kernel methods

Let us consider the regression problem of approximating a function

$$f : \mathbb{R}^d \longrightarrow \mathbb{R}$$

from a given dataset $Z = \{(x^{(i)}, y^{(i)})\}_{i=1}^n \in (\mathbb{R}^d \times \mathbb{R})^n$ obtained as

$$y^{(i)} = f(x^{(i)}) + \varepsilon_i.$$

where $\{\varepsilon_i\}_{i=1}^n$ are i.i.d. random variables, e.g. $\varepsilon_i \sim N(0, \sigma^2)$.

Note:-

Instead of NNs, we will consider today a simpler method, which can be seen as a generalization of linear regression.

2.1 Linear regression

We can consider a linear model of the form

$$f_\theta(x) = \langle \theta, x \rangle, \qquad \text{for some } \theta \in \mathbb{R}^d.$$

Every prediction $f_\theta(x)$ is based on a linear combination of the entries in the vector $x = (x_1, \ldots, x_d) \in \mathbb{R}^d$, which we refer to as **features**. The feature space is therefore the Euclidean space $\mathbb{R}^d$, which is a Hilbert space endowed with the dot product $\langle x, x' \rangle = \sum_{j=1}^d x_j x'_j$.

The coefficients $\theta \in \mathbb{R}^d$ in the model f_θ are typically obtained by minimizing the mean square error

$$\theta = \arg\min_{\theta \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^n (\langle \theta, x^{(i)} \rangle - y^{(i)})^2 + \gamma \|\theta\|^2, \tag{2.1}$$

with possibly an l^2 regularization term $\|\theta\|^2 = \langle \theta, \theta \rangle$ and $\gamma > 0$.

Exercise 2.1. Under the assumption that the parameter θ follows a normal distribution $N(0, \tau I_d)$, prove that, for a suitable parameter $\gamma > 0$ the solution to (2.1) maximizes the posterior probability

$$\Pr\left[\theta \middle| \{(x^{(i)}, y^{(i)})\}_{i=1}^n\right].$$

Note:-

Advantages:

- The minimization problem (2.1) is uniformly convex in θ.
- Very efficient if the target function f is linear. We can use optimization algorithms with theoretical guarantees or even compute the explicit solution.

Limitation:

- The model is too rigid and cannot represent non-linear functions (the hy-

pothesis set is too small).

Question: Can we construct a model which is linear in its parameters and can capture non-linear phenomena?

- In linear regression the hypothesis set is too small.
- We can enrich the hypothesis set by adding more parameters, but how can we add more parameters if we want our model to be linear in its parameters?

The only possibility is to increase the dimension of the feature space.

2.2 Rising the dimension of the feature space

We are going to consider, as feature space, a Hilbert space $\mathcal{H}$ with dimension much higher than d, potentially with infinite dimension. We also need a map

$$\varphi : \mathbb{R}^d \longrightarrow \mathcal{H},$$

which associates, to every input $x \in \mathbb{R}^d$, the corresponding features in the feature space $\mathcal{H}$.

We are going to consider a linear model of the form

$$f_\theta(x) = \langle \theta, \varphi(x) \rangle_{\mathcal{H}} \qquad \text{for some } \theta \in \mathcal{H}.$$

Important: it is convenient to consider a map $\varphi : \mathbb{R}^d \to \mathcal{H}$ which is nonlinear, so that the model f_θ is linear in the parameter θ and non-linear in the input variable x.

Now we can consider empirical risk minimization to obtain the parameter $\theta \in \mathcal{H}$ as the solution to the minimization problem

$$\min_{\theta \in \mathcal{H}} \widehat{R}_Z(\theta) + \gamma \|\theta\|_{\mathcal{H}}^2, \tag{2.2}$$

where

$$\widehat{R}_Z(\theta) := \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}\big(\langle \theta, \varphi(x^{(i)}) \rangle_{\mathcal{H}},\, y^{(i)}\big).$$

Note:-

In the case of a convex loss function $\mathcal{L}$, e.g. squared error, the above minimization problem is strictly convex in θ as soon as $\gamma > 0$. Hence, it can be addressed with theoretical guarantees by means of gradient based methods.

The following Theorem is crucial in kernel methods, and justifies the need for the feature space to have a Hilbert space structure (i.e. a dot product), and the addition of the quadratic regularization term $\|\theta\|_{\mathcal{H}}^2$.

Theorem 2.1 (Representer Theorem). Let $\gamma > 0$, $\varphi : \mathbb{R}^d \to \mathcal{H}$ and let us be given a dataset $Z = \{x^{(i)}, y^{(i)}\}_{i=1}^n \in (\mathbb{R}^d, \mathbb{R})^n$. It holds that

$$\inf_{\theta \in \mathcal{H}} \widehat{R}_Z(\theta) + \gamma \|\theta\|_{\mathcal{H}}^2 = \inf_{\theta \in \mathcal{H}_Z} \widehat{R}_Z(\theta) + \gamma \|\theta\|_{\mathcal{H}}^2,$$

where $\mathcal{H}_Z$ is the linear subspace spanned by $\{\varphi(x^{(i)})\}_{i=1}^n \in \mathcal{H}^n$, i.e.

$$\mathcal{H}_Z := \left\{ \sum_{i=1}^n \alpha_i \varphi(x^{(i)}) \ : \quad \alpha \in \mathbb{R}^n \right\}.$$

Proof. Let us denote by $\mathcal{H}_Z^\perp$ the orthogonal complement of $\mathcal{H}_Z$ in $\mathcal{H}$, i.e.

$$\mathcal{H}_Z^\perp := \left\{ \theta \in \mathcal{H} \ : \quad \langle \theta, \hat{\theta} \rangle = 0 \quad \forall \hat{\theta} \in \mathcal{H}_Z \right\}.$$

Using the Hilbertian structure of $\mathcal{H}$, every $\theta \in \mathcal{H}$ can be written as $\theta = \theta_Z + \theta_Z^\perp$, for some $\theta_Z \in \mathcal{H}_s$ and $\theta_Z^\perp \in \mathcal{H}_Z^\perp$. Then we have

$$\langle \theta, \varphi(x^{(i)}) \rangle_{\mathcal{H}} = \langle \theta_Z, \varphi(x^{(i)}) \rangle_{\mathcal{H}} + \langle \theta_Z^\perp, \varphi(x^{(i)}) \rangle_{\mathcal{H}} = \langle \theta_Z, \varphi(x^{(i)}) \rangle_{\mathcal{H}},$$

for all $x^{(i)}$, with $i = 1, \dots, n$, which implies

$$\widehat{R}_Z(\theta) = \frac{1}{n} \sum_{i=1}^n \mathcal{L}\left(\langle \theta_Z, \varphi(x^{(i)}) \rangle_{\mathcal{H}}, y^{(i)} \right) = \widehat{R}_Z(\theta_Z).$$

Moreover, using the Pythagorean Theorem, we have

$$\|\theta\|_{\mathcal{H}}^2 = \|\theta_Z\|_{\mathcal{H}}^2 + \|\theta_Z^\perp\|_{\mathcal{H}}^2.$$

Hence, for any $\theta \in \mathcal{H}$, we have

$$\widehat{R}_Z(\theta) + \|\theta\|_{\mathcal{H}}^2 \geq \widehat{R}_Z(\theta_Z) + \|\theta_Z\|_{\mathcal{H}}^2$$

for some $\theta_Z \in \mathcal{H}_Z$. Whence, since $\mathcal{H}_Z \subset \mathcal{H}$, we deduce

$$\inf_{\theta \in \mathcal{H}} \widehat{R}_Z(\theta) + \|\theta\|_{\mathcal{H}}^2 \leq \inf_{\theta \in \mathcal{H}_Z} \widehat{R}_Z(\theta) + \|\theta\|_{\mathcal{H}}^2 \leq \inf_{\theta \in \mathcal{H}} \widehat{R}_Z(\theta) + \|\theta\|_{\mathcal{H}}^2.$$

□

As a consequence of the Representer Theorem, although the Hilbert space $\mathcal{H}$ may be infinite-dimensional, we can restrict ourselves to parameters θ of the form

$$\theta = \sum_{i=1}^n \alpha_i \varphi(x^{(i)}) \qquad \text{for some } \alpha \in \mathbb{R}^n. \tag{2.3}$$

Note:-

- The optimization space has the same dimension as the size of the dataset. This might seem to be a bit brutal, but the number of parameters in NNs architectures is typically much bigger than the size of the dataset.
- No assumption is made on the loss $\mathcal{L}$ for the Representer Theorem to hold. Hence, kernel methods apply to any kind of loss function. The advantage of considering convex losses is that it makes the minimization problem (2.2) convex, which is convenient when applying gradient based methods to approximate a solution.

With the knowledge that the parameter θ has the form (2.3), the linear model f_θ

can be written as

$$f_\theta(x) = \left\langle \sum_{i=1}^n \alpha_i \varphi(x^{(i)}), \varphi(x) \right\rangle_{\mathcal{H}} = \sum_{i=1}^n \alpha_i \langle \varphi(x^{(i)}), \varphi(x) \rangle_{\mathcal{H}} = \sum_{i=1}^n \alpha_i k(x^{(i)}, x),$$

where $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is the so-called **kernel function**, defined as

$$k(x, x') = \langle \varphi(x), \varphi(x') \rangle_{\mathcal{H}}.$$

2.3 Reformulation of risk minimization problem

Given the dataset

$$\{(x^{(i)}, y^{(i)})\}_{i=1}^n \in (\mathbb{R}^d \times \mathbb{R})^n,$$

we want to solve the minimization problem

$$\min_{\theta \in \mathcal{H}_Z} \frac{1}{n} \sum_{i=1}^n \mathcal{L}\left(\langle \theta, \varphi(x^{(i)}) \rangle_{\mathcal{H}}, y^{(i)} \right) + \gamma \|\theta\|_{\mathcal{H}}^2,$$

where $\mathcal{H}_Z$ is the span of $\{\varphi(x_i)\}_{i=1}^n$.

We define the kernel matrix associated to $\{x^{(i)}\}_{i=1}^n$ as

$$K \in \mathbb{R}^{n \times n}, \qquad K_{ij} = k(x^{(i)}, x^{(j)}).$$

For any $j \in \{1, \dots, n\}$ and $\theta \in \mathcal{H}_s$ of the form $\theta = \sum_{j=1}^n \alpha_j \varphi(x^{(j)})$, we can write

$$\langle \theta, \varphi(x^{(i)}) \rangle_{\mathcal{H}} = \sum_{j=1}^n \alpha_j k(x^{(i)}, x^{(j)}) = (K\alpha)_i.$$

On the other hand we have

$$\|\theta\|_{\mathcal{H}}^2 = \langle \theta, \theta \rangle_{\mathcal{H}} = \sum_{i=1}^n \alpha_i \langle \theta, \varphi(x^{(i)}) \rangle_{\mathcal{H}} = \alpha^\top K \alpha.$$

Hence, we can write the risk minimization problem as

$$\min_{\alpha \in \mathbb{R}^n} \frac{1}{n} \sum_{i=1}^n \mathcal{L}\left((K\alpha)_i, y^{(i)} \right) + \gamma \alpha^\top K \alpha,$$

which is convex whenever the loss function $\mathcal{L}$ is convex.

Exercise 2.2. Prove that the kernel matrix $K \in \mathbb{R}^{n \times n}$ is positive semidefinite, i.e. $\alpha^\top K \alpha \geq 0$ for all $\alpha \in \mathbb{R}^n$.

Example. In the case of a quadratic loss of the form

$$\mathcal{L}(\hat{y}, y) = (\hat{y} - y)^2$$

the minimization problem can be written as

$$\min_{\alpha \in \mathbb{R}^d} \frac{1}{n} \|K\alpha - Y\|^2 + \gamma \alpha^\top K \alpha.$$

Jan 29 2024 Mon (12:00-13:00)

Lecture 3: Reproducing kernel Hilbert spaces

3.1 Definition of RKHS

We have seen that from a feature map

$$\varphi : \mathbb{R}^d \longrightarrow \mathcal{H}$$

where $\mathcal{H}$ is a Hilbert space, we can construct a kernel $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ defined as

$$k(x, x') := \langle \varphi(x), \varphi(x') \rangle_{\mathcal{H}},$$

and consider regression functions of the form

$$h(x) = \sum_{i=1}^{n} \alpha_i k(x^{(i)}, x), \qquad \text{for some } \{x^{(i)}\}_{i=1}^n \in (\mathbb{R}^d)^n \text{ and } \alpha \in \mathbb{R}^n. \tag{3.1}$$

Instead of constructing the regression functions h from a feature space $\mathcal{H}$ and a feature map φ, we can construct them directly from a suitable positive definite kernel function $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$.

Definition 3.11

A function $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is a positive definite kernel if all kernel matrices[a] $K \in \mathbb{R}^{n \times n}$ are positive semi-definite.

[a] matrices of the form $K_{ij} = k(x^{(i)}, x^{(j)})$ for some $\{x^{(i)}\}_{i=1}^n \in (\mathbb{R}^d)^n$.

Given a definite positive kernel $k(\cdot, \cdot)$, we can endow the linear space formed by the functions of the form (3.1) with the inner product

$$\langle h, h' \rangle = \sum_{i=1}^{n} \sum_{j=1}^{n'} \alpha_i \alpha'_j k(x^{(i)}, x'^{(j)}) = \alpha^\top K_{h,h'} \alpha',$$

where $K_{h,h'} \in \mathbb{R}^{n \times n'}$ is the matrix $K_{ij} = k(x^{(i)}, x'^{(j)})$. By the positivity of the kernel, we can prove that

$$\|h\|^2 = \langle h, h \rangle = \alpha^\top K_h \alpha,$$

where $K_h \in \mathbb{R}^{n \times n}$ is the positive semidefinite matrix $K_{ij} = k(x^{(i)}, x^{(j)})$, defines a semi-norm in the linear space of the functions of the form (3.1).

Definition 3.12

We define the **Reproducing Kernel Hilbert Space** (RKHS) generated by the kernel k as the completion of the functions h of the form (3.1) w.r.t the norm

$$\|h\|^2 = \alpha^\top K \alpha,$$

where $K \in \mathbb{R}^{n \times n}$ is the kernel matrix with coefficients $K_{ij} = k(x^{(i)}, x^{(j)})$.

The following Theorem establishes the equivalency between positive definite kernels and feature maps.

Theorem 3.1. A kernel function $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is positive definite if and only if there exists a Hilbert space $\mathcal{H}$ and a function $\varphi : \mathbb{R}^d \to \mathcal{H}$ such that

$$k(x, x') = \langle \varphi(x), \varphi(x') \rangle_{\mathcal{H}} \qquad \forall x, x' \in \mathcal{H}.$$

Note:-

- As a consequence of the previous theorem, any RKHS on $\mathbb{R}^d$ can be defined, either through a positive kernel or through a Hilbert space and a feature map φ.
- The feature space $\mathcal{H}$ and the feature map φ is not necessarily unique.
- For any RKHS over $\mathbb{R}^d$ there exists a Hilbert space $\mathcal{H}$ and a feature map $\varphi : \mathbb{R}^d \to \mathbb{R}$ such that **any function in the RKHS** is the limit functions of the form

$$f_\theta(x) = \langle \theta, \varphi(x) \rangle_{\mathcal{H}} \qquad \text{for some } \theta = \sum_{i=1}^{n} \alpha_i \varphi(x^{(i)}).$$

(the limit has to be taken w.r.t. the norm of the RKHS)

Important: It is not in general true that the RKHS generated by $\mathcal{H}$ and φ is isomorphic to $\mathcal{H}$. Indeed, there might exist $\theta_1, \theta_2 \in \mathcal{H}$ with $\theta_1 \neq \theta_2$ such that $\langle \theta_1, \varphi(x) \rangle_{\mathcal{H}} = \langle \theta_2, \varphi(x) \rangle_{\mathcal{H}}$ for all $x \in \mathbb{R}^d$, i.e. $f_{\theta_1}(\cdot) = f_{\theta_2}(\cdot)$.

In particular, given a definite positive kernel $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, we can choose, as Hilbert space, the RKHS generated by k and as feature map, the map

$$\varphi(x) = k(\cdot, x).$$

The RKHS $\mathcal{F}$ associated to k satisfies the so-called reproducing properties

$$h(x) = \langle k(\cdot, x), h \rangle_{\mathcal{F}} \quad \forall h \in \mathcal{H} \quad \text{and} \quad k(x, x') = \langle k(\cdot, x), k(\cdot, x') \rangle_{\mathcal{F}}.$$

This gives raise to an alternative definition of RKHS.

Definition 3.13

A Hilbert space $\mathcal{F}$ is a RKHS over $\mathbb{R}^d$ if, for every $x \in \mathbb{R}^d$, there exists a linear functional $L_x : \mathcal{F} \to \mathbb{R}$ such that

$$f(x) = L_x\big[f(\cdot)\big], \qquad \forall f(\cdot) \in \mathcal{F}.$$

By Riesz representation Theorem, for every linear functional L_x, there exists $K_x(\cdot) \in \mathcal{F}$ such that

$$L_x\big[f(\cdot)\big] = \langle K_x(\cdot), f(\cdot) \rangle_{\mathcal{F}}, \qquad \forall f(\cdot) \in \mathcal{F}.$$

The reproducing kernel associated to $\mathcal{H}$ is therefore given by

$$k(x, y) = K_x(y).$$

Example. Let $G(\cdot,\cdot):\mathbb{R}^d\times\mathbb{R}^d\to\mathbb{R}$ be a function such that

$$G(\cdot,x)\in L^2(\mathbb{R}^d)\qquad \forall x\in\mathbb{R}^d,$$

and define the positive definite kernel

$$k(x,y)=\int_{\mathbb{R}^d} G(z,x)G(z,y)dz.$$

The RKHS, denoted by $\mathcal{F}$, associated to this kernel contains all the functions of the form

$$f(x)=\int_{\mathbb{R}^d} g(y)G(y,x)dy, \tag{3.2}$$

where $g\in L^2(\mathbb{R}^d)$ is such that

$$g(x)=\sum_{i=1}^{n}\alpha_i G(x,x^{(i)})\qquad \text{for some}\quad \{x^{(i)}\}_{i=1}^n\subset\mathbb{R}^d,\ \alpha\in\mathbb{R}^n \tag{3.3}$$

The $\mathcal{F}$-norm of such f is given by

$$\|f\|_{\mathcal{F}}=\|g\|_{L^2}.$$

The space $\mathcal{F}$ is formed by all the functions of the form (3.2)–(3.3) and all the limits of such functions w.r.t. the norm $\|\cdot\|_{\mathcal{F}}$. Actually, $\mathcal{F}$ consists of all the functions of the form (3.2) with $g\in L^2(\mathbb{R}^d)$

Important: Although $\mathcal{F}$ is a Hilbert space of functions $\mathbb{R}^d\to\mathbb{R}$, $\mathcal{F}$ and $L^2(\mathbb{R}^d)$ are different spaces. Actually, $L^2(\mathbb{R})$ is not a RKHS.

We note that, using a smooth kernel regularizes the functions, and consequently, the corresponding RKHS contains functions that are more regular than those in $L^2(\mathbb{R}^d)$.

Exercise 3.1.

- Prove that $L^2(\mathbb{R}^d)$ is not a RKHS.
- Prove that

$$\mathcal{F}_{lin}:=\{f(x)=w\cdot x\ :\quad w\in\mathbb{R}^d\}$$

and

$$\mathcal{F}_{aff}:=\{f(x)=w\cdot x+b\ :\quad (w,b)\in\mathbb{R}^d\times\mathbb{R}\}$$

are two different RKHS and compute the associated kernel.

3.2 Connection between RKHS and NNs

Let us consider a FCNN with $L+1$ layers, **omitting the activation function and the biases in the last layer**. In other words, let us consider functions of the form

$$\Phi(x;w,\theta)=w\cdot\Phi_L(x,\theta)=\sum_{i=1}^{N_L} w_i\phi(x,\theta_i), \tag{3.4}$$

where $w \in \mathbb{R}^{N_L}$ and $\Phi_L(\cdot, \theta) : \mathbb{R}^d \to \mathbb{R}^{N_L}$ is a FCNN with L layers, that we can write as

$$\Phi(\cdot, \theta) = \big(\phi(\cdot, \theta_1), \dots, \phi(\cdot, \theta_{N_L})\big)$$

where $\phi(\cdot, \theta_i) : \mathbb{R}^d \to \mathbb{R}$ is a NN and the parameters $\theta = (\theta_1, \dots, \theta_{N_L}) \in \mathcal{P}^{N_L}$, where $\mathcal{P} \subset \mathbb{R}^{\#P}$, with $\#P$ being the number of parameters in the NN $\phi(x, \theta)$, i.e. the dimensionality of θ.

Given a measure $\mu(\cdot)$ on the space of parameters $\mathcal{P}$, let us denote by $C(\mathcal{P})/\mu$ the quotient space of continuous functions, under the equivalence relation of being equal μ-a.e.[1].

We define, for any pair of continuous functions $\varphi_1(\cdot), \varphi_2(\cdot) \in C(\mathcal{P})/\mu$, the inner product

$$\langle \varphi_1, \varphi_2 \rangle_{L^2(d\tau)} = \int_{\mathcal{P}} \varphi_1(\theta)\varphi_2(\theta)\mu(\theta)d\theta.$$

We now consider, as feature space $\mathcal{H}$, the Hilbert space $L^2(\mathcal{P}, \mu)$ obtained as the completion of $C(\mathcal{P})/\mu$ w.r.t the above inner product.

Since the neural network ϕ used (3.4) is continuous in its parameters, we have that the function

$$\theta \in \mathcal{P} \longmapsto \phi(x, \theta)$$

is in $L^2(\mathcal{P}, \mu)$ for all $x \in \mathbb{R}^d$. Hence, we can define the feature map

$$x \longmapsto \phi(x, \cdot).$$

With this feature map we can then define the following positive kernel over $\mathbb{R}^d$:

$$k(x_1, x_2) = \int_{\mathcal{P}} \phi(x_1, \theta)\phi(x_2, \theta)\mu(\theta)d\theta.$$

This positive kernel defines an associated RKHS, that we call $\mathcal{F}$, and we can consider the problem of minimizing

$$\frac{1}{m}\sum_{j=1}^{m} \mathcal{L}(f, (x_j, y_j)) + \|f\|^2_{\mathcal{H}}$$

over $\mathcal{H}$.

By the Representer Theorem, we have that the minimizer is of the form

$$f^*(x) = \sum_{j=1}^{n} \alpha_j k(x, x_j).$$

Now, we can approximate the kernel by sampling N_L points $\{\theta_i\}_{i=1}^{N_L} \subset \mathcal{P}$ from the probability distribution $\mu(\cdot)$, to obtain

$$\hat{k}(x, x_j) \approx \frac{1}{N_L}\sum_{i=1}^{N_L} \phi(x_j, \theta_i)\phi(x, \theta_i).$$

[1] We consider that two continous functions $\varphi_1(\cdot)$ and $\varphi_2(\cdot)$ are the same in $C(\mathcal{P})/\mu$ whenever they coincide for all $\theta \in \mathcal{P} \setminus A$, where A is a set of μ-measure 0.

This gives an approximation of the optimal function in the RKHS as

$$\begin{aligned}\hat{f}(x) &= \sum_{j=1}^{m} \alpha_j \frac{1}{N_L} \sum_{i=1}^{N_L} \phi(x_j, \theta_i)\phi(x, \theta_i) \\ &= \sum_{i=1}^{N_L} \underbrace{\frac{\alpha_j}{N_L} \sum_{j=1}^{m} \phi(x_j, \theta_i)}_{w_i} \phi(x, \theta_i).\end{aligned}$$

Note:-

The above computation tells us that using a Deep Neural Network with initial random parameters and optimizing only the parameters of the last layer is an approximation to optimizing over a RKHS, in which the kernel depends on the distribution chosen to sample the initial parameters. This technique is known as **random feature model**.

Optimizing over the parameters in the lower layers of the NN can be interpreted as optimizing over the kernel.

Jan 31 2024 Wed (12:00-13:00)

Lecture 4: Neural Tangent Kernel

As in the previous lectures, we consider the problem of approximating a function

$$f : \mathbb{R}^d \longrightarrow \mathbb{R}$$

In this lecture we consider a loss function

$$\mathcal{L} : \mathcal{M}(\mathbb{R}^d, \mathbb{R}) \times (\mathbb{R}^d \times \mathbb{R}) \to \mathbb{R},$$

of the form

$$(f, (x, y)) \longmapsto \mathcal{L}(f(x), y),$$

i.e. the loss only depends on $f(x)$ and y. It measures the discrepancy between $f(x)$ and y.

Given a hypothesis set $\mathcal{F} \subset \mathcal{M}(\mathbb{R}^d, \mathbb{R})$ and a dataset $\{(x^{(i)}, y^{(i)})\}_{i=1}^m \in (\mathbb{R}^d \times \mathbb{R})^m$, we consider the Empirical Risk Minimization (ERM) problem, that reads as follows:

$$\underset{f \in \mathcal{F}}{\text{minimize}}\ \widehat{R}(f) := \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(f(x^{(i)}), y^{(i)}).$$

In deep learning, $\mathcal{F}$ consists of a parametrized neural network with a given architecture

$$\mathcal{F} := \{f(x) = \Phi(x; \theta) \ : \quad \theta \in \mathcal{P}\},$$

where $\mathcal{P} \subset \mathbb{R}^{\#P}$, and $\#P$ is the number of parameters in the NN.

The parametrization of $\mathcal{F}$ allows us to reduce the training to a finite-dimensional optimization problem

$$\underset{\theta \in \mathcal{P}}{\text{minimize}}\ \widehat{R}(\Phi(\cdot; \theta)).$$

However, even if the empirical risk $\widehat{\mathcal{R}}(f)$ is convex w.r.t. f, the composition $\widehat{\mathcal{R}} \circ \Phi : \mathcal{P} \to \mathbb{R}$ is in general highly non-convex. This is inconvenient since gradient based methods may get stuck in undesirable local minima.

Note:-

- Our **goal** today is to prove that, during the training of a parametrized neural network, the network function $\Phi(\cdot, \theta)$ follows a descent along a kernel gradient, known as the **Neural Tangent Kernel** (NTK). It makes it possible to study the training as a gradient descent in the reproducing kernel Hilbert space generated by the NTK (instead of $\mathcal{P}$), on which the functional $\widehat{R}(\cdot)$ is convex.
- However, as a main feature when using Deep Neural Networks, the kernel evolves in time, due to the evolution of the parameters in the lower layers of the network.
- In the following lecture, we shall prove that when the parameters are randomly initialized, and the width of the layers tends to infinity, the Neural Tangent Kernel converges in probability to a deterministic kernel which stays constant during the training. This is known as the **kernel regime**.

  A more detailed presentation of the content of this lecture can be found in [18].

4.1 Kernel gradient

In this section, instead of considering $\mathcal{F}$ to be a family of NNs, we consider $\mathcal{F}$ to be a Reproducing Kernel Hilbert Space (RKHS), for some positive definite kernel $k(\cdot,\cdot) : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$. We shall denote by $\langle\cdot,\cdot\rangle_{\mathcal{F}}$ and $\|\cdot\|_{\mathcal{F}}$, the associated inner product and norm respectively.

The **goal** is to derive the gradient flow associated to the empirical risk $\widehat{\mathcal{R}}(\cdot)$ in $\mathcal{F}$. Then we will compare the expression to the gradient flow associated to a neural network, in order to eventually identify the so-called Neural Tangent Kernel.

Using the above notation, we can write the ERM problem as

$$\underset{f\in\mathcal{F}}{\text{minimize}}\ \widehat{\mathcal{R}}(f) := \frac{1}{m}\sum_{i=1}^{m}\mathcal{L}(f(x^{(i)}), y^{(i)}). \tag{4.1}$$

Given an initial function $f_0 \in \mathcal{F}$, we consider the trajectory in $\mathcal{F}$ given by the opposite direction to the gradient of $\widehat{\mathcal{R}}(\cdot)$ in $\mathcal{F}$. We denote this trajectory by the map

$$t \in [0,\infty) \longmapsto f(t,\cdot) \in \mathcal{F},$$

which is the solution to the differential equation

$$\begin{cases} \dfrac{d}{dt}f(t,\cdot) = -\nabla_{\mathcal{F}}\widehat{\mathcal{R}}(f(t,\cdot)) & t > 0 \\ f(0,\cdot) = f_0(\cdot) \end{cases}$$

Using the definition of $\widehat{\mathcal{R}}(\cdot)$ in (4.1) and the chain rule, we can write $\nabla_{\mathcal{F}}\widehat{\mathcal{R}}(f(t,\cdot))$ as

$$\nabla_{\mathcal{F}}\widehat{\mathcal{R}}(f(t,\cdot)) = \frac{1}{m}\sum_{i=1}^{m}\partial_1\mathcal{L}(f(t,x^{(i)}), y^{(i)})\nabla_{\mathcal{F}}f(t,x^{(i)}). \tag{4.2}$$

Here, $\partial_1\mathcal{L}(\hat{y}, y)$ denotes the derivative of the function $\mathcal{L}(\cdot) : \mathbb{R}\times\mathbb{R}\to\mathbb{R}$ with respect to the first variable.

Note:-

Using the reproducing property of $\mathcal{F}$, we have

$$f(t,x^{(i)}) = \langle f(t,\cdot), k(x^{(i)},\cdot)\rangle_{\mathcal{H}} \qquad \forall i \in \{1,\dots,m\},$$

which yields

$$\nabla_{\mathcal{F}}f(t,x^{(i)}) = \nabla_{\mathcal{F}}\left(\langle f(t,\cdot), k(x^{(i)},\cdot)\rangle_{\mathcal{F}}\right) = k(x^{(i)},\cdot). \tag{4.3}$$

Using this, we can write the gradient of $\widehat{\mathcal{R}}(\cdot)$ in $\mathcal{F}$ as follows.

Lemma 4.1. Let $\mathcal{F}$ be a RKHS with kernel $k(\cdot,\cdot)$. For a functional $\widehat{\mathcal{R}} : \mathcal{F} \to \mathbb{R}$

of the form (4.1), the gradient of C in $\mathcal{F}$ is given by

$$\nabla_{\mathcal{F}}\widehat{\mathcal{R}}(f(t,\cdot)) = \frac{1}{m}\sum_{i=1}^{m}\partial_1\mathcal{L}(f(t,x^{(i)}),y^{(i)})k(x^{(i)},\cdot). \tag{4.4}$$

Proof. The proof is simply the combination of (4.2) and (4.3). □

We can now derive the differential equation that describes the evolution of the cost $\widehat{\mathcal{R}}(\cdot)$, evaluated on the trajectory $f(t,\cdot)$.

$$\begin{aligned}
\frac{d}{dt}\widehat{\mathcal{R}}(f(t,\cdot)) &= \frac{1}{m}\sum_{i=1}^{m}\partial_1\mathcal{L}(f(t,x^{(i)}),y^{(i)})\frac{d}{dt}f(t,x^{(i)}) \\
&= \frac{1}{m}\sum_{i=1}^{m}\partial_1\mathcal{L}(f(t,x^{(i)}),y^{(i)})\left\langle \frac{d}{dt}f(t,\cdot), k(x^{(i)},\cdot)\right\rangle_{\mathcal{F}} \\
&= \Big\langle \underbrace{\frac{1}{m}\sum_{i=1}^{m}\partial_1\mathcal{L}(f(t,x^{(i)}),y^{(i)})k(x^{(i)},\cdot)}_{\nabla_{\mathcal{F}}C(f(t,\cdot))}, \underbrace{\frac{d}{dt}f(t,\cdot)}_{-\nabla_{\mathcal{F}}\widehat{\mathcal{R}}(f(t,\cdot))}\Big\rangle_{\mathcal{F}}.
\end{aligned}$$

Here we used the reproducing property with the function $\frac{d}{dt}f(t,\cdot)$. The above computation yields

$$\begin{cases}\dfrac{d}{dt}\widehat{\mathcal{R}}(f(t,\cdot)) = -\left\|\nabla_{\mathcal{F}}\widehat{\mathcal{R}}(f(t,\cdot))\right\|_{\mathcal{F}}^2 & t>0\\ \widehat{\mathcal{R}}(f(0,\cdot)) = \widehat{\mathcal{R}}(f_0).\end{cases}$$

Note:-

Note that this implies that the cost is monotonically decreasing in t and the stationary points are solutions to

$$\nabla_{\mathcal{F}}\widehat{\mathcal{R}}(f) = 0 \qquad \text{in } \mathcal{F}.$$

In the case $\widehat{\mathcal{R}}(\cdot)$ is convex, as it would be the case if we consider $\mathcal{L}$ to be the squared error, then any solution to this equation is a global minimizer of $\widehat{\mathcal{R}}(\cdot)$. Hence, we can conclude that the gradient flow converges to a global minimizer of $\widehat{\mathcal{R}}$.

4.2 Neural Tangent Kernel

When training Deep Neural Networks with gradient descent, the situation is very similar. The network function $\Phi_\theta(\cdot)$ evolves in the opposite direction to a kernel gradient, known as the *Neural Tangent Kernel* NTK. However, in contrast to the previous case, the realization of a Deep NN is not linear, and therefore, the NTK depends on the parameters. This implies that the kernel is random at the initialization and varies during training.

Let us consider a FCNN with L layers and architecture $N = (d, N_1, \dots, N_{L-1}, 1)$. Given the parameters $\theta \in \mathcal{P}$ for some parameter space $\mathcal{P} \subset \mathbb{R}^{\#P}$, where $\#P = \sum_{l=1}^{L} N_l(N_{l-1}+1)$ is the number of parameters, we denote the NN function as

$$\Phi(\cdot,\theta) : \mathbb{R}^d \longrightarrow \mathbb{R}$$

Given a training dataset $\{(x^{(i)}, y^{(i)})\}_{i=1}^m$, we consider the (unregularized) risk minimization problem as learning algorithm, i.e.

$$\underset{\theta \in \mathbb{R}^{P(N)}}{\text{minimize}} \ \widehat{\mathcal{R}}(\Phi(\cdot, \theta)) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}\left(\Phi(x^{(i)}, \theta), y^{(i)}\right).$$

Let us assume that we train the NN using a time-continuous version of gradient descent. The parameters θ evolve according to the differential equation

$$\theta'(t) = -\nabla_\theta \widehat{\mathcal{R}}\left(\Phi(\cdot, \theta(t))\right), \qquad \theta(0) = \theta_0,$$

where

$$\nabla_\theta C\left(\Phi(\cdot, \theta(t))\right) = \frac{1}{m} \sum_{i=1}^{m} \partial_1 \mathcal{L}\left(\Phi(x^{(i)}, \theta(t)), y^{(i)}\right) \nabla_\theta \Phi(x^{(i)}, \theta(t))$$

and the initialization parameters $\theta_0 \in \mathbb{R}^{P(N)}$ are typically randomly selected.

The evolution of the function $\Phi(\cdot, \theta(t))$ in time is then given by

$$\begin{aligned} \frac{d}{dt}\Phi(\cdot, \theta(t)) &= -\nabla_\theta \Phi(\cdot, \theta(t)) \cdot \theta'(t) \\ &= -\frac{1}{m} \sum_{i=1}^{m} \partial_1 \mathcal{L}\left(\Phi(x^{(i)}, \theta(t)), y^{(i)}\right) \nabla_\theta \Phi(\cdot, \theta(t)) \cdot \nabla_\theta \Phi(x^{(i)}, \theta(t)), \end{aligned}$$

that we can write as

$$\frac{d}{dt}\Phi(\cdot, \theta(t)) = -\frac{1}{m} \sum_{i=1}^{m} \partial_1 \mathcal{L}\left(\Phi(x^{(i)}, \theta(t)), y^{(i)}\right) k(\cdot, x^{(i)}; \theta(t)), \tag{4.5}$$

where $k(\cdot, \cdot; \theta) : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is the parametrized definite positive kernel given by

$$k(x, x'; \theta) = \nabla_\theta \Phi(x, \theta) \cdot \nabla_\theta \Phi(x', \theta).$$

In view of the expression of the kernel gradient in (4.4), we can write (4.5) as

$$\frac{d}{dt}\Phi(\cdot, \theta(t)) = -\nabla_{\mathcal{F}_\theta} \widehat{\mathcal{R}}(\Phi(\cdot, \theta(t))),$$

where $\nabla_{\mathcal{F}_\theta} \widehat{\mathcal{R}}(\cdot)$ denotes the gradient of the functional $\widehat{\mathcal{R}}(\cdot)$ in the RKHS $\mathcal{F}_\theta$ associated to the kernel $k(\cdot, \cdot; \theta)$.

Definition 4.14

For a neural network $\Phi(\cdot, \theta) : \mathbb{R}^d \to \mathbb{R}$ with parameter $\theta \in \mathbb{R}^{P(N)}$, we define the **Neural Tangent Kernel** as the θ-parametrized positive kernel

$$k(x, x'; \theta) = \nabla_\theta \Phi(x, \theta) \cdot \nabla_\theta \Phi(x', \theta).$$

The cost $\widehat{\mathcal{R}}(\Phi(\cdot,\theta(t)))$ evolves according to the following differential equation

$$\begin{aligned}
\frac{d}{dt}\widehat{\mathcal{R}}(\Phi(\cdot,\theta(t))) &= \frac{1}{m}\sum_{j=1}^{m}\frac{d}{dt}\mathcal{L}\left(\Phi(x^{(j)},\theta(t)),y^{(j)}\right)\\
&= \frac{1}{m}\sum_{j=1}^{m}\partial_1\mathcal{L}\left(\Phi(x^{(j)},\theta(t)),y^{(j)}\right)\frac{d}{dt}\Phi(x^{(j)},\theta(t))\\
&= -\frac{1}{m^2}\sum_{j=1}^{m}\sum_{i=1}^{m}\partial_1\mathcal{L}\left(\Phi(x^{(j)},\theta(t)),y^{(j)}\right)\partial_1\mathcal{L}\left(\Phi(x^{(i)},\theta(t)),y^{(i)}\right)k(x^{(j)},x^{(i)};\theta(t))\\
&= -\left\|\frac{1}{m}\sum_{i=1}^{m}\partial_1\mathcal{L}\left(\Phi(x^{(i)},\theta(t)),y^{(i)}\right)k(\cdot,x^{(i)};\theta(t))\right\|^2_{\mathcal{F}_\theta}\\
&= -\left\|\nabla_{\mathcal{F}_\theta}\widehat{\mathcal{R}}(\Phi(\cdot,\theta(t)))\right\|^2_{\mathcal{F}_\theta},
\end{aligned}$$

where $\|\cdot\|_{\mathcal{F}_\theta}$ denotes the norm of the RKHS associated to the kernel $k(\cdot,\cdot;\theta)$.

Conclusions:

- The time-continuous gradient descent of the empirical risk, with respect to the parameters of a Deep Neural Network, corresponds to the gradient flow in a parameterized RKHS, denoted by $\mathcal{F}_\theta$, the kernel of which is given by

$$k(x,x';\theta(t)) = \nabla_\theta\Phi(x,\theta(t))\cdot\nabla_\theta\Phi(x',\theta(t)).$$

- The cost functional $\widehat{\mathcal{R}}(\Phi(\cdot,\theta))$ might not be convex with respect to the parameters θ, however, $\widehat{\mathcal{R}}(\cdot)$ is convex in $\mathcal{F}_\theta$ for any $\theta\in\mathcal{P}$, provided the loss $\mathcal{L}(\hat{y},y)$ is convex with respect to $\hat{y}$.
- The cost is monotonically decreasing, which means that, if $\widehat{\mathcal{R}}(\cdot)$ is bounded from below, it will eventually converge as $t\to\infty$. This implies

$$\left\|\nabla_{\mathcal{F}_\theta}\widehat{\mathcal{R}}(\Phi(\cdot,\theta(t)))\right\|^2_{\mathcal{F}_\theta}\longrightarrow 0 \qquad \text{as} \quad t\to\infty.$$

  This means that for t large, $\Phi(\cdot,\theta(t))$ is close to the global minimizer of $C(\cdot)$ in $\mathcal{F}_\theta$. However, $\mathcal{F}_\theta$, and hence this minimizer depends on $\theta(t)$, which might be different for t large depending on the initialization. Actually, we cannot in principle guarantee that the neural tangent kernel $k(x,x';\theta(t))$ converges as $t\to\infty$.
- In the following lecture we shall prove that in the overparametrized regime, the NTK is close to a deterministic kernel, which is constant during training. This allows one to prove global convergence to the minimizier in this deterministic RKHS. That being said, it's worth noting that this RKHS depends on the random variable used to initialize the parameters of the NN.

Feb 5 2024 Mon (12:00-13:00)

Lecture 5: Kernel Regime

Let us consider a Fully–Connected Neural Network FCNN

$$\Phi(\cdot, \theta) : \mathbb{R}^d \longrightarrow \mathbb{R}$$

with parameter $\theta \in \mathcal{P} \subset \mathbb{R}^{P(N)}$, where $P(N)$ is the number of parameters in the NN.

In this lecture we describe how, when the parameters in θ are i.i.d. samples of certain probability distribution, the Neural Tangent Kernel, defined in the previous lecture as

$$k(x, x'; \theta) = \nabla_\theta \Phi(x, \theta) \cdot \nabla_\theta \Phi(x', \theta),$$

converges in the infinite width limit, to a deterministic kernel, which is constant during training. This result was proven in [18]. The first step is to prove that, also in the infinite width limit, Deep NNs with randomly initialized parameters converge in probability to a Gaussian process [11, 22].

Note:-

Contrary to the finite–parameter case that we discussed in the previous lecture, in which the neural tangent kernel NTK is randomly initialized and evolves during the gradient descent, in the infinite width limit, the NTK is deterministic at the initialization, and does not evolve during training. This in turn implies global convergence of the gradient flow to the global minimizer in the RKHS associated to the limiting kernel. **Note however that the limiting kernel depends on the probability distribution used for the initialization of the parameters.**

5.1 Deep Neural Networks and Gaussian processes

Definition 5.15

Given two functions

$$\mu : \mathbb{R}^d \to \mathbb{R} \quad \text{and} \quad k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R},$$

where $k(\cdot, \cdot)$ is a positive definite kernel, the **Gaussian Process** F, with mean $\mu(\cdot)$ and covariance $k(\cdot, \cdot)$ is a stochastic process such that, for any finite set of points $X = (x^{(1)}, \dots, x^{(m)}) \in (\mathbb{R}^d)^m$, the Gaussian Process F evaluated at those points follows a multivariate normal distribution

$$\big(F(x^{(1)}), \dots, F(x^{(m)})\big) \sim N(\mu_X, \Sigma_X)$$

with parameters

$$\mu_X = \Big(\mu(x^{(1)}), \dots, \mu(x^{(m)})\Big) \quad \text{and} \quad \Sigma_X = \begin{bmatrix} k(x^{(1)}, x^{(1)}) & \cdots & k(x^{(1)}, x^{(m)}) \\ \vdots & \ddots & \vdots \\ k(x^{(m)}, x^{(1)}) & \cdots & k(x^{(m)}, x^{(m)}) \end{bmatrix}$$

When the function μ is identically zero, we say that the Gaussian process is centred.

One-layer Neural Network

Let us start with the simplest Neural Network possible, i.e. a single layer with no activation. This is in fact a linear model that we can write as a function

$$\begin{array}{rccc} \Phi: & \mathbb{R}^d & \longrightarrow & \mathbb{R}^{N_1} \\ & x & \longmapsto & Wx+b \end{array}$$

for some matrix $W \in \mathbb{R}^{N_1 \times d}$ and $b \in \mathbb{R}^{N_1}$, where N_1 is the width of the layer.

We now consider that weights and the biases are randomly chosen as follows: every row $W_i \in \mathbb{R}^d$ in W and every entry $b_i \in \mathbb{R}$ in b are normal distributions of the form

$$W_i \sim N(0, I_d) \quad \text{and} \quad b_i \sim N(0, \tau_b^2).$$

Here, I_d is the $d \times d$ identity matrix, meaning that the entries in W_i are uncorrelated.

Note:-

We observe that the entries $\phi_i(x)$ of the vector

$$\Phi(x) = (\phi_1(x), \ldots, \phi_{N_1}(x)) \in \mathbb{R}^{N_1}$$

are i.i.d. samples of the same Normal distribution

$$x \longmapsto W_i \cdot x + b_i \sim N\left(0, \|x\|^2 + \tau_b^2\right).$$

For any $x_1, x_2 \in \mathbb{R}^d$, we can compute the covariance of the normally distributed random variables $W_i x_1$ and $W_i x_2$ as

$$\mathrm{Cov}(W_i \cdot x, W_i \cdot x') = \mathbb{E}\big[(W_i \cdot x)(W_i \cdot x')\big] = x \cdot x'.$$

Hence, each entry $\phi_i(\cdot)$ is a centred Gaussian process with covariance

$$k_0(x, x') = x \cdot x' + \tau_b^2.$$

Let us denote this Gaussian process by $\Phi_0(\cdot)$.

Two-layer Neural Network

Let us now consider a two layer Neural Network of the form

$$\Phi^{(1)}(x) = W^{(1)} \sigma\left(\Phi^{(0)}(x)\right) + b^{(1)},$$

where $\Phi^{(0)}(x)$ is a one-layer NN as before, and $W^{(1)} \in \mathbb{R}^{N_2 \times N_1}$ and $b^{(1)} \in \mathbb{R}^{N_2}$. Here, N_2 is the width of the second layer.

Important: Keep in mind that we just proved that $\Phi^{(0)}(x) \in \mathbb{R}^{N_1}$ is a vector of i.i.d. random variables

$$\Phi^{(0)}(x) = \left(\phi_1^{(0)}(x), \ldots, \phi_{N_1}^{(0)}(x)\right),$$

where $\phi_i^{(0)}(x) \sim N(0, \tau_b^2)$.

As before we consider that every row $W_i^{(1)}$ in the matrix of weights $W^{(1)}$, and every entry $b_i^{(1)}$ in the vector of biases $b^{(2)}$ are given by Normal distributions as

$$W_i^{(1)} \sim N\left(0, \frac{1}{N_1} I_{N_1}\right) \quad \text{and} \quad b_i^{(1)} \sim N(0, \tau_b^2).$$

Now, each entry in the vector

$$\Phi^{(1)}(x) = \left(\phi_1^{(1)}(x), \dots, \phi_{N_2}^{(1)}(x)\right)$$

can be written as

$$\phi_i^{(1)}(x) = W_i^{(1)}\sigma(\Phi^{(0)}(x)) + b_i^{(1)} = \sum_{j=1}^{N_1} W_{ij}^{(1)}\sigma(\phi_j^{(0)}(x)) + b_i^{(1)},$$

where $\phi_j^{(1)}(x)$ are N_1 i.i.d. Gaussian processes identically distributed.

We can write

$$Z = \sum_{j=1}^{N_1} W_{ij}^{(1)}\sigma(\phi_j^{(0)}(x)) = \frac{\sqrt{N_1}}{N_1}\sum_{j=1}^{N_1}\sqrt{N_1}W_{ij}^{(1)}\sigma(\phi_j^{(0)}(x)) = \frac{\sqrt{N_1}}{N_1}\sum_{j=1}^{N_1} Z_j.$$

Since the variance of the coefficients $W_{ij}^{(2)}$ is $1/N_1$, the variance of $\sqrt{N_1}W_{ij}^{(1)}$ is 1. This implies

$$\mathrm{Var}(Z_j) = \mathbb{E}[\sigma(\Phi_0(x))^2].$$

Also, the fact that the random variables $W_{ij}^{(1)}$ and $\sigma(\phi_j^{(0)}(x))$ are independent implies that $\mathbb{E}(Z_j) = 0$.

Hence, as $N_1 \to \infty$, by the central limit theorem, the random variable Z converges to a Normal distribution

$$N\left(0, \mathbb{E}[\sigma(\Phi_0(x))^2]\right).$$

Hence, for each $x \in \mathbb{R}^d$, in the limit $N_1 \to \infty$, we have

$$\phi_i^{(1)}(x) \sim N\left(0, \mathbb{E}[\sigma(\Phi_0(x))^2] + \tau_b^2\right)$$

Arguing similarly, we deduce that, for any $x, x' \in \mathbb{R}^d$, the covariance of the limit random variables $\phi_i^{(1)}(x)$ and $\phi_i^{(1)}(x')$ is given by

$$\mathrm{Cov}(\phi_i^{(1)}(x), \phi_i^{(1)}(x')) = \mathbb{E}[\sigma(\Phi_0(x))\sigma(\Phi_0(x'))] + \tau_b^2,$$

where the expectation is taken w.r.t.the Gaussian process $\Phi_0(\cdot)$.

Note:-

From the above arguments, we conclude that, in the infinite width limit $N_1 \to \infty$, the output of the two layer neural network

$$\Phi^{(1)}(x) = \left(\phi_i^{(1)}(x), \dots, \phi_i^{(1)}(x)\right) = W^{(1)}\sigma\left(\Phi^{(0)}(x)\right) + b^{(1)},$$

is a centred Gaussian process with covariance function

$$k_1(x, x') = \mathbb{E}[\sigma(\Phi_0(x))\sigma(\Phi_0(x'))] + \tau_b^2.$$

We denote the Gaussian process associated to this kernel by $\Phi_1(\cdot)$.

Deep NNs

We can now argue recursively using the same argument as in the two layer case. For any $l \in \{1, \dots, L-1\}$, the output of the $(l+1)$-th layer is an N_{l+1}-dimensional vector

$$\Phi^{(l)}(x) = \left(\phi_1^{(l)}(x), \dots, \phi_{N_{l+1}}^{(l)}(x)\right),$$

where each entry is given by

$$\phi_i^{(l)}(x) = \sum_{j=1}^{N_l} W_{ij}^{(l)} \sigma\left(\phi_j^{(l-1)}(x)\right) + b_i^{(l)}.$$

Under the recursive assumption that each $\phi_j^{(l-1)}(\cdot)$ is an i.i.d. Gaussian process $\Phi_{l-1}(\cdot)$, and that the parameters $W_{ij}^{(l)} \in \mathbb{R}$ and $b_i^{(l)} \in \mathbb{R}$ are i.i.d. samples from

$$W_{ij}^{(l)} \sim N\left(0, \frac{1}{N_l}\right) \qquad \text{and} \qquad b_i^{(l)} \sim N(0, \tau_b^2),$$

we can use the law of large numbers again to deduce that as $N_l \to +\infty$, each $\phi_i^l(\cdot)$ converges in probability to a Gaussian process $\Phi_l(\cdot)$ with covariance

$$k_l(x, x') = \mathbb{E}[\sigma(\Phi_{l-1}(x))\sigma(\Phi_{l-1}(x'))] + \tau_b^2.$$

Applying the same argument recursively, we can deduce that the output of the NN, which is a one-dimensional random variable of the form

$$\Phi(x, \theta) = \sum_{i=1}^{N_L} W_i^{(L)} \sigma\left(\Phi_{L-1}(x)\right) + b^L,$$

where $\Phi_{L-1}(\cdot)$ is a Gaussian process and

$$W_i^{(L)} \sim N\left(0, \frac{1}{N_L}\right) \qquad \text{and} \qquad b^{(L)} \sim N(0, \tau_b^2),$$

converges in probability, as $N_L \to \infty$, to a Gaussian process with kernel

$$k_L(x, x') = \mathbb{E}[\sigma(\Phi_{L-1}(x))\sigma(\Phi_{L-1}(x'))] + \tau_b^2.$$

Theorem 5.1. For a FCNN of depth L, with Lipschitz nonlinearity and random parameters $W_{ij}^{(l)} \sim N(0, 1/N_l)$ and $b_i^{(l)} \sim N(0, \tau_b^2)$, the NN function $\Phi(\cdot, \theta)$ converges in probability as $N_1, N_2, \dots, N_L \to +\infty$ to a centred Gaussian process with kernel $k_L(x, x')$ given recursively as

$$\begin{cases} k_0(x, x') = x \cdot x' + \tau_b^2 \\ k_l(x, x') = \mathbb{E}[\sigma(\Phi_{l-1}(x))\sigma(\Phi_{l-1}(x')] + \tau_b^2, \end{cases}$$

where the expectation is taken w.r.t. the centred Gaussian Process $\Phi_{l-1}(\cdot)$ associated to the kernel $k_{l-1}(\cdot, \cdot)$.

See [22, 11] for a complete proof of the above result.

5.2 Convergence of the Neural Tangent Kernel

Using a similar recursive argument, it is proven in [18] that the NTK with Gaussian i.i.d. parameters converges, as the width goes to infinity, to a deterministic positive definite kernel.

Theorem 5.2. For a FCNN of depth L, with Lipschitz activation function and random parameters $W_{ij}^{(l)} \sim N(0, 1/N_l)$ and $b_i^{(l)} \sim N(0, \tau_b^2)$, the NTK of the NN $\Phi(\cdot, \theta)$, defined as

$$\Theta(x, x'; \theta) = \nabla_\theta \Phi(x, \theta) \cdot \nabla_\theta \Phi(x', \theta),$$

converges in probability, as $N_1, N_2, \ldots, N_L \to +\infty$, to a deterministic limiting kernel $\Theta_\infty^{(L)}(x, x')$ defined recursively as

$$\begin{cases} \Theta_\infty^{(0)} = k_0(x, x') \\ \Theta_\infty^{(l+1)}(x, x') = \Theta_\infty^{(l)}(x, x')\dot{k}_{l+1}(x, x') + k_{l+1}(x, x'), \end{cases}$$

where $k_{l+l}(x, x')$ is the limiting kernel defined in Theorem 5.1, and

$$\dot{k}_{l+1}(x, x') = \mathbb{E}[\dot{\sigma}(\Phi_l(x))\dot{\sigma}(\Phi_l(x'))],$$

where the expectation is taken with respect to the Gaussian process $\Phi_l(\cdot)$ with kernel $k_l(\cdot, \cdot)$.

Feb 7 2024 Wed (12:00-13:00)

Lecture 6: The Optimisation Landscape

Once more, let us recall from previous lectures that, in order to address any learning task, one needs to provide three elements:

1. The hypothesis set $\mathcal{F} \subset \mathcal{M}(\mathcal{X}, \mathcal{Y})$.
2. The loss function $\mathcal{L} : \mathcal{M}(\mathcal{X}, \mathcal{Y}) \times \mathcal{Z} \to \mathbb{R}$.
3. The learning algorithm $\mathcal{A} : \bigcup_{m\in\mathbb{N}} \mathcal{Z}^m \to \mathcal{F}$.

Focusing on the third part, remember from previous lectures that the learning algorithm in principle comes down to a solving a minimisation problem, e.g., minimising empirical risk on a data set $s = \{z^{(i)}\}_{i=1}^m \in \mathcal{Z}^m$

$$\mathcal{Z}^m \ni s \mapsto \mathcal{A}(s) \in \operatorname*{arg\,min}_{f\in\mathcal{F}} \ \widehat{R}_s(f) := \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}(f, z^{(i)}). \tag{6.1}$$

This lecture will be the first of two lectures on optimisation in deep learning, where we will focus on actually solving the optimisation problem 6.1. We refer the reader to [6, Section 5.1] as reference material. During these two lectures on optimisation we will discuss some existing approaches that have attempted to explain the success of *stochastic gradient descent* (SGD). That is, in this lecture we will talk about the behaviour of the *optimisation landscape* and the next lecture will be on *lazy training* as a consequence of the provable convergence of gradient descent, which is the finite dimensional counterpart of the result we saw in the previous lecture.

6.1 Optimisation landscape analysis: paths and level sets

If we assume that 6.1 is an arbitrary non-smooth and non-convex optimisation problem, we do not expect that there is any hope of finding a good minimiser. In particular, saddle points might slow down convergence arbitrarily badly and spurious local minimisers trap the iterates. However, in practice SGD performs really well, which cannot be explained from classical optimisation theory. So something else might be going on.

In the following we will explore several simple, but insightful cases of the topology of the loss landscape. It is worth mentioning that there is a lot of mythology around the optimisation landscape. In particular, we will be concerned with answering the following question:

Question 1

"Are there paths of decreasing loss in the network parameter landscape to global minimisers?".

In this part we will consider some special cases and see to what extend this myth holds up. For that, let us consider regression, i.e., $\mathcal{Z} = \mathbb{R}^d \times \mathbb{R}$ and $\mathcal{L}(f, (x, y)) = (f(x) - y)^2$. Next, for the following it will be useful to define – for a given neural network architecture – $\widehat{r} : \mathcal{P} \to \mathbb{R}$ from the empirical loss

$$\widehat{r}(\theta) := \widehat{R}_s(\Phi(\cdot, \theta)) = \frac{1}{m}\sum_{i=1}^{m} (\Phi(x^{(i)}, \theta) - y^{(i)})^2, \tag{6.2}$$

where $\mathcal{P} \subset \mathbb{R}^{P(N)}$ and $P(N)$ is the number of parameters of the NN.

We can characterize bad or spurious local minima from the topology of sub-level sets.

Definition 6.16

The *sub-level sets* of a function $F : \mathbb{R}^N \to \mathbb{R}$ are defined as

$$\Omega_F(\lambda) := \{\theta \in \mathbb{R}^N \mid F(\theta) \leq \lambda\}. \tag{6.3}$$

Theorem 6.1. Let $F : \mathbb{R}^N \to \mathbb{R}$ be any function. If $\Omega_F(\lambda)$ is connected for all λ then every local minimum of F is a global minimum.

Example. Consider the linear neural network $\Phi(\cdot, \theta) : \mathbb{R}^d \to \mathbb{R}$ given by

$$\Phi(x; \theta) = W_L \dots W_1 x, \quad \theta = (W_1, \dots, W_L), \tag{6.4}$$

where $W_l \in \mathbb{R}^{N_l \times N_{l-1}}$ for $l = 1, \dots, L$ with $N_0 = d$ and $N_L = 1$, and the corresponding empirical risk $\widehat{r} : \mathcal{P} \to \mathbb{R}$ under this network as in 6.2.

If $N_l > 1$ for $l = 1, \dots, L-1$, it can be shown[a] that $\Omega_{\widehat{r}}(\lambda)$ is connected for all λ.

[a] cf. [12]

In the example above, there are no poor local minima. Moreover, any θ can be connected to the lowest energy level through a path of decreasing loss.

However, in the case of deep learning with non-linear activation, the sub-level sets of the loss landscape are generally *not connected*. However, for *some combinations of neural network architecture and loss function*, it can be shown that there exist paths where the risk increases only by a small amount $\varepsilon > 0$.

We will consider one such following example.

Theorem 6.2. Let $(x^{(i)}, y^{(i)})_{i=1}^m \in (\mathbb{R}^d \times \mathbb{R})^m$ be a data set. Consider the function $\widehat{r} : \mathbb{R}^{n \times d} \times \mathbb{R}^n \to \mathbb{R}$ given by

$$\widehat{r}(\theta) := \frac{1}{m} \sum_{i=1}^{m} (\Phi(x^{(i)}, \theta) - y^{(i)})^2, \tag{6.5}$$

where $\Phi(\cdot; \theta) : \mathbb{R}^d \to \mathbb{R}$ is a two-layer neural network of the form

$$\Phi(x; \theta) := W_2 \cdot \sigma(W_1 x), \quad \theta = (W_1, W_2), \ \sigma(x) := \max\{0, x\}, \tag{6.6}$$

with $W_1 \in \mathbb{R}^{n \times d}$, $W_2 \in \mathbb{R}^n$.

Then, for any $\theta^A, \theta^B \in \mathbb{R}^{n \times d} \times \mathbb{R}^n$, there is an almost convex continuous path γ with $\gamma(0) = \theta^A$, $\gamma(1) = \theta^B$, i.e., $t \mapsto \widehat{r}(\gamma(t))$ is convex on $t \in [t_n^A, t_n^B] \subset [0, 1]$ for $0 < t_n^A < t_n^B < 1$.

Proof. (Sketch)

Note that we may assume that all the rows of W_1 are normalized, i.e., elements of the unit sphere. Next, define the matrix $\bar{W}_1 \in \mathbb{R}^{n\times d}$

$$(\bar{W}_1)_{j,:} := \begin{cases} (W_1^A)_{j,:} & j \in [n/2] \\ (W_1^B)_{j,:} & j \in [n]/[n/2] \end{cases}, \tag{6.7}$$

and the linear space

$$\mathcal{F}_{\bar{W}_1} := \{\theta := (W_1, W_2) \in \mathbb{R}^{n\times d} \times \mathbb{R}^n \mid W_1 = \bar{W}_1\}. \tag{6.8}$$

It suffices to construct a path

1. from θ^A to some $\bar{\theta}^A \in \mathcal{F}_{\bar{W}_1}$
2. from $\bar{\theta}^A \in \mathcal{F}_{\bar{W}_1}$ to $\bar{\theta}^B \in \mathcal{F}_{\bar{W}_1}$
3. from $\bar{\theta}^B \in \mathcal{F}_{\bar{W}_1}$ to θ^B

with empirical risk behaving in the claimed way, i.e., convexity in (2). We also want to get insight what happens in the regions of of non-convexity (1) and (3).

(1) First step is to construct the matrix $\tilde{W}_1^A \in \mathbb{R}^{n\times d}$ whose rows are defined as

$$(\tilde{W}_1^A)_{j,:} := \tilde{w}_j^A \in \operatorname*{arg\,min}_{w\in\{(W_1^A)_{k,:}\}_{k=1}^{n/2}} \|w - (W_1^A)_{j,:}\|_2, \tag{6.9}$$

and define $\varepsilon_n^A > 0$ as

$$\varepsilon_n^A := \max_{j\in[n]} \inf_{w\in\{(W_1^A)_{k,:}\}_{k=1}^{n/2}} \|w - (W_1^A)_{j,:}\|_2. \tag{6.10}$$

We have

$$\max_{t\in[0,1]} \widehat{r}((t\tilde{W}_1^A + (1-t)W_1^A, W_2^A)) \in \mathcal{O}(\varepsilon_n^A), \tag{6.11}$$

because σ is Lipschitz, W_2^A is fixed and $\hat{y} \mapsto (\hat{y} - y)^2$ is locally Lipschitz. We will call the intermediate point $\tilde{\theta}^A := (\tilde{W}_1^A, W_2^A)$.

Next, we can find $\bar{W}_2^A$ such that $(\bar{W}_2^A)_j = 0$ for $j \in [n]/[n/2]$ and have

$$\Phi(\cdot, (\tilde{W}_1^A, W_2^A)) = \Phi(\cdot, (\tilde{W}_1^A, t\bar{W}_2^A + (1-t)W_2^A)), \quad t \in [0,1]. \tag{6.12}$$

Finally, by our choice of $\bar{W}_2^A$ we have

$$\Phi(\cdot, (\tilde{W}_1^A, \bar{W}_2^A)) = \Phi(\cdot, t(\bar{W}_1 + (1-t)\tilde{W}_1^A, \bar{W}_2^A)), \quad t \in [0,1]. \tag{6.13}$$

So to conclude, we have a path

$$\theta^A = (W_1^A, W_2^A) \to (\tilde{W}_1^A, W_2^A) \to (\tilde{W}_1^A, \bar{W}_2^A) \to (\bar{W}_1, \bar{W}_2^A) =: \bar{\theta}^A, \tag{6.14}$$

on which the total loss is always less than $\widehat{r}(\theta^A) + C^A\varepsilon_n^A$ for some $C^A > 0$.

(3) Similarly, we can construct a path from θ^B to an analogously constructed $\bar{\theta}^B = (\bar{W}_1, \bar{W}_2^B)$ for some $\bar{W}_2^B \in \mathbb{R}^n$ the total loss is always less than $\widehat{r}(\theta^B) + C^B\varepsilon_n^B$, where ε_n^B is analogously defined as ε_n^A and $C^B > 0$.

(2) It remains to connect $\bar{\theta}^A$ and $\bar{\theta}^B$. As both have the same $W_1 = \bar{W}_1$ and the mapping $W_2 \mapsto \widehat{r}((\bar{W}_1, W_2))$ is convex. □

The proof can easily be extended to more general losses than the regression loss $\mathcal{L}(f,(x,y)) = (f(x)-y)^2$.

Exercise 6.1. Show that the result above holds for any loss $\mathcal{L} : \mathcal{F} \times (\mathbb{R}^d \times \mathbb{R}) \to \mathbb{R}$ such that for fixed $(x,y) \in \mathbb{R}^d \times \mathbb{R}$ the mapping $f \mapsto \mathcal{L}(f,(x,y))$ is convex with Lipschitz gradients.

Note:-

If θ^A is randomly initialized and θ^B is the global minimiser, we expect that

- the first contribution diminishes for wide enough networks as it can be shown that ε_n^A scales as $\mathcal{O}(n^{-\frac{1}{(d-1)}})$ if $x^{(i)} \neq x^{(j)}$ for all $i \neq j \in [m]$
- the loss along the second part of the curve is decreasing due to the convexity of $\widehat{r}$ and assuming that $\bar{\theta}^B \approx \theta^B$

In other words, for wide enough randomly initialized networks, there is a decreasing path to the global minimizer (up to $\mathcal{O}(\varepsilon_n^B)$) in the close vicinity of the initialisation.

Feb 12 2024 Mon (12:00-13:00)

Lecture 7: Lazy Training and Linear Convergence

In the previous, we discussed that training neural networks with gradient-based methods performs better than is to be expected from the classical optimisation literature. We have seen that one way of investigating this discrepancy is to look at the optimisation landscape. We discussed several scenarios:

- In general, the optimisation landscape of a neural network will have various spurious local minimisers and gradient-based methods are not expected to converge to the global minimiser
- For a two-layer neural network and a convex loss there exist almost convex paths to the global minimiser
- For a randomly initialized network the optimisation landscape becomes increasingly nicer as the width of the network increases. Also, the distance to a path that is decreasing towards the global minimiser vanishes

Today we will continue this discussion and ask a natural follow up question. We will do so through an adaptation of the strategy presented in [6, Section 5.2].

Question 2

"Do gradient-based algorithms in the overparametrized network regime actually overcome the initial bump and take a path to the global minimiser?".

Throughout this lecture we will again assume the simple example of the two-layer neural network $\Phi(\cdot;\theta) : \mathbb{R}^d \to \mathbb{R}$ given by

$$\Phi(x;\theta) = W_2 \cdot \sigma(W_1 x), \quad \theta = (W_1, W_2)\,, \tag{7.1}$$

where $W_1 \in \mathbb{R}^{n\times d}$, $W_2 \in \mathbb{R}^n$ and where $\sigma : \mathbb{R} \to \mathbb{R}$ is *smooth* (non-polynomial). We will also assume the empirical risk $\widehat{r} : \mathbb{R}^{n\times d} \times \mathbb{R}^n \to \mathbb{R}$ given by

$$\widehat{r}(\theta) := \widehat{R}_s(\Phi(\cdot,\theta)) = \frac{1}{m}\sum_{i=1}^{m}(\Phi(x^{(i)},\theta) - y^{(i)})^2. \tag{7.2}$$

To get some intuition for the question above consider Figure 3, where several networks have been trained with different n. Now, the right-most plot in Figure 3 suggests that this answer to this lecture's question is a resounding yes. Moreover, from the left-most plot we also see that the distance to the initial weights gets smaller for wider networks.

In this lecture we will try to understand this behaviour. In particular, we will try to proof that when training highly overparametrized two-layer NNs of the form 7.1 through minimizing 7.2 through GD, the parameters

- barely change during training, if initialized randomly
- converges linearly to a zero-loss configuration

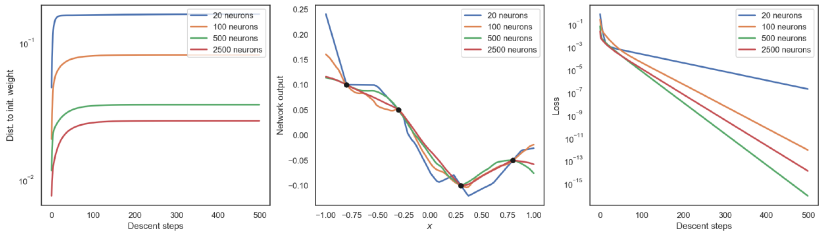

Figure 3: Four two-layer networks that are initialized with random weights, but with different layer widths were trained by gradient descent to fit four points that are shown in the middle figure as black dots.

7.1 Control of the gradient at random initializations

Our first step is to understand what the gradient $\nabla_\theta \widehat{r}(\Theta)$ looks like with high probability over the initialization $\Theta = (\mathbf{W}_1; \mathbf{W}_2) \sim \mathcal{N}(0,1)^{n\times d} \otimes \mathcal{N}(0,1/n)^n$.

We will give a sketch of the proof of the following theorem

Lemma 7.1. Let $(x^{(i)}, y^{(i)})_{i=1}^m \in (\mathbb{R}^d \times \mathbb{R})^m$ be a data set with $x^{(i)} \neq x^{(j)}$ for all $i \neq j \in [m]$. Consider the function $\widehat{r} : \mathbb{R}^{n\times d} \times \mathbb{R}^n \to \mathbb{R}$ given by

$$\widehat{r}(\theta) := \frac{1}{m}\sum_{i=1}^m (\Phi(x^{(i)}, \theta) - y^{(i)})^2, \tag{7.3}$$

where $\Phi(\cdot, \theta) : \mathbb{R}^d \to \mathbb{R}$ is a two-layer neural network of the form

$$\Phi(x; \theta) := W_2 \cdot \sigma(W_1 x), \quad \theta = (W_1, W_2)\,, \tag{7.4}$$

with smooth non-polynomial activation function $\sigma : \mathbb{R} \to \mathbb{R}$.

Then, with high probability as $n \to \infty$

$$c\frac{n}{m}\widehat{r}(\Theta) \leq \|\nabla_\theta \widehat{r}(\Theta)\|_2^2 \leq C\frac{n}{m}\widehat{r}(\Theta), \quad \Theta = (\mathbf{W}_1, \mathbf{W}_2) \sim \mathcal{N}(0,1)^{n\times d} \otimes \mathcal{N}(0,1/n)^n \tag{7.5}$$

where $0 < c < C$ are constants independent of m and n.

Proof. We will sketch the proof for the lower bound.

We first note that

$$\begin{aligned}
\|\nabla_\theta \widehat{r}(\Theta)\|_2^2 &= \|\frac{2}{m}\sum_{i=1}^m \nabla_\theta \Phi(x^{(i)}, \Theta)(\Phi(x^{(i)}, \Theta) - y^{(i)})\|_2^2 \\
&\geq \frac{4}{m^2}\|\sum_{i=1}^m \nabla_{W_2} \Phi(x^{(i)}, \Theta)(\Phi(x^{(i)}, \Theta) - y^{(i)})\|_2^2 \\
&= \frac{4}{m^2}((\Phi(x^{(i)}, \Theta) - y^{(i)})_{i=1}^m)^\top \bar{K}_\Theta (\Phi(x^{(i)}, \Theta) - y^{(i)})_{i=1}^m, \quad (7.6)
\end{aligned}$$

where $\bar{K}_\Theta \in \mathbb{R}^{m\times m}$ is the random matrix given entry-wise by

$$(\bar{K}_\Theta)_{i,j} = \nabla_{W_2}\Phi(x^{(i)},\Theta)(\nabla_{W_2}\Phi(x^{(j)},\Theta))^\top \tag{7.7}$$

Evaluating $\nabla_{W_2}\Phi(x^{(i)},\Theta) \in \mathbb{R}^n$ gives

$$(\nabla_{W_2}\Phi(x,\Theta))_k = \sigma((\mathbf{W}_1 x)_k). \tag{7.8}$$

In other words, we can write $\bar{K}_\Theta$ as a sum of rank-1 matrices

$$\bar{K}_\Theta = \sum_{k=1}^{n} v_k v_k^\top, \quad \text{where } v_k = (\sigma((\mathbf{W}_1 x^{(i)})_k)_{i=1}^m \in \mathbb{R}^m. \tag{7.9}$$

Then, the kernel $\bar{K}_\Theta$ is symmetric and positive semi-definite by construction. Furthermore, it is positive definite if it is non-singular, which is the case with high probability[a].

Finally, we have with high probability that $\lambda_{\min}(\bar{K}_\Theta) > 0$ and it is clear that $\lambda_{\min}(\bar{K}_\Theta) \in \Omega(n)$. This allows us to find a tighter lower bound on 7.6. That is,

$$\begin{aligned}\|\nabla_\theta \widehat{r}(\Theta)\|_2^2 &\geq \frac{4}{m^2}((\Phi(x^{(i)},\Theta) - y^{(i)})_{i=1}^m)^\top \bar{K}_\Theta (\Phi(x^{(i)},\Theta) - y^{(i)})_{i=1}^m \\ &\geq \frac{4}{m^2}\lambda_{\min}(\bar{K}_\Theta)\|(\Phi(x^{(i)},\Theta) - y^{(i)})_{i=1}^m\|_2^2 \geq c\frac{n}{m}\widehat{r}(\Theta),\end{aligned} \tag{7.10}$$

for some $c > 0$, which yields the claim on the lower bound.

□

[a] Here we use the assumption that the data points are unique and the activation function non-polynomial cf. [6, Prop. 1.1]

Exercise 7.1. Show the upper bound in 7.5.

Note:-

Note that the proof above can be generalized fairly easily, e.g., by incorporating non-smooth activation such as ReLU or replacing the loss by any other loss $\mathcal{L} : \mathcal{F} \times (\mathbb{R}^d \times \mathbb{R}) \to \mathbb{R}$ such that for fixed $(x, y) \in \mathbb{R}^d \times \mathbb{R}$ the mapping $f \mapsto \mathcal{L}(f, (x, y))$ is strongly convex with Lipschitz gradients. This has been omitted to make the proof more readable.

In order to control the gradient throughout an entire GD procedure, we need to be able to have upper and lower bounds such as in Lemma 7.1 in a neighbourhood around a random initialzation.

We will use the following lemma without proof, but note that it relies on the result we obtained in Lemma 7.1.

Lemma 7.2. Under the assumptions in Lemma 7.1, the upper and lower bounds can be extended to a local neighbourhood.

In particular, for $\Theta = (\mathbf{W}_1, \mathbf{W}_2) \sim \mathcal{N}(0,1)^{n\times d} \otimes \mathcal{N}(0,1/n)^n$

$$\bar{c}\frac{n}{m}\widehat{r}(\Theta+\bar{\theta}) \leq \|\nabla_\theta \widehat{r}(\Theta+\bar{\theta})\|_2^2 \leq \bar{C}\frac{n}{m}\widehat{r}(\Theta+\bar{\theta}), \quad \bar{\theta} \in \mathbb{B}_1(0), \tag{7.11}$$

where $0 < \bar{c} < \bar{C}$ are constants independent of m and n.

7.2 Linear convergence of randomly initialized gradient descent

We are now ready to proof linear convergence of gradient descent in the overparametrized network regime.

Theorem 7.1. Let $(x^{(i)}, y^{(i)})_{i=1}^m \in (\mathbb{R}^d \times \mathbb{R})^m$ be a data set with $x^{(i)} \neq x^{(j)}$ for all $i \neq j \in [m]$. Consider the function $\widehat{r} : \mathbb{R}^{n\times d} \times \mathbb{R}^n \to \mathbb{R}$ given by

$$\widehat{r}(\theta) := \frac{1}{m}\sum_{i=1}^m (\Phi(x^{(i)}, \theta) - y^{(i)})^2, \tag{7.12}$$

where $\Phi(\cdot, \theta) : \mathbb{R}^d \to \mathbb{R}$ is a two-layer neural network of the form

$$\Phi(x;\theta) := W_2 \cdot \sigma(W_1 x), \quad \theta = (W_1, W_2)\,, \tag{7.13}$$

with smooth non-polynomial activation function $\sigma : \mathbb{R} \to \mathbb{R}$.

Then, with high probability as $n \to \infty$ gradient descent initialised with $\Theta^{(0)} = \left(\mathbf{W}_1^{(0)}, \mathbf{W}_2^{(0)}\right) \sim \mathcal{N}(0,1)^{n\times d} \otimes \mathcal{N}(0,1/n)^n$ and with step size $\eta = \alpha \frac{m}{\bar{c}n\sqrt{\widehat{r}(\Theta^{(0)})}}$ for small enough $\alpha \in (0, \sqrt{\widehat{r}(\Theta^{(0)})})$ for minimising

$$\inf_{\theta \in \mathbb{R}^{n\times d}\times\mathbb{R}^n} \widehat{r}(\theta) \tag{7.14}$$

converges linearly – with rate $(1 - \frac{\alpha}{2\sqrt{\widehat{r}(\Theta^{(0)})}})$ – to an error $\varepsilon_{opt} := \varepsilon\widehat{r}(\Theta^{(0)})$ for $\varepsilon \in (0,1)$ in $K' = \lceil \frac{\log(\varepsilon)}{\log(1-\alpha/(2\sqrt{\widehat{r}(\Theta^{(0)})}))} \rceil$ iterations if $K' \leq K := \lfloor \frac{\bar{c}}{\alpha}\sqrt{\frac{n}{\bar{C}m}} \rfloor$, where $\bar{c}, \bar{C} > 0$ are the constants in Lemma 7.2.

Proof. We'll first show that for $k \leq K := \lfloor \frac{\bar{c}}{\alpha}\sqrt{\frac{n}{\bar{C}m}} \rfloor$ we have

$$\|\Theta^{(k)} - \Theta^{(0)}\|_2 \leq 1, \quad \text{and} \quad \widehat{r}(\Theta^{(k+1)}) \leq (1 - \frac{\alpha}{2\sqrt{\widehat{r}(\Theta^{(0)})}})^k \widehat{r}(\Theta^{(0)}). \tag{7.15}$$

For $k = 0$ we have

$$\|\Theta^{(0)} - \Theta^{(0)}\|_2 = 0 \leq 1, \tag{7.16}$$

and

$$\begin{aligned}\widehat{r}(\Theta^1) &= \widehat{r}(\Theta^0) - \eta\|\nabla_\theta \widehat{r}(\Theta^{(0)})\|_2^2 + \mathcal{O}(\eta^2)\\ &\overset{\alpha \text{ small enough}}{\leq} \widehat{r}(\Theta^0) - \frac{\eta}{2}\|\nabla_\theta \widehat{r}(\Theta^{(0)})\|_2^2\\ &\overset{\text{Lemma 7.1}}{\leq} (1 - \frac{\bar{c}\eta n}{2m})\widehat{r}(\Theta^{(0)})\\ &= (1 - \frac{\alpha}{2\sqrt{\widehat{r}(\Theta^{(0)})}})\widehat{r}(\Theta^{(0)}). \quad (7.17)\end{aligned}$$

Then, by induction, i.e., assuming that 7.15 is true for $k-1$

$$\begin{aligned}\|\Theta^{(k)} - \Theta^{(0)}\|_2 &= \| - \eta\sum_{k'=0}^{k-1}\nabla_\theta \widehat{r}(\Theta^{(k')})\|_2 \leq \eta\sum_{k'=0}^{k-1}\|\nabla_\theta \widehat{r}(\Theta^{(k')}\|_2\\ &\overset{\text{Lemma 7.2}}{\leq} \eta k\sqrt{\bar{C}\frac{n}{m}\widehat{r}(\Theta^{(0)}} \leq (\alpha\frac{m}{\bar{c}n\sqrt{\widehat{r}(\Theta^{(0)})}})(\frac{\bar{c}}{\alpha}\sqrt{\frac{n}{\bar{C}m}})\sqrt{\bar{C}\frac{n}{m}\widehat{r}(\Theta^{(0)}} = 1. \quad (7.18)\end{aligned}$$

and

$$\begin{aligned}\widehat{r}(\Theta^{k+1}) &= \widehat{r}(\Theta^k) - \eta\|\nabla_\theta \widehat{r}(\Theta^{(k)})\|_2^2 + \mathcal{O}(\eta^2)\\ &\overset{\alpha \text{ small enough}}{\leq} \widehat{r}(\Theta^k) - \frac{\eta}{2}\|\nabla_\theta \widehat{r}(\Theta^{(k)})\|_2^2\\ &\overset{\text{Lemma 7.2}}{\leq} (1 - \frac{\bar{c}\eta n}{2m})\widehat{r}(\Theta^{(k)}) \leq (1 - \frac{\bar{c}\eta n}{2m})^k\widehat{r}(\Theta^{(0)})\\ &= (1 - \frac{\alpha}{2\sqrt{\widehat{r}(\Theta^{(0)})}})^k\widehat{r}(\Theta^{(0)}). \quad (7.19)\end{aligned}$$

Linear convergence for all $k \in [K]$ follows directly from the above. The claimed number of iterations K' needed to reach the error $\varepsilon_{opt} := \varepsilon\widehat{r}(\Theta^{(0)})$ can easily be verified from here and it is also clear that $K' \leq K$. □

The theorem above is in line with the behaviour we have seen in 3, i.e.,

- we predict linear convergence with a rate independent of the layer width n
- for larger n we predict to use a smaller step size. So in the end, the solution is expected to be closer to the initialization, since the total number of iterations is fixed

Finally, remember from lecture 2 that the total error can be controlled by the approximation error, the generalisation error and the optimisation error, i.e.,

$$\mathcal{R}(f_s) - \mathcal{R}^* \leq \varepsilon_{approx} + 2\varepsilon_{gen} + \varepsilon_{opt}.$$

In this lecture we have seen that we can get $\varepsilon_{opt} \to 0$ by letting the network width tend to infinity. Note that this is not always a good thing. In particular, if we are given very noisy data and we get (almost) 0 optimisation error, the generalization error ε_{gen} will inevitably increase. In other words, we still need to be careful with finding a good balance between the errors.

Feb 14 2024 Wed (12:00-13:00)

Lecture 8: Universal approximation with two-layer NNs

We recall from previous lectures that we aim to **approximate a target function** $f^* \in \mathcal{M}(\mathcal{X}, \mathcal{Y})$, which minimizes a risk defined as

$$\mathcal{R}(f) := \mathbb{E}_Z[\mathcal{L}(f, Z)],$$

where Z is a random variable following an unknown probability distribution $\mathbb{P}_Z$ over $\mathcal{Z}$. In order to simplify the problem, we **choose a hypothesis set** $\mathcal{F} \subset \mathcal{M}(\mathcal{X}, \mathcal{Y})$, typically finite-dimensional, and consider the problem[2]

$$\underset{f \in \mathcal{F}}{\text{minimise}}\ \mathcal{R}(f). \tag{8.1}$$

The class of functions $\mathcal{F}$ has to be large enough, so that the solution to (8.1) is close to the risk of the target function $\mathcal{R}(f^*)$. In the Deep Learning wordy, we say that the hypothesis set has to be expressible enough.

The error induced by the lack of expressivity of the Neural Network's architecture is known as approximation error, and is given by

$$\varepsilon_{approx} := \inf_{f \in \mathcal{F}} \mathcal{R}(f) - \mathcal{R}^*, \qquad \text{where } \mathcal{R}^* := \inf_{\mathcal{M}(\mathcal{X}, \mathcal{Y})} \mathcal{R}(f).$$

Question 3

"Is the class of Neural Networks large enough to ensure a small approximation error?".

We consider the problem of approximating a continuous function

$$f : \Omega \subset \mathbb{R}^d \longrightarrow \mathbb{R}$$

where Ω is a compact set of $\mathbb{R}^d$. The class of Neural Networks in the hypothesis set that we are going to consider is the two-layer feedforward fully connected networks with a single activation after the first layer, i.e. functions of the form

$$\Phi_\theta(x) = W_2 \sigma(W_1 x + \mathbf{b}_1),$$

where the parameters $\theta = (W_1, W_2, b_1)$ are of the form $W_1 \in \mathbb{R}^{n \times d}$, $W_2 \in \mathbb{R}^{1 \times n}$ and $b_1 \in \mathbb{R}^n$. Here, $n \in \mathbb{N}$ represents the number of neurons in the hidden layer. If we denote

$$W_2 = [\alpha_1, \alpha_2, \dots, \alpha_n] \in \mathbb{R}^n, \qquad W_1 = [w_1, w_2, \dots, w_n]^T, \quad \text{with } y_j \in \mathbb{R}^d$$
$$\text{and} \qquad \mathbf{b}_1 = [b_1, b_2, \dots, b_n] \in \mathbb{R}^n,$$

we can write $\Phi_\theta(x)$ as

$$\Phi_\theta(x) = \sum_{j=1}^{n} \alpha_j \sigma(w_j \cdot x + b_j). \tag{8.2}$$

[2] Of course, the problem (8.1) cannot be addressed in practice since we do not have access to the risk function $\mathcal{R}(\cdot)$.

8.1 Universal approximation with sigmoidal activation

Here we review a classical result by Cybenko in 1989 [10] which ensures that two-layer feedforward fully connected neural networks with a sigmoidal activation function can approximate any continuous function in a compact set at arbitrary precision.

As for the activation function σ, we consider any **continuous sigmoidal function**, which is any function $\sigma \in C(\mathbb{R})$ satisfying

$$\lim_{s\to+\infty} \sigma(s) = 1 \qquad \text{and} \qquad \lim_{s\to-\infty} \sigma(s) = 0.$$

Actually, the proof applies to a more general class of activation functions, known as **discriminatory** functions.

Definition 8.17

We say that $\sigma \in C(\mathbb{R})$ is discriminatory if for any signed measure μ on Ω,

$$\int_\Omega \sigma(w \cdot x + b)\mu(x)dx = 0 \quad \forall (w,b) \in \mathbb{R}^d \times \mathbb{R}, \quad \text{implies that } \mu \equiv 0.$$

Exercise 8.1. Prove that any continuous sigmoidal function is discriminatory.

The following theorem by Cybenko ensures that the space of functions of the form (8.2), that we shall denote by $\mathcal{F}$, is dense in $C(\Omega)$, the space of continuous functions in Ω. More precisely, the closure of $\mathcal{F}$ with respect to the uniform topology is $C(\Omega)$, i.e.

$$\forall f \in C(\Omega) \text{ and } \forall \varepsilon > 0, \quad \exists \Phi_\theta \in \mathcal{F} \quad \text{s.t.} \quad \|f - \Phi_\theta\|_\infty < \varepsilon.$$

Theorem 8.1 (Cybenko, 1989 [10]). Let $\Omega \subset \mathbb{R}^d$ and σ be any continuous discriminatory function. Then the set of functions

$$\mathcal{F} := \text{span}\left\{\sigma(w \cdot x + b) \ : \quad (w,b) \in \mathbb{R}^d \times \mathbb{R}\right\}$$

is dense in $C(\Omega^d)$.

Proof. The result is an application of Hahn-Banach Theorem (see Theorem 8.2 and Exercise 8.2). We see that $\mathcal{F}$ is a linear subspace of $C(\Omega)$. Let us prove that the closure[a] of $\mathcal{F}$ is the whole space of continuous functions $C(\Omega)$, which is a Banach space.

Suppose for a contradiction that $\overline{\mathcal{F}} \neq C(\Omega)$. By Hahn-Banach Theorem (see Theorem 8.2 and Exercise 8.2), there exists a linear functional $F \in C(\Omega)^\star$ such that $F \not\equiv 0$ and $F(\Phi_\theta) = 0$ for all $\Phi_\theta \in \mathcal{F}$.

Now, since Ω is compact, the dual space of $C(\Omega)$ is the space of measures in Ω, and by Riesz representation Theorem, there exists a measure $\mu \not\equiv 0$ such that

$$F(f) = \int_\Omega f(x)\mu(x)dx.$$

Since F vanishes on $\mathcal{F}$ and any function of the form $x \mapsto \sigma(w \cdot x + b)$ for some

$(w, b) \in \mathbb{R}^d \times \mathbb{R}$ is in $\mathcal{F}$, we deduce that

$$\int_\Omega \sigma(w \cdot x + b)\mu(x)dx = 0 \qquad \forall (w, b) \in \mathbb{R}^d \times \mathbb{R}.$$

However, since σ is discriminatory, we deduce that $\mu \equiv 0$, and then $F \equiv 0$, which yields a contradiction, implying that $\overline{F} = C(\Omega)$. □

[a] The closure is taken with respect to the topology induced by the uniform norm $\|\cdot\|_\infty$.

Let us recall the Hahn-Banach Theorem in its geometric form, which is used in the previous proof.

Theorem 8.2 (Hahn-Banach Theorem). Let E be a normed vector space and let $A \subset E$ and $B \subset E$ be two non-empty convex subsets such that

$$A \cap B = \emptyset, \ A \text{ is closed and } B \text{ is compact.}$$

Then, there exists a closed hyperplane that strictly separates A and B, i.e.

$$\exists F \in E^\star \text{ and } \exists \alpha \in \mathbb{R} \text{ such that } F(a) < \alpha < F(b) \quad \forall (a, b) \in A \times B.$$

Exercise 8.2. Prove the following statement as a consequence of Hahn-Banach Theorem:

Let $A \subset E$ be a linear subspace of E such that $\overline{A} \neq E$. Then there exists some $F \in E^*$, with $F \not\equiv 0$ such that $F(a) = 0$ for all $a \in A$.

Note:-

1. We proved that any continuous function can be approximated by a fully connected neural network with a single hidden layer of the form (8.2). However, in order to achieve arbitrary small accuracy we need to allow the number of neurons in the hidden layer, represented by n, to be arbitrarily large.
2. There exist many results about universal approximation by means of two-layers NNs. For instance in [23], where the strategy of the proof consists in proving that two-layer NNs can approximate any polynomial, and then applying Stone-Weierstrass Theorem to extend the result to continuous functions.

Example. Consider a regression task with $\mathcal{X} = \Omega$ and $\mathcal{Y} = \mathbb{R}$ and a sampling probability distribution $\rho(x, y)$ over $\mathcal{X} \times \mathcal{Y}$. We recall that the Bayes optimal function is given by

$$f^*(x) = \mathbb{E}\big[Y|X = x\big], \quad \text{and} \quad \mathcal{R}^* = \mathcal{R}(f^*).$$

One can readily prove that

$$\begin{aligned}\mathcal{R}(f)-\mathcal{R}^* &= \mathbb{E}_X\left[(f(X)-f^*(X))^2\right]\\ &= \int_\Omega (f(x)-f^*(x))^2 d\rho_X(x)\\ &\leq \|f-f^*\|_\infty.\end{aligned}$$

Let us assume that the probability distribution ρ is such that $f^*(x)$ is continuous in Ω. Then, by means of Theorem 8.1, for any $\varepsilon>0$, there exists a two-layer neural network Φ_θ, with sufficiently many neurons in the hidden layer, such that

$$\mathcal{R}(\Phi_\theta)-\mathcal{R}^*<\varepsilon.$$

This provides an upper bound for the approximation error.

8.2 Quantitative estimates depending on the regularity

A question which naturally arises in the context of approximation by NNs is how many parameters are sufficient to achieve a specified accuracy in the approximation of a given function. As we will see, to answer this question, one needs to make precise **what kind of function we want to approximate.** For instance, it is not the same to approximate a linear function, a piece wise linear function or a general continuous function.

A quantitative version of Theorem 8.1 can be found in [25]. In broad terms, it says that the more regular the target function is (in terms of the Sobolev norm $W^{k,p}$) the less parameters we need to approximate it in the L^p distance.

Let us recall some elementary definitions. Let $\Omega\subset\mathbb{R}^d$ be a compact set and $p\in[1,\infty]$. For any measurable function $f:\Omega\to\mathbb{R}$, we define its L^p-norm as

$$\|f\|_{L^p}=\left(\int_\Omega |f(x)|^p dx\right)^{\frac{1}{p}}.$$

The Lebesgue space $L^p(\Omega)$ is defined as the space of function with finite L^p-norm, i.e.

$$L^p(\Omega):=\{f\in\mathcal{M}(\Omega,\mathbb{R})\quad\text{s.t.}\quad \|f\|_p<\infty\}.$$

For any $k\in k\in\mathbb{N}$, we define the Sobolev space $W^{k,p}(\Omega)$ as

$$W^{k,p}(\Omega):=\{f\in L^p(\Omega)\quad\text{s.t.}\quad D^r f\in L^p(\Omega)\ \forall|\alpha|\leq k\},$$

where $r=(r_1,r_2,\dots,r_d)\in\mathbb{N}^d$ is a multi-index with $|r|=\sum_{i=1}^d r_i$ and $D^r f$ denotes the $|r|$-th order derivative of f w.r.t. the d variables as represented in the multi-index r. The $W^{k,p}$-Sobolev norm is then defined as

$$\|f\|_{W^{k,p}}:=\sum_{r\leq k,|r|\leq k}\|D^r f\|_p.$$

Theorem 8.3. Let $d,k\in\mathbb{N}$ and $p\in[1,\infty]$. Let $\sigma\in C^\infty(\Omega)$ be a sigmoidal function. Then there exists a constant $c\in(0,\infty)$ such that, for any $n\in\mathbb{N}$ with $n\geq O(k^d)$, there exist parameters $\{(w_j,b_j)\}_{j=1}^n\in[\mathbb{R}^d\times\mathbb{R}]^n$ such that, for every

$f \in W^{k,p}(\Omega)$ it holds that

$$\|\Phi_\alpha - f\|_{L^p} \leq cn^{-\frac{k}{d}} \|f\|_{W^{k,p}}.$$

for some $\alpha := (\alpha_1, \alpha_2, \ldots, \alpha_n) \in \mathbb{R}^n$, where

$$\Phi_\alpha(x) := \sum_{j=1}^{n} \alpha_j \sigma(w_j x + b_j).$$

Note:-

- In the statement of the above Theorem, we see that the choice of the parameters of the first layer (i.e. w_j and b_j) is independent of the target function f. Hence, the approximation of f consists on a linear combination functions which approximates a basis of $W^{k,p}(\Omega)$.
- Indeed, the strategy of the proof in [25] consists on two steps:
  1. First they prove that, for all $m \geq k$, any function $f \in W^{k,p}(\Omega)$ can be approximated by a polynomial $P_m[f](x)$ of degree m such that
  $$\|P_m[f] - f\|_{L^p} \leq cm^{-k} \|f\|_{W^{k,p}},$$
  where the polynomial $P_m[f]$ is a linear combination of Chebyshev polynomials T_k, which are independent of f.
  2. The second step is to construct an approximation to every polynomial T_k used in the first step, by means of a function of the form (8.2) with $n \sim m^d$ terms.
- Since the proof of Theorem 8.3 consists in approximating a classical approximation method by means of NNs, the upper bound for the approximation error cannot be better than that for the classical method.
- The main drawback of the approximation bound in Theorem 8.3 is the fact that, in order to ensure an ε accuracy in the L^p-norm, we need to consider a two-layer NN with
$$n \sim \left(\frac{\|f\|_{W^{k,p}}}{\varepsilon} \right)^{\frac{d}{k}},$$
which means that the number of neurons increases a lot in high-dimensional cases $d \gg 1$. In the machine learning literature this phenomenon is sometimes referred to as the **curse of dimensionality**. In the next lecture, we will introduce a functional framework which is more suitable to obtain approximation results by means of Neural Networks.

Feb 19 2024 Mon (12:00-13:00)

Lecture 9: Variation norm and Barron space

We consider the problem of approximating a function

$$f : \Omega \longrightarrow \mathbb{R}$$

by means of a two-layer feedforward fully connected neural network. We consider Ω to be a bounded subset of $\mathbb{R}^d$.

We recall that a two-layer NN with $n \in \mathbb{N}$ neurons in the hidden layer can be written as

$$\Phi_\theta(x) = \sum_{j=1}^{n} \alpha_j \sigma(w_j \cdot x + b_j), \tag{9.1}$$

where the set of parameters $\theta \in \mathbb{R}^n \times \mathbb{R}^{n\times d} \times \mathbb{R}^n$ have the form

$$\theta = [\alpha_j, w_j, b_j]_{j=1}^n \in [\mathbb{R} \times \mathbb{R}^d \times \mathbb{R}]^n.$$

Let us consider the rectified linear unit (ReLU) as activation function, which is the function $\mathbb{R} \to \mathbb{R}$ defined as

$$\sigma(x) = \max\{0, x\} = (x)_+.$$

Note:-

The main feature of this activation function, as compared with other activation function, is the 1-homogeneity, i.e.

$$\sigma(\lambda x) = \lambda\sigma(x) \qquad \forall \lambda \geq 0.$$

This implies that, for any $(\lambda_1, \ldots, \lambda_n) \in \mathbb{R}_+^n$, the parameters

$$\theta = [\alpha_j, w_j, b_j]_{j=1}^n \qquad \text{and} \qquad \theta' = \left[\lambda_j \alpha_j, \frac{w_j}{\lambda_j}, \frac{b_j}{\lambda_j}\right]_{j=1}^n$$

give rise to the same prediction function. Indeed, for all $x \in \Omega$, we have

$$\Phi_\theta(x) = \sum_{j=1}^{n} \alpha_j \sigma(w_j \cdot x + b_j) = \sum_{j=1}^{n} \lambda_j \alpha_j \sigma\left(\frac{w_j \cdot x + b_j}{\lambda_j}\right) = \Phi_{\theta'}(x).$$

Important: This means that when using ReLU activation, we can restrict the choice of the parameters in the first layer (w, b) to a compact set $K \subset [\mathbb{R}^d \times \mathbb{R}]^n$ without changing the expressivity of the architecture.

In the past lecture we saw approximation properties of two-layer FCNNs in Sobolev spaces $W^{k,p}$. In this functional space, the approximation bounds suffer from the curse of dimensionality (the number of neurons increases exponentially with d). Today we aim at introducing a more suitable functional space to obtain approximation bounds for two-layer FCNNs. Namely, we shall introduce the Banach space generated by this kind of architectures and endow it with a suitable norm.

9.1 Universal approximation property in one dimension

Let us consider $d = 1$ and Ω a bounded interval, e.g. $[-R, R]$ for some $R > 0$. We aim to approximate a function $f : [-R, R] \longrightarrow \mathbb{R}$ by means of a linear combination of functions of the form

$$x \mapsto \alpha(wx + b)_+ \quad \text{for some} \quad \alpha, w, b \in \mathbb{R}. \tag{9.2}$$

Since the function in (9.2) is continuous and piecewise affine for any parameters α, w, b, any linear combination of functions of this form is continuous and piecewise affine as well. Indeed, one can readily prove that any continuous piecewise affine function in $[-R, R]$ with a finite number of pieces can be represented as a linear combination of functions of the form (9.2).

Exercise 9.1. Let $f : [-R, R] \to \mathbb{R}$ be a continuous piecewise affine function with m pieces, i.e. there exist

$$-R = a_0 < a_1 < a_2 < \ldots < a_{m-1} < a_m = R$$

such that, for any $j \in \{1, \ldots, m\}$, $f(x) = v_j x + c_j$ for all $x \in [a_{j-1}, a_j]$ for some $(v_j, c_j) \in \mathbb{R}^2$.

Prove that there exists a function of the form

$$\Phi_\theta(x) = \sum_{j=0}^{m} \alpha_j (w_j x + b_j)_+$$

such that $f(x) = \Phi_\theta(x)$ for all $x \in [-R, R]$.

Once we have proved that we can exactly represent any continuous piecewise affine function by means of a two-layer ReLU NN, we can use the "density" of this class of functions in a larger functional space, to guarantee universal approximation of $\mathcal{F}$ in the larger functional space (for instance the continuous functions).

Note:-

When we talk about the "density" of one class of functions A in a larger class of functions X, we mean that the closure of A is equal to X. Hence, one needs to make precise the topology on X.

For instance, piecewise affine functions are dense in the space of continuous functions with respect to the topology induced by the L^∞-norm. They are also dense in $L^2(\Omega)$ with respect to the weaker topology induced by the L^2-norm.

Obviously, if one aims to provide quantitative approximation bounds (for instance in terms of number of hidden neurons), one needs to make regularity assumptions on the objective function.

Recall from the previous lecture: if we want to approximate a function $f \in W^{k,p}(\Omega)$ by a two-layer FCNN with n neurons in the hidden layer, we can upper bound the approximation error by

$$\varepsilon_{approx} \lesssim n^{-\frac{k}{d}} \|f\|_{W^{k,p}}.$$

9.2 Variation norm and the Barron space

We refer to [4] for more details about this section.

Here, we consider again the multi-dimensional case $d \geq 1$. For any $m \in \mathbb{N}$, we consider two-layer NNs of the form

$$\Phi_\theta(x) = \sum_{j=1}^{m} \alpha_j \sigma(w_j \cdot x + b_j). \tag{9.3}$$

This time we are going to constraint the parameters in the first layer to be in a compact set. Namely, for some $R > 0$ fixed, we require

$$(w_j, b_j/R) \in K, \qquad \text{for some compact subset } K \subset \mathbb{R}^{d+1}. \tag{9.4}$$

Remember that, by the homogeneity of ReLU function, this constraint does not affect the expressivity of the NN architecture. Typical choices are $w_j \in \overline{B(0,1)}$ and $b_j \in [-1,1]$. Roughly, the constant $R > 0$ determines the radius of the domain Ω, which is assumed to be bounded.

We denote by $\mathcal{F}_m$ the set of functions $\Omega \to \mathbb{R}$ that can be represented by a NN of the form (9.3) with m neurons, and parameters satisfying (9.4). For any function $f \in \mathcal{F}_m$, let us define the quantity

$$\gamma_1^{(m)}(f) = \inf \left\{ \|\alpha\|_1 \ : \quad \Phi_\theta(x) = f(x) \ \forall x \in \Omega, \ \theta = [\alpha, (w,b)] \in [\mathbb{R} \times K]^m \right\}.$$

When f is not representable by m neurons we write $\gamma_1^{(m)}(f) = +\infty$.

We clearly have

$$\mathcal{F}_m \subset \mathcal{F}_{m+1} \quad \text{and} \quad \gamma_1^{(m)}(f) \geq \gamma_1^{(m+1)}(f) \quad \text{for all } m \geq 1.$$

Goal: We aim at studying the limit of the set of functions $\mathcal{F}_m$ and the quantity $\gamma_1^{(m)}$ as $m \to \infty$, i.e. as the number of layers tends to infinity.

For any $m_0 \geq 1$ and any function $f \in \mathcal{F}_{m_0}$, the sequence $\{\gamma_1^{(m)}(f)\}_{m \geq m_0}$ is bounded, non-negative and non-increasing, which implies that it converges as $m \to \infty$. Hence, for any function $f \in \bigcup_{m \geq 0} \mathcal{F}_m$, we can define the quantity

$$\gamma_1(f) := \lim_{m \to \infty} \gamma_1^{(m)}(f) < +\infty.$$

We observe that $\bigcup_{m \geq 0} \mathcal{F}_m$, the set of functions that can be represented by a finite number of neurons, is clearly a vector space over $\mathbb{R}$.

> **Exercise 9.2.** Prove that, for any $f, g \in \bigcup_{m \geq 0} \mathcal{F}_m$, the following statements hold true:
>
> 1. $\gamma_1(\lambda f) = \lambda \gamma_1(f) \qquad \forall \lambda \geq 0.$
> 2. $\gamma_1(f+g) \leq \gamma_1(f) + \gamma_1(g)$

3. $\gamma_1(f) = 0$ implies $f = 0$.

The above exercise implies that $\gamma_1(\cdot)$ defines a norm on the vector space $\bigcup_{m\geq 0} \mathcal{F}_m$. Hence, we can complete this space to obtain the so-called Barron space, which is the smallest Banach space endowed with the norm $\gamma_1(\cdot)$ that contains all the functions which can be represented by a finite number of neurons.

Definition 9.18

We call **Barron space**, and denote $\mathcal{F}_1$, to the completion of $\bigcup_{m\geq 0} \mathcal{F}_m$ with respect to the norm $\gamma_1(\cdot)$. This norm is often called **variation norm**.

9.3 Construction of $\mathcal{F}_1$ through measures

We have defined $\mathcal{F}_1$ as the limit of functions of two-layer FCNNs with respect to the variation norm $\gamma_1(\cdot)$ above defined. Let us see how these limits look like.

Any function $f : \Omega \to \mathbb{R}$ that can be represented by means a two-layer FCNN with finitely many layers can be written as

$$f(x) = \sum_{j=1}^{m} \alpha_j \sigma(w_j \cdot x + b_j),$$

for some $(\alpha_1, \dots, \alpha_m) \in \mathcal{R}^m$ and $\{(w_j, b_j)\}_{j=1}^m \in K^m$.

Notation:

1. For any $(w, b) \in K$ let us denote by $\delta_{(w,b)}(\cdot)$ the Dirac delta distribution in K centred at (w, b).
2. The space of Radon measures on the compact set K will be denoted by $\mathcal{M}(K)$. Recall that $\mathcal{M}(K)$ is the dual space of $C(K)$.
3. We denote by $\mathcal{M}_a(K)$ the set of Radon measures consisting of a finite combination of Dirac deltas, i.e. $\nu \in \mathcal{M}_a(\Omega)$ if for some $m \in \mathbb{N}$, there exist $\{\alpha_j\}_{j=1}^m \in \mathbb{R}^m$ and $\{(w_j, b_j)\}_{j=1}^m \in K^m$ such that

$$\nu(\cdot) = \sum_{j=1}^{m} \alpha_j \delta_{(w_j, b_j)}(\cdot).$$

The subscript a stands for "atomic" mass, as the measures in this class consist of a finite combination of atomic masses.

Since the activation function $\sigma : \mathbb{R} \to \mathbb{R}$ is continuous, for any $x \in \Omega$, the function

$$\begin{array}{rccc} \sigma_x : & K & \longrightarrow & \mathbb{R} \\ & (w, b) & \longmapsto & \sigma(w \cdot x + b) \end{array}$$

is continuous as well. Hence, we can write f as

$$f(x) = \int_K \sigma_x(w, b)\nu(w, b)d(w, b), \qquad \forall x \in \Omega,$$

for some measure $\nu \in \mathcal{M}_a(K)$.

For any $\nu(\cdot) \in \mathcal{M}_a(K)$, we can write the total variation of ν, as

$$|\nu|(K) = \int_K d|\nu(w,b)| = \sum_{j=1}^m |\alpha_j|.$$

Hence, if $f \in C(\Omega)$ is such that it can be written as a two-layer FCNN with finitely many neurons, we can define its variation norm as

$$\gamma_1(f) = \inf\left\{ |\nu|(K) \ : \ \nu \in \mathcal{M}_a(K) \text{ s.t. } f(x) = \int_K \sigma_x(w,b)\nu(w,b)d(w,b) \quad \forall x \in \Omega \right\}.$$

The following Theorem generalizes the above argument to all the functions in $\mathcal{F}_1$, also those which are not necessarily representable by a two-layer FCNN with finitely many neurons in the hidden layer. The proof is based on the density of $\mathcal{M}_a(K)$ in $\mathcal{M}(\mathcal{K})$ with respect to the weak-$*$ topology.

Theorem 9.1. The Barron space $\mathcal{F}_1$ is the set of functions of the form

$$f(x) = \int_K \sigma_x(w,b)\nu(w,b)d(w,b) \qquad \text{for some } \nu \in \mathcal{M}(K).$$

Moreover, $\mathcal{F}_1$ is a Banach space endowed with the norm

$$\gamma_1(f) = \inf\left\{ |\nu|(K) \ : \ \nu \in \mathcal{M}(K) \text{ s.t. } f(x) = \int_K \sigma_x(w,b)\nu(w,b)d(w,b) \quad \forall x \in \Omega \right\}.$$

Proof. Let $f \in \mathcal{F}_1$ be a function in the Barron space. By definition, f can be written as the γ_1-limit of a sequence of continuous functions f_n that can be represented by a two-layer FCNN, which implies that there exists a sequence of atomic masses $\nu_n \in \mathcal{M}_a(K)$ such that

$$f_n(x) = \int_K \sigma_x(w,b)\nu_n(w,b)d(w,b) \quad \text{and} \quad \lim_{n\to\infty} \gamma_1(f_n) = \lim_{n\to\infty} |\nu_n|(K) = \gamma_1(f) < +\infty.$$

By the weak-$*$ compactness (see [1, Theorem 1.59] of the space $\mathcal{M}(K)$, there exists a measure $\nu \in \mathcal{M}(K)$ which is the weak-$*$ limit of $\{\nu_n\}_{n\geq 1}$, i.e.

$$\lim_{n\to\infty} \int_K \phi(w,b)d\nu_n(w,b) = \int_K \phi(w,b)\nu(w,b)d(w,b) \qquad \forall \phi \in C(K).$$

Since $\sigma_x(\cdot) \in C(K)$ for all $x \in \Omega$, this implies that the limit function f can be written as

$$f(x) = \int_K \sigma_x(w,b)\nu(w,b)d(w,b) \qquad \forall x \in \Omega. \tag{9.5}$$

We can also deduce that $\gamma_1(f) = \lim_{n\to\infty} \gamma_1(f_n)$ and that

$$\gamma_1(f) = \inf\left\{ |\nu|(K) \ : \ \nu \in \mathcal{M}(K) \text{ s.t. } f(x) = \int_K \sigma_x(w,b)\nu(w,b)d(w,b) \quad \forall x \in \Omega \right\}.$$

We finally need to prove that any function of the form (9.5) can is the γ_1-limit of functions which can be represented by finitely many neurons. By a density

argument of atomic masses in $\mathcal{M}(K)$ with respect to the weak-$*$ topology, we can prove that any Radon measure $\nu \in \mathcal{M}(K)$ can be obtained as the weak-$*$ limit of a sequence of atomic masses in $\mathcal{M}_a(K)$, any function of the form

$$f(x) = \int_K \sigma_x(w,b)\nu(w,b)d(w,b) \qquad \text{for some } \nu \in \mathcal{M}(K)$$

can be obtained as the γ_1-limit of two-layer FCNN. Hence $f \in \mathcal{F}_1$. □

9.4 The Hilbert space $\mathcal{F}_2$

Given $K \subset \mathbb{R}^{d+1}$ compact, a positive measure $\mu \in \mathcal{M}(K)$, and a continuous activation function $\sigma : \mathbb{R} \to \mathbb{R}$, we consider the Hilbert space $L^2(K,\mu)$ and the feature map

$$\varphi : \Omega \longrightarrow L^2(K,\mu)$$

defined for each $x \in \Omega$ as the function

$$\varphi(x) : (w,b) \longmapsto \phi(x;w,b) := \sigma(w \cdot x + b).$$

Note:-

The space $L^2(K,\mu)$ is the Hilbert space of μ-measurable functions $\nu : K \to \mathbb{R}$ such that

$$\int_K |\nu(w,b)|^2 \mu(w,b)d(w,b) < +\infty,$$

which is endowed with the inner product

$$\langle \nu_1(\cdot), \nu_2(\cdot) \rangle_{L^2(K,\mu)} = \int_K \nu_1(w,b)\nu_2(w,b)\mu(w,b)d(w,b).$$

The fact that σ is continuous ensures that $\varphi(x) \in L^2(K,\mu)$ for all $x \in \mathbb{R}^d$.
The squared norm of $L^2(K,\mu)$ is given by

$$\|\nu(\cdot)\|^2_{L^2(K,\mu)} = \int_K |\nu(w,b)|^2 \mu(w,b)d(w,b).$$

We now define $\mathcal{F}_2$ as the set of functions $f : \Omega \to \mathbb{R}$ such that

$$f(x) = \langle \nu(\cdot), \varphi(x) \rangle_{L^2(K,\mu)} = \int_K \nu(w,b)\sigma(w \cdot x + b)\mu(w,b)d(w,b),$$

for some $\nu(\cdot) \in L^2(K,\mu)$.

We endow $\mathcal{F}_2$ with the norm

$$\gamma_2(f) = \inf \left\{ \|\nu\|_{L^2(K,\mu)} \ : \quad \nu \in L^2(K,\mu) \text{ s.t. } f(x) = \langle \nu(\cdot), \varphi(x) \rangle_{L^2(K,\mu)} \quad \forall x \in \Omega \right\}. \tag{9.6}$$

Theorem 9.2. The vector space $\mathcal{F}_2$ with the norm $\gamma_2(\cdot)$ is the reproducing kernel Hilbert space associated to the kernel

$$k(x,y) = \langle \varphi(x), \varphi(y) \rangle_{L^2(K,\mu)} = \int_K \sigma(w \cdot x + b)\, \sigma(w \cdot y + b)\mu(w,b)d(w,b).$$

Proof. We define the linear operator $U : L^2(K,\mu) \to \mathcal{F}_2$ given by

$$U\phi(x) = \langle \nu(\cdot), \varphi(x)(\cdot)\rangle_{L^2(K,\mu)}.$$

Let $\mathcal{N}$ denote the null space of U, i.e.

$$\mathcal{N} := \{\nu \in L^2(K,\mu) \ : \quad U\nu(\cdot) \equiv 0\}.$$

The restriction of U to the orthogonal complement of $\mathcal{N}$, denoted by $\mathcal{N}^\perp$, is a bijection from $\mathcal{N}^\perp$ to $\mathcal{F}_2$.

Exercise: Prove that $U|_{\mathcal{N}^\perp} : \mathcal{N}^\perp \to \mathcal{F}_2$ is a bijection.

We can then define the following inner product on $\mathcal{F}_2$ as

$$\langle f_1(\cdot), f_2(\cdot)\rangle_{\mathcal{F}_2} = \langle U^{-1}f_1(\cdot), U^{-1}f_2(\cdot)\rangle_{L^2(K,\mu)}, \qquad \forall f_1, f_2 \in \mathcal{F}_2.$$

We need to prove that this inner product induces the norm $\gamma_2(\cdot)$ defined in (9.6). Let $f \in \mathcal{F}_2$. For any $\nu \in L^2(K,\mu)$ such that $f = U\nu$, we have

$$\nu = U^{-1}f + \tilde{\nu}, \qquad \text{for some } \tilde{\nu} \in \mathcal{N}.$$

Since $U^{-1}f \in \mathcal{N}^\perp$ and $\tilde{\nu} \in \mathcal{N}$, the Pythagorean Theorem yields

$$\|\nu\|^2_{L^2(K,\mu)} = \|U^{-1}f\|^2_{L^2(K,\mu)} + \|\tilde{\nu}\|^2_{L^2(K,\mu)}.$$

This implies that $U^{-1}f$ minimises $\|\nu\|_{L^2(K,\mu)}$ among all functions $\nu \in L^2(d\tau)$ such that $f(x) = \langle \nu(\cdot), \varphi(x)\rangle_{L^2(K,\mu)}$, which gives

$$\gamma_2(f)^2 = \|U^{-1}f(\cdot)\|^2_{L^2(K,\mu)} = \langle U^{-1}f(\cdot), U^{-1}f(\cdot)\rangle_{L^2(K,\mu)}.$$

We finally need to check that $\mathcal{F}_2$ satisfies the reproducing property with the kernel defined in the statement of the Theorem. Note that, by taking

$$\nu(w,b) = \varphi(x),$$

we have that, for any $x \in \mathbb{R}^d$, the function $k(x,\cdot)$ is in $\mathcal{F}_2$. Now, let $\nu_x = U^{-1}k(x,\cdot) \in \mathcal{N}^\perp$. Since $U\nu = U\nu_x$, we have $\nu - \nu_x \in \mathcal{N}$. Hence, for any $f \in \mathcal{F}_2$, we have

$$\begin{aligned}
\langle f(\cdot), k(x,\cdot)\rangle_{\mathcal{F}_2} &= \langle U^{-1}f(\cdot), \nu_x(\cdot)\rangle_{L^2(K,\mu)} \\
&= \langle U^{-1}\Phi(\cdot), \nu_x(\cdot)\rangle_{L^2(K,\mu)} + \langle U^{-1}f(\cdot), \nu(\cdot) - \nu_x(\cdot)\rangle_{L^2(K,\mu)} \\
&= \langle U^{-1}f(\cdot), \nu(\cdot)\rangle_{L^2(K,\mu)} \\
&= \langle U^{-1}f(\cdot), \varphi(x)\rangle_{L^2(K,\mu)} \\
&= UU^{-1}f(x) = f(x).
\end{aligned}$$

□

Example. Let us consider that K_f is a finite subset of K, i.e. $K_f = \{(w_i, b_i)\}_{i=1}^m \in (\overline{B_d(0,1)} \times [-R, R])^m$ and that μ is the uniform probability distribution over K_f. Then, $\mathcal{F}_2$ is composed by the functions of the form

$$f(x) = \frac{1}{m} \sum_{i=1}^m \alpha_i \sigma(w_i \cdot x + b_i), \qquad \text{for some } \alpha \in \mathbb{R}^m,$$

and the norm $\gamma_2(\cdot)$ is given by

$$\gamma_2(f)^2 = \inf \left\{ \frac{1}{m} \sum_{i=1}^m \alpha_i^2 \ : \quad \{\alpha_i\}_{i=1}^m \in \mathbb{R}^m \text{ s.t. } f(x) = \frac{1}{m} \sum_{i=1}^m \alpha_i \sigma(w_i \cdot x + b_i) \right\}.$$

This space corresponds to two-layer FCNNs with fixed parameters in the inner layer.

9.5 Comparison between $\mathcal{F}_1$ and $\mathcal{F}_2$

In order to compare $\mathcal{F}_1$ and $\mathcal{F}_2$, let us take $K = \overline{B_d(0,1)} \times [-R, R]$ and consider μ to be the uniform probability distribution on K. Up to a scaling factor, we can assume without loss of generality that K has Lebesgue measure equal to 1. Under this assumption, both spaces contain functions of the form

$$f(x) = \int_K \sigma(w \cdot x + b) \nu(w, b) d(w, b), \qquad \text{for some } \nu(\cdot) \in \mathcal{M}(K).$$

- In the case of $\mathcal{F}_1$ functions, the measure $\nu(\cdot)$ needs to have finite total variation.
- In the case of $\mathcal{F}_2$ functions, the measure $\nu(\cdot)$ must be a function in $L^2(K)$.

Single neurons: We see that single neurons

$$\Phi(x) = \alpha_0 \sigma(w_0 \cdot x + b_0) \qquad \text{for some } \alpha \in \mathbb{R} \text{ and } (w_0, b_0) \in K$$

are in $\mathcal{F}_1$, with variation norm $\gamma_1(\Phi) = |\alpha_0|$, but are not in $\mathcal{F}_2$, since any measure $\nu(\cdot) \in \mathcal{M}(K)$ associated to this function needs to be singular at (w_0, b_0). More generally, any finite combination of neurons is not in $\mathcal{F}_2$, which implies that none of the two-layer FCNN with finitely many hidden neurons is in $\mathcal{F}_2$.

Embedding: Using Jensen's inequality we have

$$\left(\int_K |\nu(w, b)| d(w, b) \right)^2 \leq \int_K |\nu(w, b)|^2 d(w, b) \qquad \forall \nu(\cdot) \in L^2(K),$$

which implies that $\gamma_1(f) \leq \gamma_2(f)$ for all $f \in \mathcal{F}_2$. This means that $\mathcal{F}_1$ contains more functions than $\mathcal{F}_2$, which are typically more regular. Considering $\mathcal{F}_2$ has advantages in terms of generalization and optimization due to the Hilbert space structure, however it lacks expressivity (approximation error is larger) since it is too small.

Feb 21 2024 Wed (12:00-13:00)

Lecture 10: Quantitative estimates in the Barron space

Now that we have defined the space $\mathcal{F}_1$ of functions of the form

$$f(x) = \int_K \sigma(w \cdot x + b)\nu(w,b)d(w,b) \qquad \text{for some } \nu \in \mathcal{M}(K),$$

which is a Banach space endowed with the finite variation norm

$$\gamma_1(f) := \inf\left\{ |\nu|(K) \ : \ \nu \in \mathcal{M}(K) \text{ s.t. } f(x) = \int_K \sigma(w \cdot x + b)\nu(w,b)d(w,b) \quad \forall x \in \Omega \right\},$$

we need to **relate this norm to classical smoothness properties, and provide approximation error bounds in terms of number of neurons and the variation norm.**

We recall that we are considering the ReLU activation function

$$\sigma(s) = \max\{0, x\}.$$

10.1 Variation norm in one dimension

We start with the one-dimensional case $\Omega = [-R, R]$, and consider a twice continuously differentiable function $f \in C^2(\Omega)$. Using the Taylor formula with integral reminder, centred at $-R$, we can write

$$\begin{aligned} f(x) &= f(-R) + f'(-R)(x+R) + \int_{-R}^{x} f''(b)(x-b)db \\ &= f(-R) + f'(-R)(x+R)_+ + \int_{-R}^{R} f''(b)(x-b)_+db \end{aligned}$$

Using the same Taylor formula, this time centred at $+R$, we obtain

$$\begin{aligned} f(x) &= f(R) + f'(R)(x-R) + \int_{R}^{x} f''(b)(x-b)db \\ &= f(R) - f'(R)(-x+R) + \int_{x}^{R} f''(b)(-x+b)db \\ &= f(R) - f'(R)(-x+R)_+ + \int_{-R}^{R} f''(b)(-x+b)_+db. \end{aligned}$$

Averaging both expressions we obtain

$$\begin{aligned} f(x) &= \frac{f(-R)+f(R)}{2} + \frac{f'(-R)}{2}(x+R)_+ - \frac{f'(R)}{2}(-x+R)_+ \\ &\quad + \frac{1}{2}\int_{-R}^{R} f''(b)(x-b)_+db + \frac{1}{2}\int_{-R}^{R} f''(b)(-x+b)_+db, \end{aligned}$$

and noting that $\frac{(x+R)_+ + (-x+R)_+}{2R} = 1$ for all $x \in [-R, R]$, it follows

$$\begin{aligned} f(x) &= \frac{1}{2}\left(\frac{f(-R)+f(R)}{2R} + f'(-R)\right)(x+R)_+ \\ &\quad + \frac{1}{2}\left(\frac{f(-R)+f(R)}{2R} - f'(R)\right)(-x+R)_+ \\ &\quad + \frac{1}{2}\int_{-R}^{R} f''(b)(x-b)_+ db + \frac{1}{2}\int_{-R}^{R} f''(b)(-x+b)_+ db. \end{aligned}$$

Hence, we have

$$\gamma_1(f) \leq \frac{1}{2}\left|\frac{f(-R)+f(R)}{2R} + f'(-R)\right| + \frac{1}{2}\left|\frac{f(-R)+f(R)}{2R} - f'(R)\right| + \int_{-R}^{R} |f''(b)| db.$$

Note:-

1. The above inequality gives a bound of the variation norm of f in terms of the L^1-norm of its second derivatives.
2. In higher dimension $d > 1$, using the Fourier transform representation of the function, we can obtain a similar bound

$$\gamma_1(f) \leq \frac{2}{(2\pi)^d R} \int_{\mathbb{R}^d} |\hat{f}(\xi)|(1 + 2R^2\|\xi\|_2^2) d\xi.$$

Recall that $\widehat{\Delta f}(\xi) = -\|\xi\|_2^2 \hat{f}(\xi)$. Hence, the upper bound depends on the regularity of f and its second derivative. See [21] for more details.

10.2 Approximation error in $\mathcal{F}_1$

The **goal** is now to obtain an estimate of the number of hidden neurons needed to achieve a prescribed approximation error, in terms of the variation norm of the target function. We want to upper bound the L^2 distance from the approximation by m neurons to the target function in terms of its variation norm.

Let us consider a function $g \in \mathcal{F}_1$, which can be written as

$$g(x) = \int_K \sigma(w \cdot x + b)\nu(w, b) d(w, b), \qquad \text{for some measure } \nu \in \mathcal{M}(K).$$

The choice of ν might not be unique.

The target function g is in

$$\mathcal{K} := \{f \in L^2(\Omega) \ : \quad \gamma_1(f) \leq \gamma_1(g)\}$$

which is a bounded convex subset of $L^2(\Omega)$.

Exercise 10.1. Prove that the extreme points of $\mathcal{K}$ are the functions of the form

$$x \mapsto \pm\gamma_1(g)\sigma(w \cdot x + b) \qquad \text{for some } (w, b) \in K.$$

Recall: Let A be a closed convex subset of a vector space X. A point $a \in A$ is an extreme point of A if it cannot be written as a convex combination of any

other two points of A.

Note:-

1. The function g can be seen as a (infinite) convex combination of the extreme points of $\mathcal{K}$.
2. We aim at approximating g as a finite convex approximation of the extreme points of $\mathcal{K}$.
3. In finite dimension, the number of points that suffices to obtain a exact representation is equal to the dimension.
4. However, we are in infinite dimension ($\mathcal{K}$ is a subset of $L^2(\Omega)$), and the situation is more subtle.

In order to achieve a finite (or sparse) approximation of g, we use a conditional gradient descent algorithm (Frank-Wolfe algorithm [19, 3]) applied to the minimization problem

$$\min_{f\in\mathcal{K}} \|f-g\|_{L^2}^2.$$

The solution to this problem is obviously $f^* = g$, however, we are looking for an approximate solution as a finite combination of extreme points of $\mathcal{K}$. The convergence rate of the Frank-Wolfe algorithm will be used to obtain the bound for the approximation error by two-layer FCNN with m neurons in the hidden layer.

Frank-Wolfe algorithm: Let $\mathcal{K}$ be a bounded convex set of a Hilbert space $\mathcal{H}$, and let J be a convex smooth functional on $\mathcal{H}$. Namely, there exists a function $J' : \mathcal{H} \to \mathcal{H}$ and a constant $L > 0$ such that

$$J(g) + \langle J'(g), f-g\rangle \le J(f) \le J(g) + \langle J'(g), f-g\rangle + \frac{L}{2}\|f-g\|_{\mathcal{H}}^2, \qquad \forall f, g \in \mathcal{H}.$$

The algorithm minimizes J over the convex bounded set $\mathcal{K}$. We consider an initial guess $f_0 \in \mathcal{K}$, and then at every step $t \in \mathbb{N}$, the update is computed as

$$f_t = \frac{t-1}{t+1} f_{t-1} + \frac{2}{t+1}\overline{f}_t = f_{t-1} + \frac{2}{t+1}(\overline{f}_t - f_{t-1}),$$

with

$$\overline{f}_t \in \arg\min_{f\in\mathcal{K}} \langle J'(f_{t-1}), f - f_{t-1}\rangle.$$

- Every update is a convex combination of two elements of $\mathcal{K}$, so $f_t \in \mathcal{K}$ for all $t \in \mathbb{N}$.
- Since $\overline{f}_t$ is the solution of a linear minimization problem in a convex set $\mathcal{K}$, we can assume that $\overline{f}_t$ is an extreme point of $\mathcal{K}$ for all $t \in \mathbb{N}$.
- Hence, by taking an extreme point of $\mathcal{K}$ as initial guess, we can deduce that f_t is a convex combination of $t+1$ extreme points of $\mathcal{K}$.

It can be shown that

$$J(f_t) - \inf_{f\in\mathcal{K}} J(f) \le \frac{2L}{t+1}\operatorname{diam}_{\mathcal{H}}(\mathcal{K})^2, \tag{10.1}$$

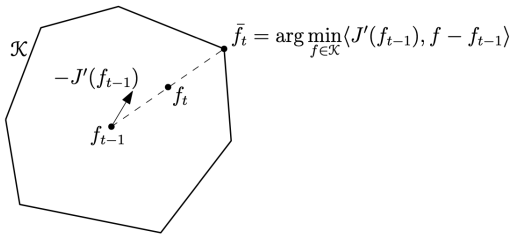

Figure 4: Illustration of one step of the Frank-Wolfe algorithm.

with

$$\mathrm{diam}_{\mathcal{H}}(\mathcal{K})^2 := \sup_{f,g\in\mathcal{K}} \|f-g\|_{\mathcal{H}}^2.$$

See [5, page 240] for a proof of this convergence rate.

Application to our problem: We can apply the above result to obtain an upper bound for the approximation error. Given a target function $g \in L^2(\Omega)$ with finite variation norm $\gamma_1(g)$, we consider the Hilbert space $\mathcal{H} = L^2(\Omega)$, the bounded convex set

$$\mathcal{K} := \{f \in L^2(\Omega) \ : \quad \gamma_1(f) \le \gamma_1(g)\}, \tag{10.2}$$

and the smooth convex functional

$$J(f) = \|f-g\|_{L^2}^2.$$

We have $L = 2$, and the diameter of $\mathcal{K}$ can be proven to be

$$\mathrm{diam}_{\mathcal{H}}(\mathcal{K})^2 = 4\gamma_1(g)^2 \sup_{(w,b)\in K} \|\phi(\cdot;w,b)\|_{L^2}^2,$$

where $\phi(\cdot;w,b) : x \mapsto \sigma(w\cdot x + b)$.

Hence, applying the convergence rate (10.1), taking into account that $\inf_{f\in\mathcal{K}} J(f) = 0$ since $g \in \mathcal{K}$, we have

$$\|f_t - g\|_{L^2}^2 \le \frac{16\gamma_1(g)^2}{t+1} \sup_{(w,b)\in K} \|\phi(\cdot;w,b)\|_{L^2}^2.$$

Since f_t is a convex combination of t functions of the form $x \mapsto \sigma(w\cdot x + b)$ with $(w,b) \in K$, we deduce that the approximation error for the hypothesis set

$$\mathcal{F}_m := \left\{ x \mapsto \sum_{j=1}^m \alpha_j \sigma(w_j \cdot x + b_j) \ : \quad \{\alpha_j\}_{j=1}^m \in \mathbb{R}^m \quad \{(w_j,b_j)\}_{j=1}^m \in K^m \right\}$$

satisfies

$$\varepsilon_{approx} \le \frac{16\gamma_1(g)^2}{m} \sup_{(w,b)\in K} \|\phi(\cdot;w,b)\|_{L^2}^2.$$

Remark: The upper bound of the approximation error does not depend on the dimension, it only depends on the variation norm and the set of parameters $K \subset \overline{B(0,1)} \times [-R,R]$.

Feb 26 2024 Mon (12:00-13:00)

Lecture 11: Convolutional Neural Networks

So far, we have considered problems in which the goal is to approximate an unknown function

$$f^* : \mathbb{R}^d \longrightarrow \mathcal{Y},$$

that takes d-dimensional vectors as input. Although this setting covers most of the cases in practical applications (note that the stored data is typically finite-dimensional), in some situations, the input data comes in some structured way which can be exploited to more efficiently train the NN model. This is the case when the output of the target function is known to satisfy certain properties with respect to transformations in the input space. These properties can be implemented in the design of the NN architecture, notably reducing the number of parameters in the NN and therefore the complexity of the problem. This kind of aspects belong to the field known as *geometric Deep Learning* [7, 13].

Example. Let us consider that the input are RGB images represented in a $N \times N$ pixel grid. Each image is represented by a map

$$X : \mathbb{Z}_N \times \mathbb{Z}_N \longrightarrow [0,1]^3, \quad (\mathbb{Z}_N \text{ denotes the cyclic group of } N \text{ elements}).$$

Let us denote by $\mathcal{X} := L^2(\mathbb{Z}_N \times \mathbb{Z}_N; [0,1]^3)$ the space of functions $\mathbb{Z}_N \times \mathbb{Z}_N \to [0,1]^3$. Every element in $\mathcal{X}$ can be represented by a tensor $\left([0,1]^{N\times N}\right)^3$. We can always flatten the images so that the input space is $\mathbb{R}^d$, with $d = 3N^2$. However, it is not always a good idea, as we illustrate in the following examples.

Definition 11.19

Let $\mathcal{X}$ be the input space and T be a transformation of $\mathcal{X}$, i.e., a map $T : \mathcal{X} \to \mathcal{X}$. We say that a function $f : \mathcal{X} \to \mathcal{Y}$ is invariant under T if

$$f(TX) = f(X) \qquad \forall X \in \mathcal{X}.$$

Example. In a binary classification task, in which the NN must determine whether an image contains a dog or not, the target function is invariant under translations, i.e.

$$f^*(TX) = f^*(X), \qquad \text{for all } X \in \mathcal{X} \text{ and } T \in \mathbb{Z}_N \times \mathbb{Z}_N,$$

where T acts on the space of functions $X : \mathbb{R}^{N\times N} \to [0,1]^3$ by

$$TX(z) = X(z - T), \qquad \forall z \in \mathbb{Z}_N \times \mathbb{Z}_N.$$

Definition 11.20

Let $\mathcal{X}$ be the input space and T be a transformation of $\mathcal{X}$. We say that a function $f : \mathcal{X} \to \mathcal{Y}$ is equivariant with respect to T if there exists tranformation T' of $\mathcal{Y}$ such that

$$f(TX) = T'f(X) \qquad \forall X \in \mathcal{X}.$$

Example. In a segmentation task, in which the NN must determine which pixels in the image correspond to a human face, the target function

$$f^* : \mathcal{X} \longrightarrow L^2(\mathbb{Z}_N \times \mathbb{Z}_N; [0, 1]),$$

associates, to any image, a grey-scale image in which darker pixels correspond to the segmented region (i.e. the face of the human). In this case, the NN function must satisfy

$$f^*(TX) = Tf^*(X) \qquad \text{for all } X \in \mathcal{X} \text{ and } t \in \mathbb{Z}_N \times \mathbb{Z}_N.$$

11.1 Group equivariant Convolutional Neural Networks

A simple way to construct equivariant NNs is to use convolutional neural networks CNNs. Inspired by the above examples, let us consider that the input space is the space of measurable functions

$$X : G \longrightarrow \mathbb{R}^c,$$

where G is a group and $c \in \mathbb{N}$. We also consider a left-invariant measure on G, and denote the integral of $X(\cdot)$ with respect to such measure as $\int_G X(h)dh$. The fact of the measure being left-invariant implies that

$$\int_G X(h)dh = \int_G X(gh)dh, \qquad \forall g \in G.$$

We denote by $L^1(G; \mathbb{R}^c)$ the space of integrable functions $G \to \mathbb{R}^c$ with respect to that measure.

Note:-

In the case G is a finite group with N elements, then the input space is isomorphic to $\mathbb{R}^{c\times N}$, which is finite dimensional.

As discussed in previous lectures, in DL models, the NN makes predictions based on a set of input features. In this case, we consider that the input features are represented by a map defined on a group, which gives some structure to the input data. This structure is crucial in the design of convolutional neural networks.

Let us define the left group action of G over $L^1(G; \mathbb{R}^c)$ given by

$$(g, X) = gX(\cdot), \quad \text{where} \quad gX(z) = X(g^{-1}z) \quad \forall z \in G. \tag{11.1}$$

Note that this group action defines, for any $g \in G$, a transformation of the input space $L^1(G; \mathbb{R}^c)$.

Definition 11.21

Let us consider a map $\kappa : G \longrightarrow \mathbb{R}^{c_2 \times c_1}$, which associates, to any $g \in G$, a linear operator $\mathbb{R}^{c_1} \to \mathbb{R}^{c_2}$, where c_1 and c_2 are positive integers.

We define the convolution operation of κ with $X \in L^1(G; \mathbb{R}^{c_1})$ as

$$[\kappa \star X](g) = \int_G \kappa(g^{-1}h) X(h) dh.$$

The convolution with $\kappa(\cdot)$ defines a linear operator

$$\begin{array}{rccc} W_\kappa : & L^1(G;\mathbb{R}^{c_1}) & \longrightarrow & L^1(G;\mathbb{R}^{c_2}) \\ & X(\cdot) & \longmapsto & [\kappa \star X](\cdot) \end{array}$$

We shall refer to a linear operator of this form as **convolutional layer**.

The map $\kappa : G \longrightarrow \mathbb{R}^{c_2 \times c_1}$ is sometimes referred to as kernel or filter.

We can now prove that convolutional layers are equivariant with respect to the group action defined in (11.1).

Lemma 11.1. For any $\kappa : G \longrightarrow \mathbb{R}^{c_2 \times c_1}$, the convolutional layer W_κ as per Definition 11.1 satisfies

$$g(W_\kappa X) = W_\kappa(gX), \qquad \forall g \in G \text{ and } X \in L^2(G; \mathbb{R}^{c_1}),$$

where $gF(z) = F(g^{-1}z)$ for any $F \in L^2(G; \mathbb{R}^{c_1})$ or $F \in L^2(G; \mathbb{R}^{c_2})$.

Proof. For any $z \in G$, we can write $W_\kappa(gX)(z)$ as

$$W_\kappa(gX)(z) = \int_G \kappa(z^{-1}h) gX(h) dh = \int_G \kappa(z^{-1}h) X(g^{-1}h) dh.$$

Using the fact that the measure is left-invariant, we obtain

$$\begin{aligned} W_\kappa(gX)(z) &= \int_G \kappa(z^{-1}gh) X(h) dh \\ &= \int_G \kappa\left((g^{-1}z)^{-1}h\right) X(h) dh \\ &= [\kappa \star X](g^{-1}z) = (W_\kappa X)(g^{-1}z), \end{aligned}$$

which in turn implies that $W_\kappa(gX)(z) = g(W_\kappa X)(z)$ for all $z \in G$. □

Note:-

A convolutional layer is a special type of feedforward NN with one layer. The difference with respect to a fully connected layer feedforward NN is that by restricting the linear operation to be of the form as in Definition 11.1, the number of parameters in the layer is significantly smaller.

A Convolutional Neural Network CNN consists on the concatenation of several convolutional layers, composed with nonlinear transformations

$$\Phi(X;\theta) = W_{\kappa_L} \circ \sigma \circ W_{\kappa_{L-1}} \circ \ldots \circ \sigma \circ W_{\kappa_0}(X),$$

where the parameter $\theta = (\kappa_0, \ldots, \kappa_L)$ are the convolutional filters and $\sigma(\cdot)$ is a non-linear transformation.

Note that when $\sigma(\cdot)$ is an element wise activation function, $\sigma \circ W_\kappa$ is equivariant with respect to the group action, and thus so is the whole NN. However, other nonlinearities like max-pooling are not equivariant.

Example. Let us consider the simple case $c_1 = c_2 = 1$ and G being a finite group with N elements. Note that the input and output space coincide $\mathcal{X} = \mathcal{Y} = L^1(G;\mathbb{R})$ and are isomorphic to $\mathbb{R}^N$.

In this case, the convolution filter is a function $\kappa : G \to \mathbb{R}$, which can be parameterized with only N parameters. This means that a convolutional layer has at most N parameters. This number can be even smaller, as one may choose a filter with small support to only allow local interactions.

On the contrary, the linear transformation in a fully connected layer, which in this case would correspond to a linear map $\mathbb{R}^N \to \mathbb{R}^N$, would need of N^2 parameters.

In the case of $N = 64 \times 64$ images, the number of parameters of a fully connected layer would be $(64 \times 64)^2 \sim 16 \cdot 10^6$, whereas for a convolutional layer, we would only need, at most, $64 \times 64 = 4096$ parameters. In practice, if we consider a 16×16 filter, then we would only need 256 parameters.

11.2 Scattering Transform

One of the first theoretical contributions to the understanding of the mathematical properties of CNNs is [24] (see also [8]). The approach in this work is to consider convolutional layers with fixed filters (without learnable parameters) that work as feature extractors of the input signal. The goal is to then feed this representation in classifier such as a FCNN. The idea is that the convolutional layers should extract features from the signal, factoring out invariances that are irrelevant for the subsequent classification.

The most typical example of translation invariant representation of a signal is the modulus of the Fourier transform

$$\begin{array}{rccc} \Psi: & L^2(\mathbb{R}^2) & \longrightarrow & L^2(\mathbb{R}^2) \\ & X(\cdot) & \longmapsto & \left|\widehat{X}\right|(\cdot) \end{array}$$

Note that if, for some $\tau \in \mathbb{R}^2$, we denote by $L_\tau X(\cdot) \in L^2(\mathbb{R}^2)$ the τ-translation of $X(\cdot)$, defined as $L_\tau X(z) = X(z-\tau)$, then it holds that

$$\widehat{L_\tau X}(\omega) = \widehat{X}(\omega) e^{-2\pi i \langle \tau, \omega \rangle}, \qquad \forall \omega \in \mathbb{R}^2.$$

Hence, we have that $\Psi(X) = \Psi(L_\tau X)$.

The modulus of the Fourier transform is invariant under translations, and therefore, any signal classifier that takes as input $\Psi(X)$ is invariant under translations as well.

The idea of the scattering transform, introduced in [24] is to provide a signal representation $\Psi(\cdot)$ which is not only translation invariant, but also approximately invariant under small deformations, i.e. if we consider a vector field $\tau : \mathbb{R}^2 \to \mathbb{R}^2$, then the deformed signal

$$X_\tau(z) = X(x - \tau(z))$$

satisfies

$$\|\Psi(X) - \Psi(X_\tau)\| \leq C(\tau, \nabla\tau, \nabla^2\tau).$$

Definition of the scattering transform

Roughly speaking, the scattering transform is defined by applying iteratively a wavelet transform, followed by a modulus point wise non-linearity and a subsequent low-pass filter. Let ψ be a wavelet[3] on $\mathbb{R}^2$ of the form

$$\psi(u) = \theta(u)e^{i\eta\cdot u},$$

for some $\eta \in \mathbb{R}^2$ and $\theta(\cdot)$ a real function with fast decay both in space and frequency (for instance $\theta(u) = e^{-u^2}$)).

We also consider a finite subgroup G of $O(2)$, the orthogonal group of 2×2 matrices, and the set $\Lambda_J = \{2^j r \ ; \quad j \in \{0, \dots, J\},\ r \in G\}$. We define the family of dilated and rotated copies of $\psi(\cdot)$ for each element $2^j r \in \Lambda_J$ as

$$\psi_{2^j r}(u) = 2^{2j}\psi(2^j r^{-1} u).$$

The wavelet transform of $X \in L^2(\mathbb{R}^2)$ is defined as the collection of wavelet coefficients

$$W[\lambda]X(\cdot) = X \star \psi_\lambda(\cdot), \qquad \text{for } \lambda \in \Lambda_J.$$

For any $\lambda \in \Lambda$, we define the scattering coefficient $S_J[\lambda]$ as

$$S_J[\lambda]X = \int_{\mathbb{R}^2} |W[\lambda]X(u)|du$$

These coefficients provide features of the signal X which are translation invariant. However, the integration looses a lot of information about the signal. In order to extract more features, we can consider higher order scattering coefficients, defined for paths $(\lambda_1, \lambda_2, \dots, \lambda_m) \in \Lambda^m$ by

$$S_J[\lambda_1, \lambda_2, \dots, \lambda_m] = \int_{\mathbb{R}^2} W_m(x)du,$$

where $W_m(\cdot)$ is obtained iteratively as

$$\begin{cases} W_1(\cdot) = |X \star \psi_{\lambda_1}(\cdot)| \\ W_i(\cdot) = |W_{i-1} \star \psi_{\lambda_i}(\cdot)|. \end{cases}$$

See Figure 5 for a diagram of the scattering coefficients.

[3] A wavelet on $\mathbb{R}^2$ is a function $\mathbb{R}^2 \to \mathbb{R}$ which is localised in space and frequency. It typically looks like a sinusoidal function in a neighborhood of the origin and decays rapidly to 0.

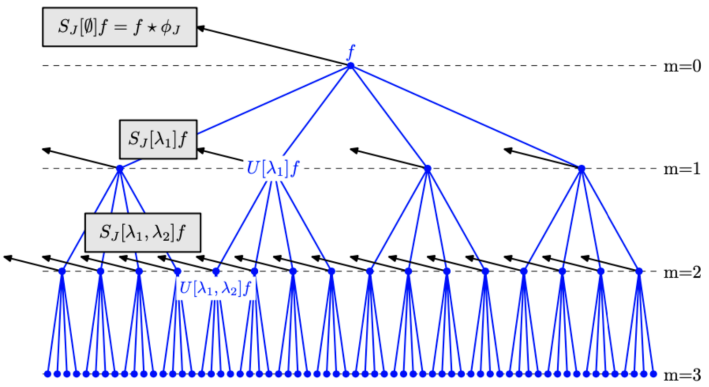

Figure 5: Diagram of the Scattering Neural Network. Figure taken from [8].

Note:-

We can see that the scattering transform is a multilayer NN with convolutional layers in which the parameters are fixed and therefore non-trainable.

- The output of the scattering transform consists of the scattering coefficients $S_J[\lambda_1, \ldots, \lambda_n]$ for different paths $(\lambda_1, \ldots, \lambda_n)$. Differently to a conventional NN, the outputs are generated after every layer, so different coefficients may correspond to different depths.
- The scattering coefficients provide approximately deformation invariant features of the input image, associated to different localized frequencies. These features can then be used for classification or regression tasks, in which the output is equivariant with respect to deformations.
- In general, one doesn't use all the possible paths up to a certain depth. Moreover, one can constrain the choice of paths to frequency decreasing paths. Indeed, when convolving the signal with a wavelet of certain signal, the output cannot have higher frequencies, so it doesn't make sense to apply a convolution to the output with a wavelet of higher frequency.

Feb 28 2024 Wed (12:00-13:00)

Lecture 12: Neural Networks and its dynamics

DNNs integrate low/mid/high level features from data. The level of the features can be enriched with the depth, i.e. by stacking more and more layers. However, the specific form of feedforward FCNNs as compositions of affine transformations and non-linearities

$$\Phi_L(x;\theta) = W_L\sigma\left(W_{L-1}\sigma\left(\cdots W_1\sigma(W_0x)\right)\right)$$

brings us another question concerning the training of such NNs by means of gradient based methods.

Question: Is training Deeper NNs as easy as their shallower counterparts?

Indeed, when computing derivatives of these "nested" functions (specially partial derivatives w.r.t. parameters in the first layers), by the chain rule, these derivatives happen to be the product of the derivatives of many functions. This may result in partial derivatives which become too big or too small, preventing gradient methods from being efficient. This phenomenon is known as **exploding and vanishing gradient** and can be an obstacle during the training of very deep NNs.

Next, we illustrate this phenomenon in a simple case (deep NNs with a single neuron in each layer), in which the computations are simpler.

12.1 Vanishing/exploding gradient

Let us consider a $L+1$-layer NN with 1 neuron in each layer and no biases, i.e. we consider functions of the form

$$\Phi_L(x;\theta) = w_L\sigma\left(w_{L-1}\sigma\left(\cdots w_1\sigma(w_0x)\right)\right).$$

where the parameter θ is a vector $\theta = (w_0, w_1, \ldots, w_L) \in \mathbb{R}^{L+1}$.

The output of the NN can be written as $\Phi_L(x,\theta) = \phi_L$, where ϕ_L is obtained by means of the following recursive formula:

$$\begin{cases} \phi_0 = w_0x \\ \phi_i = w_i\sigma\left(\phi_{i-1}\right), \quad i \in \{1,\ldots,L\}. \end{cases} \tag{12.1}$$

Note that, for each $i \in \{1,\ldots,L\}$, the output of the i-th layer of the NN, denoted by ϕ_i is a function of the parameter w_i and the output of the previous layer ϕ_{i-1}.

Let us compute the partial derivatives of Φ_L with respect to the parameters in the last layers, and then deduce the recurrence formula for all the other derivatives.

Derivative with respect to w_L: Since $\Phi_L(x;\theta) = w_L\phi_{L-1}$, and ϕ_{L-1} does not depend on the parameter w_L, the partial derivative of $\Phi_L(x,\theta)$ with respect to w_L is given by

$$\frac{d}{dw_L}\Phi_L(x,\theta) = \sigma\left(\phi_{L-1}\right).$$

Derivative with respect to w_{L-1}: Likewise, $\phi_{L-1} = w_{L-1}\sigma(\phi_{L-2})$, and ϕ_{L-2} does not depend on w_{L-1}, so

$$\begin{aligned}\frac{d}{dw_{L-1}}\Phi_L(x,\theta) &= \frac{d}{d\phi_{L-1}}\phi_L \frac{d}{dw_{L-1}}\phi_{L-1} \\ &= w_L\dot{\sigma}(\phi_{L-1})\sigma\left(\phi_{L-2}\right)\end{aligned}$$

Derivative with respect to w_{L-2}: Again, $\phi_{L-2} = w_{L-2}\sigma(\phi_{L-3})$, and ϕ_{L-3} does not depend on w_{L-2}, so

$$\begin{aligned}\frac{d}{dw_{L-2}}\Phi_L(x,\theta) &= \frac{d}{d\phi_{L-2}}\phi_L \frac{d}{dw_{L-2}}\phi_{L-2} \\ &= \frac{d}{d\phi_{L-1}}\phi_L \frac{d}{d\phi_{L-2}}\phi_{L-1}\frac{d}{dw_{L-2}}\phi_{L-2} \\ &= w_L\dot{\sigma}(\phi_{L-1})w_{L-1}\dot{\sigma}(\phi_{L-2})\sigma\left(\phi_{L-2}\right)\end{aligned}$$

Derivative with respect to w_i: In general, we can write

$$\frac{d}{dw_i}\Phi_L(x,\theta) = \frac{d}{d\phi_i}\phi_L \frac{d}{dw_i}\phi_i,$$

where $\dfrac{d}{d\phi_i}\phi_L$ satisfies

$$\frac{d}{d\phi_i}\phi_L = \frac{d}{d\phi_{i+1}}\phi_L \frac{d}{d\phi_i}\phi_{i+1} = w_{i+1}\dot{\sigma}(\phi_i)\frac{d}{d\phi_{i+1}}\phi_L$$

and

$$\frac{d}{dw_i}\phi_i = \sigma(\phi_{i-1}).$$

Note:-

In conclusion, the partial derivative of $\Phi_L(x;\theta)$ with respect to the parameter w_i is given by

$$\frac{d}{dw_i}\Phi_L(x;\theta) = \widehat{\phi}_i\sigma(\phi_{i-1}), \tag{12.2}$$

where $\widehat{\phi}_i$ is given by the recursive formula

$$\begin{cases}\widehat{\phi}_L = 1 \\ \widehat{\phi}_i = \widehat{\phi}_{i+1}w_{i+1}\dot{\sigma}(\phi_i), \quad i \in \{0,\dots,L-1\}.\end{cases} \tag{12.3}$$

Note that the ϕ_i's are obtained by using the forward-recursive formula (12.1), and the $\widehat{\phi}_i$'s are obtained by means of the backward recursive formula (12.3), using the values ϕ_i. This method to compute the gradients iteratively is known in the DL literature as **back propagation**.

Using (12.2) and (12.3), we have that the partial of $\Phi_L(x;\theta)$ derivatives are given by

$$\frac{d}{dw_i}\Phi_L(x;\theta) = \sigma(\phi_{i-1})\prod_{j=i}^{L-1} w_{j+1}\dot{\sigma}(\phi_j)$$

We observe that if, for instance, we take weights in the unit ball $B(0,1)$, then the product in the right-hand-side becomes very small for i large, which implies that the parameters in the first layers of the NN change very little at every step of SGD.

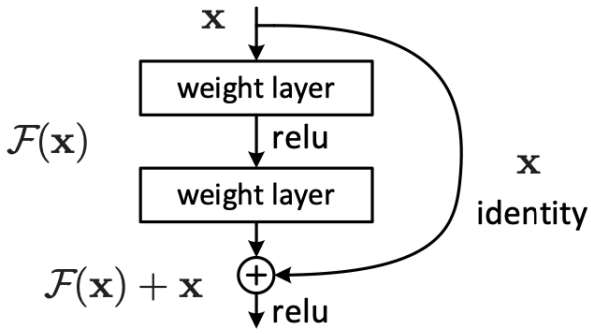

Figure 6: Diagram of a Residual block, which constitutes one layer of a Deep Residual NN. Figure taken from [16]

12.2 Residual Neural Networks

One of the possibilities to deal with the problems of training Deep Neural Networks is to use Residual Neural Networks (ResNet), proposed in [16] in 2016, which are NNs with skip connections in a residual fashion, i.e. instead of stacking layer as in (12.1), the output of the NN

$$\Phi_L(x;\theta) = w_L \phi_L,$$

where ϕ_L is obtained by the following recursive formula:

$$\begin{cases} \phi_0 = x \\ \phi_{i+1} = \phi_i + \gamma\sigma(w_i\phi_i) \quad \text{for } i \in \{0,\dots,L-1\}, \end{cases} \tag{12.4}$$

for some $\gamma > 0$ fixed, and $\theta = (w_0, w_1, \dots, w_L) \in \mathbb{R}^L$ are the parameters of the NN. See Figure 6 for an illustration of a residual block. An important observation of ResNet networks is that (12.4) can be seen as a dynamical system with time-step equal to γ.

Next we compute the partial derivatives of $\Phi_L(x;\theta)$. We have

$$\frac{d}{dw_L}\Phi_L(x;\theta) = \phi_L$$

and for all $i \in \{0,1,\dots,L-1\}$ we have

$$\frac{d}{dw_i}\Phi_L(x;\theta) = w_L \frac{d}{d\phi_{i+1}}\phi_L \frac{d}{dw_i}\phi_{i+1}.$$

Using (12.4), one obtains

$$\frac{d}{dw_i}\phi_{i+1} = \gamma\dot{\sigma}(w_i\phi_i)w_i.$$

On the other hand, if we denote $\hat{\phi}_i = \dfrac{d}{d\phi_i}\phi_L$, one can obtain the recursive formula

$$\begin{aligned} \hat{\phi}_i &= \frac{d}{d\phi_{i+1}}\phi_L \frac{d}{d\phi_i}\phi_{i+1} \\ &= \hat{\phi}_{i+1}\left(1 + \gamma\dot{\sigma}(w_i\phi_i)\right) w_i \\ &= \hat{\phi}_{i+1} + \gamma\dot{\sigma}(w_i\phi_i)w_i\hat{\phi}_{i+1} \end{aligned}$$

Note:-

In conclusion, the partial derivative of $\Phi_L(x;\theta)$ with respect to the parameter w_i, for $i \in \{0, 1, \dots, L-1\}$ is given by

$$\frac{d}{dw_i}\Phi_L(x;\theta) = \gamma w_L \dot{\sigma}(w_i\phi_i) w_i \widehat{\phi}_{i+1}, \tag{12.5}$$

where $\widehat{\phi}_i$ is given by the recursive formula

$$\begin{cases} \widehat{\phi}_L = 1 \\ \widehat{\phi}_i = \widehat{\phi}_{i+1} + \gamma \dot{\sigma}(w_i\phi_i) w_i \widehat{\phi}_{i+1}, \quad i \in \{0, \dots, L-1\}. \end{cases} \tag{12.6}$$

Note that the ϕ_i's are obtained by using the forward-recursive formula (12.4), and the $\widehat{\phi}_i$'s are obtained by means of the backward recursive formula (12.6), using the values ϕ_i. We observe that in this case, the **back propagation** has also a ResNet architecture, i.e. it can be seen as a backward in time dynamical system with time-step equal to γ.

Using (12.6), we can compute

$$\begin{aligned} \widehat{\phi}_i - \widehat{\phi}_L &= \sum_{j=i}^{L-1} \left(\widehat{\phi}_j - \widehat{\phi}_{j+1} \right) \\ &= \gamma \sum_{j=i}^{L-1} \dot{\sigma}(w_j\phi_j) w_j \widehat{\phi}_{j+1}, \end{aligned}$$

which yields

$$\widehat{\phi}_i = 1 + \gamma \sum_{j=i}^{L-1} \dot{\sigma}(w_j\phi_j) w_j \widehat{\phi}_{j+1}.$$

We see that the partial derivatives of the NN with respect to the parameters are simpler when one considers a ResNet architecture instead of fully connected (FC) NNs. The most important feature is the fact that it doesn't involve the product of the derivatives along the layers, but rather the sum. The choice of the hyperparameter $\gamma > 0$ can be used to control the size of the gradient, a typical choice being to take γ proportional to $1/L$. In the next section we will see why this is a natural choice.

Note:-

However, ResNet architectures have a main **limitation** as compared to FC networks. The fact that the output of each layer is the summation of the activation function and the output of the previous layer implies that the outputs of all the layers must have the same dimension, or in other words, all the layers must have the same number of neurons.

Let us consider a ResNet NN $\Phi(\cdot;\theta) : \Omega \subset \mathbb{R}^d \to \mathbb{R}$. The output of the ResNet is given by

$$\Phi(x;\theta) = W_L \cdot \phi_L + b_L, \tag{12.7}$$

where $\phi_L \in \mathbb{R}^d$ is obtained as

$$\begin{cases} \phi_0 = x \\ \phi_{i+1} = \phi_i + \gamma\sigma(W_i\phi_i + b_i) \quad \text{for } i \in \{0, \dots, L-1\}. \end{cases} \tag{12.8}$$

Here the parameters $\theta = \{(W_i, b_i)\}_{i=0}^L$ satisfy

$$W_i \in \mathbb{R}^{d\times d}, \quad b_i \in \mathbb{R}^d \quad \text{for } i \in \{0, 1, \dots, L-1\}$$

and $W_L \in \mathbb{R}^d$ and $b_L \in \mathbb{R}$.

When using ResNets, we do not have the freedom to change the number of neurons over the layers, which has to be equal to the input dimension. This is quite undesirable, specially in high-dimensional problems like imaging, where the input dimension is of the order of the number of pixels. **In practice**, one can always use a few layers to reduce the dimension of the input data (for instance convolutional layers in the case of images), and then stack a very Deep ResNet to enjoy the power of depth without having to care much about the vanishing/exploding gradient phenomenon during training.

12.3 Infinite-depth limit and Neural ODEs

Another interesting feature of ResNet architectures (12.8) is the fact that they can be seen as parametrized dynamical systems, or controlled dynamical systems, which allows one to study Deep Learning problems from an optimal control theory perspective [14].

We can indeed look at the index in the system (12.8) as a pseudo-time. For any depth $L \in \mathbb{N}$, consider the uniform grid in the interval $[0, 1]$ denoted by

$$\{t_n\}_{n=0}^L \subset [0, 1], \qquad \text{with } t_n = \frac{n}{L},$$

and set the parameter γ in (12.8) as

$$\gamma = \Delta t = \frac{1}{L}.$$

The output of the ResNet NN defined in (12.7) can then be written as

$$\Phi_L(x; \theta) = W_L \cdot \phi_L(x, 1),$$

where $\phi(x, \cdot)$ is the solution to the discrete dynamical system

$$\begin{cases} \phi_L(x, t_{n+1}) = \phi_L(x, t_n) + \Delta t F(\phi_L(x, t_n), W(t_n), b(t_n)) \quad \text{for } n \in \{0, 1, \dots, L-1\} \\ \phi_L(x, 0) = 0, \end{cases}$$

where

$$F(\phi, W, b) = \sigma(W\phi + b), \qquad \text{for any } \phi \in \mathbb{R}^d,\ (W, b) \in \mathbb{R}^{d\times d} \times \mathbb{R}^d.$$

Here, the parameters are functions

$$(W, b) : [0, 1] \longrightarrow K \subset \mathbb{R}^{d\times d} \times \mathbb{R}^d,$$

where K is a compact parameter space.

If we look at the recursive formula for $\phi_L(x, t_n)$, we recognize an explicit Euler scheme for ODEs, i.e.

$$\frac{\phi_L(x, t_{n+1}) - \phi_L(x, t_n)}{t_{n+1} - t_n} = F(\phi_L(x, t_n), W(t_n), b(t_n)).$$

Hence, by standard arguments, one can prove that, as the number of layers goes to infinity $L \to +\infty$, and hence $t_{n+1} - t_n = \Delta t = 1/L$ goes to 0, the function ϕ_L converges to the solution of the non-linear ODE

$$\begin{cases} \dfrac{d}{dt}\phi(x,t) = F(\phi(x,t), W(t), b(t)) \quad \text{for } t \in (0,1) \\ \phi(x,0) = 0. \end{cases} \tag{12.9}$$

This known in the literature as Neural ODE. The empirical risk minimization problem for regression can then be formulated as

$$\underset{\substack{(W,b)\in L^1((0,1);K) \\ W_L \in \mathbb{R}^d}}{\text{minimize}} \ \frac{1}{n}\sum_{i=1}^{n} \left(W_L \cdot \phi(x^{(i)}, 1) - y^{(i)}\right)^2 \qquad \text{subject to} \quad \phi(x^{(i)}, t) \text{ solves (12.9)}.$$

This shows that, in the infinite-depth limit, the risk minimization problem associated to Deep NNs with ResNet architecture can be seen as an approximation of an Optimal Control Problem.

12.4 Stability and dissipativity

A desirable property of a neural network is **stability**. Roughly speaking, this property means that a small perturbation in the input of the neural network should have a small effect in the output.

Definition 12.22

Let $\Phi(x;\theta)$ be a neural network with parameter $\theta \in \mathcal{P}$. We say that Φ is **stable** if there exists $C > 0$ such that

$$\|\Phi(x_1;\theta) - \Phi(x_2;\theta)\| \le C\|x_1 - x_2\| \qquad \forall (x_1, x_2) \in \mathcal{X}.$$

Let us consider the case of a NN of the form

$$\Phi(x;\theta) = W_L Z_\theta(T;x),$$

for some $W_L \in \mathbb{R}^d$, and where $Z_\theta(T;x) = z(T)$, the solution to the neural ODE

$$\begin{cases} \dot{z}(t) = \sigma(W(t)z(t)) \quad t \in (0,T) \\ z(0) = x. \end{cases} \tag{12.10}$$

The parameters θ of such NN are the vector W_L along with the continuous map $W : (0,T) \to \mathbb{R}^{d\times d}$.

In this case, we have the following estimate

$$\|\Phi(x_1;\theta) - \Phi(x_2;\theta)\| = \|W_L Z_\theta(T;x_1) - W_L Z_\theta(T;x_2)\| \le \|W_L\|\|x_1 - x_2\| e^{TL},$$

where $L = \mathrm{Lip}(\sigma) \max_{t\in(0,T)} \|W(t)\|$. Here, $\mathrm{Lip}(\sigma)$ is the Lipschitz constant of the activation function, and for every $W \in \mathbb{R}^{d\times d}$, $\|W\|$ represents the operator norm.

When we look at a Neural Network from a dynamical system perspective, an interesting question is whether the trajectory associated to the NN is **contractive**. Consider an ODE of the form

$$\begin{cases} \dot{z}(t) = f(z(t), w(t)) \quad t > 0 \\ z(0) = z_0. \end{cases} \tag{12.11}$$

Suppose $\exists \nu \in \mathbb{R}$ such that

$$\langle f(z_2, w(t)) - f(z_1, w(t)),\, z_2 - z_1 \rangle \leq \nu \|z_2 - z_1\|^2, \qquad \forall t > 0, \tag{12.12}$$

then it holds that

$$\|z_2(t) - z_1(t)\| \leq \|z_2(0) - z_1(0)\| e^{\nu t},$$

where $z_1(t)$ and $z_2(t)$ are the solutions to (12.11) with initial condition z_1 and z_2 respectively. We say that the dynamical system (12.11) is dissipative if it holds with $\nu < 0$.

12.4.1 Dissipative models and gradient flows

Here we give some guidelines that allow one to construct a Neural ODE which is dissipative. One can easily prove that, if we consider a matrix trajectory $t \mapsto A(t)$, such that $A(t)$ is skew-symmetric for all $t > 0$, then the flow $\dot{z} = A(t)z(t)$ preserves the L^2-norm, i.e.

$$\|z(t)\| = \|z(0)\| \qquad \forall t > 0.$$

Let us consider

$$W(t) = \underbrace{S(t) - S(t)^T}_{A(t)} - \gamma I_d,$$

where $\gamma > 0$, $S(t)$ is an arbitrary matrix for any $t > 0$, and I is the identity in $\mathbb{R}^{d\times d}$. One can prove prove that the flow

$$\dot{z}(t) = (A(t) - \gamma I) z(t)$$

satisfies

$$\|z(t)\| = \|z(0)\| e^{-\gamma t} \qquad \forall t > 0.$$

This dissipative behaviour can however change when we add a non-linearity to our model.

In [31], it is proposed the following construction of dissipative nerual ODEs. For a Lipschitz activation function $\sigma(\cdot)$, let us consider the flow

$$\dot{z}(t) = -A(t)^T \sigma(A(t)z(t) + b(t)), \tag{12.13}$$

which can be rewritten as

$$\dot{z}(t) = -\nabla_z V(z(t); A(t), b(t)), \quad \text{where} \quad V(z; A, b) = \langle \gamma(Az + b), \mathbf{1} \rangle,$$

and where $\gamma : \mathbb{R} \to \mathbb{R}$ is a differentiable function such that $\gamma'(s) = \sigma(s)$ for all $s \in \mathbb{R}$.

The following result is proven in [31].

Theorem 12.1. 1. Let $V(z; A(t), b(t))$ be twice differentiable and uniformly convex with respect to z for all $t > 0$. Then

$$f(z; A(t), b(t)) = -\nabla_z V(z(t); A(t), b(t))$$

satisfies (12.12) for all $t > 0$ with $\nu < 0$.

2. Suppose $\sigma(s)$ is absolutely continuous and $0 \leq \sigma'(s) \leq 1$ for a.e. $s \in \mathbb{R}$. Then, (12.13) satisfies (12.12) with $-\mu_*^2 \leq \nu_\sigma \leq 0$, where $\mu_* = \min_t \mu(t)$, where $\mu(t)$ is the smallest eigenvalue of $A(t)$.

The first result is a general property of gradient flows associated to convex functionals. The second result uses that the flow in (12.13) can be written as a gradient flow associated to a convex functional.

12.4.2 Hamiltonian flows

In the previous subsection we saw how it is possible to construct Neural ODEs that are a gradient flow of a convex functional. One may take inspiration from mechanical systems and introduce the Hamiltonian framework. Let us consider a separable Hamiltonian of the form

$$H(t, z, p) = T(t, p) + V(t, z),$$

where $T(t, p)$ and $V(t, z)$ represent the kinetic and the potential energy respectively. This leads to the following Hamiltonian system

$$\begin{cases} \dot{z}(t) = \partial_p H(t, z(t), p(t)) = \partial_p T(t, p(t)) \\ \dot{p}(t) = -\partial_z H(t, z(t), p(t)) = -\partial_z V(t, z(t)). \end{cases} \tag{12.14}$$

In [9], the following model is suggested

$$\begin{cases} \dot{z}(t) = A_1(t)^T \sigma_1(A_1(t)p(t) + b_1(t)) \\ \dot{p}(t) = -A_2(t)^T \sigma_2(A_2(t)z(t) + b_2(t)) \end{cases} \tag{12.15}$$

which is a system of ODEs, whose dynamics correspond to the Hamiltonian system associated to the Hamiltonian $H(t, z, p) = T(t, p) + V(t, z)$ with

$$T(t, p) = \langle \gamma_1(A_1(t)p + b_1(t)), \mathbf{1} \rangle$$

and

$$V(t, z) = \langle \gamma_2(A_2(t)z + b_2(t)), \mathbf{1} \rangle,$$

where $\gamma_1, \gamma_2 : \mathbb{R} \to \mathbb{R}$ are differentiable functions such that $\gamma_1'(s) = \sigma_1(s)$ and $\gamma_2'(s) = \sigma_2(s)$ for a.e. $s \in \mathbb{R}$.

A simple example is to take $A_1(t) = I$, $b_1(t) = 0$ and $\sigma_1(t) = t$ for all t. In this case, using the fact that $\dot{z}(t) = p(t)$, the dynamics can be written as the following second-order differential equation

$$\ddot{z}(t) = -A_2(t)^T \sigma_2(A_2(t)z(t) + b_2(t)).$$

The finite-difference approximation of this ODE gives raise to a NN architecture with finite number of layers, which is known in the literature as *momentum ResNet*.

Mar 4 2024 Mon (12:00-13:00)

Lecture 13: Generative models I: Diffusion models

This is the first of two lectures about **generative models**. The main **goal** of generative models is to learn a complex probability distribution (typically in a high-dimensional domain) from a finite number of samples, which are assumed to be drawn form the target distribution. Once the model is trained, it can be used to generate new data from the target distribution which is not necessarily in the dataset. Generative models can also be used to do inference, help the training of other models or to solve inverse problems. The most popular approaches to construct generative models are Normalizing Flows [27], Generative Adversarial Networks [15], Variational Autoencoders [20] and Diffusion Models [28, 29].

13.1 Diffusion models

The **key idea** in diffusion models is to consider a stochastic process in the data space which, after a long run, the probability distribution of the process converges to a known simple distribution, which is also independent of the initial state of the process. This process, that we will refer to as forward diffusion, can be seen as adding noise to the data points. The generative model is then obtained by approximating the reverse stochastic process. In this way, if one considers the stationary distribution associated to the forward diffusion process, one can recover the target distribution by running the reverse stochastic process.

Definition 13.23: Forward diffusion

Let $\mathcal{X}$ be a measurable space. Let $q(\cdot)$ be the probability density function associated to an unknown probability distribution on $\mathcal{X}$, and let $\pi(\cdot)$ be a known simple probability distribution on $\mathcal{X}$. We call **forward diffusion process** to a stochastic process $\{x_t\}_{t\in\mathbb{N}}$ satisfying the three following properties:

1. x_0 is a random variable with probability density function $q(\cdot)$.
2. For all $t \geq 1$, the random variable $x_t|x_{t-1}$ follows a simple distribution.
3. For any $x_0 \in \mathcal{X}$, the probability density function associated to the random variable $x_t|x_0$ converges to $\pi(\cdot)$ as $t \to \infty$.

The conditional probability distribution for $x_t|x_{t-1}$ has to be chosen in such a way that the unique equilibrium distribution is π, no matter what the initial distribution q is.

The **goal** is to approximate the reverse diffusion process through a parametrized model $p_\theta(x_{t-1}|x_t)$ which represents the probability distribution of x_{t-1} conditioned to x_t. Then, the generative model is obtained as the random variable $\hat{X}_0 = x_0$, where x_0 is obtained as

$$\begin{cases} x_T \sim \pi \\ x_{t-1} \sim p_\theta(x_{t-1}|x_t). \end{cases}$$

In order to train the model, i.e. choose a suitable parameter θ such that the random variable $\hat{X}_0$ has a probability density close to $q(\cdot)$, one can use the dataset $\{x^{(i)}\}_{i=1}^n$,

which is assumed to be an i.i.d. sample from $q(\cdot)$.

13.2 Denoising diffusion

Here we review one of the possible constructions of a diffusion model. This construction is proposed in [28, 17].

Forward diffusion: Let $\beta_1, \dots, \beta_T \in (0,1)$ be a sequence of T constants, with $T \in \mathbb{N}$ large. Given the state of the process at time $t-1$, denoted by x_{t-1}, the state at time t, denoted by x_t, follows a normal distribution as follows:

$$x_t | x_{t-1} \sim N(\sqrt{1-\beta_t} x_{t-1}, \beta_t I).$$

This process can equivalently be written as

$$x_t = \sqrt{1-\beta_t} x_{t-1} + \beta_t z_t,$$

where $z_t \sim N(0, I)$. We denote the conditional distribution associated to this diffusion process as $q_t(x_t | x_{t-1})$.

One can readily prove that for any $t \geq 1$, it holds that

$$x_t | x_0 \sim N(\sqrt{\overline{\alpha}_t} x_0, (1-\overline{\alpha}_t) I),$$

where $\overline{\alpha}_t := \prod_{i=1}^{t} (1-\beta_i)$.

Note:-

Since $\beta_t \in (0,1)$ for all t, we have that $\overline{\alpha}_t \to 0$ as t increases. Hence, if we consider T large enough, we have that the probability distribution of $x_t | x_0$ is close to a normal distribution $N(0, I)$, for any $x_0 \in \mathcal{X}$.

Backward diffusion: The backward process is modelled as a normal distribution with parametrized mean and variance:

$$p_\theta(x_{t-1} | x_t) = N(\mu_\theta(x_t), \Sigma_\theta(x_t, t)). \tag{13.1}$$

Here, $\mu_\theta(\cdot)$ and $\Sigma_\theta(\cdot)$ are parametrized functions

$$\mu_\theta : \mathcal{X} \to \mathcal{X} \quad \text{and} \quad \Sigma_\theta : \mathcal{X} \times \mathbb{N} \to \mathbb{R}^{d \times d},$$

where d is the dimension of $\mathcal{X}$.

Training: In order to train the model using the dataset $\{x^{(i)}\}_{i=1}^n$, we need to define a functional which will then be minimized by means of a gradient method. Let us denote by $p_\theta(x_0)$ the probability distribution associated to the stochastic process with transition probabilities (13.1) and with $x_T \sim \pi = N(0, I)$. During the training, we minimize the cross entropy $\mathbb{E}_{x_0 \sim q}(-\log p_\theta(x_0))$, which is equivalent to minimizing the KL divergence between q and p_θ. The cross entropy can be upper estimated by using the so-called upper confidence bound for $p_\theta(x_0)$:

$$\begin{aligned}
\mathbb{E}_{x_0 \sim q}(-\log p_\theta(x_0)) &\leq \mathbb{E}_{x_0, x_1, \dots, x_T \sim q_t} \left(-\log \left(\frac{p_\theta(x_0, \dots, x_T)}{\mathbb{P}(x_1, \dots, x_T | x_0)} \right) \right) \\
&= \mathbb{E}_{x_0, x_1, \dots, x_T \sim q_t} \left(-\log(\pi(x_T)) - \sum_{t=1}^{T} \log \frac{p_\theta(x_{t-1} | x_t)}{q_t(x_t | x_{t-1})} \right) \\
&= L(\theta).
\end{aligned}$$

Here, the expectations are taken with respect to the forward diffusion process q_t with $x_0 \sim q$.

When minimizing the loss functional $L(\theta)$, we can use a Monte Carlo approximation of the expectation. Since we know the distribution of x_t conditioned to x_0, we can sample x_t for different t's by sampling x_0 uniformly from the dataset $\{x^{(i)}\}_{i=1}^n$, and then sampling x_t from $N(\sqrt{\overline{\alpha}_t}x_0, (1-\overline{\alpha}_t)I)$.

13.3 Score-based diffusion models

Here we briefly described the construction proposed in [30, 29].

Forward diffusion: The forward diffusion model is given by the following stochastic differential equation (SDE)

$$dx = f(x,t)dt + g(t)dw(t), \tag{13.2}$$

where $f(\cdot,t) : \mathbb{R}^d \to \mathbb{R}^d$ is a vector field for every $t > 0$, known as the drift coefficient, $g(t) \in \mathbb{R}$ is known as the diffusion coefficient, and $w(t)$ is standard Brownian motion in $\mathbb{R}^d$.

Note:-

Let us denote by $q_t(x)$ the probability density function associated to the random variable $x(t)$, solution to the SDE (13.2) with initial condition $x(0) \sim q$. This function is the solution to the following Fokker-Planck equation

$$\begin{cases} \dfrac{d}{dt}q_t(x) - \mathrm{div}_x(q_t(x)f(x,t)) - g(t)^2\Delta_x q_t(x) = 0 \quad (t,x) \in (0,\infty)\times\mathbb{R}^d \\ q_0(x) = q(x). \end{cases} \tag{13.3}$$

The functions f and g have to chosen in such a way that the unique stationary state associated to this evolution PDE is the known probability density $\pi(\cdot)$. In this way, one can ensure that $q_t(\cdot) \to \pi(\cdot)$ as $t \to \infty$.

Example. One possible choice for the choice of the diffusion process is the so-called Langevin dynamics, in which

$$dx = -\frac{\beta}{2}\nabla_x V(x)dt + \sqrt{\beta}dw,$$

where $\beta > 0$ is the diffusion coefficient and $V : \mathbb{R}^d \to \mathbb{R}^d$ is a potential, e.g. $V(x) = \dfrac{\|x\|^2}{2}$.

Backward diffusion: Next we state and important result by Anderson in 1982, see [2], which characterises the backward diffusion process in terms of the density function $q_t(\cdot)$ associated to the forward diffusion.

Theorem 13.1. For any $T > 0$, the diffusion process (13.2) with $x(0) \sim q$, has an

associated reverse-time diffusion process given by

$$\begin{cases} dx = (f(x,t) - g(t)^2 \nabla_x \log q_t(x))dt + g(t)d\overline{w}(t) & t \in (0,T) \\ x(T) \sim q_T(x), \end{cases} \tag{13.4}$$

where $q_T(\cdot)$ is the density function associated to the forward diffusion process at time T, and $\overline{w}(t)$ is a backward-in-time Brownian motion.

The probability density function associated to the random variable $x(0)$ given by the above reverse diffusion process is $q(\cdot)$.

In view of the above Theorem, we can recover an approximation of the target distribution $q(\cdot)$ by approximating the backward diffusion process (13.4). Since we already know the functions f and g, that were chosen during the construction of the forward diffusion, we only need to approximate $q_T(\cdot)$ and $\nabla_x \log q_t(\cdot)$.

1. The probability density $q_T(\cdot)$ can be approximated by the equilibrium density $\pi(\cdot)$ by selecting $T > 0$ large enough.
2. The function $\nabla_x \log q_t(\cdot)$ is known as the **score function**, and will be approximated by means of a Neural Network, that we denote by

$$s_\theta(x,t) \approx \nabla_x \log q_t(\cdot).$$

Training: In order to train our model, we minimize the **score-matching functional**, given by

$$J_{SM}(\theta) := \frac{1}{2}\int_0^T \mathbb{E}_{q_t(x)}\left[\lambda(t)\|\nabla_x \log q_t(x) - s_\theta(x,t)\|_2^2\right] dt, \tag{13.5}$$

where $\lambda : [0,T] \to (0,\infty)$ is a positive weighting function, that will be determined in the sequel.

In practice, we don't have access to the actual score function $\nabla_x \log q_t(x)$, which makes it impossible to even evaluate our functional. Therefore, instead of minimizing $J_{SM}(\theta)$, we minimize the following functional, known as the denoising score matching functional, which is given by

$$J_{DSM}(\theta) := \frac{1}{2}\int_0^T \mathbb{E}_{x_0 \sim q, x \sim q_t(x|x_0)}\left[\lambda(t)\|\nabla_x \log q_t(x|x_0) - s_\theta(x,t)\|_2^2\right] dt,$$

where $q_t(x|x_0)$ represents the probability density function associated to the forward diffusion process (13.2) with $x(0) = x_0$. This functional is equal to J_{SM} up to a constant. Note that this expectation is accessible to us since, on one hand, we can sample x_0 from $q(\cdot)$ by using the dataset $\{x^{(i)}\}_{i=1}^n$, and on the other hand, the probability density $q_t(x|x_0)$ can be analytically computed for suitable choices of $f(x,t)$. Note that $q_t(x|x_0)$ is the solution to the Fokker-Planck equation (13.3) with initial condition $\delta_{x_0}(\cdot)$, the Dirac distribution centred at x_0. For instance, if $f(x,t)$ is linear with respect to x, it is known that $q_t(x|x_0)$ is a Gaussian.

13.4 Two probabilistic models

Once the score function is trained, we obtain the probabilistic model given by the aforementioned backward diffusion process. This is a generative model in which the random variable, denoted by $\widehat{X}_\theta^{SDE}$ is obtained as

$$\widehat{X}_\theta^{SDE} = x_\theta(0),$$

where $x_\theta(0)$ is obtained through the stochastic process

$$\begin{cases} dx_\theta = (f(x_\theta, t) - g(t)^2 s_\theta(x_\theta, t))dt + g(t)d\overline{w} \quad t \in [0, T] \\ x_\theta(T) \sim \pi. \end{cases}$$

On the other hand, one can also consider the model associated to the probability density flow associated to the above stochastic process, see [29]. In this case, the samples are obtained through a backward deterministic ODE, with random final condition from the distribution π. We denote it by

$$\tilde{X}_\theta^{ODE} = \tilde{x}_\theta(0),$$

where $\tilde{x}_\theta(0)$ is given by

$$\begin{cases} d\tilde{x}_\theta = (f(\tilde{x}_\theta, t) - \frac{1}{2} g(t)^2 s_\theta(\tilde{x}_\theta, t))dt \quad t \in [0, T] \\ \tilde{x}_\theta(T) \sim \pi. \end{cases}$$

This is an example of a normalizing flow. If the approximation of the score function $s_\theta(x, t)$ would actually be a score function, then both models would be equivalent, however, since $s_\theta(x, t)$ might not be a valid score function, then both models are different, in the sense that the probability distribution of $\widehat{X}_\theta^{SDE}$ and $\tilde{X}_\theta^{ODE}$ are not the same.

13.5 Likelihood estimates for score-based diffusion models

The following result is proved in [29]. It proves that, under a suitable choice of the weighting parameter $\lambda(t)$, minimizing the score-matching functional is an approximation for the maximum likelihood estimator for θ.

Theorem 13.2. Let $\lambda(t) = g(t)^2$ in (13.5), and denote by $p_\theta^{SDE}(\cdot)$ the probability density function for the probabilistic model $\widehat{X}_\theta^{SDE}$. Then

$$D_{KL}(q \| p_\theta^{SDE}) \leq J_{SM}(\theta) + D_{KL}(q_T \| \pi).$$

We see in the above Theorem that KL divergence between the target probability density q and the probability density of the probabilistic model p_θ^θ is upper bounded by the score-matching functional and the KL divergence between q_T and π. The latter term can be ensured to be small by taking T sufficiently big, since the probability density $q_T(\cdot)$ converges to $\pi(\cdot)$ as T goes to infinity. Then, by minimizing the score-matching functional, one can control the KL divergence between p_θ^{SDE} and the target q.

Note:-

As a corollary of the above result we have

$$-\mathbb{E}_{q(x)}(\log p_\theta^{SDE}(x)) \leq J_{SM}(\theta) + C_T,$$

for some C_T which decreases with T. Some remarks are in order:

- The minimizer of $J_{SM}(\theta)$ provides, approximately, a maximum likelihood estimator for the parameter in the generative model $\widehat{X}_\theta^{SDE}$.

- Since $s_\theta(x,t)$ might not be a valid score function for a stochastic process, the probabilistic models $\widehat{X}_\theta^{SDE}$ and $\tilde{X}_\theta^{ODE}$ are not necessarily equal.
- Minimizing the score matching functional with $\lambda(t) = g(t)^2$ is guaranteed to improve the performance of the model $\widehat{X}_\theta^{SDE}$ but not necessarily the performance of the model $\tilde{X}_\theta^{ODE}$.

Mar 6 2024 Wed (12:00-13:00)

Lecture 14: Generative models II: Variational Autoencoders

In this lecture we give a description of Variational autoencoders (VAEs), following the work [20]. VAEs are a special case of probabilistic generative models, which are characterized by the structure in two parts: an encoder, which maps the data space $\mathcal{X}$ to a latent space $\mathcal{Z}$, and the decoder, which maps the latent space $\mathcal{Z}$ to the data space $\mathcal{X}$. The latent space $\mathcal{Z}$ is typically chosen to be a simple low-dimensional space. Heuristically, the encoder is supposed to extract the relevant information from the data and encode it in the latent space, in such a way that the no meaningful information is lost, so that the decoder is able to recover the data from the latent space representation. Let us describe VAEs in a more rigorous way.

14.1 Problem formulation

Let $\mathcal{X}$ be a measurable space and let $q(\cdot)$ be a probability density over $\mathcal{X}$. We are given a dataset $\{x^{(i)}\}_{i=1}^n$ which is assumed to be an i.i.d. sample from q. The **goal** is to construct a probabilistic model of the two following elements:

1. $p_\theta(z)$ is a probability density on $\mathcal{Z}$.
2. $p_\theta(x|z)$ is a conditional probability density on $\mathcal{X}$ for all $z \in \mathcal{Z}$.

These are typically simple probability distributions which are parametrized by the parameter θ.

Example. A simple case is to choose the latent space $\mathcal{Z} = \mathbb{R}^{d^*}$, with d^* relatively small. As the prior distribution, we can use a normal distribution, i.e. $p_\theta = N(0, I)$. We see that the prior don't need to be parametrized. As the conditional distribution, a typical choice is to consider a normal distribution, in which the mean is given through a neural network $E_\theta(z)$, i.e., $p_\theta(\cdot|z) = N(E_\theta(z), \Sigma)$. The neural network $E_\theta(\cdot) : \mathcal{Z} \to \mathcal{X}$ is typically called the decoder.

We can compute the marginal distribution in $\mathcal{X}$ associated to the above probabilistic model, which is given by

$$p_\theta(x) = \int_{\mathcal{Z}} p_\theta(x|z)p_\theta(z)dz.$$

The goal is to choose θ such that $p_\theta(x) \approx q(x)$ for all $x \in \mathcal{X}$.

We are also interested in computing the posterior distribution associated to our probabilistic model p_θ, which is given by

$$p_\theta(z|x) = \frac{p_\theta(x|z)p_\theta(z)}{p_\theta(x)}.$$

The posterior gives a probability distribution in the latent space for any given data point. The posterior is used to train the model, but can also be used to do inference or to solve inverse problems. However, the posterior involves the quantity $p_\theta(x)$, which involves an integral in the latent space. Since we can't afford to approximate this

integral everytime we need to compute a posterior (specially during the training), we approximate the posterior by means of a neural network, that we denote by $q_\phi(z|x)$.

Definition 14.24

Let $\mathcal{X}$ and $\mathcal{Z}$ be two measurable spaces, representing the data space and the latent space respectively. A Variational Autoencoder if composed of the three following elements:

1. A **prior distribution** $p_\theta(z)$ over $\mathcal{Z}$.
2. A **probabilistic decoder** $p_\theta(x|z)$, which defines a probability distribution over $\mathcal{X}$ for every $z \in \mathcal{Z}$.
3. A **probabilistic encoder** $q_\phi(z|x)$, which defines a probability distribution over $\mathcal{Z}$ for any $x \in \mathcal{X}$. This is sometimes also known as the **recognition model**.

14.2 Training

Here we describe the training procedure. As in most of Deep Learning tasks, it consists in minimizing (or maximizing) a suitable loss functional. In this case, the training consists in maximizing the log-likelihood of the given dataset $\{x^{(i)}\}_{i=1}^n \in \mathcal{X}^n$. This can be written as

$$\max_\theta \log(p_\theta(x^{(1)}, x^{(n)})) = \sum_{i=1}^n p_\theta(x^{(i)}).$$

As we mentioned before, computing the marginal probability density $p_\theta(x)$ is intractable in practice. Instead, we will maximize the so-called Evidence Lower Bound (ELBO).

Exercise 14.1. Using the identity $p_\theta(x) = \frac{p_\theta(x,z)}{p_\theta(z|x)}$, prove

$$\log(p_\theta(x)) = \mathbb{E}_{z\sim q_\phi(\cdot|x)}\left(\log\left(\frac{p_\theta(x,z)}{q_\phi(z|x)}\right)\right) + D_{KL}\left(q_\phi(\cdot|x)\|p_\theta(\cdot|x)\right).$$

Since the KL divergence is always non-negative, we have the following lower bound for the log-likelihood:

$$\log(p_\theta(x)) \geq \mathcal{L}(x,\theta,\phi) := \mathbb{E}_{z\sim q_\phi(\cdot|x)}\left(\log\left(\frac{p_\theta(x,z)}{q_\theta(z|x)}\right)\right).$$

Next, using the identity $p_\theta(x,z) = p_\theta(x|z)p_\theta(z)$, we can re-write $\mathcal{L}(x,\theta,\phi)$ as

$$\mathcal{L}(x,\theta,\phi) = \mathbb{E}_{z\sim q_\phi(\cdot|x)}\left(\log p_\theta(x|z)\right) - D_{KL}\left(q_\phi(\cdot|x)\|p_\theta(\cdot)\right).$$

Note:-

In the above expression, the first term can be seen as a data fidelity term, which maximizes the log-likelihood of the data. The second term in turn enhances the posterior $q_\phi(\cdot|x)$ to be close tho the prior distribution $p_\theta(\cdot)$ for all $x \in \mathcal{X}$, and can be seen as a regularization term.

If one aims to use a gradient based method to maximize the function $\mathcal{L}(x,\theta,\phi)$ with respect to the parameters θ and ϕ, one has to take into account that the expectation

is taken w.r.t. the distribution $q_\phi(\cdot|x)$, which depends on the parameters. This is very inconvenient when computing the gradient with respect to ϕ. Indeed, one can use a Monte Carlo approximation of the expectation, however, after every gradient iteration, the probability distribution $q_\phi(\cdot|z)$ is different since we have updated ϕ. This means that we need to resample from the updated distribution after every iteration. This makes the minimization method computationally very expensive. Next we describe a common solution to overcome this issue.

14.3 Reparametrization

Let $g_\phi : \Omega \times \mathcal{X} \to \mathcal{Z}$ be a parametrized deterministic function, where Ω is an arbitrary measurable space. Consider also a probability distribution $p(\cdot)$ over Ω. For any $x \in \mathcal{X}$, we can construct the posterior distribution by puching forward the distribution p through the function $g_\phi(\cdot, x) : \Omega \to \mathcal{Z}$, i.e. for any measurable set $Z \subset \mathcal{Z}$, we have

$$q_\phi(Z|x) = p(g_\phi^{-1}(Z, x)),$$

where $g_\phi^{-1}(Z, x)$ is the pre-image of Z by the function $g_\phi(\cdot, x) : \Omega \to \mathcal{Z}$.

With this parametrization of the posterior $q_\phi(z|x)$, we can re-write $\mathcal{L}(x, \theta, \psi)$ as

$$\mathcal{L}(x, \theta, \phi) = \mathbb{E}_{\varepsilon \sim p(\cdot)} \left(\log p_\theta(x|g_\phi(\varepsilon, x)) \right) - D_{KL} \left(q_\phi(\cdot|x) \| p_\theta(\cdot) \right).$$

In this way, the expectation is taken with respect to a probability distribution which does not depend on the parameters, making the approximation of the gradient much computationally cheaper. Note that we don't need to re-sample ε after every gradient iteration.

Mar 11 2024 Mon (12:00-13:00)

Lecture 15: Deep Learning meets inverse problems I

One of the most remarkable and successful empirical applications of Deep Learning DL techniques in the recent years has been in the field of image reconstruction, in which an image of interest has to be recovered from its incomplete and noisy observation. The data generation process is typically governed by an underlying physical process, that can be modelled by an operator (often referred to as the forward operator), plus the addition of noise, which accounts for error in the measurements or other physical effects which are neglected by the physical model.

Mathematically, image reconstruction can be formulated as an inverse problem consisting on recovering $x^* \in \mathcal{X}$ from a given observation $y \in \mathcal{Y}$ which is assumed to be related to x^* through the equation

$$y = \mathcal{A}x^* + e,$$

where $\mathcal{A} : \mathcal{X} \to \mathcal{Y}$ is the so-called forward operator, which is typically known, and $e \in \mathcal{Y}$ represents the observation error and is modelled as a random variable.

The inverse problem consists in finding a reconstruction method

$$\mathcal{R} : \mathcal{Y} \to \mathcal{X},$$

which given an observation $y \in \mathcal{Y}$, returns the state x^* that gave rise to the measurement y.

However, in most real applications, inverse problems happen to be very challenging due to the fact that they are typically **ill-posed** in one or both of the following senses:

- On one hand, the problem might be **under-determined**, meaning that two different reconstructions x_1 and x_2 in $\mathcal{X}$ may be compatible with the observation y.
- On the other hand, the problem might also be **ill-conditioned**, leading to an unstable reconstruction method. This happens when the level of noise in the measurement is big as compared to the variance of the clean signal $\mathcal{A}x^*$. In this case, the noise can dramatically affect the reconstruction, which is rather undesirable.

There are two main approaches to deal with the issue of ill-posedness, which in some cases are equivalent. See [26] for a survey about Deep Learning techniques applied to inverse problems, especially in image reconstruction.

In the **functional analytical setting**, the reconstruction x^* is typically obtained through a minimization problem of the form

$$\underset{x \in \mathcal{X}}{\text{minimize}} \ \mathcal{L}(x, y) + \mathcal{S}_\theta(x), \tag{15.1}$$

where $\mathcal{L}(x, y)$ is the data fidelity term, and $\mathcal{S}_\theta(\cdot)$ is an appropriately chosen regularizer, which stabilizes the reconstruction method by enforcing certain regularity on the reconstruction. Without this term, the reconstruction would be based solely on data fidelity, which may lead to overfitting. The choice of the regularization term $\mathcal{S}_\theta(\cdot)$

with possibly hyperparameters θ allows to encode prior knowledge about desirable reconstructions.

In the **Bayesian setting** the ill-posedness of the problem is circumvented by assuming a prior distribution on $x \in \mathcal{X}$, which determines how the reconstruction should look like at a population level. In this case the reconstruction is obtained by maximizing the posterior distribution, i.e. $\mathbb{P}(x|Y = y)$, which by virtue of Bayes theorem is equivalent to

$$\underset{x\in\mathcal{X}}{\text{maximize}}\ \mathbb{P}(y|X = x)\mathbb{P}_X(x),$$

where $\mathbb{P}_X(\cdot)$ is the prior distribution on $\mathcal{X}$. The idea is to prevent the reconstruction methods from producing undesirable reconstructions.

Solutions to (15.1) can be interpreted as maximum-a-posteriori estimators if the data consistency term is $\mathcal{L}(x, y)$ is proportional to the negative log-likelihood of the conditional probability, i.e., $-\log(\mathbb{P}(y|X = x))$, and the regularizer is proportional to the negative log-likelihood of the prior distribution, i.e. $\log(\mathbb{P}_X(x))$.

15.1 Model-based vs data-driven reconstruction

Model-based reconstructions rely on the availability of the forward operator $\mathcal{A}$, or an approximation thereof. The goal is to construct a pseudo-inverse of $\mathcal{A}$ that maps every measurement $y \in \mathcal{Y}$ to an appropriate reconstruction $x \in \mathcal{X}$. In a functional analytical setting, a typical approach is to obtain the reconstruction through a variational problem of the form

$$\underset{x\in\mathcal{X}}{\text{minimize}} \|\mathcal{A}x - y\|_2^2 + \mathcal{S}_\theta(x), \tag{15.2}$$

where the first terms quantifies consistency in the space of data $\mathcal{Y}$ and the second term penalizes undesirable solutions.

Note:-

One of the advantages of this approach is that it can be generalized to different forward operators $\mathcal{A}$ in a **plug-and-play** manner. The key element in this approach is to find an appropriate regularizer $\mathcal{S}_\theta(\cdot)$ which encodes the how the reconstructions should look like (prior knowledge). Then, the same regularizer can be used to address different tasks in the same dataset by only changing the forward operator $\mathcal{A}$. The regularizer can be for instance a trained NN.

Most of the physical phenomena in practice are of non-linear nature, which leads to problems when solving (15.2) due to the lack of convexity. In practice, linear approximations of $\mathcal{A}$ can be used instead, or alternatively, the full non-linear model, at expenses that convergence guarantees only hold locally. Moreover, the dimension of x can be very large, leading to a high-dimensional optimization problem, which can be computationally expensive or unfeasible.

It's worth noting that there are several issues to take into account when using this approach:

1. What happens if the forward operator is inaccurate or unknown?
2. How should one hand-craft the regularizer $\mathcal{S}_\theta(\cdot)$ and choice the parameters?
3. The computational time to solve the variational problem (15.2) makes the reconstruction method unsuitable for certain application where one needs immediate response.

Data-driven approaches offer a new and promising avenue to address the above challenges. Instead of handcrafting a reconstruction method, data-driven methods use training data to learn an optimal reconstruction method based on statistical learning.

Learned reconstructions

$$\mathcal{R}_\theta : \mathcal{Y} \longrightarrow \mathcal{X}$$

are typically parameterized by some suitably chosen deep neural network (DNN) and thus learning refers to selecting optimal parameters θ based on the training data. Depending on the type of training data available, we will use a different approach during training.

15.2 Supervised learning

In this case one has access to ground truth and measurements $\{(x^{(i)}, y^{(i)})\}_{i=1}^n$, and the reconstruction method $\mathcal{R}_\theta$ is obtained by empirical risk minimization (ERM), i.e.

$$\widehat{\theta} \in \arg\min_{\theta \in \mathcal{P}} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_{\mathcal{X}}(\mathcal{R}_\theta(y^{(i)}), x^{(i)}).$$

Note that the (ERM) problem does not directly use the forward operator $\mathcal{A}$, although it is implicitly incorporated in the training data. However, one can select a parametrized reconstruction method of the form

$$\mathcal{R}_\theta = \Phi_\theta \circ \mathcal{A}^\dagger$$

that combines a pseudo-inverse $\mathcal{A}^\dagger : \mathcal{Y} \to \mathcal{X}$ with a NN $\Phi_\theta : \mathcal{X} \to \mathcal{X}$, which is used as a learned post-processing operator.

15.3 Unsupervised learning

Here we find two different scenarios depending whether the data contains ground truths $x^{(i)}$ or measurements $y^{(i)}$:

1. When the training data consists of ground truths $x^{(i)}$, one can use it to learn a regularizer $\mathcal{S}_\theta$ and then consider model based reconstructions as in (15.2). The regularizer

   $$\mathcal{S}_\theta : \mathcal{X} \longrightarrow \mathbb{R}$$

   is typically learned from the data by generative modelling.

   Generative models consist training an NN that transforms a known (simple) distribution into an unknown one. We consider a low-dimensional space $\mathcal{Z}$, sometimes referred to as latent space, and a simple probability distribution $\mu_{\mathcal{Z}}$ on $\mathcal{Z}$. Typically a Gaussian or a uniform distribution. Then, a neural network

   $$\Phi_\theta : \mathcal{Z} \longrightarrow \mathcal{X}$$

   is trained so that the pushforward measure $\Phi_{\theta *}(\mu_{\mathcal{Z}})$ approximates the measure $\mu_{\mathcal{X}}$, i.e.

   $$\mu_{\mathcal{Z}}(\Phi_\theta^{-1}(A)) \approx \mu_{\mathcal{X}}(A) \qquad \forall A \subset \mathcal{X}.$$

   Note that here $\Phi_\theta^{-1}(A)$ denotes the pre-image of the set A by the NN, not the inverse. Examples of generative models are variational auto-encoders (VAEs) or generative adversarial networks (GANs).

If we consider $\mu_{\mathcal{Z}}$ for instance to be Gaussian, then once the generator $\Phi_\theta(\cdot)$ is trained, the reconstruction method is formulated as a variational problem of the form

$$\underset{z\in\mathcal{Z}}{\text{minimize}} \ \|\mathcal{A}\Phi_\theta(z) - y\|_2^2 + \|z\|^2.$$

Note that in this case, the reconstruction problem is reduced to a low dimensional minimization problem.

2. In the other case, when the training data consists of measurements $\{y^{(i)}\}_{i=1}^n$, one option is to learn a data-driven solver for optimization problems of the type (15.2). A natural choice is to construct $\mathcal{R}_\theta$ by *unrolling* an optimization solver for (15.2). The idea is to start with some iterative scheme to solve (15.2) with the different $y^{(i)}$'s in the training data. Then, the iterative scheme is truncated after a fixed number of iterations and unrolled by replacing the updates with a NN.

References

[1] L. Ambrosio, N. Fusco, and D. Pallara. *Functions of bounded variation and free discontinuity problems*. Oxford Mathematical Monographs, 2000.

[2] B. D. Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982.

[3] F. Bach. Duality between subgradient and conditional gradient methods. *SIAM Journal on Optimization*, 25(1):115–129, 2015.

[4] F. Bach. Breaking the curse of dimensionality with convex neural networks. *The Journal of Machine Learning Research*, 18(1):629–681, 2017.

[5] F. Bach. Learning theory from first principles. *Online version*, 2021.

[6] J. Berner, P. Grohs, G. Kutyniok, and P. Petersen. The modern mathematics of deep learning. *arXiv preprint arXiv:2105.04026*, 2021.

[7] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

[8] J. Bruna and S. Mallat. Invariant scattering convolution networks. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1872–1886, 2013.

[9] B. Chang, L. Meng, E. Haber, L. Ruthotto, D. Begert, and E. Holtham. Reversible architectures for arbitrarily deep residual neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[10] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[11] A. G. de G. Matthews, J. Hron, M. Rowland, R. E. Turner, and Z. Ghahramani. Gaussian process behaviour in wide deep neural networks. In *International Conference on Learning Representations*, 2018.

[12] C. D. Freeman and J. Bruna. Topology and geometry of half-rectified network optimization. *arXiv preprint arXiv:1611.01540*, 2016.

[13] J. E. Gerken, J. Aronsson, O. Carlsson, H. Linander, F. Ohlsson, C. Petersson, and D. Persson. Geometric deep learning and equivariant neural networks. *Artificial Intelligence Review*, pages 1–58, 2023.

[14] B. Geshkovski and E. Zuazua. Turnpike in optimal control of pdes, resnets, and beyond. *Acta Numerica*, 31:135–263, 2022.

[15] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[16] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[17] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[18] A. Jacot, F. Gabriel, and C. Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

[19] M. Jaggi. Revisiting Frank-Wolfe: Projection-free sparse convex optimization. In *International conference on machine learning*, pages 427–435. PMLR, 2013.

[20] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[21] J. M. Klusowski and A. R. Barron. Approximation by combinations of relu and squared relu ridge functions with ℓ^1 and ℓ^0 controls. *IEEE Transactions on Information Theory*, 64(12):7649–7656, 2018.

[22] J. Lee, Y. Bahri, R. Novak, S. S. Schoenholz, J. Pennington, and J. Sohl-Dickstein. Deep neural networks as gaussian processes. *arXiv preprint arXiv:1711.00165*, 2017.

[23] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural networks*, 6(6):861–867, 1993.

[24] S. Mallat. Group invariant scattering. *Communications on Pure and Applied Mathematics*, 65(10):1331–1398, 2012.

[25] H. N. Mhaskar. Neural networks for optimal approximation of smooth and analytic functions. *Neural computation*, 8(1):164–177, 1996.

[26] S. Mukherjee, A. Hauptmann, O. Öktem, M. Pereyra, and C.-B. Schönlieb. Learned reconstruction methods with convergence guarantees: A survey of concepts and applications. *IEEE Signal Processing Magazine*, 40(1):164–182, 2023.

[27] D. Rezende and S. Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pages 1530–1538. PMLR, 2015.

[28] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. PMLR, 2015.

[29] Y. Song, C. Durkan, I. Murray, and S. Ermon. Maximum likelihood training of score-based diffusion models. *Advances in neural information processing systems*, 34:1415–1428, 2021.

[30] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

[31] L. Zhang and H. Schaeffer. Forward stability of resnet and its variants. *Journal of Mathematical Imaging and Vision*, 62:328–351, 2020.