Topics in the Foundations of Artificial Intelligence

A Reader

Draft from July 10, 2024.
Please don't cite or distribute without permission.
Comments welcome!

Levin Hornischer
Levin.Hornischer@lmu.de
https://levinhornischer.github.io/FoundAI/

Contents

Preface

This is the reader for the course *Advanced Topics in the Foundations of AI* given during the summer semester 2024 at *LMU Munich* as part of the *Master in Logic and Philosophy of Science*. The reader is written as the course progresses. A website for the course is found at

`https://levinhornischer.github.io/FoundAI/`.

Comments I'm happy about any comments: spotting typos, finding mistakes, pointing out confusing parts, or simply questions triggered by the material. Just send an informal email to `Levin.Hornischer@lmu.de`.

Content In recent years, artificial intelligence and, in particular, machine learning made great—but also disconcerting—progress. However, their foundations are, unlike other areas of computer science, less well understood. This situation is sometimes compared to being able to build steam engines without having a theory of thermodynamics.

This seminar is about the mathematical foundation of AI. After a review of the classical theory (Computability Theory, No-Free-Lunch Theorem, Universal Approximation Theorem, etc.), we read some recent research papers to get an overview of some current approaches to the foundations of AI.

Objectives In terms of content, the course aims to convey and overview of the foundations of AI—including both classic material and cutting-edge research. In terms of skills, the course aims to teach the ability to both mathematically and philosophically assess the different approaches to the foundations of AI.

Prerequisites In order to appreciate the literature, the course requires basic familiarity with mathematics (calculus, linear algebra, probability theory), logic (including, ideally, computability theory), and AI (neural networks). Some papers also use more advanced concepts from topology, probability theory, or category theory, so you should also be prepared to read up on those. But they are not assumed: the seminar sessions are,

among others, meant to get clearer on these concepts. Programming skills will of course be useful, but will not be assumed.

Schedule and organization The course is organized as a seminar. Hence, for each session, we have assigned readings, which we then discuss during the session. The reading for each week is announced in the schedule on the course's website. The readings are roughly organized by topic, forming the chapters of this reader.

Background material Some helpful short explainer videos on AI are found here. A excellent mini series on (the mathematics of) neural networks is found here.

Moreover, the material of my companion course on the Philosophy and Theory of AI might also be helpful. You can take the present course independently of that companion course and vice versa, but they do complement each other. The companion course is more introductory and looks at a broader range of philosophical issues connected to AI and how to theorize about them, while the present course focuses specifically on the more mathematical foundations of neural networks.

A recent edited collection on the mathematical foundations of AI is Grohs and Kutyniok (2022).

Layout These notes are informal and partially still under construction. For example, there are margin notes to convey more casual comments that you'd rather find in a lecture but usually not in a book. Todo notes indicate, well, that something needs to be done. References are found at the end.

This is a margin note.

This is a todo note

Notation Throughout, 'iff' abbreviates 'if and only if'.

1 Introduction

Summary

We give an outlook of the course and of this document.

The field of AI is typically characterized along the lines of aiming to build "machines that can compute how to act effectively and safely in a wide variety of novel situations" (Russell and Norvig 2021, p. 19). In chapter 2, we briefly collect basic terminology and concepts in AI, to make sure we're all on the same page.

Regardless of the definition, it is helpful to distinguish two main traditions in AI. They go by varying names, with different connotations depending on the community that uses them, including the following.

1. *Symbolic AI*: classicist, logic-based, Good Old-Fashioned AI (GOFAI), etc.

   Example: An algorithm or computer program that, given as input a position in a game of chess, outputs the next best move. This algorithm was written by a programmer.

2. *Subsymbolic AI*: connectionist, non-logicist, machine learning, deep learning, etc.

   Example: A neural network that, given as input a pixel image of a handwritten digit, outputs the digit depicted on the image. The neural network was trained to become better at this mapping using thousands of data points, i.e., input images labeled with the digit depicted on them.

(Some might distinguish a third tradition—*statistical AI*—which, in a sense, sits between the two preceding traditions: Like symbolic AI it typically has 'interpretable' variables, but they are now continuous random variables, and like subsymbolic AI it typically processes the information in a continuous way.)

For symbolic AI, we have a pretty good theory due to mathematical/philosophical logic and computability theory. We review this in chapter 3. It gives a good idea of what we also would expect of a theory of AI.

For subsymbolic AI—which is most of modern AI—we, however, lack a good theory. So in this course, we mostly focus on approaches to provide such a theory. In chapter 4, we review the main results that the standard theory of machine learning—which is mostly statistical learning theory—can deliver. But we also look at what is still missing for the concrete case of the neural networks that modern AI is built on.

The remaining chapters—as listed in the table of contents—then are about different approaches to fill these gaps in the standard theory of machine learning, or approaches that rethink this theory all together. There is more material than we will be able to cover: especially from the later chapters we will pick the topics based on your interests.

2 Background

Readings

- A textbook introduction to the field of AI: Russell and Norvig (2021, ch. 1).

Key concepts

- History of AI: Ada Lovelace, Alan Turing, McCulloch & Pitts, Logic Theorist, Dartmouth workshop, summers and winters, big data, deep learning revolution.
- Types of AI: symbolic, subsymbolic, statistical
- Definitions of AI: acting humanly (Turing test), thinking humanly (cognitive modeling), thinking rationally (logic, probability), acting rationally (rational agent; perfect vs limited rationality)
- Types of learning tasks: Supervised learning, unsupervised learning, reinforcement learning. Machine learning pipeline (conceptualization, data, model, deployment).
- Key concepts of artificial neural networks: neurons, layers, feedforward/recurrent, weights, activation function, loss function, backpropagation, learning rate, local/global minima (equilibrium), regularization, overfitting/underfitting.

In this session, we discuss the main reading to get an understanding of each of the key concepts—the basic AI terminology—mentioned above. These key concepts are further illuminated in the additional material mentioned below.

Elaborating on the rational agent idea, Russell and Norvig (2021) write:

> In a nutshell, AI has focused on the study and construction of agents that *do the right thing*. What counts as the right thing is defined by the objective that we provide to the agent. This general paradigm is so pervasive that we might call it the *standard*

> *model*. . . . [Though,] the standard model assumes that we will supply a fully specified objective to the machine. . . . The problem of achieving agreement between our true preferences and the objective we put into the machine is called the *value alignment problem* . . . Ultimately, we want agents that are *provably beneficial* to humans (Russell and Norvig 2021, 22–21, emphasis altered).

Further material

- A very accessible overview, written at the beginning of the deep learning revolution: Boden (2016, ch. 1 and 4).
- The Stanford Encyclopedia of Philosophy entry on artificial intelligence: Bringsjord and Govindarajulu (2024).
- Also see the background material mentioned in the preface (explainer videos and companion course).

3 Foundations of symbolic AI

Readings

- A short overview of symbolic AI: Flasiński (2016, ch. 2)
- An overview of computability and complexity theory: Immerman (2021)

Key concepts

- Turing machine
- Church–Turing thesis
- Halting problem
- Entscheidungsproblem
- Tiling problems
- Gödel's incompleteness theorems
- Computational complexity theory: P vs NP

I might add the contents of the slides that I presented in class

4 Standard theory of machine learning

Readings

- On statistical learning theory and the No-Free-Lunch theorem: Shalev-Shwartz and Ben-David (2014), chapters 2–3 (basic set up), chapter 5 (No-Free-Lunch), sections 6.1–4 (fundamental theorem of statistical learning theory).
- On the approximation theorem: One of the classics, Hornik et al. (1989).

Key concepts

- Statistical learning theory
- PAC learnability
- No-Free-Lunch theorem
- Bias-complexity tradeoff
- VC-dimension and the fundamental theorem of PAC learning
- Universal approximation theorem

Statistical learning theory Statistical learning theory was developed as the theory of machine learning. There is some disconnect between theory and practice, which we will discuss in the next chapter, but first we introduce this theory here.
The statistical learning framework:

- A learner gets input x from some domain X (e.g., a particular papaya) and they need to label this input with a label $h(x) = y$ from the label space Y, here the only labels are 0 (e.g., not tasty) and 1 (e.g., tasty). The learner will see finitely many training data $(x_1, y_1), \ldots, (x_m, y_m)$ of input-output pairs, and based on that suggest a general rule $h : X \to Y$. We now formalize this idea.

- The *domain set* X. Typically the elements are vectors, e.g., $x = (0.1, 0.7)$ saying that the object x is described completely by feature 0 (e.g., the papayas color) having value 0.1 and feature 1 (e.g., the papayas softness) having value 0.7.
- The *label set* Y. Typically $Y = \{0, 1\}$.
- The *training data* $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ which is a finite subset of $X \times Y$. These are the examples that the learner has access to (e.g., the m-many papayas they have bought, checked the color and softness, and then tasted them to determine whether they were tasty or not). Elements of S are also called data points or training examples, and S is also called training set.
- The *learner's output*: Based on the training data, the learner has to output a prediction rule, i.e., a rule for how to label points from the domain set. This rule is described as a function $h : X \to Y$, which is also called predictor, hypothesis, or classifier.
- The *learning algorithm* A: Typically, the learner will follow a general learning algorithm that works not just for the specific training set S but also for other ones. So it is a function A that takes as input finite subsets of $X \times Y$ and outputs a predictor $A(S) : X \to Y$. One learning algorithm that we introduce below is empirical risk minimization.
- *Sampling*: We assume there is a probability distribution D on the domain set X which describes how likely it is that we see a particular point x (e.g., how likely it is that the learner gets papaya x when going to the market). One says D is a (probabilistic) data-generation model. Importantly, the learner has *no* access to this probability distribution. Rather D describes how the world actually is.
- *True risk.* The error of a classifier h is the probability that it does not predict the correct label. Formally, we assume there is a true labeling function $f : X \to Y$ (that the learner aims to find) and the error of h is

  $$L_{D,f}(h) := D(\{x \in X : h(x) \neq f(x)\}).$$

  This is also called generalization error, risk, or true error (to distinguish it from the empirical error/risk that we introduce later). (Below, in the NFL Theorem, we generalize the assumption of a true function f and instead work with a probability distribution over $X \times Y$.)

- *Empirical risk*. The true risk is defined with respect to the distribution D and the true labeling function f, both of which the learner has no access to. The learner can only calculate the error of their predictions on the training dataset:

$$L_S(h) := \frac{1}{m}\left|\{i \in \{1,\dots,m\} : h(x_i) \neq y_i\}\right|.$$

- *Empirical Risk Minimization* (ERM) is the learning paradigm of coming up with a predictor h that minimizes the empirical risk $L_S(h)$. This has to be restricted though: If the learner is allowed to pick any predictor h, they can pick the h which, on input x, predicts y if (x, y) is in the training set and 0 otherwise. This minimizes the empirical risk (it is 0), but it generalizes badly to points x outside of the training set (it all assigns them the same label): one says h *overfits* the data. To avoid this, one fixes a set H of allowed predictors (and the just mentioned h wouldn't usually be allowed). This H is called the *hypothesis class*. The learner has to choose this in advance, before seeing the data. This choice of H is the *inductive bias* (or, positively, prior knowledge) of the learner: they are biased to certain predictors before seeing data (or know a priori that they will better fit the data). (This will lead to the bias-complexity trade-off that we'll discuss later.)

PAC learning:

- What would it mean for a learner—with their choice of hypothesis class H—to be 'good', i.e., to produce correct prediction rules? We want that, no matter what the true distribution D and labeling function f are, given a required confidence parameter δ and accuracy parameter ϵ, there is a number of samples $m = m(\delta, \epsilon)$ such that, if we sample m-many examples from D labeled with f, then with confidence $1 - \delta$ the learner knows that they are correct up to ϵ, provided there is a correct hypothesis in the first place. One says: we are *probably approximately correct* (PAC). The formal definition is:

Though one might ask: why quantify over all D and f and not just those that are likely for the task? E.g., for the papayas some distributions and labelings are more likely than others. (Cf. margins theory *discussed by Belkin (2021, sec. 3.3).*

- A hypothesis class H is *PAC-learnable* if there is a function $m_H : (0,1) \times (0,1) \to \mathbb{N}$ and a learning algorithm A such that: For every $\epsilon, \delta \in (0,1)$, for every probability distribution D over X, and for every labeling function $f : X \to \{0,1\}$, if the realizability assumption holds (i.e., there is $h^* \in H$ with $L_{D,f}(h^*) = 0$), then, when running A on $m \geqslant m(\epsilon, \delta)$ i.i.d. (independently and identically distributed)

examples sampled with D and labeled with f, the algorithm return a hypothesis h such that, with probability of at least $1 - \delta$ (over the choice of the examples), $L_{D,f}(h) \leqslant \epsilon$.

- This can be generalized by dropping the realizability assumption, going beyond the binary label case (multiclass and regression, using more general loss functions).

The No-Free-Lunch and the PAC Learning theorems No-Free-Lunch Theorem (NFL Theorem):

- We avoided the overfitting problem of the ERM learning paradigm by making explicit the inductive bias/prior knowledge of the learner in the form of a hypothesis class H. One may ask: is this really necessary, i.e., can there be a learner who is successful without using any task-specific prior knowledge and can thus solve any task? One way to interpret the NFL theorem is that it answers 'no': there cannot be such a *universal learner*.

For a discussion of interpretations of the NFL theorem, see the further reading below (Sterkenburg and Grünwald 2021).

- A bit more precisely, the NFL Theorem takes a learning task to be given by an unknown distribution P over $X \times Y$, and the goal of the learner is to find a predictor $h : X \to Y$ whose risk $L_P(h)$ is small. As mentioned, this generalizes the framework so far: So far we had a distribution D on X and a true label function f. This determines the distribution P on $X \times Y$ according to which the probability of (x, y) is 0 if $y \neq f(x)$ and otherwise the probability of x according to D. Now we allow any distribution P on $X \times Y$, so we don't assume there is a single true label function, but only a conditional probability of how likely a label is given the input. Accordingly, the loss is

$$L_P(h) = P(\{(x, y) \in X \times Y : h(x) \neq y\},$$

  which is also known as *0-1 loss*. The NFL Theorem then says: It is not the case that there is a learning algorithm A and a training set size m such that, for every distribution P over $X \times Y$, if A receives m-many i.i.d. samples from P, there is a high chance it outputs a predictor h that has low risk. In other words, for every learner, there is a task on which it fails, even though another learner succeeds.

- Formally, the NFL theorem is stated as follows. Let A be any learning algorithm for the task of binary classification with respect to the 0-1 loss over a domain X. Let m be any number smaller than $|X|/2$,

representing a training set size. Then, there exists a distribution P over $X \times \{0,1\}$ such that:

1. There exists a function $f : X \to \{0,1\}$ with $L_P(f) = 0$, but
2. With probability of at least $1/7$ over the choice of $S \sim P^m$ we have that $L_P(A(S)) \geqslant 1/8$.

So A fails on this task while the ERM learner with hypothesis class $H = \{f\}$ succeeds.

Bias-complexity tradeoff

- Since we don't have a universal learner—who performs best possible on any task—, we cannot get around analyzing a learner on a specific task. So we want to get some guarantee how bad the true error of the learner is. As we'll describe now, we can split this error up into two parts, but there is a tradeoff in that improving one part tends to worsen the other, and vice versa.

- *Error decomposition*. We can analyze the error $L_P(A(S))$ of our learning algorithm/ERM predictor A on a given dataset S as $\epsilon_{app} + \epsilon_{est}$ with

$$\epsilon_{app} := \min_{h \in H} L_P(h) \qquad \epsilon_{est} := L_P(A(S)) - \epsilon_{app},$$

  where ϵ_{app} is the *approximation error* (the minimal risk achievable by a predictor from the hypothesis class) and ϵ_{est} is the *estimation error* (the difference between the empirical minimal risk $L_P(A(S))$ and the true minimal risk ϵ_{app}).

- *Bias-complexity tradeoff*. To reduce the error $L_P(A(S))$ we hence want to reduce both the approximation error and the estimation error. However, this comes at a tradeoff:

  1. To reduce the approximation error, we want a large, more *complex* hypothesis class; but, as we saw, this may lead to *overfitting* (empirical risk is low, but true risk is high), so the the estimation error can be high.
  2. To reduce the estimation error, we might rather choose a small, more *biased* hypothesis class, but this results in a high approximation error, because the predictors are now *underfitting* the data.

So learning theory studies how to find rich hypothesis classes H for which we still have reasonable estimation errors. We will further discuss this for neural networks in chapter 5 (the universal approximation theorem will show that—as the name suggests—their approximation error is zero).

VC-dimension and the fundamental theorem of PAC learning:

- How can we ensure our learner has a low error? Reducing the approximation error is a matter of picking the right prior knowledge encapsulated as a hypothesis class. For the estimation error, the key is to realize that PAC learnability bounds the estimation error. So we would like to have a guarantee for PAC learnability. This is provided by the so-called VC-dimension, which is defined as follows.

Under the realizability assumption, the approximation error is zero, so the estimation error can be bounded by ϵ with probability $1-\delta$ when sampling a dataset of size $\geqslant m(\delta, \epsilon)$.

- First, some helpful terminology: If H is a hypothesis class and $C \subseteq X$, then H *shatters* C if $\{h \restriction C : h \in H\} = Y^C$, where $h \restriction C$ is the restriction of the function $h : X \to Y$ to the set C, and Y^C is the set of all functions from C to Y.

- The *VC-dimension* of a hypothesis class H is the maximal size of a finite set $C \subseteq X$ that can be shattered by H. If H shatters sets of arbitrarily large size, it has infinite VC-dimension.

The idea is: If H shatters a set C of size $2m$, then we cannot learn H using m examples: all ways of labeling the other m instances in C are possible according to some hypothesis in H, i.e., prior knowledge doesn't exclude any of those.

- The *fundamental theorem*: Let H be a hypothesis class of functions from X to $\{0, 1\}$, and let the loss function be the 0-1 loss. Then the following are equivalent

  1. H is PAC learnable
  2. Any ERM rule is a successful PAC learner for H
  3. H has a finite VC-dimension.

  In fact, if the VC-dimension of H is $d < \infty$, there are constants C_1 and C_2 such that H is PAC learnable with sample complexity

$$C_1 \frac{d + \log(1/\delta}{\epsilon} \leqslant m(\epsilon, \delta) \leqslant C_2 \frac{d \log(1/\epsilon) + \log(1/\delta)}{\epsilon}.$$

The full version of the theorem has some further equivalent conditions.

The Universal Approximation Theorem There are several universal approximation theorems. Here we discuss an early one by Hornik et al. (1989).[1] (For a modern textbook version including a proof sketch, see

[1] Interestingly, Hornik et al. (1989, p. 360) mention Kolmogorov's superposition theorem which has a similar form of the approximation theorem but would require a possibly

Berner et al. (2022, thm. 1.16 on p. 16); and the further reading Kratsios 2021 below.)

It considers the set Σ^r of functions $\mathbb{R}^r \to \mathbb{R}$ that can be realized by a feed-forward neural network with r-many input neurons, one output neuron, and one hidden layer using an activation function $G : \mathbb{R} \to \mathbb{R}$ that is non-decreasing and converges to 0 (resp., 1) as its argument goes to $-\infty$ (resp., $+\infty$).

Intuitively, the universal approximation theorem should say that the functions in Σ can approximate any non-pathological function $f : \mathbb{R}^r \to \mathbb{R}$ arbitrarily well. This is made precise as follows.

The 'non-pathological' functions are the measurable functions. We don't state the precise definition here, since this would take us a bit astray. For us it suffices to know that basically any function that naturally occurs is measurable. So let M^r be the set of all measurable functions $\mathbb{R}^r \to \mathbb{R}$. Any function realized by a neural network is measurable, so $\Sigma^r \subseteq M^r$.

Now how to say that Σ^r can approximate any function in M^r arbitrarily well? For that, we will define a metric ρ on M^r, i.e., a way to measure the distance $d = \rho(f, g) \in \{x \in \mathbb{R} : x \geqslant 0\}$ between two functions $f, g \in M^r$. Then to say that Σ^r can approximate any function in M^r is to say that Σ^r is *dense* in M^r, i.e., for every $f \in M^r$ and for every $\epsilon > 0$, there is $g \in \Sigma^r$ such that $\rho(f, g) < \epsilon$.

So it remains to define the metric ρ on M^r. The idea is that two functions $f, g \in M^r$ are close if "there is only a small probability that they differ significantly" (Hornik et al. 1989, p. 361). This is made precise as by fixing a probability measure μ on $\mathbb{R}^r$ and defining

$$\rho_\mu(f, g) := \inf\Big\{\epsilon > 0 : \mu\big(\{x : |f(x) - g(x)| > \epsilon\}\big) < \epsilon\Big\}.$$

So for a given 'significance level' $\epsilon > 0$, we check on how many inputs $x \in \mathbb{R}^r$ our functions f and g differ by more than ϵ. If the probability of encountering such an x is low, i.e., smaller than ϵ, then our functions pass the test and are at least ϵ-close; otherwise they are more than ϵ far apart. Now we look for the smallest ϵ's for which our functions are at least ϵ-close. The infimum of all of them then is the distance between f and g.

Now the universal approximation theorem says (Hornik et al. 1989, thm. 2.4 on p. 362):

different activation function for every neuron (rather than a single one for all neurons as it is done in feed-forward neural networks). This idea has been taken up very recently (Liu et al. 2024) investigating a neural network architecture with different, learnable activation functions.

- For any probability measure μ on $\mathbb{R}^r$, the set Σ^r of functions realized by a neural network is ρ_μ-dense in M^r.

Further material

- A discussion of the no-free-lunch theorem: Sterkenburg and Grünwald (2021).
- A modern, more general approach to the universal approximation theorem: Kratsios (2021).

5 Refining statistical learning theory

Readings

- Berner et al. (2022), pages 1–31.
- Belkin (2021), sections 1–3.

Key concepts

- Approximation error, generalization error, and optimization error.
- The generalization puzzle/problem
- Interpolation and over-parametrization
- Double descent generalization curve

Berner et al. (2022) describe in section 1.1.2 (i.e., pages 5–23) the insights that statistical learning theory can provide when it is applied to neural networks. On page 13, they analyze the true error of a neural network into three components. (The first two are closely related to the approximation and estimation error that we considered in the general setting; the third is new.)

- The approximation error: how far the best possible prediction rule of the network is away from the truly best possible prediction rule, i.e., the Bayes-optimal function.
- The (uniform) generalization error: among the prediction rules of the network, the maximal difference between the true error of the prediction rule and the empirical error of the prediction rule.
- The optimization error: the difference in empirical error between the prediction rule found by the neural network after training and the prediction rule that minimizes the empirical risk.

The third type of error is added because with neural networks we do *algorithmic* empirical risk minimization (in the words of Belkin (2021, p. 217)):

backpropagation (i.e., the implementation of stochastic gradient descent for neural networks) is an algorithm that aims to find the prediction rule (i.e., the set of weights) that minimizes empirical risk (i.e., the error on the training set). The general theory that we covered previously is *algorithm-independent* in the sense that it only assumes *that* an (ERM-) learner outputs a prediction rule with minimal empirical risk but doesn't say so *how* this rule is found.

On pages 13–23, the then provide theorems to bound these errors.

- The approximation error is bound by the universal approximation theorem, which we already discussed.
- The generalization error is bound—using the VC-dimension—with high probability by

  $$\sqrt{\frac{p \log(p)}{m}},$$

  where p is the number of parameters of the neural network and m is the size of the training set. (But: when there are more parameters than training samples, as is common for neural networks, this bound is vacuous. This is picked up as an open question below.)
- The optimization error is bound by convergence guarantees of stochastic gradient descent. (But: the assumptions of these guarantees are typically some form of convexity that, however, is usually not satisfied for neural networks. Again this is picked up as an open question.)

For such convergence guarantees also see the Polyak–Łojasiewicz condition in Belkin (2021, sec. 4.1).

In section 1.1.3 (i.e., pages 23–31), Berner et al. (2022) overview the questions that are left open by classical statistical learning theory when applied to neural networks. In particular, this includes:

- The *generalization puzzle* (as Berner et al. (2022, p. 25) call it): Neural networks are typically *over-parametrized*, i.e., have more parameters than training samples. This means that they can typically *interpolate* (or overfit) the training data, i.e., fit exactly to the training data (hence have zero empirical risk). Hence classical bounds on the generalization error are vacuous, and indeed the bias-complexity tradeoff of classical statistical learning theory would predict neural networks to perform poorly. Nonetheless, they perform well. Explaining this is the generalization puzzle.

- The *wonders of optimization* (as Belkin (2021, sec. 4) calls it): Stochastic gradient descent, as it is performed by neural networks, typically converges to good local minima, despite typically being non-convex. Hence the classical convergence guarantees from optimization theory cannot be given. Why is it that neural networks still successfully optimize?

Belkin (2021, sec. 3.7) extends the classical picture of the U-shape trade-off between bias (i.e., small hypothesis class) and complexity (i.e., complex/interpolating hypothesis class). Belkin extends this 'classical regime' further to the 'modern regime' with an even more complex hypothesis class that is over-parametrized. The result is the *double decent generalization curve* (for the figure, see the top of page 3 here): Not only can we descent from the interpolation threshold (where we have a hypothesis class that can just overfit/interpolate the training data) into the classical regime by reducing the complexity of the hypothesis class. We can also descent into the modern regime by making the hypothesis class even more complex.

Belkin (2021, sec. 3.6) provides a heuristic for why this works: In this modern, interpolating regime, there are many prediction rules of the learner which interpolate the training data. Among those, the over-parametrized neural network learner tends to pick the *smoothest* one. (A smooth function is one with, intuitively speaking, no sharp edges.) Being smooth is a sense of being simple, so we can understand this inductive bias of the neural network as an instance of Occam's razor: among the explanation that are consistent with the evidence (i.e., the prediction rules with zero empirical risk) pick the simplest one (i.e., the smoothest one).

Further material

- Rest of the papers.

6 Computability theory of machine learning

Readings

- Colbrook et al. (2022)

Key concepts

- Inverse problems
- Smale's 18th problem: the limits of AI
- Stability vs accuracy tradeoff
- Adversarial attacks

With the universal approximation theorems, we know that neural networks can approximate any (measurable) function arbitrarily well. But knowing that a neural network must exist that approximates a given function is one thing. Another thing is to *actually find* this network. The paper by Colbrook et al. (2022) shows that these two things really can come apart: that although there must an approximating network, it is impossible for us to find it.

An analogous difference is described by intuitionistic logic: between truth/existence and provability/construction.

Concretely, they consider neural networks aiming to solve *inverse problem*. Some reality x determines, via a linear map A and some random noise e, the sensor readings $y = Ax + e$. The task is to reconstruct, given the sensor readings y, what the most minimal reality x must have been giving rise to these sensor readings.

Their result (theorem 1 and 2 combined) then states: There are mappings that map training data of an inverse problem to a neural network solving it, but no training algorithm (like, e.g., stochastic gradient descent) can compute such a mapping (producing neural nets with reasonable accuracy).

The broader context is the question: Why do neural networks tend to be unstable (adversarial attacks, hallucinations, etc.), even in situations—like inverse problems—where stable and accurate neural nets exist (since the universal approximation theorem guarantees that some neural network can approximate the stable solution)?

This suggest developing for neural networks an analogous theory like computability and complexity theory for symbolic computation.

Further material

- Related papers to the above: Bastounis et al. (2022), Boche et al. (2022)
- On decidability of learnability: Caro (2023)
- An older overview of computability theory of neural networks: Šíma and Orponen (2003).
- On formal language theory and neural networks: Delétang et al. (2023), Strobl et al. (2024)

7 Using statistical mechanics

Readings

- Roberts and Yaida (2022)
- Bahri et al. (2020)

Key concepts

- Analogy between statistical mechanics in physics and neural networks in machine learning

Further material

- The book by Roberts and Yaida (2022) has been presented in a course (`https://deeplearningtheory.com/lectures/`).

8 Topological data analysis

Readings

- Naitzat et al. (2020)
- Overview: Hensel et al. (2021)

Key concepts

- TBA

Further material

- For a short explanation of persistent homology, see here.
- For a popular science application to neuroscience, see here.

9 Geometric deep learning

Readings

- Bronstein et al. (2021)

Key concepts

- Applying the Erlangen Program to Machine Learning

Further material

- A website of the project is found here.

10 Category theory as a language of machine learning

Readings

- A categorical framework to describe how Large Language Models move from next-word-probability distributions to syntax and semantics of language: Bradley et al. (2021)
- Overview interactions of category theory and machine learning: Shiebler et al. (2021)

Key concepts

- TBA

Further material

- The paper by Bradley et al. (2021) is also covered on the first author's great blog here.
- More on categories for AI in this course (`https://cats.for.ai/`) and this list of papers on the topic (`https://github.com/bgavran/Category_Theory_Machine_Learning`).
- A(nother) language for design patterns of AI architectures: van Bekkum et al. (2021).

11 Dynamical systems

Readings

- Overview: Bournez and Pouly (2021)

Key concepts

- Dynamical system
- Monoid action, differential equation
- Classification of models of computation into space-discrete vs space-continuous and time-discrete vs time-continuous.
- Characterization of expressive power of different neural network architectures in terms of symbolic models of computation
- General purpose analog computer, equivalence with computable analysis

The key take-away from this chapter is that basically all forms of computation, be it symbolic or not, can be regarded as dynamical systems. A dynamical system consists out of a state space (consisting of the states that the system can be in) and a dynamics (describing how the system evolves over time from the current state to the next states). And, indeed, any computing machine is, at any given time, in some state, and to compute, it updates its state step-by-step according to some procedure or algorithm.

Thus, dynamical systems theory provides a rich and all-encompassing framework to talk about different models of computation. The paper provides an overview of how this perspective furthers our understanding of specific models of computation, focusing on the analog ones (including neural networks) rather than the digital ones (like Turing machines).

Further material

- Further reading: Saxe et al. (2014) and E et al. (2022).
- Popular science videos on analog computation: here and here

12 Verification of neural networks

Readings

- Albarghouthi (2021)

Key concepts

- TBA

Further material

- TBA

13 Post-hoc explainability

Readings

- Bilodeau et al. (2024)

Key concepts

- TBA

Further material

- TBA

Bibliography

Albarghouthi, A. (2021). *Introduction to Neural Network Verification*. arXiv: 2109.10317 [cs.LG] (cit. on p. 26).

Bahri, Y. et al. (2020). "Statistical Mechanics of Deep Learning." In: *Annual review of condensed matter physics* 11.1, pp. 501–528. DOI: 10.1146/annurev-conmatphys-031119-050745 (cit. on p. 21).

Bastounis, A., A. C. Hansen, and V. Vlačić (2022). *The extended Smale's 9th problem – On computational barriers and paradoxes in estimation, regularisation, computer-assisted proofs and learning*. arXiv: 2110.15734 [math.OC] (cit. on p. 20).

Belkin, M. (2021). "Fit without fear: remarkable mathematical phenomena of deep learning through the prism of interpolation." In: *Acta Numerica* 30, pp. 203–248 (cit. on pp. 10, 16 sqq.).

Berner, J. et al. (2022). "The Modern Mathematics of Deep Learning." In: *Mathematical Aspects of Deep Learning*. Ed. by P. Grohs and G. Kutyniok. Cambridge: Cambridge University Press, pp. 1–111. DOI: 10.1017/9781009025096.002 (cit. on pp. 14, 16 sq.).

Bilodeau, B. et al. (2024). "Impossibility theorems for feature attribution." In: *Proceedings of the National Academy of Sciences* 121.2, e2304406120 (cit. on p. 27).

Boche, H., A. Fono, and G. Kutyniok (2022). *Inverse Problems Are Solvable on Real Number Signal Processing Hardware*. arXiv: 2204.02066 [eess.SP] (cit. on p. 20).

Boden, M. A. (2016). *AI: Its nature and future*. Oxford: Oxford University Press (cit. on p. 6).

Bournez, O. and A. Pouly (2021). "A Survey on Analog Models of Computation." In: *Handbook of Computability and Complexity in Analysis*. Ed. by V. Brattka and P. Hertling. Cham: Springer International Publishing, pp. 173–226. DOI: 10.1007/978-3-030-59234-9_6 (cit. on p. 25).

Bradley, T.-D., J. Terilla, and Y. Vlassopoulos (2021). *An enriched category theory of language: from syntax to semantics*. arXiv: 2106.07890 [math.CT] (cit. on p. 24).

Bringsjord, S. and N. S. Govindarajulu (2024). "Artificial Intelligence." In: *The Stanford Encyclopedia of Philosophy*. Ed. by E. N. Zalta and U. Nodelman. Summer 2024. Metaphysics Research Lab, Stanford University (cit. on p. 6).

Bronstein, M. M. et al. (2021). *Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges*. arXiv: 2104.13478 [cs.LG] (cit. on p. 23).

Caro, M. C. (12/2023). "From undecidability of non-triviality and finiteness to undecidability of learnability." In: *International Journal of Approximate Reasoning* 163, p. 109057. DOI: 10.1016/j.ijar.2023.109057. URL: https://arxiv.org/abs/2106.01382 (cit. on p. 20).

Colbrook, M. J., V. Antun, and A. C. Hansen (2022). "The difficulty of computing stable and accurate neural networks: On the barriers of deep learning and Smale's 18th problem." In: *Proceedings of the National Academy of Sciences* 119.12, e2107151119. DOI: 10.1073/pnas.2107151119. eprint: https://www.pnas.org/doi/pdf/10.1073/pnas.2107151119. URL: https://www.pnas.org/doi/abs/10.1073/pnas.2107151119 (cit. on p. 19).

Delétang, G. et al. (2023). *Neural Networks and the Chomsky Hierarchy*. arXiv: 2207.02098 [cs.LG] (cit. on p. 20).

E, W., J. Han, and Q. Li (2022). "Dynamical Systems and Optimal Control Approach to Deep Learning." In: *Mathematical Aspects of Deep Learning*. Ed. by P. Grohs and G. Kutyniok. Cambridge University Press, pp. 422–438. DOI: 10.1017/9781009025096.011 (cit. on p. 25).

Flasiński, M. (2016). *Introduction to Artificial Intelligence*. Cham: Springer. DOI: https://doi.org/10.1007/978-3-319-40022-8 (cit. on p. 7).

Grohs, P. and G. Kutyniok, eds. (2022). *Mathematical Aspects of Deep Learning*. Cambridge University Press. DOI: 10.1017/9781009025096 (cit. on p. 2).

Hensel, F., M. Moor, and B. Rieck (2021). "A Survey of Topological Machine Learning Methods." In: *Frontiers in Artificial Intelligence* 4. DOI: 10.3389/frai.2021.681108 (cit. on p. 22).

Hornik, K., M. Stinchcombe, and H. White (1989). "Multilayer feedforward networks are universal approximators." In: *Neural Networks* 2.5, pp. 359–366. DOI: https://doi.org/10.1016/0893-6080(89)90020-8 (cit. on pp. 8, 13 sq.).

Immerman, N. (2021). "Computability and Complexity." In: *The Stanford Encyclopedia of Philosophy*. Ed. by E. N. Zalta. Winter 2021. Metaphysics Research Lab, Stanford University (cit. on p. 7).

Kratsios, A. (2021). "The Universal Approximation Property." In: *Annals of Mathematics and Artificial Intelligence* 89.435-469. DOI: https://doi.org/10.1007/s10472-020-09723-1 (cit. on pp. 14 sq.).

Liu, Z. et al. (2024). *KAN: Kolmogorov-Arnold Networks*. arXiv: 2404.19756 [cs.LG] (cit. on p. 14).

Naitzat, G., A. Zhitnikov, and L.-H. Lim (2020). "Topology of Deep Neural Networks." In: *Journal of Machine Learning Research* 21.184, pp. 1–40. URL: http://jmlr.org/papers/v21/20-345.html (cit. on p. 22).

Roberts, D. A. and S. Yaida (2022). *The Principles of Deep Learning Theory: An Effective Theory Approach to Understanding Neural Networks*. Cambridge: Cambridge University Press. DOI: 10.1017/9781009023405. URL: https://arxiv.org/abs/2106.10165 (cit. on p. 21).

Russell, S. J. and P. Norvig (2021). *Artificial Intelligence: A Modern Approach*. Pearson series in artificial intelligence. Harlow: Pearson (cit. on pp. 3, 5 sq.).

Saxe, A. M., J. L. McClelland, and S. Ganguli (2014). *Exact solutions to the nonlinear dynamics of learning in deep linear neural networks*. arXiv: 1312.6120 (cit. on p. 25).

Shalev-Shwartz, S. and S. Ben-David (2014). *Understanding Machine Learning: From Theory to Algorithms*. A copy for personal use only at https://www.cs.huji.ac.il/~shais/UnderstandingMachineLearning/copy.html. New York: Cambridge University Press (cit. on p. 8).

Shiebler, D., B. Gavranović, and P. Wilson (2021). *Category Theory in Machine Learning*. URL: https://arxiv.org/abs/2106.07032 (cit. on p. 24).

Šíma, J. and P. Orponen (2003). "General-Purpose Computation with Neural Networks: A Survey of Complexity Theoretic Results." In: *Neural Computation* 15, pp. 2727–2778 (cit. on p. 20).

Sterkenburg, T. F. and P. D. Grünwald (2021). "The no-free-lunch theorems of supervised learning." In: *Synthese* 199.3-4, pp. 9979–10015. DOI: 10.1007/s11229-021-03233-1 (cit. on pp. 11, 15).

Strobl, L. et al. (2024). *What Formal Languages Can Transformers Express? A Survey*. arXiv: 2311.00208 [cs.LG] (cit. on p. 20).

Van Bekkum, M. et al. (2021). "Modular design patterns for hybrid learning and reasoning systems: a taxonomy, patterns and use cases." In: *Applied Intelligence* 51, pp. 6528–6546. DOI: 10.1007/s10489-021-02394-3 (cit. on p. 24).