Contents

— 1 —

Introduction

From its roots, machine learning embraces the anything goes principle of scientific discovery. Machine learning benchmarks become the iron rule to tame the anything goes. But after decades of service, a crisis grips the benchmarking enterprise.

"The only principle that does not inhibit progress is: anything goes," the philosopher Paul Feyerabend proclaimed half a century ago. Taking on the giants of scientific method, like Popper and Kuhn, the mantra became a point of debate in philosophy circles and at dinner parties. Feyerabend advocated for flexibility and creativity in knowledge production to a degree he deemed *anarchic*. No single way of doing science is superior, he insisted.

The early days of machine learning, then called pattern recognition, certainly resonated with Feyerabend's words. Pattern recognition was a child of the cybernetics era, when science and science fiction came close. Emerging concepts from control, communication, and computation blended in ambitious research programs aimed at building intelligent machines.

Frank Rosenblatt, the inventor of perceptrons—a precursor to modern artificial neural networks—was emblematic of the time. His work on perceptrons was unconstrained by scientific orthodoxy; instead, he embraced exploration, intuitive leaps, and interdisciplinary thinking. A psychologist by training, Rosenblatt took inspiration from, among many others, the ideas of the economist Friedrich Hayek about how intelligence emerges in distributed computation. Hayek, too, opposed what he called *scientism*, the

excessive application of scientific methods and principles to domains where they may be inappropriate. It's hard not to see Rosenblatt's 600-page tome, *Principles of Neurodynamics*, as a testament to the unconstrained scientific enterprise Feyerabend demanded.

In some significant ways, today's machine learning research has stayed true to its roots. To this day, the field doesn't prescribe how to get to a result. It does not insist on rigor, method, or science in the way that researchers derive what they propose. If tomorrow a breakthrough result appeared that was somehow inspired by child development, quantum chemistry, or cell biology, no one would raise an eyebrow. Researchers are completely unconstrained in what techniques they may apply. Any model architecture, any optimizer, and any tweak is fair game.

As a result, it's been difficult to build a science of the things that people actually do in practice. The model builders always seem to outpace all attempts to theorize their doings. In a search space with infinitely many degrees of freedom, no single thing is necessarily fixed. Any particular design choice can be compensated elsewhere. Hundreds of model architectures came only to give way to a different architecture within months. What seems necessary at any point in time—different kinds of regularization or weight normalization—looks dated eventually. A flurry of papers promising to reveal "all you need" only culminated in the consensus that "all you need is all you need". This truism characterizes the craft. In some sense, machine learning is never more than whatever set of tricks you currently need.

There's an exhilarating freedom to "anything goes" that has surely attracted many to the field of artificial intelligence. But the early days already experienced the tragedy of anything goes. The initial excitement about a machine that could separate squares from triangles created the field of pattern recognition. The explosively growing field—brimming with activity—rapidly spawned myriad ways of attacking different pattern recognition problems. Research teams claimed all sorts of advances, illustrated by specific experiments and eclectic demonstrations. It became hard to tell what worked best in an ocean of possibilities. For a practically minded field, not knowing a clear answer to what works best is a torturous predicament.

One ingredient was missing to tame the "anything goes".

The iron rule

To tame the "anything goes" there had to be some kind of test. The test had better be empirical and quantitative. Aesthetics, theory, and subjective opinion ought not to play a role in the test. Since the goal of pattern recognition was to do things like classify objects in a scene, it made sense to score an algorithm by how often it succeeded in doing so. Classification accuracy was a natural target. Researchers quickly realized, however, that any model did a lot better on data points it had encountered during training than those it hadn't. So, they agreed to separate training and test cases. Soon, model builders began to compete over who could achieve the highest test accuracy. This common sense agreement among researchers was the starting point of machine learning benchmarks.

In developing benchmarks, pattern recognition discovered its own instance of the iron rule of modern science. A term coined by the philosopher Michael Strevens, the iron rule asserts that all disputes between scientists must ultimately be settled by competitive empirical testing. In this view, modern scientific communities organize around empirical protocols that lay out the rules of scientific competition. These are a lot like the rules in a sporting competition. Scientists are free to think and do whatever they want, but for the purposes of scientific competition, they stick to the rules.

The iron rule makes a virtue out of what might seem like a problem: relentless competition among scientists. By making empirical testing the objective, scientists accumulate knowledge as they compete. Scientific institutions—funding agencies, journals, and universities alike—reinforce the rule by rewarding those who come out ahead in the metrics. Deciding who gets what via empirical testing lowers friction in the gears of science, as it avoids drawn-out debate and keeps personal opinions in check. What results, Strevens argues, is an efficient knowledge machine that powers modern science.

Benchmarking is the iron rule of machine learning research and a radically simple contract at that: Anything goes on the training set, competitive ranking on the test set. The recipe is simple. What's surprisingly hard is to explain why and when it should work as an engine of progress. Strevens calls the idea that science should run on the iron rule unreasonable on the face of it. Likewise, academics have made a good case against benchmarking.

Critics argue that competitive testing promotes narrow research objectives that stifle more creative scientific activities. They say benchmarks incentivize gaming and overfitting, ultimately misleading scientists about model capabilities. Besides, benchmarks give a structural advantage to industry labs that can build the biggest models.

The critics aren't wrong. The scientific case *for* benchmarks is weak. And researchers might've indeed dismissed the idea of benchmarking as unreasonable, if it hadn't been so successful.

The ImageNet era

Benchmarks emerged from little more than common sense and intuition. They appeared in the late 1950s, had some life during the 1960s, hibernated throughout the 1970s, and sprang to popularity in the late 1980s when machine learning began to look like pattern recognition. Today, benchmarks are so ubiquitous, we take them for granted. And we expect them to do their job. After all, they have in the past.

The deep learning revolution of the 2010s was a triumph for the benchmarking enterprise. The ImageNet benchmark was at the center of all the cutting-edge advances in image classification with deep convolutional neural networks. Despite massive competitive pressure, it reliably supported model improvements for nearly a decade. Throughout its long life, a sprawling software ecosystem grew around ImageNet, making it ever simpler to develop and test models on the benchmark.

Even its tiny cousin, CIFAR-10, did surprisingly well for itself. Model builders often put CIFAR-10 into the development loop for architecture search. Once they found a promising candidate architecture, they would then scale it up to ImageNet. Folklore has it that some of the best ImageNet architectures were first developed on CIFAR-10. Even though CIFAR-10 features only 10,000 tiny pixelated test images from ten classes, such as frogs, trucks, and ships, the dataset was any model builder's Swiss Army knife for many years. The platform PapersWithCode counts more than 15,000 papers published with CIFAR-10 evaluations. This does not count the numerous evaluations that went into every single one of these papers. It also doesn't count the enormous amount of engineering work that used the dataset in one way or another.

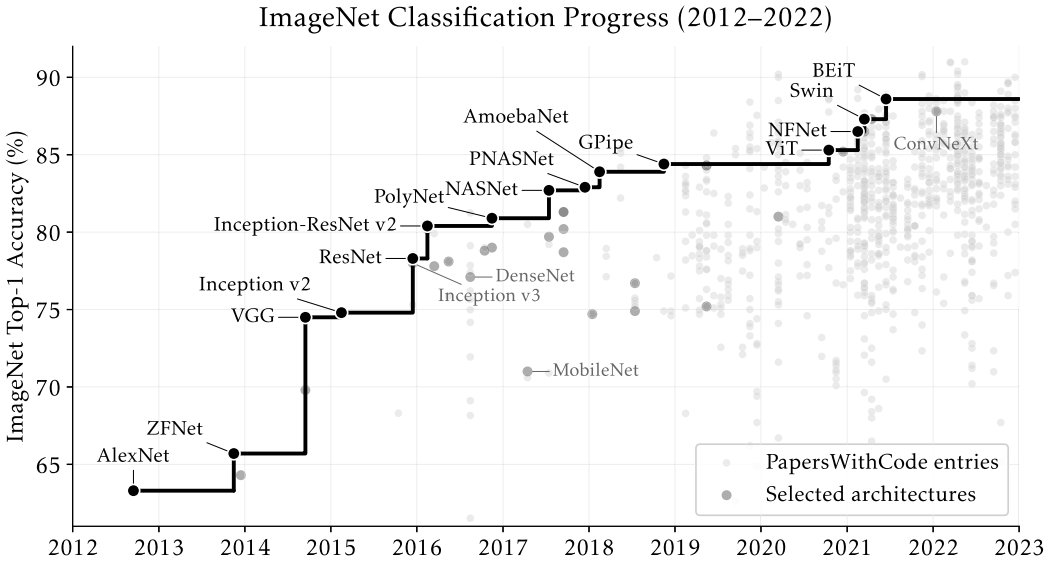

Figure 1.1: A decade of progress on the ImageNet ILSVRC 2012 benchmark.

It might seem reckless that so much work should turn on so little data. With the benefit of hindsight, though, we can even verify that the ranking of popular models on CIFAR-10 largely agrees with the ranking of the same models on ImageNet. Model rankings on ImageNet in turn transfer well to many other datasets. Researchers even created a toy dataset called ImageNot, full of noisy web crawl data, designed to stray as far as possible from ImageNet, while matching only its scale and diversity. Retraining all key ImageNet era architectures on ImageNot from scratch, model rankings turn out to be the same.

The stability of model rankings is a robust empirical fact of the ImageNet era. Numerous papers show that model accuracies change erratically from one benchmark to the next. At the same time, relative comparisons between models turn out to be surprisingly reliable. If one model beats another in one benchmark, it's likely to beat the same model in a different context, too. There's evidently a certain kind of robustness to beating the previous best—the key mechanism of the iron rule. This wasn't clear to begin with. It's just something researchers got used to.

When rankings work, benchmarks free you from second-guessing yourself: Suppose you take the highest-ranked model off the shelf and you spend a few months adapting it to your application. You find out that it doesn't

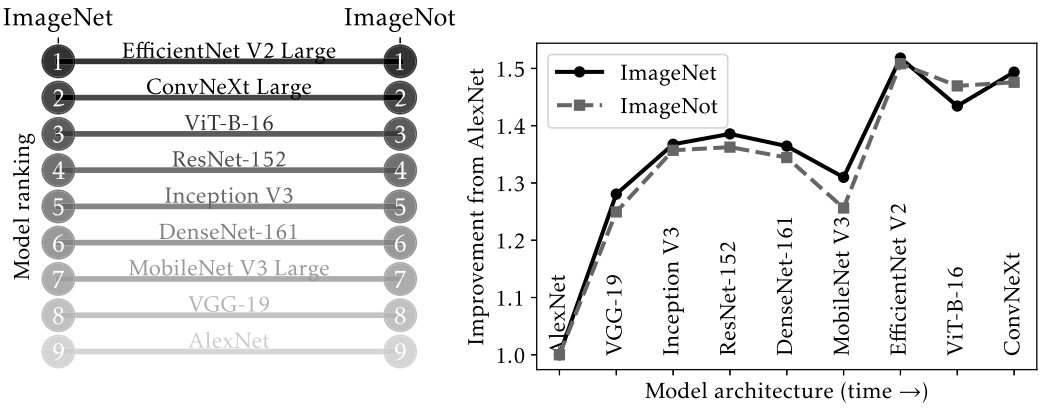

Figure 1.2: Stability of ImageNet model rankings (left) and relative accuracy improvements over AlexNet (right).

work. A good benchmark should reassure you that probably no other model would've worked either. This saves you a lot of time, since you don't have to try out the other few hundred models on the shelf. This is the powerful promise of a good ranking. The benchmark may not anticipate how well a model will work for your application. But it can tell you what's the best model to start from, thus reducing trial and error.

Faith in the top of the leaderboard was the religion of the deep learning revolution. As is often the case, though, faith is strongest just before the crisis.

The LLM era

Eventually, attention shifted from image classification to natural language processing (NLP), as the new transformer architecture scored a victory over the sluggish recurrent neural networks that had long been the workhorse for sequence problems of all kinds. Transformers were much easier to scale up and quickly took over. A simple training objective—repeatedly predict the next token in a sequence of text—avoided costly human data annotation. Companies quickly scraped up any sequence they could find on the internet, from chat messages and Reddit rants to humanity's finest writing. New scaling laws suggested empirical relationships between the dataset size, the number of model parameters, and pretraining compute that jointly minimize training loss. For a while, it seemed the only thing left to do was to sit back and watch new capabilities emerge in explosively growing models.

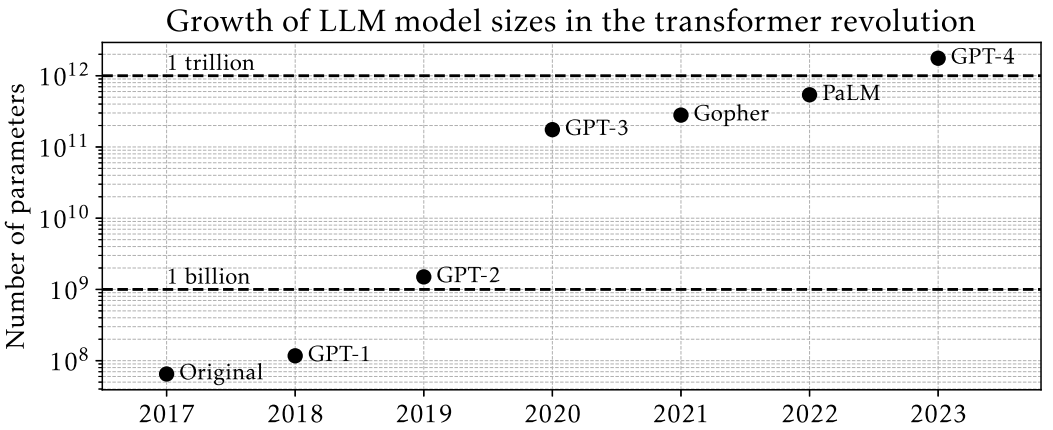

Figure 1.3: Model size growth in the transformer revolution. GPT-4 model size according to rumors.

Unlike in image classification, in NLP there has never been a single central benchmark task. Translation, parsing, entailment, and sentiment are only a few of the many tasks we associate with language understanding. Each has all sorts of different benchmarks. Most of these test tasks are a leap removed from how the model trains by predicting the next token in a never-ending stream of text. This divergence between training objective and test task spelled trouble. Right after training, large language models tend to ramble on, displaying disregard for utility, truthfulness, or manners. Hallucination is the default rather than the exception. It's part of what makes these models so intriguing: They can make up stuff. Yet, they are still reluctant to follow instructions and solve any concrete task reliably. Something was missing.

The paradigm of alignment became the catch-all cure for whatever problems models have right after training. Alignment fine-tunes pretrained models in one way or another on data such as human demonstrations, comparisons, or thumbs-up clicks from chatbot platforms. Computationally, alignment costs only a tiny fraction of the resources allocated to training. Yet, it has a transformative impact on benchmark performance and subjective human evaluations. Alignment is now part of a broader post-training pipeline that aims to make a model useful for concrete use cases. Post-training is what you do with benchmark performance in mind.

If language has no single most important task, what benchmark should we pay attention to? An exploding task plurality directly threatens the iron rule

of modern science. What exactly should scientists compete over when there are hundreds of options on the menu? If decathlon isn't exactly a crowd favorite at the Olympics, imagine watching "centathlon"—an eclectic and somewhat arbitrary collection of a hundred events with no clear winner in the end.

The growing unease with language benchmarks fueled the desire for one authoritative ranking compiling all available evaluation data into one. Researchers hoped that piling up more and more tasks would uncover a *true* ranking. This is the premise of multi-task benchmarks. But aggregation of diverse rankings is a notoriously tricky problem—the bane of all voting systems—and there are no perfect solutions. Evidence soon emerged that aggregation breaks the stability of model rankings we've come to expect from our ImageNet upbringing.

And that was only the beginning of the trouble.

As multi-task benchmarks provided no clear solution to a growing evaluation crisis, much of the competition began to gravitate toward only a few benchmarks. Among them was the MMLU benchmark, which stands for Massive Multitask Language Understanding. MMLU consists of thousands of college-level multiple-choice questions from numerous subjects. In a departure from traditional benchmarks, there is no conventional training dataset; after all, large language models are trained by predicting next tokens on some chunk of the internet.

A general knowledge question in MMLU might ask:

> As of 2016, about what percentage of adults aged 18 years or older were overweight? A: 10%, B: 20%, C: 40%, D: 80%.

A question from high school computer science could look like this:

> Let x = 1. What is x « 3 in Python 3? A: 1, B: 3, C: 8, D: 16.

Scoring high on MMLU hinges on knowledge the model acquires during training, as well as its ability to understand the prompt. The latter turns out to be surprisingly tricky. Progress on the benchmark catapulted from barely better than random guessing to over 90% in just a few years. What mesmerized observers is that accuracy gains only picked up as models reached a certain size. This sudden increase in accuracy became known as *emergent abilities*, fueling narratives about sudden unpredictable AI advances.

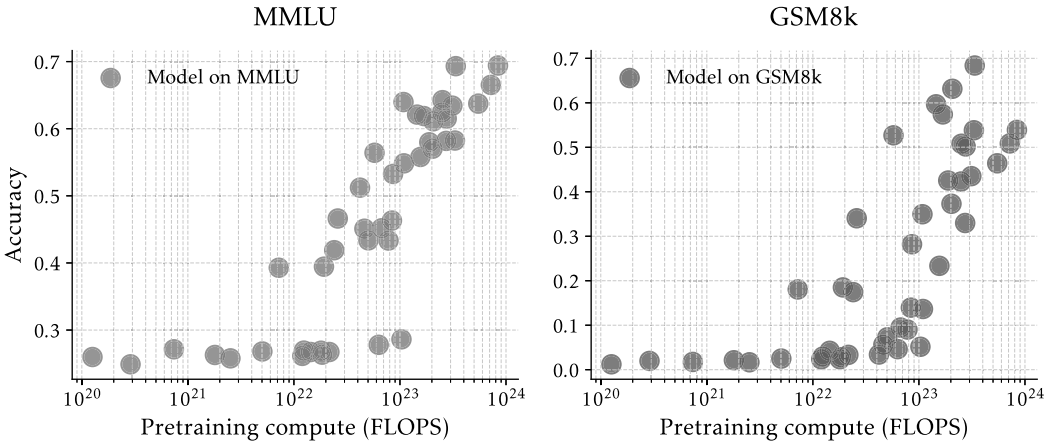

Figure 1.4: Compute versus accuracy on MMLU and GSM8k (Grade School Math 8k), a popular math benchmark.

The fact that models have reached 90% accuracy on MMLU may be an indication that progress on the benchmark has saturated and we should stop using it. However, once established, benchmarks have a long life. Researchers continued to use CIFAR-10 actively well after the 90% mark had been surpassed. Model comparisons don't necessarily need wide margins. Even small signals can indicate that one model is better than another.

It's not like people think that CIFAR-10 and MMLU are particularly faithful representations of the real-world challenges you'll encounter. Rather, these benchmarks create a point of reference for the community. They lay out the rules of the game. The right question is not how to make benchmarks more realistic. The better question is how to set up a game whose winner we should care about. CIFAR-10 turned out to be a good game to play. But there was a deeper reason why MMLU could never become CIFAR-10.

Unlike CIFAR-10, MMLU has no training data. It's just a test set like most other LLM benchmarks. Providing training data for such benchmarks might seem pointless anyway, given that models train on the internet. In the ImageNet era, researchers primarily compared models trained on the same training data. The LLM era quietly departed from this implicit agreement by making data part of the competition. Each competitor can now choose training data that maximally improve benchmark performance—a problem the AI community dubbed *benchmaxxing*. Even without explicitly training on the test set, model providers can still train on the *test task* by choosing data that best anticipates the test set.

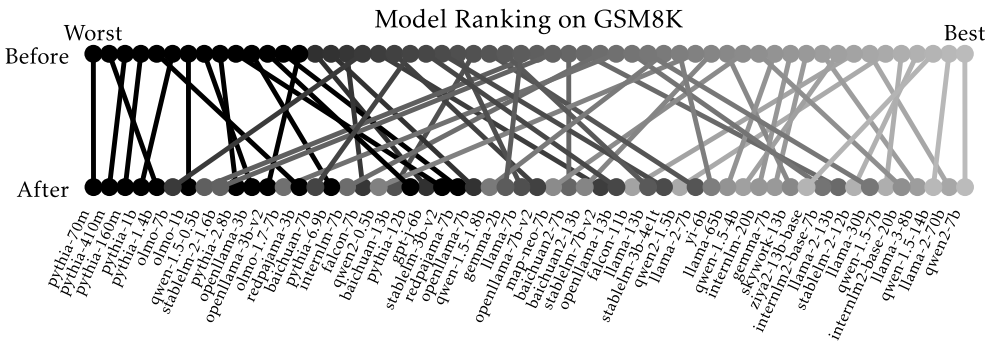

Figure 1.5: Model rankings on the GSM8k benchmark before and after giving each model the same preparation for the benchmark.

The lack of a shared training set has profound consequences for model comparisons and benchmarking more broadly. Every competitor now has their own data recipes, often as closely guarded a secret as the Coca-Cola recipe. When we compare two models on a test set, we therefore can't know how much each model specifically prepared for this benchmark. The top model on the leaderboard may have simply studied more for this specific test than others. Looking at it one way, this is fair game. Why shouldn't knowledge about the downstream test inform your training practices? From a scientific perspective, however, it breaks the no second-guessing guarantee of a good benchmark. A lower-ranking model may well have been your best choice, but it showed up less prepared for the test.

The validity of model rankings was an empirical phenomenon of the ImageNet era, but it's not clear that we should get this lucky again. Indeed, Figure 1.5 illustrates on the GSM8k grade school math benchmark what is true more broadly across different benchmarks: Giving each model the same task-specific preparation results in rankings that differ strongly from those we get when we directly evaluate the model as it is. Which model ranking we should trust is a sticky question in the LLM era.

What further troubles evaluation in the chatbot era is that models deployed at scale influence future data, a phenomenon called *performativity*. It's a century-old problem that has gained new urgency. Performativity challenges model evaluation, in particular, since there is no longer model-independent ground truth. Data—from text to labels—are now contingent on the model. Many worry that ChatGPT will drown the internet in its own text generations, leading to a vicious cycle of data and model degradation.

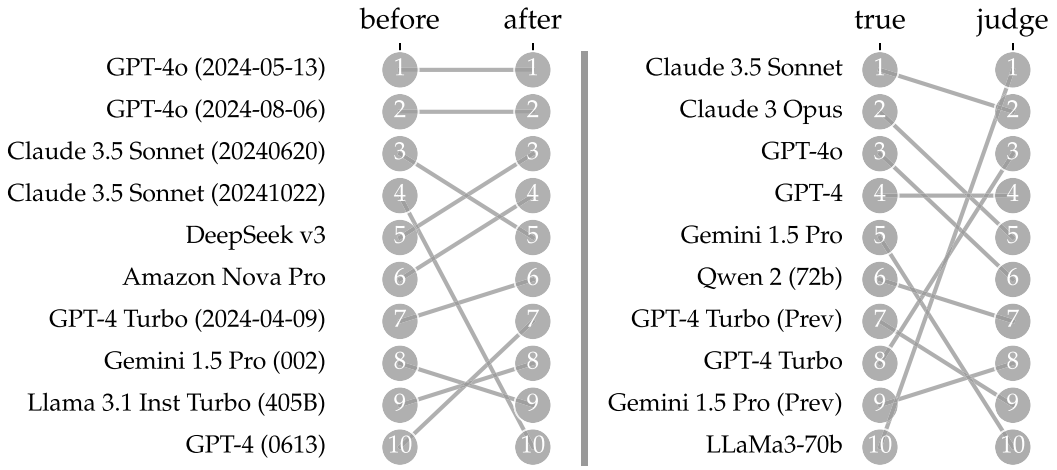

Figure 1.6: Instability of model rankings. Left: Effect of adding weak models to the HELM (Holistic Evaluation of Language Models) multi-task benchmark. Right: Effect of using an LLM (Llama 3) as a judge.

The more we use ChatGPT, the more text will look like ChatGPT. Research on performativity sheds light on this problem of data feedback loops that many see as a fundamental risk to the data ecosystem.

Where some see a problem, others see opportunity. Dynamic benchmarks try to make a virtue out of data feedback loops by creating tests that evolve as models improve. Adding examples where models fail, the benchmark steadily becomes more difficult. Whether model-data feedback loops will be vicious or virtuous is uncertain, but further progress depends on the answer.

The final problem benchmarking now faces is an existential one. As model capabilities exceed those of human evaluators, researchers are running out of ways to test new models. There's hope that models might be able to evaluate each other: Take the best available model and use it to judge new candidates. But this idea of using models as judges runs into serious hurdles. LLM judges are biased, unsurprisingly, in their own favor. Attempts to debias evaluations ultimately point back at what's missing in the first place: a reliable benchmark.

Will the iron rule—the old engine of progress in machine learning—grind to a halt?

In a moment of crisis, we tend to accelerate. We do more of the same, hoping that the problem will go away on its own. What if instead we step back

and ask why we expected benchmarks to work in the first place—and what for? This question leads us into uncharted territory. For the longest time, we took benchmarks for granted and didn't bother to work out the method behind them. We got away with it mostly by sheer luck, but we might not be as lucky this time again. Over the last decade, however, a growing body of work has started to map out the foundations of a science of machine learning benchmarks. What emerges is a rich set of observations—both theoretical and empirical—raising intriguing open problems that deserve the community's attention.

If benchmarks are to serve us well in the future, we must put them on solid scientific ground.

— 2 —

Populations and predictions

The mathematical foundations of machine learning follow the astronomical conception of society: Populations are probability distributions. Optimal predictors minimize loss functions on a probability distribution.

The philosopher Ian Hacking called it the *astronomical conception of society*. It's the idea that we can think of populations as probability distributions and find regularities in them in the same way that an astronomer finds patterns by observing the universe.

The idea goes back to the 19th century Belgian astronomer Adolphe Quetelet, one of the founders of quantitative social science, who called it *social physics*. Quetelet believed that there were laws to be discovered in society, such as stable crime rates in France, and that we could find these laws by fitting statistical models to data. Modern statistics adopted the perspective and the discipline built the formalisms around it that students now learn in virtually every class that touches on data. By the end of the 1930s, statistics had settled on its foundations. Scientists are so used to the idea of populations as probability distributions that they no longer question it. Machine learning, in particular, follows the same astronomical conception, especially when it comes to the bulk of work on model building and evaluation, from linear regression to ChatGPT.

What was radical about the idea when it came up is that human populations clearly are not probability distributions. In a sense, they couldn't be further. The world we live in is a complex, interconnected dynamical system of people, institutions, infrastructure, and nature. What we observe changes

over time and in response to our actions. What we see even changes in response to the models that we deploy in the world. It's important to keep this in mind to understand why sometimes things fail. Datasets collected at different times or different locations generally don't follow the same distributions. Machine learners call this *distribution shift*, but the expression starts from a wrong premise. There is no distribution to begin with. The dynamic nature of society contradicts the stationarity of the astronomical conception. When things go wrong in machine learning, it's often fundamentally this contradiction in disguise.

Yet, the astronomical conception is vindicated by its relentless utility. From its early applications in insurance pricing to recommender engines on digital platforms and AI chatbots, fitting statistical models to datasets has always made someone rich. It's hard to argue with success. It's also hard to argue with its simplicity. The formal setup needed for much of this book fits in a few pages and is well worth knowing in your sleep.

What makes the astronomical conception so useful is that it gives us a simple way to make predictions. A prediction is an educated guess about something we don't know for sure given information we do have. Bernoulli called it the *art of conjecturing*. In this sense, prediction is not just about the future. It also applies to things that have already happened, but we're uncertain about them. A camera took an image, but we're unsure from the pixels whether we're looking at a mouse or a hamster. A company filed their quarterly report with the Securities and Exchange Commission, but we're unsure from the numbers if financial fraud has occurred. In both cases, there's a definitive truth value about which we're unsure.

Prediction also applies in cases where there is no definitive answer. To give an example, suppose you know somebody's age, but nothing else. Make a guess about whether the person has a valid driver's license. Statistics lets you do that, once you fix a population. For any given age, say 18, you look up if the majority of 18-year-olds in the population have a driver's license. If so, you guess yes, otherwise no. This common sense rule turns out to be the optimal predictor if the goal is to maximize the probability of a correct guess. Here the probability refers to a random draw of an individual from that fixed population.

The example makes it clear that your best guess is specific to a population. It depends on whether you consider the population of all citizens of

the United States of America or the population of Tokyo residents. It also depends on time. Your best guess in 2025 may not be your best guess in 1998.

2.1 Prediction

To formalize prediction, we start from a distribution $\mathcal{D}$ called *data-generating distribution* or *population*. A draw from the data-generating distribution gives us one data point. A labeled data point is a pair (x, y), where $x \in \mathcal{X}$ is an array of feature values that could describe a row in a table, a text sequence, or an image. The value $y \in \mathcal{Y}$ is the label assigned to x. The set $\mathcal{Y}$ contains the possible labels that can occur. The set $\mathcal{X} \times \mathcal{Y}$ of all possible data points is called the *universe*. The support of the data-generating distribution is some subset of the universe. A *predictor* is a function $f\colon \mathcal{X} \to \mathcal{Y}$ that maps an input x to a *prediction* $f(x)$.

It's often convenient to write the data-generating distribution as a pair (X, Y) of jointly distributed random variables denoting a random draw from $\mathcal{D}$. This lets us write things succinctly, like $\mathbb{P}\{f(X) = Y\}$, the probability of a correct guess. The random variable Y is then called *target variable* as it is the target of our predictor. Throughout, I omit mentioning the population when it is clear from context.

There are two common settings of prediction:

- In a classification problem, the set $\mathcal{Y}$ contains categorical values, such as $\{\texttt{hamster}, \texttt{mouse}\}$. These are the *classes* of the classification problem. In the context of classification, a predictor is also called *classifier*. The most well-studied setting in learning theory is binary classification, where there are two classes that we can write as 0 and 1. Sometimes it's mathematically convenient to instead work with the numbers -1 and 1 for the two classes. A classification problem with more than two classes is a multi-class problem.
- In a regression problem, the label set $\mathcal{Y} = \mathbb{R}$ is the real line. Rather than predicting a discrete value, our goal is to estimate quantitative information, such as the income of a person, the temperature of an engine, or the stock price of a company share.

In a classification problem, we try to be correct, in a regression problem, we try to be close. Here are some examples of typical prediction problems:

(continued...)

Index