Philosophy and Theory of Artificial Intelligence

A Reader

Draft from January 24, 2025.
Please don’t cite or distribute without permission.
Comments welcome!

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

Contents

Preface

This is the reader for the course "Philosophy and Theory of Artificial Intelligence" given during the winter semester 2024/25 at *LMU Munich* as part of the *Master in Logic and Philosophy of Science*. The reader is updated as the course progresses. A website with all the course material is found at

> `https://levinhornischer.github.io/PhilTheoAI/`.

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 This course provides, as its title suggests, an introduction to both the *philosophy* and the *theory of artificial intelligence*. This field researches the philosophical foundations of artificial intelligence (AI). It recently gained much prominence because of its urgent relevance. AI made astonishing but also disconcerting technological progress. For a recent example, just think of *ChatGPT*. However, we are lacking a theoretical understanding of AI. We would like to answer questions like the following. Why are neural networks—that underlie modern AI—so good at learning from data? And what kind of knowledge do they have? How do they compare to the human-interpretable symbolic AI models that have been used previously? What is even a good language to talk about AI models and the computation that they perform? What are the possibilities and limitations of AI models? We will in particular also investigate the problems of modern AI: How to do deal with its ethical issues like bias or fairness. We look at the black-box problem of neural networks: that they are difficult to interpret and hard to explain. And we consider their lack of robustness: that in similar situations they unexpectedly might behave incorrectly. Answering these questions is not just an engineering task: it crucially also is a philosophical task—which we undertake in this course.

The course title was inspired by the conference series of the same name.

Objectives In terms of content, the course aims to convey an overview of the questions, methodology, and results of the philosophy and theory of AI. We cover both classic material and cutting-edge research. In terms of skills, the course aims to teach: (1) the basic ability to program an AI model, (2) the ability to critically reflect on the many issues of AI by relating it to established theory in philosophy, and (3) the ability to apply results from the theory of AI to assess its power and limits.

Prerequisites The course does not assume any programming knowledge. It assumes basic familiarity with philosophy (first-year university level), logic (e.g., an introductory course) and mathematics (though not really beyond high-school level). None of these are strictly necessary: by far most of the reader can be understood also without, it will mostly be helpful to appreciate, e.g., remarks about connected and more advanced topics.

Schedule and organization The course is organized as a seminar. Hence, for each session, we have assigned readings. During the session, we first make sure that we all have understood the provided key AI concepts relevant for the session (by arriving at an explanation in the group), and then we critically discuss the readings. Study questions in this reader are meant to both help engagement with the readings and as starting points for discussion in class. The schedule for the readings is found on the course's website. In sum, the readings aim to provide an overview of the field of philosophy and theory of AI.

After an introduction to modern AI, the organizational principle for selecting the readings was 'question-based'. Each chapter concerns one 'big question' about the philosophy of AI. See the table of contents for a list of those chapters. As usual, there is much more possible content than time, and during the course we can still decide on which of the readings we will focus on.

Other organizational principles would be possible, too; e.g., 'method-based'.

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

Further study material In addition to the provided papers and further material, some helpful short explainer videos on AI are found here. And on philosophy of neuroscience here. References for 'classical' philosophy of AI (i.e., up to the early 1990s) are, e.g., Boden (1990) and Copeland

(1993). A recent book is M. Mitchell et al. (2019), as talk here. A new 'Philosophical Glossary of AI' is found here.

Notation Throughout, 'iff' abbreviates 'if and only if'. Study questions are marked by

$\rightsquigarrow$ This is a study question.

Acknowledgement I have taken great inspiration in designing this course from other courses on this topic both by Stephan Hartmann and Timo Freiesleben and by Cameron Buckner.

1 Introduction to AI

This chapter's big question

What actually is an AI system and, in particular, a neural network?

Key concepts

- History of AI: Ada Lovelace, Alan Turing, McCulloch & Pitts, Logic Theorist, Dartmouth workshop, division of AI into life (cybernetics, connectionism, differential equations) and mind (symbolic computing, logic), big data, deep learning revolution.
- Types of AI: classical/symbolic vs subsymbolic/neural networks/connectionism.
- Definitions of AI: Turing test (more on this in chapter 2 and figure 1.1), technological vs scientific aim of AI, virtual vs physical machines
- Types of learning tasks: Supervised learning, unsupervised learning, reinforcement learning. Machine learning pipeline (conceptualization, data, model, deployment).
- Artificial neural networks: neurons, layers, feedforward vs recurrent, weights, activation function, loss function (as your way of telling the neural network what to optimize for), backpropagation, learning rate, local/global minima (equilibrium), regularization, overfitting/underfitting.

Before one can do *philosophy of* X, one needs a good understanding of X. So we start with an introduction to AI, both practically and theoretically.

Lecture 1 For a practical introduction, we see how an AI system is actually built in practice. We consider the standard example of training a neural network to classify hand-written digits (on the MNIST dataset). Thus, we get a concrete idea of what an 'AI system' really is and this does not remain an abstract term in future discussion. We build the system in the form of a coding exercise, which is purpose-built for this course and available on the course website. But—fear not—you do not need any coding experience for this! In class, we go through the parts of the coding exercise. As a study question, you are asked to:

⇝ Change the parameters and see how the performance of the neural network changes.

⇝ Your challenge is to find some parameters with which the networks reaches an accuracy of 98%.

In the next lecture, we will discuss your observations.

Lecture 2 For a theoretical introduction, the readings below and also the coding exercise introduce the central concepts of modern AI, which are summarized in this chapters list of 'key concepts'.

⇝ Make sure that, by doing the readings, you know what these concepts mean.

We discuss them in the second lecture of the course.

The readings also mentioned the following further advanced concepts. You can skip them on a first reading and come back to them at a later stage:

- biological plausibility (backprop too global, Boden's 'too neat, too simple, too few, too dry', neuromodulation like GasNet),
- Hebbian learning (fire together, wire together),
- predictive coding (Helmholtz's 'unconscious inference, the 'Bayesian brain', cognition as predicting incoming low-level sense information from higher-level neural layers),
- perceptron (XOR problem),
- localist (concepts represented by single neurons) vs distributed networks (concepts stored across the whole system)

Readings

- A very accessible overview, written at the beginning of the deep learning revolution: M. A. Boden (2016). *AI: Its nature and future*. Oxford: Oxford University Press. Chapters 1 and 4.
- Short explainer videos of central concepts in AI are found here.

	empiricist	*rationalist*
thought process	thinking humanly (cognitive modeling)	thinking rationally (logic & probability)
behavior	acting humanely (Turing test)	acting rationally (decision theory)

Figure 1.1: The four definitions of artificial intelligence of Russell and Norvig (2021, ch. 1) according to whether an AI system should realize thought processes (first row) or behavior (second row) and whether the benchmark is human (left column) or ideal (right column) performance.

An excellent detailed mini-series explaining neural networks is found here.

- The coding exercise on the course website.

Further material

- A great interactive visualization of neural networks is found here.
- A concise introduction to deep learning and its philosophical aspects: C. Buckner (2019). “Deep learning: A philosophical introduction.” In: *Philosophy Compass* 14.10, e12625. DOI: `https://doi.org/10.1111/phc3.12625`.
- An introduction to AI from the standard textbook: Russell and Norvig (2021, ch. 1). They have a fourfold definition of AI: acting humanly (Turing test), thinking humanly (cognitive modeling), thinking rationally (logic, probability), acting rationally (rational agent; perfect vs limited rationality).
- An encyclopedia entry on AI: S. Bringsjord and N. S. Govindarajulu (2022). “Artificial Intelligence.” In: *The Stanford Encyclopedia of Philosophy*. Ed. by E. N. Zalta and U. Nodelman. Fall 2022. Metaphysics Research Lab, Stanford University
- A great introduction to AI: M. Mitchell et al. (2019). “Artificial intelligence: A guide for thinking humans.” In

2 AI and philosophy of mind

This chapter's big question

Can AI systems think? Like humans?

Key concepts

- Turing test
- Symbol grounding problem
- Octopus test/Chinese room though experiment
- Classicist vs connectionist theories of mind
- Levels of analysis: neural, subsymbolic, symbolic; Marr's levels
- Physical symbol system hypothesis (Newell–Simon) vs Connectionist dynamical system hypothesis (Smolensky)

Lecture 3 A classic text on the question whether an AI system—or, simply, a 'machine'—can think is Turing's 1950 paper, in which he introduces what is now known as the *Turing test*. Some even take this as the very definition of artificial intelligence, i.e., when a machine should be considered intelligent (the other three are in figure 1.1). The Turing test operationalizes the intuitive question 'Can machines think?' in a behavioristic way: can you distinguish whether answers to your questions where produced by the machine or by a human?

*Turing is a giant of computer science. There is even a Hollywood movie (*The Imitation Game*, 2014) portraying Turing's eventful and also tragic life (e.g., he was prosecuted in 1952 for his sexual orientation).*

⇝ Can we really only test for intelligence in a purely behavioristic way (up to 'extension') and not also capture aspects of thought processes ('intensional' aspects)?

⇝ Are response times (or computational complexity) aspects of thought processes? If the question was to multiply 793 and 868, the machine could add an artificial pause to mimic a human response; however, the machine is exposed if it is not as quick as humans in task they find easy (e.g., pattern recognition).

⇝ What is the relation between the two sentences (1) ‘This machine can think’ and (2) ‘This machine passes the Turing test’? According to Turing, we can only say something about (2), which is the extensional/observable reflection of (1). Is (2) or *rational reconstruction* of (1), serving as a good definition of AI? Or can one also ask if (2) is a *conceptual analysis* (1), i.e., is (2) sufficient and necessary for (1)? A infamous argument against necessity is the Chinese room thought experiment: A person who doesn’t speak Chinese is in a room and responds to incoming Chinese message according to a long look-up table; from the ‘outside’, this generates the correct behavior, but on the ‘inside’ there is no understanding—or so the argument goes. (For more, see Oppy and Dowe (2021).)

See Glymour (2015, p. 357) for the idea that the person in the Chinese room also has to match the computation time of a Chinese speaker, not just the input-output relation.

Fast-forward 70+ years, seeing ‘machines’ like ChatGPT, what about Turing’s test? Is it solved? The next paper discusses (and denies) whether large language models (LLMs) can, apart from producing the sensible text response, also be said to understand the meaning of this text and in this sense be intelligent.

The first author, Emily M. Bender, is an influential researcher in Natural Language Processing and AI Ethics, who is also well-known for the stochastic parrot *paper (Bender, Gebru, et al. 2021).*

> We argue that, independently of whether passing the Turing test would mean a system is intelligent, a system that is trained only on form would fail a sufficiently sensitive test, because it lacks the ability to connect its utterances to the world (Bender and Koller 2020, p. 5188).

So the system cannot solve the *symbol grounding problem*, a term coined by Harnad (1990).

This reminiscent of Quine’s indeterminacy of translation (for an overview, see, e.g., here or here), so you may discuss connections.

⇝ Do you find the Octopus though experiment convincing that is to establish the above cited conclusion? (We also will further discuss this in chapter 5.)

⇝ What about the concrete case of training a LLM on all available Java code but without information about compilers or input-output relations of specific programs? Is there no way to learn the meaning relation $J \subseteq E \times I$, which relates a piece e of Java code to the mathematical function i that it computes?

Lecture 4 On the question of how the symbol grounding problem can be solved, the next paper by Smolensky (1988) discusses how a neural network based approach to modeling intelligence can give rise to symbolic meaning by developing the concept of a *subsymbol*.

The background of the debate is the strong divide, especially at the time, in cognitive science between the traditional symbolic approach and the connectionist approach. Here cognitive science—or cognitive modeling—is understood as the building of formal models of how the human mind/intelligence solves various cognitive tasks like vision, language, planning, motor control, etc. As such, the goal is the same as for AI, modulo the question of whether the formal model should operating similar to a human (cf. scientific aim of AI) or not necessarily (cf. technological aim of AI).

The symbolic approach conceptualizes a cognitive task as an input-output function and the formal model is a general algorithm of computing this function. For example, for the cognitive task of route planning, the input is a map with a current position and a desired destination and the output is a route of how to get to this destination—the algorithm would need to describe a general procedure of computing such a rule. The conviction that this is the right approach to cognitive modeling is expressed in:

> "*The Physical Symbol System Hypothesis*. A physical symbol system has the necessary and sufficient means for general intelligent action" (Newell and Simon 1976, p. 116).

The connectionist approach, on the other hand, conceptualizes a cognitive task still as an input-output function, but at a lower level: The input is a long list of real numbers describing, e.g., some sensory input or maybe a numerical encoding of some symbolic information (e.g., a pixel image of the map) and the output similarly is a numeric encoding for the intended answer to the cognitive task. This input-output function is computed by a neural network, as we have discussed them so far.

> "*The connectionist dynamical system hypothesis*: The state of the intuitive processor at any moment is precisely defined by a vector of numerical values (one for each unit). The dynamics of the intuitive processor are governed by a differential equation. The numerical parameters in this equation constitute the processor's program or knowledge. In learning systems, these parameters change according to another differential equation" (Smolensky 1988, p. 6, emphasis added).

The question is then how the symbolic approach and the connectionist approach are related, i.e., how symbolic meaning may be represented and processed in neural networks. That is discussed at length in the paper, but the key idea is:

> “The entities in the intuitive processor with the semantics of conscious concepts of the task domain are complex patterns of activity over many units. Each unit participates in many such patterns” (Smolensky 1988, p. 6).

So a symbolic meaning (like ‘the digit 7’) is represented by a large number of subsymbols, i.e., activation patterns in the neural network (all the ways neurons can be activated to give high probability to the neuron in the output layer representing the digit 7). While the symbolic paradigm hence has a simple semantics for its symbols but allows complicated operations on these symbols, the connectionist paradigm only allows simple operations on the neurons (weighted sums plus ReLU) hence the semantics for the subsymbols must be complicated, to match the computational power of the symbolic paradigm.

- ⇝ The symbolic approach operates at a higher level than the connectionist approach. But Smolensky also introduces a third, yet lower level: the neural level, i.e., the activity in an actual brain. In what way are the artificial neural networks of connectionism more high-level than the biological neural networks studied in neuroscience? How do these three levels relate to Marr’s tri-level hypothesis?

- ⇝ In (13)(c–d), Smolensky hypothesizes when it may be possible to give a symbolic semantics to a subsymbolic system, namely according to whether or not the cognitive task involves conscious rule application or just intuition. Can you think of examples? How does this distinction line up with Kahneman’s “Thinking, Fast and Slow” (2011): System 1 (implicit, fast, parallel, instinctive, emotional) vs system 2 (explicit, slow, sequential, deliberative, logical)? For a concrete example of a (symbolic) causal abstraction of a subsymbolic neural network, see e.g. (Geiger et al. 2021).

- ⇝ If the cognitive agent (or intuitive processor) is a dynamical system, which dynamical systems then are cognitive? In (19), Smolensky mentions a necessary condition: the system should be stable under a wide range of environmental conditions. Compare this to the discussion of when a dynamical system performs computation (Piccinini and Maley 2021).

Readings

- A. M. Turing (1950). “Computing Machinery and Intelligence.” In: *Mind* 59.236, pp. 433–460. DOI: https://doi.org/10.1093/mind/LIX.236.433.

This text contains comments about race (e.g., p. 448) and gender stereotypes (e.g., p. 434) which should be reflected on critically. Given the classic status of the text, it is included in the syllabus here, but this is to flag that these comments are not silently endorsed.

- E. M. Bender and A. Koller (2020). “Climbing towards NLU: On Meaning, Form, and Understanding in the Age of Data.” In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 5185–5198.
- P. Smolensky (1988). “On the proper treatment of connectionism.” In: *Behavioral and brain sciences* 11.1, pp. 1–74. DOI: https://doi.org/10.1017/S0140525X00052791. Note: You only need to read the paper by Smolensky, i.e., pages 1–23, *not* the replies following it!

This is quite a dense text: on a first reading focus on understanding the key concepts mentioned above. This is a text worth coming back to over and over again.

Further material

- For an overview of the discussion around the Turing test, see: G. Oppy and D. Dowe (2021). “The Turing Test.” In: *The Stanford Encyclopedia of Philosophy*. Ed. by E. N. Zalta. Winter 2021. Metaphysics Research Lab, Stanford University.
- Further reading on the idea that cognitive agents are dynamical systems: T. van Gelder (1998). “The dynamical hypothesis in cognitive science.” In: *Behavioral and Brain Sciences* 21.5, pp. 615–628. DOI: 10.1017/S0140525X98001733.
- As mentioned above, an example of (symbolic) causal abstraction of a subsymbolic neural network: A. Geiger et al. (2021). “Causal abstractions of neural networks.” In: *Advances in Neural Information Processing Systems* 34, pp. 9574–9586. URL: https://arxiv.org/abs/2106.02997.

3 AI and epistemology

This chapter's big question

How do AI systems gain knowledge, if any?

Key concepts

- Empiricists/nurture vs rationalists/nature
- Domain-general vs domain-specific cognitive systems
- Moderate empiricism: allow 'innate' general inductive biases to learn domain-specific knowledge.
- Control Problem: If several general modules (representing various cognitive faculties) are posited, how do they fruitfully interact?

Lecture 5 When it comes to the question what kind of knowledge deep neural networks gain, it makes sense to turn to philosophy. The subfield of epistemology has discussed the question of how we gain knowledge for millennia. There are two positions: Empiricists (like Locke) say all knowledge comes from sensory experience, while rationalists (like Leibniz) say that in getting knowledge we rely on our innate concepts about the basic structure of the world. (And Kant aimed to combine these two traditions.) Thus, it is suggestive to associate deep learning with empiricists (neural networks only learn from data) and symbolic AI with rationalists (the hard coded program of solving the task is the innate knowledge about the world). (Neuro-symbolic integration then follows Kant in combining the two approaches: see the further reading box.) However, this chapter argues for a more nuanced moderate position. It endorses the "new empiricist DoGMA [that a] (Do)main General Modular Architecture is the best hope for modeling rational cognition in AI" (p. 26).

As remarked in footnote 28 of the text, this is in reference to the famous paper of Quine (1951) "Two Dogmas of Empiricism" (here's a lecture on it).

⇝ What is the main argument against the simple association of deep learning with empiricism and symbolic AI with rationalism? Can you think of examples (e.g., CNNs)? Relate that to the concept of *inductive bias* (we'll come back to that in chapter 7).

⇝ Explain why "the current incarnation of the nativist-empiricist debate in artificial intelligence presents us with a similar golden opportunity, in which we might attempt one of the rarest feats of intellectual alchemy: the conversion of a timeless philosophical riddle into a testable empirical question" (p. 8)? Do you agree? How does this relate to the continuum of views between empiricism and rationalism?

⇝ Which of the proposed theories in section 1.5 of how a (computational) model of cognition relates to the mind do you find most convincing?

Readings

- C. J. Buckner (2023). *From Deep Learning to Rational Machines: What the History of Philosophy Can Teach Us about the Future of Artificial Intelligence*. New York: Oxford University Press. DOI: `https://doi.org/10.1093/oso/9780197653302.001.0001`. Chapter 1 "Moderate Empiricism and Machine Learning".

Further material

- Given this distinction between rationalist/symbolic and empiricist/subsymbolic approaches to AI, wouldn't it make sense to combine the two (as Kant attempted)? Especially in light of the fact that they have complementary benefits? To get the best of both worlds, rather than sharp opposition? There is a movement aiming to do precisely this: known as *neurosymbolic computation*. (Buckner cites 'cognitive' versions of this—namely ACT-R, SOAR, Sigma, and CMC—on page 39.) A recent overview is:

  A. d. Garcez and L. C. Lamb (2023). "Neurosymbolic AI: the 3rd wave." In: *Artificial Intelligence Review*. DOI: `https://doi.org/10.1007/s10462-023-10448-w`

  The main idea is that of neurosymbolic computation: "combine robust learning in neural networks with reasoning and explainability by offering symbolic representations for neural models" (quoted from the abstract). Ideally in a neural-symbolic cycle: compile a neural network from symbolic knowledge (in weights or semantic loss function) and decompile the neural network into symbolic knowledge.

4 AI and philosophy of science

This chapter's big question

Is machine learning the 'end of theory'?

Referring to a 2008 Wired article by Chris Anderson with the title 'The End of Theory: The Data Deluge Makes the Scientific Method Obsolete'.

Key concepts

- Opacity problem
- elementwise/localist vs holistic/distributed representation
- Model audit vs scientific inference

Lecture 6 We have discussed how neural networks can gain knowledge about the world from the data they are trained on. This also is the goal of science—understanding the world based on observations of it. So can neural networks replace scientific theorizing, as the provocative quote above suggests? We will discuss this in this chapter.

Long before the deep learning revolution, it has been noted that there is an intriguing similarity between the fields of machine learning and philosophy of science. Philosophy of science investigates the best way of doing *scientific induction*: building a theory or model from observed data. And machine learning does something very similar: training a model (e.g., neural network) on collected data. Because of this, some even identify the two fields (Korb 2004), while others describe their relation in a more nuanced way as a *dynamic interaction* (Williamson 2004). (Some new progress in machine learning may inform philosophy of science, and at other times philosophy of science may help theorizing in machine learning.)

However, there also is a crucial difference between modern deep learning and science: the former produces opaque models that solely focus on prediction, but scientific models should also offer explanations of the phenomenon they describe. This point is forcefully made by Noam Chomsky in the discussion of ChatGPT:

> Perversely, some machine learning enthusiasts seem to be

> proud that their creations can generate correct "scientific" predictions (say, about the motion of physical bodies) without making use of explanations (involving, say, Newton's laws of motion and universal gravitation). But this kind of prediction, even when successful, is pseudoscience. While scientists certainly seek theories that have a high degree of empirical corroboration, as the philosopher Karl Popper noted, "we do not seek highly probable theories but explanations; that is to say, powerful and highly improbable theories."
>
> The theory that apples fall to earth because that is their natural place (Aristotle's view) is possible, but it only invites further questions. (Why is earth their natural place?) The theory that apples fall to earth because mass bends space-time (Einstein's view) is highly improbable, but it actually tells you why they fall. True intelligence is demonstrated in the ability to think and express improbable but insightful things. (Chomsky et al. 2023)

The main reading discusses this problem that even if modern deep learning models predict the scientific phenomenon near perfectly, they are still not the *kind of* models that scientists favor. This is because these AI models are very complex (so it is hard to 'understand' the model) and it is unclear how the parts of the model relate to the parts of the phenomenon. This is known as the *opacity problem* (Boge 2021; Sullivan 2022). Interpretable machine learning and explainable artificial intelligence aim to make AI models more 'understandable'. But it is not clear whether these methods can be used to draw scientific inference about the real phenomenon from the AI model. The paper discusses why and how this problem can be addressed.

This recently turned into an online book, of which you can also read, e.g., the first part.

- ⇝ Why is 'just' giving the right predictions (like an ML model) not enough in science? Or do you think it actually is?
- ⇝ Explain the distinction between elementwise/localist representation and holistic/distributed representation. Isn't it a problem if we do not have a compositional understanding of our model—isn't this the hallmark of science?
- ⇝ What is the differences between "auditing ML models" and "leveraging them for scientific inference".

⇝ How can the 'property descriptors' bridge this gap, intuitively speaking?

Readings

- T. Freiesleben, G. König, et al. (2024). "Scientific Inference with Interpretable Machine Learning: Analyzing Models to Learn About Real-World Phenomena." In: *Minds and Machines* 34.3. DOI: https://doi.org/10.1007/s11023-024-09691-z

This recently turned into an online book.

Further material

- The above mentioned papers on the relation between machine learning and philosophy of science:

  K. B. Korb (2004). "Introduction: Machine Learning as Philosophy of Science." In: *Minds and Machines* 14, pp. 433–440. DOI: https://doi.org/10.1023/B:MIND.0000045986.90956.7f.

  J. Williamson (2004). "A Dynamic Interaction Between Machine Learning and the Philosophy of Science." In: *Minds and Machines* 14, pp. 539–549. DOI: https://doi.org/10.1023/B:MIND.0000045990.57744.2b.

- The book on machine learning for science:

  T. Freiesleben and C. Molnar (2024). *Supervised Machine Learning for Science. How to stop worrying and love your black box.* URL: https://ml-science-book.com/.

5 AI and philosophy of language

This chapter's big question

Do Large Language Models have linguistic and cognitive competence or are they just stochastic parrots?

The term 'stochastic parrot' is from the title of the paper by Bender, Gebru, et al. (2021).

Key concepts

- The Blockhead thought experiment.
- Symbolic vs statistical approaches to language (generative grammars vs distributional hypothesis).
- Word embeddings (Word2Vec).
- Transformer-based LLMs: tokens, self-attention, next-token prediction.
- Pre-training then fine-tuning with reinforcement learning from human feedback (RLHF), three Hs: helpfulness, harmlessness, honesty.
- In-context learning, few-shot learning, zero-shot learning.
- Re-description fallacy (a computer only manipulates 0s and 1s, so it cannot possibly do . . .), need for probing inner mechanisms.
- Compositional generalization.
- Continuity principle.
- Poverty of the stimulus.
- Externalism in philosophy of language.
- Benchmarks (Goodhart's Law, data contamination, performance vs competence)

- Opening black box: probing, feature attribution, causal intervention.
- Mechanistic interpretability: activation patching, linear representation hypothesis, causal abstraction.
- Grokking (initially overfit, but later in training suddenly generalize well)

Lecture 7 While neural networks for computer vision were most dominant in AI during the 2010s, this shifted to Large Language Models (LLMs) at the beginning of the 2020s. Their inputs (prompts) and outputs (generated response) are both text in natural language, so LLMs readily connect to philosophy of language. We focus on three classical debates.

1. Compositionality: Can neural networks learn compositional structure (e.g., treat complex sentences consistently with how their parts are treated)? Dilemma: either they cannot and symbolic approaches to mind (which easily can) are better, or they can but (a) only implement an explicit symbolic architecture or (b) they learn a proper, but non-classical representation of compositional structure (continuity principle). Evidence for meta-learning points to the second horn of the dilemma, but distinguishing (a) and (b) needs to consider the internal mechanism.
2. Language acquisition: Is it possible to learn a language from scratch just by exposition or do you need some innate rules about the language? Nativism says the latter, but comes in two forms. (1) Strong learnability: no amount of linguistic data is enough to learn syntactic knowledge without innate grammar; (2) weak learnability: the amount of data that children get during language learning is not enough to learn syntactic knowledge without innate grammar ('poverty of the stimulus'). LLMs challenge (1), but open regarding (2) which needs more knowledge of the internal mechanism.
3. Language understanding: Coming back to the octopus thought experiment (from chapter 2), can neural networks learn linguistic meaning, beyond form? Three worries: (1) meaning cannot be learned just from form, (2) without grounding internal representation to the external world, an LLM can only gain meaning through an external interpreter (e.g., user), (3) words only get meaning when the speaker has communicative intentions. Distinction: inferential vs referential

aspects of semantic competence. The former seems within reach, the latter depends on conception of reference; easier with externalist conception in philosophy of language, potentially aided by indirect causal ties to the world via RLHF.

See, e.g., here.

Some study questions on these issues:

⇝ What are the symbolic and the statistical approach to language? Do they contradict each other or do they capture two different aspects of language? Which do you think is more plausible?

⇝ How could one distinguish between options (a) and (b) in the second horn of the dilemma about compositionality?

⇝ If LLMs should be more than fancy lookup tables (more than blockheads), what can or should this 'more' consist in?

Lecture 8 As we saw in the previous lecture, many of the classical debates in the philosophy of language and related fields are informed by LLMs, but often a more substantive answer would require more knowledge about the internal mechanisms of LLMs. For example, regarding language acquisition, LLMs refute the strong claim that language can never be learned without innate grammar, but the weaker claim allowing only the amount of data that children get was left open. In this lecture, we review ways to probe the inner mechanisms of LLMs and neural networks more generally.

A theme that came up several time already is the distinction between the *externally* observable behavior of an AI-system (typically its input output behavior) and the *internal* process of the AI-system that realizes this behavior. The former has the advantage of being directly accessible allowing for direct comparisons to *benchmarks* provided either by human behavior or other AI-systems. But it also has three disadvantages:

1. AI-system typically are built to outperform a given benchmark, but when a measure becomes a target, it ceases to be a good measure (aka Goodhart's Law).
2. Since LLMs are trained on text on the internet and benchmarks are online, the test data can leak into the training data (aka *data contamination*), making the benchmark even less of a good measure.
3. A benchmark has trouble distinguishing *performance* from *competence*. For example, a LLM may perform well on Theory of Mind task from

cognitive science (that test whether one attributes mental states to others that are different to one's own); but one cannot conclusively tell whether this is just due memorization after seeing such examples during training and not due true achieving of a theory of mind (performance without competence).

But how, then, can we open the *black box* and access the inner process of neural networks? There are three approaches: probing, feature attribution, and causal intervention.

In probing, one tries to predict, from activation patterns in a part of the neural network, some—e.g., linguistic—feature. If one can do this with high accuracy, this suggests that the neural network represents this feature in this part. Though, this does not mean that the feature plays a causal role or that it is a genuine representation and not just an incidental association.

In feature attribution, one tries to identify which parts of the input to the AI-model was most important for generating the output. For example, when identifying a cat in an image, we expect the pixel showing the cat (its shape, fur, paws, etc.) were more important to the classification than, say the grass in the background. If this is not the case, this suggests that the neural network does not have the right internal process for generating the correct performance. On the other hand, these attributions also only offer limited information: it does not say how the neural network really processes the input information to get to the output.

In causal intervention, one responds to the problems of the the previous two approach by checking if the probe or attribution actually is used by the internal process: namely, if it really makes a difference. By intervening on the neural network one aims to unravel the causal mechanism behind the internal processing, and thus provide a *mechanistic explanation* of the inner workings. In other words, when we ask when an activation pattern in a neural network genuinely represents a given feature, philosophical theories of representation answer:

1. the activation pattern should carry information about the feature,
2. the activation pattern should influence the behavior of the neural network in a task-relevant way, and
3. the model should be capable of misrepresenting the feature (Harding 2023).

Probing and attribution deliver the first (and maybe the last) item, but the second item needs causal interventions. There are three increasingly stronger ways to do so:

- *Ablation*, i.e., disabling groups of neurons to see if this relevantly influences the behavior (though this disregards that neurons are *polysemantic*, i.e., may simultaneously represent several features in 'superposition' (Elhage et al 2022))
- *Iterative nullspace projection* (Ravfogel et al 2020), i.e., after identifying with a probe a particular part of the neural network, test if it is causally influential by adjusting the neural activation pattern in a way that adds or removes the identified information and see if this makes a difference to the behavior.
- *Mechanistic interpretabiltiy* (Geiger et al 2021, Elhage et al 2021, etc.), i.e., describe the entire processing of the neural network (not just a part of it) at an algorithmic level (as a human understandable causal mechanism).

For more information, see the great blogpost on this here.

Some key words for the mechanistic interpretability approach: a commonly used intervention method goes under the name of *activation patching*, aka interchange interventions. The *linear representation hypothesis* says that high-level concepts or features are represented linearly as directions in activation space. *Causal abstraction* is the idea that the macro-level description of the neural network is a causal abstraction of the micro-level process: the micro-level variables (e.g., activations of specific neurons) can be partitioned into sets corresponding to macro-level variables such that a translation function maps values of the variables in a set to a value of the macro-level variable, in a way that is consistent with interventions (i.e., intervening on both the micro-level and the macro-level yields equivalent results).

⇝ Millière and C. Buckner (2024b, p. 16) conclude that the *grokking phenomenon* in LLMs speaks against the skeptical view that LLMs are just giant look up tables (i.e., blockheads) but can actually learn general and 'smart' algorithms to solve a task. Explain that phenomenon and see if you agree with this assessment.

⇝ What would that mean for the questions left open from last lecture? What would potential experiments look like to settle these?

Readings

- First lecture in this chapter: R. Millière and C. Buckner (2024a). “A Philosophical Introduction to Language Models – Part I: Continuity With Classic Debates.” In: arXiv: 2401.03910 [cs.CL].

URL: https://arxiv.org/abs/2401.03910

- Second lecture (especially section 2): R. Millière and C. Buckner (2024b). "A Philosophical Introduction to Language Models – Part II: The Way Forward." In: arXiv: 2405.03207 [cs.CL]. URL: https://arxiv.org/abs/2405.03207

Further material

- Great introductory videos to LLMs: part 1, part 2, and part 3 (there also is a short summary and a talk of this material).

6 AI and ethics

This chapter's big question

AI systems are just objective computer models, so they must be fair, right?

There already are dedicated courses on the ethics of AI at LMU, so we focus here on some specific aspect: *algorithmic fairness*. We do still provide references to general overviews on ethics of AI.

Key concepts

- Policy goals as prediction tasks, and modeling assumptions behind this (overarching goal, population choice, decision space).
- Bias in data: statistical and societal.
- Model architecture: interpretability, perturbation, choice of features.
- Model evaluation assumptions (no interference, uniform, simultaneous).
- Identifying advantaged and disadvantaged groups: intersectionality (Crenshaw).
- Fairness definitions: oblivious (purely probabilistic) vs non-oblivious (also including similarity metric between individuals and causality).
- Impossibility result: equalized odds and predictive parity are inconsistent under realistic assumptions (Kleinberg et al. 2016, Chouldechova 2017).

Cf. the machine learning pipeline.

Lecture 9 Prediction-based decision making refers to using machine learning models (or statistical models) to make decisions. For example, train a machine learning model on existing data about which customers of a bank paid back their loan in order to assist the bank's decisions for new customers based on the prediction that the machine learning model given for them. This is notoriously used also in other critical domains: pre-trial detention, child maltreatment screening, welfare eligibility, etc.

Especially since the mid-2010s, there has been a discussion—both in science and in society—about how these prediction models are *biased*. While it is difficult precisely define 'bias', it roughly means that "the model's performance (however defined) unjustifiably differs along social axes such as race, gender, and class" (S. Mitchell et al. 2021, p. 142). The field of *algorithmic fairness* developed to define, detect, and rectify bias in prediction models.

How are prediction models built? To answer that, we can recall the machine-learning pipeline (from chapter 1) and make it more specific to the prediction task:

- *Task conceptualization*: What is the social objective of deploying the model (the policy goal)? For example, maximizing profit for the bank, minimizing pre-trial detentions while guaranteeing public safety, etc. How is the objective measured? (Just in terms of arrest and appearance in court, or also more broadly in terms of well-being of the defendant?) To which subpopulation is the model going to be applied to? What qualifies to be part of the subpopulation? Is this decision free from objectionable social structures? What are the actions available to the decision maker, and which of those should the model predict on?

- *Training data*: Which features/covariatates of the inputs are used? Which features of a person are relevant to predicting creditworthiness? Which are sensitive attributes/protected characteristic? Which data is used to train the prediction model and how is the data gathered? What kind of bias can be in the data? Statistical: e.g., non-representative sampling and measurement error. Societal: objectionable social structures and past injustices represented in the data? And connections between the two (e.g., societal bias leads to over-policing people of color which leads to statistical bias in arrest data).

- *Model choice*: Which statistical/machine learning model should be used? An interpretable one or a deep neural network? How robust to changes in input is the model? (So stakeholder have an idea how perturbations/uncertainties in the input features affect the model's predictions.) How is the model evaluated? Typically with a loss function, but this comes with three important assumptions: (1) no interference: evaluation just for a single individual, not effects between individuals (denying a loan may impact a family member's ability to repay their loan). (2) uniform: all individuals are considered the same (whether the loan is for education or a vacation home does not matter). (3) simultaneous: downstream effects of the decision are not considered; cf. performativtiy (Perdomo et al. 2020).

- *Deployment*: In which contexts is the model used in practice, and is well-trained for that? Are the sensitive attributes of the model capture discrimination in reality? Importantly, much algorithmic fairness work considered just one sensitive attribute (belong to the advantaged or the disadvantages group), but in reality "discrimination might affect members at the intersection of two groups, even if neither group experiences discrimination in isolation" (S. Mitchell et al. 2021, p. 148). Famously, Crenshaw (1989) coined the term *intersectionality* for this, showing that Black women could not establish their discrimination as sex discrimination (if white women are not affected) and also not as race discrimination (if black men are not affected).

Keeping in mind these difficult choices for the prediction model pipeline, algorithmic fairness now aims to give formal and quantitative definitions of fairness of a prediction model. To do so, it formalizes the prediction task as follows:

- The set of variables V describes the input (e.g., credit history, age, occupation, gender, etc. of a person applying for a loan). It is split into sensitive variables A and insensitive variables X.
- The goal of the prediction model is to predict the true outcome Y (e.g., if the applicant will pay back the loan or not) based on the values for the variables in V.
- Typically, the prediction model will output a score S for a given list of values for the variables V; if the value exceeds a fixed threshold, the decision is $D = 1$ and otherwise $D = 0$. This yields the following possible matches or mismatches of prediction:

	$Y = 1$	$Y = 0$
$D = 1$	true positive	false positive
$D = 0$	false positive	true negative

With this, one can formulate various fairness criteria:

- *Equalized odds*: $D \perp A | Y$. In words, no matter the true outcome ($Y = 0$ or $Y = 1$), the probability for a positive decision for an individual does not depend on whether or not they are part of the advantaged group. Hence false positive and false negative rates are independent of protected characteristics.

- *Predictive parity*: $Y \perp A | D$. In words, no matter the decision ($D = 0$ or $D = 1$), whether or not the prediction was correct should not depend on whether or not the individual was part of the advantaged group. Hence the predictive value of the model (i.e., that it predicts correctly) is independent of protected characteristics.

- *Fairness through unawareness*: $D \perp A | X$. In words, individuals with the same insensitive variables are treated the same, regardless of their sensitive variables (Kusner et al. 2017).

Here $A \perp B | C$ means that the random variable A and B are conditionally independent given C, i.e., $P(A|B, C) = P(A|C)$.

However, *impossibility* results like those of Kleinberg et al. 2016 or of Chouldechova 2017 show that, under realistic assumptions, equalized odds and predictive parity are inconsistent. (For attempts at reconciliation, see Pleiss et al. 2017.)

The above fairness definitions are *oblivious* in the sense that they only refer to statistical relationships. Non-oblivious definitions also include, e.g., similarity metrics between individuals or causal relationships; for the latter, with a counterfactual fairness notions using causal models, see Beigang 2023 (further reading below).

⇝ It is (by now) fairly well known that machine learning systems can be (and usually are) biased: they learn from data and typically the data is biased. But where does the bias in the data come from? Discuss, for example, the distinction between statistical and societal bias. Are there further sources of bias in the model? What about inductive bias?

⇝ As an exercise in mathematical philosophy, think about mismatches between the intuitive notion of fairness/bias and its formal explication in the above definitions.

Readings

- An overview of the field of algorithmic fairness.

  S. Mitchell et al. (2021). "Algorithmic Fairness: Choices, Assumptions, and Definitions." In: *Annual Review of Statistics and Its Application* 8.1, pp. 141–163. DOI: 10.1146/annurev-statistics-042720-125902. eprint: https://doi.org/10.1146/annurev-statistics-042720-125902. URL: https://doi.org/10.1146/annurev-statistics-042720-125902.

Further material

- An encyclopedia overview on Ethics of AI:

  J.-S. Gordon and S. Nyholm (n.d.). "Ethics of Artificial Intelligence." In: *The Internet Encyclopedia of Philosophy*. Available at: https://iep.utm.edu/ethic-ai/ (accessed: 6 Mar 2022).

- A famous paper on the ethics of Large Language Models.

  E. M. Bender, T. Gebru, et al. (2021). "On the Dangers of Stochastic Parrots: Can Language Models Be Too Big?" In: *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, pp. 610–623.

- A discussion of the interplay between formalized technical discourse of fairness and informal ethical discourse of fairness.

  P. Schwöbel and P. Remmers (2022). "The Long Arc of Fairness: Formalisations and Ethical Discourse." In: *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*. FAccT '22. Seoul, Republic of Korea: Association for Computing Machinery, pp. 2179–2188. DOI: 10.1145/3531146.3534635. URL: https://doi.org/10.1145/3531146.3534635.

- A recent impossibility result showing that three plausible requirements for a predictive algorithm to be fair (equalized odds, predictive parity, and counterfactual fairness) are in fact jointly inconsistent. It can be seen as a continuation of the quite short section 5 of S. Mitchell et al. (2021) discussing causal reasoning in fairness definitions.

  F. Beigang (2023). "Yet Another Impossibility Theorem in Algorithmic Fairness." In: *Minds and Machines*. DOI: https://doi.org/10.1007/s11023-023-09645-x

- A philosophical discussion of the role of statistical algorithmic fairness criteria:

W. Fleisher (forthcoming). “Algorithmic Fairness Criteria as Evidence.” In: *Ergo*. URL: https://philarchive.org/rec/FLEAFC

7 Theory of AI: Power and Limits

This chapter's big question

Is there anything AI models cannot do?

Key concepts

- Turing machine and Church–Turing thesis
- Halting problem, Entscheidungsproblem, tiling problems
- Gödel's incompleteness theorems
- Computational complexity theory: P vs NP
- Statistical learning theory
- PAC learnability
- No-Free-Lunch theorem
- Bias-complexity tradeoff
- Universal approximation theorems

We split this section into two lectures. The first one is on the theory of symbolic AI. Here we focus on computability theory, which is the most central part of the theory of symbolic AI. The second lecture is on the theory of non-symbolic. Here we focus on statistical learning theory, which is, if any, the most established theory of non-symbolic AI—though we also point out where it is still lacking. The theory of symbolic AI, and especially computability theory, is very well developed, so it can serve as a guiding example for what a full theory of non-symbolic AI—which is still missing—should deliver.

Lecture 10 In a remarkable feat of mathematical philosophy, Alan Turing explicated in 1936 the intuitive notion of (symbolic) computation with an abstract machine, now known as the *Turing machine*. It processes symbols, that are written on a tape, according to its program, which is simple a table of instructions. At any given time step, the machine is in some internal state and positioned at some cell of the tape. Based on its internal state and the symbol written in that cell, the program says (a) in which internal state the machine should go into next, (b) which symbol the machine should write into the present cell instead, and (c) whether it should move to the left or the right cell next. The machine has one designated internal state known as the halting state: if it enters that, it stops its computation. The Turing machine then computes the following function: It takes as input a string of symbols (consisting only of symbols that also the machine uses). They are written on the otherwise blank tape. Then the machine start processing according to its program. If it enters its halting state, then the string of symbols that is written on the tape at that moment is the output of the function. If the machine never enters a halting state and 'loops forever', then no output is ever given.

A function that maps strings to strings over a given finite set of symbols is *Turing-computable*, if there is a Turing machine that computes that function. The expected functions indeed turn out to be computable. For example, the addition function that maps the string of digits representing the number n together with the string of digits representing the number m to the string of digits representing the number $n + m$. Similarly for multiplication, exponentiation, ordering, etc.

Many other models of computation have been developed (e.g., Church's lambda calculus or Gödel's recursive functions). Remarkably, though, they all turned out to be equivalent: a function is computable in one model if and only if it is computable in the other model. This lead to the conviction that the formal concept of Turing computability really coincides with the intuitive concept of (symbolic) computabilility. This is known as the *Church–Turing thesis*: that a function is Turing computable if and only if it is computable in the intuitive sense. Note that this is a *philosophical* and not a mathematical claim. It is a claim about an informal philosophical concept: namely, that of computability. It cannot be proven by formal mathematical means.

But Turing was asking: are there functions that are not computable?[1] In other words, are there things that no computer can ever solve? He considered the so-called *halting problem*: Given a Turing machine e and an input n, decide if the Turing machine on this input halts or not. A

[1] Turing machines are finite objects, there are only countably many Turing machines, so there are also only countably many computable functions. On the other hand, there are uncountably many function $\mathbb{N} \to \mathbb{N}$ (by Cantor's diagonalization theorem). So there must be *some* uncomputable functions. But explicitly providing one is more difficult.

Turing machine that solves this problem is a 'meta-program' that checks whether an inputted programs ever gets stuck in a loop. So it would be very desirable to have such a meta-program, but Turing showed that this cannot exist.

Theorem 7.1 (Turing)**.** *There is no Turing machine that solves the halting problem. In other words, the* $\mathtt{h}$ *function that maps a (code of a) Turing machine* $\mathtt{e}$ *and an input* $\mathtt{n}$ *to*

$$\mathtt{h}(\mathtt{e},\mathtt{n}) := \begin{cases} 1 & \textit{Turing machine } \mathtt{e} \textit{ halts on input } \mathtt{n} \\ 0 & \textit{otherwise} \end{cases}$$

is not computable.

Proof. Toward a contradiction, assume $\mathtt{h}$ is computable. Then we can define the Turing machine $\mathtt{e}$ that, on input $\mathtt{n}$, computes $\mathtt{h}(\mathtt{n},\mathtt{n})$ and if this is 0 it outputs 0 and otherwise loops forever. Now, we plug $\mathtt{e}$ into $\mathtt{h}$. Since $\mathtt{h}$ always outputs 0 or 1, there are two cases:

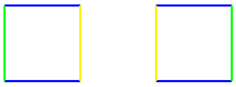
- Case 1: $\mathtt{h}(\mathtt{e},\mathtt{e}) = 0$. Then, by definition, the Turing machine $\mathtt{e}$ with input $\mathtt{e}$ outputs 0, i.e., halts, so $\mathtt{h}(\mathtt{e},\mathtt{e}) = 1$.
- Case 2: $\mathtt{h}(\mathtt{e},\mathtt{e}) = 1$. Then, by definition, the Turing machine $\mathtt{e}$ with input $\mathtt{e}$ runs forever, i.e., does not halt, so $\mathtt{h}(\mathtt{e},\mathtt{e}) = 0$.

Thus, we indeed get a contradiction. □

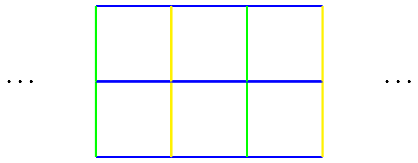
Knowing the non-computability of the halting problem is a stepping stone for other non-computability results. A common method to prove a given problem to be non-computable is a *reduction argument*: assume there is a Turing machine solving the given problem and the use this to build a Turing machine that solves the halting problem. Since the latter is impossible, the former is, too.

This is how the historically important *Enscheidungsproblem* was answered in the negative: there is no algorithm to decide whether a given formula in first-order logic is logically valid (i.e., true in all interpretations). If there was such an algorithm, we could solve the halting problem: given Turing machine $\mathtt{e}$ and input $\mathtt{n}$, there is a clever way to build a formula φ of first-order logic such that φ is valid iff $\mathtt{e}$ halts on input $\mathtt{n}$, so if we can decide validity we can also decide halting.

Another nicely visual problem that can be shown to be non-computable is the *tiling problem*. As an input one gets finitely many square tiles (all of the same size) and each of their edges has some color. For example:



Then one should decide if it is possible to tile the plane with them, given infinitely many copies of each tile. So one has to put them together so they match colors at adjacent edges. For example, with the above tiles, once can do it:



But with other sets of tiles, one cannot tile the plane. Again one can show with a reduction argument that there is no Turing machine that, given a set of tiles as input, output 'yes' or 'no' according to whether or not the set of tiles tiles the plane.

In fact, one of the most famous theorems of 20th century mathematics can also be proven with a reduction argument: Gödel's first incompleteness theorem. (Gödel's original proof was different: Turing machines did not exist yet at that time.)

The formulation that we state here uses the concept of a computable enumeration. A set of string is *computably enumerable* if there is a Turing machine that, as it keeps processing, writes on the tape strings separated by a blank cell such that all written strings are in the set and each string of the set eventually is written on the tape. This is weaker than the set of string being *decidable*: for that we require a Turing machine that take a string as input and outputs 'yes' or 'no' depending on whether or not the string is in the set. In the computably enumerable case, we only get confirmation if the string is in the set. If the string is not in the set, then, as we follow the enumeration, we never know if the string actually is in the set but has not yet appeared or actually is not in the set and because of that did not appear. On the other hand, in the decidable case, we also get confirmation if the string is not in the set.

Theorem 7.2 (Gödel's first incompleteness theorem)**.** *There is no computably enumerable list of axioms that prove all and only the true statements of elementary mathematics (including, say, basic arithmetic, finite combinatorics, etc.).*

This version is discussed, e.g., in this online lecture.

Just arithmetic is enough, because other 'elementary mathematics' can be encoded with natural numbers, so questions there reduce to questions about arithmetic.

Proof. For a reduction argument, assume there is such a list of axioms. By systematically applying all the deduction rules, we can computably enumerate all and only the theorems of the theory (not just the axioms).

But then we can solve the halting problem: Given a Turing machine e and an input n, consider the statement 'Turing machine e halts on input n'. This is a statement of elementary mathematics. So either this statement or its negation shows up on the enumeration of all theorems (one of the two has to). So we start the enumeration and wait until one of the two shows up, and depending on which, we say the Turing machine halts or does not halt. This hence would be a computable procedure to solve the halting problem, which is impossible. □

This theorem is so important because it shattered Hilbert's dream of writing down, in a systematic—and hence computably enumerable—way all the axioms of mathematics, so that every mathematical truth can be immutably established by deriving it from the axioms. But it also played a big role in the philosophy of AI in the 1980s and 90s, when Roger Penrose revived an argument of Lucas with the conclusion that no machine can ever reach human-level intelligence. The idea of the argument is that any such machine would have to do mathematical reasoning and, qua machine, it would do it in a computable way, but, by Gödel's incompleteness result, there would be statements (e.g., about the halting behavior of some Turing machine) that we, with our human intelligence, can see to be true, but the machine cannot establish. This argument had a controversial discussion: for a short overview, see here.

So far, we have talked about the distinction between computable functions (or problems) and non-computable ones. However, even if one knows that that the problem one attempts to solve is computable, it is still important to know just how *complex* it is, i.e., how long (or more generally, how many resources) the best possible Turing machine would take to solve it. This is studied by *complexity theory*. It has a whole zoo of classes of problems that have a similar complexity. But the two most famous classes are:

- The class P of polynomial-time decision problems: the time to compute 'yes' or 'no' for a given input n takes at most $p(n)$ time steps, for some polynomial p like, e.g., n^2 (rather than exponentially long, i.e., 2^n).
- The class NP of problems where it can at least be verified in polynomial time that the provided answer for a given input is indeed correct.

Intuitively one might expect that $\mathsf{P} \neq \mathsf{NP}$, i.e., that there are problems where one cannot quickly find the correct answer, but for a suggested answer one can at least quickly verify that it is correct (think, e.g., of solving a puzzle). And it is indeed widely believed that $P \neq NP$—a lot of modern cryptography is even built on this belief. However, despite the simplicity of this question, it is one Millenium Prize Problems whether $P = NP$ or not.

Lecture 11 Statistical learning theory was developed as the theory of machine learning. If any, it is the theory of non-symbolic AI, even though, as we discuss at the end of the lecture, there is some disconnect between theory and practice. It aims to provide a framework that is as general as possible to ask and answer questions about what is and what is not learnable.

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.

So statistical learning theory focuses on supervised learning! That's not too much of a restriction, though, since this covers most of modern AI, and some prima facie unsupervised tasks can also be recast as supervised tasks (Berner et al. 2022, remark 1.3).

- 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.

Note that no computability constraints are put on the function A, so it is a bit of a stretch to call it an 'algorithm'. But there also is computable statistical learning theory; and, in practice, the A is computed.

- *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 $X \times Y$.)

For a subset A of X, we write $D(A)$ for the probability that a randomly sampled x is in the set A.

- *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:

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

If A is a set, $|A|$ is the number of elements in that set (aka its cardinality).

- *Empirical Risk Minimization* (ERM) is the learning paradigm of coming up with a predictor h that minimizes the empirical risk $L_S(h)$. So an ERM learning algorithm takes a training set S and outputs a predictor $A(S)$ that minimizes the empirical risk. 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.

Again, no computability constraints are imposed: how *this empirical risk minimizer $A(S)$ is found is not specified, and in practice this is far from trivial.*

> 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/believe a priori that they will better fit the data).

PAC learning. With this general framework in place, we can ask: when does a learner succeed? In other words, 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 $\delta > 0$ and accuracy parameter $\epsilon > 0$, there is a number of samples $m = m(\delta, \epsilon)$ such that, if the learner samples m-many examples from D labeled with f, then with a high confidence of $1 - \delta$ the learner knows that they are correct up to an error of at most ϵ, provided there is a correct hypothesis in the first place. One says: the learner is *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).*

Definition 7.3. A learner A with hypothesis class H is a *PAC-learner* if there is a function $m_H : (0, 1) \times (0, 1) \to \mathbb{N}$ 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 there is $h^* \in H$ with $L_{D,f}(h^*) = 0$ (in which case one says: the realizability assumption holds), 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 returns a hypothesis h such that, with probability of at least $1 - \delta$ (over the choice of the examples), $L_{D,f}(h) \leqslant \epsilon$.

One says that a hypothesis class H is *PAC-learnable* if there is a learning algorithm A with hypothesis class H that is a PAC-learner.

This definition certainly is a mouthful! Take some time to parse it and see how it spells out the intuitive idea that the learner is probably approximately correct.

This can be generalized by dropping the realizability assumption, going beyond the binary label case (multiclass and regression, using more general loss functions). But here we stick to the basic setting.

The 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? The NFL theorem says 'no': there cannot be such a *universal learner*.

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

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

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:

Theorem 7.4 (No-Free-Lunch Theorem). *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 P 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 ERM learner A on a given dataset S as $\epsilon_{\text{app}} + \epsilon_{\text{est}}$ with

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

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 true minimal risk ϵ_{app} and the true risk of the prediction rule $A(S)$ that minimizes the empirical risk $L_P(A(S))$).

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, the *bias-complexity tradeoff*:

In practice, when we approximate the ERM prediction rule through learning, we can also measure how far off the rule found by learning is from the actual empirical risk minimizer. This is known as the optimization error.

- 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.
- 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.

This is illustrated in the blue U-curve of figure 7.1. If we start with a learner that has a very simple hypothesis class, it will have both high true risk and empirical risk, because it does not have a prediction rule that is close to the real world. If we now increase the complexity of the hypothesis class by adding more prediction rules, the learner eventually fits the data well and also generalizes well outside the training set. But if we add too many prediction rules, the learner will overfit: picking prediction rules that have perfect empirical risk but are too idiosyncratic and hence fail to generalize. (We later get to the red part, the 'double descent', that is observed for neural networks.)

Statistical learning theory studies how to find rich hypothesis classes H for which we still have reasonable estimation errors. With a rich enough class, the approximation error is 0 or very close to 0. As we will see next, the universal approximation theorem which—as the name suggests—says that this is the case for neural networks. 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*, and the *fundamental theorem of PAC learning* says that a hypothesis class is PAC learnable iff it has finite VC-dimension (see Shalev-Shwartz and Ben-David 2014, ch. 4).

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 $\geqslant m(\delta, \epsilon)$.

The Universal Approximation Theorem. Actually, there are several universal approximation theorems. Here we discuss an early one by Hornik et al. (1989).[2] Intuitively, it states that a neural network learner has zero

For a modern textbook version including a proof sketch, see Berner et al. (2022, thm. 1.16 on p. 16); and for further discussion, see Kratsios (2021).

[2] 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 different activation function for every neuron (rather than a single one for all neurons as it

approximation error. So the theorem considers the set $H = \Sigma^r$ 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., $+\infty$). The theorem then says that the functions in Σ can approximate any non-pathological function $f : \mathbb{R}^r \to \mathbb{R}$ arbitrarily well. So whatever the true function f of the world is, and whatever accuracy should be achieved, the neural network learner has a prediction rule $h \in H$—which is given by some choice of hidden layer size and weights—such that h is like f up to the desired accuracy. So the approximation error is minimized. This is made precise as follows.

Other neural network learners (i.e., other architectures) are considered in generalizations of this theorem. But here we focus on the simple case.

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):

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.

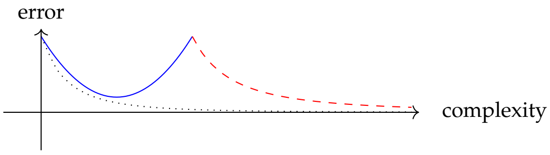


Figure 7.1: The empirical error (dotted) and the true error (solid and dashed) against the model complexity. In blue-solid: The classical bias-complexity tradeoff of statistical learning theory. In red-dashed: The extension to the double-descent curve of Belkin (2021).

Theorem 7.5 (Universal approximation theorem)**.** *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.*

It is important to note that the theorem only guarantees the existence of a hidden layer size and weights with which the neural network approximates the given true function to the desired accuracy. It does *not* tell us how to find these parameters. In practice, we choose a hidden layer size and then train the neural network on the data set S using backpropagation. We hope that, this way, we find weights with which the neural network computes a function that is close to the true function, also outside the training samples. We can then test whether we have found good ones on the held-out test set: which then estimates the true error. However, sometimes this also goes wrong and, e.g., the neural networks overfits, which we can see if it had good accuracy on the training set but bad accuracy on the test set. In fact, a result of Colbrook et al. (2022) shows that there are natural machine learning tasks where, by the universal approximation theorem, neural networks exist that solve the task (i.e., approximate the true function very closely) but backpropagation—and in fact no learning algorithm—will ever find weights with which the neural network will solve the task!

What's missing for a theory of modern deep learning? There are various phenomena in modern deep learning that cannot be explained with the classical statistical learning theory (for an overview, see Berner et al. (2022)). Maybe the most pressing one is the *generalization problem*: Why is it that neural networks generalize well? The bias-variance tradeoff would say that, because neural networks are so over-parametrized (i.e., have many more parameters than training samples), they are on the 'too complex' side and hence overfit: that they minimize the empirical error but not the true error because they match the data too closely at the expense of finding a good general pattern. In practice, however, these over-parametrized neural networks are actually found to generalize well, i.e., also minimize the true error. Belkin (2021) calls this the double-descent curve: further increasing the complexity of the learner's model class beyond the point where the classical bias-complexity tradeoff stops actually brings down again the true error. Explaining why this is the case is one of the big open problems in the theory of machine learning.

We explore this and other questions in the course 'Advanced Topics in the Foundations of AI'.

Readings

- For lecture 10: An overview of computability theory: Immerman (2021).
- For lecture 11: Shalev-Shwartz and Ben-David (2014), chapters 2–3 (set up of statistical learning theory) and chapter 5 (No-Free-Lunch Theorem).

Further material

- A textbook overview of (the theory of) symbolic AI: Flasiński (2016, ch. 2).
- A video lecture on computability and on Gödel incompleteness.
- For the use of computability theory—and especially complexity theory—in the study of human intelligence (aka cognitive science :-)), see Van Rooij (2008).
- For a discussion of interpretations of the No-Free-Lunch Theorem: Sterkenburg and Grünwald (2021).
- The result about task-solving neural networks that exist but cannot be found: Colbrook et al. (2022).

8 Explainable AI

This chapter's big question

What can we do about the black box nature of neural networks?

Key concepts

- Reasons for interpretability: trust, causality, transferability, fairness, privacy, robustness.
- Interpretability as transparency/by design vs interpretatability as post-hoc explanation.
- Explainable AI
- Robustness in AI

It is a commonplace that neural networks are black boxes: they work but we don't understand what they are doing. However, it is actually not a trivial matter to make precise exactly what we do not understand about them. This is because, in a sense, we understand them perfectly: They are well-defined mathematical functions and their parameters are updated according to a well-defined algorithm (namely backpropagation). In fact, we understand them so well that we can even implement them on computers! So there is a different sense at play when we say that we don't really understand them.

A good way to describe these different senses is in terms of levels: We perfectly understand the micro-level behavior of the neural networks (its activation values, weight setting, etc.). However, what we are missing is a macro-level understanding of this behavior: a description of the micro-level process in human-understandable terms. This is sometimes compared with thermodynamics: if we have, say, a box of gas, we have a micro-level description of how the individual gas particles move around as dictated by the laws of mechanics, but we also have a macro-level description in terms of temperature and pressure, which are meaningful to us.

The field of *Explainable AI* (XAI) aims to provide various forms of such macro-level descriptions of neural networks. The field goes by various names: interpretable AI, reliable AI, robust AI, trustworthy AI—and different people might means slightly different things by these names. Here we understand 'explainable AI' broadly to include all these aspects.

We focus on two such aspects in two lectures. The first lecture is on interpretability: what exactly do we mean when we say that we want to interpret the behavior of the network in understandable terms? The second lecture is on explainability: what makes a good explanation of the behavior of the network?

Lecture 12 The goal of interpretability is to explain or present the behavior of the AI-model in human-understandable terms. But why, we should first ask, is this even needed? After all, for some systems, say, a thermostat, we don't require interpretability if we know that it does exactly what we want it to do.

Doshi-Velez and Kim (2017, p. 3) argue that the need for interpretability comes from an incompleteness in the task conceptualization. For example, the task may be to predict loan approval in a fair way. However, we don't have a precise mathematical formulation of 'fair' that we could use to train and evaluate the AI system. So we could train the AI system based on accuracy with respect to historic data about loan approval. But then, after training, we still need to 'look into the inner workings' of the AI system to check that it reaches its predictions in a fair way—so we need some interpretability. Such reasons for requiring interpretability are the following:

The first three items are from Lipton (2018), the others are added by Doshi-Velez and Kim (2017).

1. *Trust*: confidence of human users that model will perform well.
2. *Causality*: infer true causal relation in the world from the correlations detected by the model; interventions/perturbations that change the model's prediction also correspond to real changes in the world.
3. *Transferability*: to new domains of application, still knowing that the model will work.
4. *Fairness*: that protected groups are not discriminated against.
5. *Privacy*: that the model does not reveal sensitive information in the training data.
6. *Robustness/reliability*: the model keeps performing well also despite perturbations of the inputs or parameters.

There are two main approaches toward achieving interpretability: interpretability by design (aka ante-hoc or intrinsically interpretable) vs post-hoc interpretability. According to the former, we build AI system using only interpretable components (e.g., only linear classifiers). According to the latter, we allow non-interpretable but well-performing AI systems (like neural networks), but after having built them, we also provide a way to interpret they behavior. Lipton (2018) caches this out as follows:

- Interpretability as *transparency*: *simulatability* (a human can simulate the model at once), *decomposability* (each part of the model admits intuitive interpretation; cf. elementwise/localist representation), *algorithmic transparency* (the learning algorithm with which the model is built is understood well: provably converges to best solution, etc.). Humans aren't transparent in any of these senses. Linear models are in general only interpretable in the last sense, and the input features that they used might be processed while they are readily interpretable for neural networks.

- Interpretability as *post-hoc interpretation*: Provide additional information to elucidate why the model provided a certain output, without necessarily being faithful to the underlying mechanism producing the output (hence this can be misleading). If humans are interpretable, it is in this sense. Examples: *text explanation* (provide a verbal explanation of model output; no guarantee of correctness), *visualization* (e.g., visualize high-dimensional representations in 2D images), *local explanation* (e.g., saliency maps showing which parts of the input were most important to the output in the sense that changing them will most likely change the output; can be misleading), *explanation by example* (e.g., which datapoints are most similar/important for the model behavior).

Since the publication of the paper, many more local explanation methods are known: counterfactual explanation, SHAP, integrated gradients, etc. But they also face certain impossibilities: see Bilodeau et al. (2024).

Once an interpretability method is provided, an important issue is to evaluate it. Doshi-Velez and Kim (2017, p. 3) provide various empirical approaches to do that (application grounded, human-grounded, functionally-grounded). One notion that is important here is that the interpretability method should be *faithful*. It is difficult to make this precise, but intuitively this means that the model, at the micro-level, is actually doing what, at the macro-level, the interpretability method says it is doing. We encountered this idea already in the discussion of mechanistic interpretability in chapter 5.

⇝ Consider the two different approach to interpretable AI models:

interpretable by design vs post-hoc interpretability. Discuss pros and cons.

⇝ How would you make precise the faithfulness requirement for interpretability methods? One approach that we have seen in chapter 5 is in terms of causal abstraction, i.e., that interventions on the micro-level match the corresponding ones on the macro-level.

Lecture 13 Explainability.

Readings

- For lecture 12: F. Doshi-Velez and B. Kim (2017). *Towards A Rigorous Science of Interpretable Machine Learning*. arXiv: 1702.08608 [stat.ML].
- For lecture 13: T. Miller (2019). "Explanation in artificial intelligence: Insights from the social sciences." In: *Artificial Intelligence* 267. DOI: https://doi.org/10.1016/j.artint.2018.07.007.

Further material

- More on interpretable AI: Lipton (2018) and Rudin (2019).
- A hands-on introduction to various interpreability techniques: Molnar (2022)
- More on explainable AI: Woodward and Ross (2021) and Mittelstadt et al. (2019).
- Robustness in AI: Freiesleben and Grote (2023)

Bibliography

Beigang, F. (2023). “Yet Another Impossibility Theorem in Algorithmic Fairness.” In: *Minds and Machines*. DOI: https://doi.org/10.1007/s11023-023-09645-x (cit. on p. 30).

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. 39, 43 sq.).

Bender, E. M., T. Gebru, et al. (2021). “On the Dangers of Stochastic Parrots: Can Language Models Be Too Big?” In: *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, pp. 610–623 (cit. on pp. 8, 19, 30).

Bender, E. M. and A. Koller (2020). “Climbing towards NLU: On Meaning, Form, and Understanding in the Age of Data.” In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 5185–5198 (cit. on pp. 8, 11).

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. 37, 41, 43).

Bilodeau, B. et al. (2024). “Impossibility theorems for feature attribution.” In: *Proceedings of the National Academy of Sciences* 121.2, e2304406120 (cit. on p. 47).

Boden, M. A., ed. (1990). *The Philosophy of Artificial Intelligence*. Oxford Readings in Philosophy. New York: Oxford University Press (cit. on p. 2).

— (2016). *AI: Its nature and future*. Oxford: Oxford University Press (cit. on p. 5).

Boge, F. J. (2021). "Two Dimensions of Opacity and the Deep Learning Predicament." In: *Minds and Machines* 32.1, pp. 43–75. DOI: https://doi.org/10.1007/s11023-021-09569-4 (cit. on p. 17).

Bringsjord, S. and N. S. Govindarajulu (2022). "Artificial Intelligence." In: *The Stanford Encyclopedia of Philosophy*. Ed. by E. N. Zalta and U. Nodelman. Fall 2022. Metaphysics Research Lab, Stanford University (cit. on p. 6).

Buckner, C. (2019). "Deep learning: A philosophical introduction." In: *Philosophy Compass* 14.10, e12625. DOI: https://doi.org/10.1111/phc3.12625 (cit. on p. 6).

Buckner, C. J. (2023). *From Deep Learning to Rational Machines: What the History of Philosophy Can Teach Us about the Future of Artificial Intelligence*. New York: Oxford University Press. DOI: https://doi.org/10.1093/oso/9780197653302.001.0001 (cit. on p. 13).

Chomsky, N., I. Roberts, and J. Watumull (2023). "Noam chomsky: The false promise of chatgpt." In: *The New York Times* 8. URL: https://www.nytimes.com/2023/03/08/opinion/noam-chomsky-chatgpt-ai.html (cit. on p. 17).

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 pp. 43 sq.).

Copeland, J. (1993). *Artificial Intelligence: A Philosophical Introduction*. Wiley-Blackwell (cit. on p. 2).

Doshi-Velez, F. and B. Kim (2017). *Towards A Rigorous Science of Interpretable Machine Learning*. arXiv: 1702.08608 [stat.ML] (cit. on pp. 46 sqq.).

Flasiński, M. (2016). *Introduction to Artificial Intelligence*. Cham: Springer. DOI: https://doi.org/10.1007/978-3-319-40022-8 (cit. on p. 44).

Fleisher, W. (forthcoming). "Algorithmic Fairness Criteria as Evidence." In: *Ergo*. URL: https://philarchive.org/rec/FLEAFC (cit. on p. 31).

Freiesleben, T. and T. Grote (2023). "Beyond generalization: a theory of robustness in machine learning." In: *Synthese* 202.109. DOI: https://doi.org/10.1007/s11229-023-04334-9 (cit. on p. 48).

Freiesleben, T., G. König, et al. (2024). "Scientific Inference with Interpretable Machine Learning: Analyzing Models to Learn About Real-World Phenomena." In: *Minds and Machines* 34.3. DOI: https://doi.org/10.1007/s11023-024-09691-z (cit. on p. 18).

Freiesleben, T. and C. Molnar (2024). *Supervised Machine Learning for Science. How to stop worrying and love your black box*. URL: https://ml-science-book.com/ (cit. on p. 18).

Garcez, A. d. and L. C. Lamb (2023). "Neurosymbolic AI: the 3rd wave." In: *Artificial Intelligence Review*. DOI: https://doi.org/10.1007/s10462-023-10448-w (cit. on p. 14).

Geiger, A. et al. (2021). "Causal abstractions of neural networks." In: *Advances in Neural Information Processing Systems* 34, pp. 9574–9586. URL: https://arxiv.org/abs/2106.02997 (cit. on pp. 10 sq.).

Glymour, C. (2015). *Thinking Things Through: An Introduction to Philosophical Issues and Achievements*. 2nd ed. Cambridge, Massachusetts: The MIT Press (cit. on p. 8).

Gordon, J.-S. and S. Nyholm (n.d.). "Ethics of Artificial Intelligence." In: *The Internet Encyclopedia of Philosophy*. Available at: https://iep.utm.edu/ethic-ai/ (accessed: 6 Mar 2022) (cit. on p. 30).

Harnad, S. (1990). "The Symbol Grounding Problem." In: *Physica D* 42, pp. 335–346 (cit. on p. 8).

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. 41 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. 44).

Korb, K. B. (2004). "Introduction: Machine Learning as Philosophy of Science." In: *Minds and Machines* 14, pp. 433–440. DOI: https://doi.org/10.1023/B:MIND.0000045986.90956.7f (cit. on pp. 16, 18).

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 p. 41).

Lipton, Z. C. (09/2018). "The Mythos of Model Interpretability." In: *Commun. ACM* 61.10, pp. 36–43. DOI: 10.1145/3233231. URL: https://doi.org/10.1145/3233231 (cit. on pp. 46 sqq.).

Liu, Z. et al. (2024). *KAN: Kolmogorov-Arnold Networks*. arXiv: 2404.19756 [cs.LG] (cit. on p. 42).

Miller, T. (2019). "Explanation in artificial intelligence: Insights from the social sciences." In: *Artificial Intelligence* 267. DOI: https://doi.org/10.1016/j.artint.2018.07.007 (cit. on p. 48).

Millière, R. and C. Buckner (2024a). "A Philosophical Introduction to Language Models – Part I: Continuity With Classic Debates." In: arXiv: 2401.03910 [cs.CL]. URL: https://arxiv.org/abs/2401.03910 (cit. on p. 24).

— (2024b). "A Philosophical Introduction to Language Models – Part II: The Way Forward." In: arXiv: 2405.03207 [cs.CL]. URL: https://arxiv.org/abs/2405.03207 (cit. on pp. 23, 25).

Mitchell, M. et al. (2019). "Artificial intelligence: A guide for thinking humans." In: (cit. on pp. 3, 6).

Mitchell, S. et al. (2021). "Algorithmic Fairness: Choices, Assumptions, and Definitions." In: *Annual Review of Statistics and Its Application* 8.1, pp. 141–163. DOI: 10.1146/annurev-statistics-042720-125902. eprint: https://doi.org/10.1146/annurev-statistics-042720-125902. URL: https://doi.org/10.1146/annurev-statistics-042720-125902 (cit. on pp. 27–30).

Mittelstadt, B., C. Russell, and S. Wachter (2019). "Explaining Explanations in AI." In: *Proceedings of the Conference on Fairness, Accountability, and Transparency*. FAT* '19. Atlanta, GA, USA: Association for Computing

Machinery, pp. 279–288. DOI: 10.1145/3287560.3287574. URL: https://doi.org/10.1145/3287560.3287574 (cit. on p. 48).

Molnar, C. (2022). *Interpretable Machine Learning. A Guide for Making Black Box Models Explainable*. 2nd ed. URL: https://christophm.github.io/interpretable-ml-book (cit. on p. 48).

Newell, A. and H. A. Simon (1976). "Computer Science as Empirical Inquiry: Symbols and Search." In: *Communications of the ACM* 19.3, pp. 113–126 (cit. on p. 9).

Oppy, G. and D. Dowe (2021). "The Turing Test." In: *The Stanford Encyclopedia of Philosophy*. Ed. by E. N. Zalta. Winter 2021. Metaphysics Research Lab, Stanford University (cit. on pp. 8, 11).

Perdomo, J. et al. (2020). "Performative Prediction." In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by H. D. III and A. Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, pp. 7599–7609. URL: https://proceedings.mlr.press/v119/perdomo20a.html (cit. on p. 28).

Piccinini, G. and C. Maley (2021). "Computation in Physical Systems." In: *The Stanford Encyclopedia of Philosophy*. Ed. by E. N. Zalta. Summer 2021. Metaphysics Research Lab, Stanford University (cit. on p. 10).

Quine, W. V. O. (1951). "Two Dogmas of Empiricism." In: *Philosophical Review* 60.1, pp. 20–43 (cit. on p. 12).

Rudin, C. (2019). "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead." In: *Nature Machine Intelligence* 1, pp. 206–215. DOI: https://doi.org/10.1038/s42256-019-0048-x (cit. on p. 48).

Russell, S. J. and P. Norvig (2021). *Artificial Intelligence: A Modern Approach*. Pearson series in artificial intelligence. Harlow: Pearson (cit. on p. 6).

Schwöbel, P. and P. Remmers (2022). "The Long Arc of Fairness: Formalisations and Ethical Discourse." In: *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*. FAccT '22. Seoul, Republic of Korea: Association for Computing Machinery, pp. 2179–2188. DOI: 10.1145/3531146.3534635. URL: https://doi.org/10.1145/3531146.3534635 (cit. on p. 30).

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 pp. 41, 44).

Smolensky, P. (1988). "On the proper treatment of connectionism." In: *Behavioral and brain sciences* 11.1, pp. 1–74. DOI: https://doi.org/10.1017/S0140525X00052791 (cit. on pp. 8–11).

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. 39, 44).

Sullivan, E. (2022). "Understanding from Machine Learning Models." In: *The British Journal for the Philosophy of Science* 73.1, pp. 109–133. DOI: 10.1093/bjps/axz035 (cit. on p. 17).

Turing, A. M. (1950). "Computing Machinery and Intelligence." In: *Mind* 59.236, pp. 433–460. DOI: https://doi.org/10.1093/mind/LIX.236.433 (cit. on p. 11).

Van Rooij, I. (2008). "The Tractable Cognition Thesis." In: *Cognitive Science* 32.6, pp. 939–984. DOI: 10.1080/03640210801897856. URL: https://onlinelibrary.wiley.com/doi/abs/10.1080/03640210801897856 (cit. on p. 44).

Van Gelder, T. (1998). "The dynamical hypothesis in cognitive science." In: *Behavioral and Brain Sciences* 21.5, pp. 615–628. DOI: 10.1017/S0140525X98001733 (cit. on p. 11).

Williamson, J. (2004). "A Dynamic Interaction Between Machine Learning and the Philosophy of Science." In: *Minds and Machines* 14, pp. 539–549. DOI: https://doi.org/10.1023/B:MIND.0000045990.57744.2b (cit. on pp. 16, 18).

Woodward, J. and L. Ross (2021). "Scientific Explanation." In: *The Stanford Encyclopedia of Philosophy*. Ed. by E. N. Zalta. Summer 2021. Metaphysics Research Lab, Stanford University (cit. on p. 48).