CMPUT 654: Mathematical Foundations of Modern AI Systems Fall 2026

Lecture 2: Learning, information, and structural generalization

September 3, 2026

Lecturer: Csaba Szepesvári *Note prepared by: Csaba Szepesvári*

About these lecture notes. These notes develop a self-contained account of learning, information, and structural generalization. They examine the role of learning in constructing AI systems and use the boxes question throughout: some inputs have been observed, while many others remain unseen. What allows the observations to constrain the missing answers?

Scope relative to class. The class reviewed final-exam arrangements and the use of Canvas and Slack; the course channels contain the exact administrative details. We discussed the usefulness and limitations of asymptotics, the motivation for learning, the boxes theorem, private coin flips, and the example in which all boxes share one label. The meeting stopped before the proof. These notes also include the context-counting calculation, the proof of the boxes theorem, threshold generalization, hypothesis classes, and Bayesian and minimax extensions. The appendices provide a more detailed proof and probability background.

Reading guide. Read through Section 2.9 before the next class, which will begin by reviewing this material. Appendices A and B provide reference material to consult as needed. Section 2.8 contains optional reading before the take-home messages. Numbered endnotes, bibliographical notes, the glossary, one collection of exercises, the appendices, and references follow in that order. The actual homework will sample from these exercises and may add others.

2.1 Why use machine learning to build AI?

We begin with a practical design question:

> Can the desired behavior be specified reliably by explicit rules, or is it better learned from data?

Rules and learning form a continuum of design choices. Explicit programming works (relatively) well when people can state the relevant objects, rules, exceptions, and objective. Broad perception, action, and specifically language tasks create a specification problem: the useful regularities and exceptions are not given in any explicit form and we lack the knowledge to to articulate them without either over, or underconstraining. Learning moves part of that specification burden into data and designing methods to turn data into algorithms. A critical point in using learning to achieve some goals is access to appropriate data that, together with the algorithms that process the data, contains enough information that produces the desired outcomes.

A learned system is still extensively programmed. People choose the model class, architecture, loss, optimizer, data pipeline, interfaces, evaluation procedure, and surrounding software. Data supplies additional constraints that would be difficult to encode one by one. The useful question is which parts of the desired behavior we can specify directly and which parts we ask an algorithm to infer from examples.

Historically, a major strand of AI research emphasized human-written symbolic representations, rules, and search procedures. Learning became increasingly central where useful behavior could be demonstrated in data more readily than it could be exhaustively written as rules. Present systems combine both approaches: learned models operate inside explicitly programmed pipelines. Programming is there, but it is at a different "layer" than before. Future systems are likely to depend even less on human-written code. Just like compilers can be used to create a compiler for the very same language that the compiler supports, future systems may write all the code that is used to produce the models, removing any direct connection to human-written code.

Hardware–software coevolution resembles local search: improving systems around designs that already work (see Lecture 1). Across the research community, a broader search continues alongside these refinements. Many people explore different architectures, learning methods, and ways of posing problems. In this sense, global search is also active, though there is no guarantee that all promising directions are explored. Science is incremental: new ideas build on earlier ones. Resources, evidence, and research fashions influence which ideas are pursued and which older ideas are revisited.

Automating this search extends a familiar idea: generate candidate programs, evaluate them, and retain promising ones. The difficult questions include what to measure and what counts as an improvement. A system may become more accurate but slower, or better at one task and worse at another. With multiple objectives, there need not be a single best system. The choice depends on the intended use and the tradeoffs we accept. Automating the search does not settle these choices, and improvements in measured performance may not carry over to what we actually care about.

Similarly, automating the design of a system does not remove the limits on what it can do. Some computational problems admit no general algorithm; others may require more time or memory than we can supply. Physical systems also face constraints on energy, communication, and dissipation. Interaction brings further difficulties: feedback can be delayed or noisy, and errors can accumulate or destabilize a system. Finally, a system may lack the information needed to choose the right answer. Learning can exploit available information and structure; we will study what can be guaranteed when these are limited. Nevertheless, automation, when done "just right" tends to improve systems and produce better outcomes.

2.1.1 The development loop

The simplified pipeline

$$\boxed{\text{data}} + \boxed{\text{training algorithm}} \longrightarrow \boxed{\text{model}} \longrightarrow \boxed{\text{deployment}}$$

omits the evaluation and decisions that precede deployment. Evaluation assesses the model, for example using *benchmarks*: specified test tasks with scoring procedures. A development team then decides whether to revise the system or deploy it. Fig. 1 shows these decisions and their feedback paths in a simplified manner.

Evaluation supplies evidence for a development decision. Performance may improve on some benchmarks and worsen on others. The team weighs these results against the intended use and decides whether to revise the system or deploy it. When evaluation results guide revisions, they become part of the information used to construct the next system.

Evaluation itself develops through use. People notice failures, discover needs they had not anticipated, and devise new tests. This feedback continually brings development into closer agreement with what they want the system to become. Removing human feedback closes that source of guidance. For a stable, well-specified task, this may be acceptable. When our aims are still taking shape, however, a system can improve against its existing criteria while missing what we have come to care about. The development loop when fully automated is optimizing the benchmarks; the recent

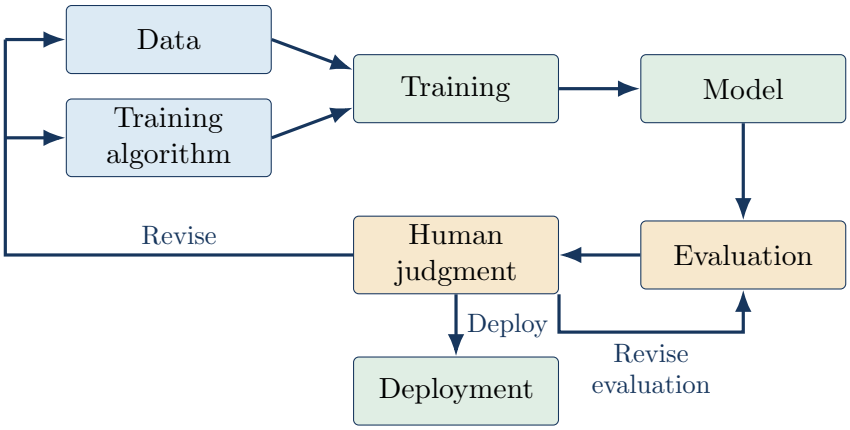


Figure 1: Evaluation informs human judgment about whether to revise the system or deploy it. Human judgment can revise the data, algorithm, and evaluation.

OpenAI-Huggingface incident is a recent reminder of that this optimization may result in unintended outcomes, a vivid example of the importance of human oversight and responsibility.

2.1.2 Pretraining, post-training, and uneven improvements

Pretraining fits a broad predictive model to large collections of data. For a language model, it supplies linguistic regularities, factual associations, styles, patterns of reasoning, and many other statistical regularities present in the training distribution. It does not by itself specify how a deployed assistant should interpret instructions, refuse requests, use tools, or prefer one acceptable response over another.

A pretrained language model continues text by drawing on patterns learned during pretraining—a "stochastic parrot" that can imitate many voices and genres. A question may elicit an answer, further questions, or an imagined conversation. For example, base GPT-3 responded to a request for a story by generating more story-writing instructions, and to a question about code by supplying multiple-choice options. These are plausible continuations of the text, but they do not fulfil the request.[1]

Post-training aims to make the pretrained model useful for its intended tasks. Post-training consumes data of various forms. Instruction examples teach desired input–output behavior; preference or reward signals rank possible responses; task-specific training can target particular capabilities. These stages exist because next-token prediction and desired deployed behavior are different objectives, even though post-training acts on a representation built by pretraining.

These stages raise several issues.

- Improving coding performance need not improve creative writing on every benchmark. Transfer depends on whether the two tasks use shared internal structure and whether the update preserves it. With enough capacity and an appropriate procedure, simultaneous improvement may be possible. This creates a pressure for larger models. Historically people were vary of large models, but with the right data, contrary to some beliefs, there is no theoretical reason to stay away from large models. Theory predicts that in the worst-case, large models with powerful training are a bad choice. This does not mean that on a particular problem they will necessarily do poorly.

- A large model can still overfit an objective, a dataset, or a repeatedly consulted benchmark. More parameters change what can be represented; they do not remove the distinction between training

evidence and fresh evidence.

- Training data can contain contradictions. A probabilistic model can represent competing continuations and uncertain outcomes. It does not decide by itself which contradiction should be resolved in which direction; the objective, the data, context, and post-training signals determine the learned compromise.

Reliance on indirect interventions complicates the development of learned systems: we steer their behavior mostly through indirect interventions—change data, objectives, architectures, optimization, or feedback and observe what behavior changes. Why machine learning is excellent for transforming information about desired behavior into software when there is no other alternative, machine learning struggles to produce software that meets strict criteria.

An alternative to the indirect interventions is "brain surgery" on the learnt models. Reverse engineering can reveal useful features or computations, but learned representations are not designed to be easily "editable" (let alone human-readable). Thus, the success of this direct model-surgery tends to be limited.

2.2 The alignment needed for generalization

Most inputs an AI system will encounter are absent from its training data. *Generalization* is therefore essential: it is the mechanism by which finite experience can support behavior on a much larger set of possible inputs.

The central principle is:

> Learning works on unseen inputs when the target, data, learner, and information protocol share exploitable structure.

Theory abstracts away details so that we can identify assumptions and derive their consequences. A proved theorem remains true under its assumptions as hardware, methods, and research fashions change. Its usefulness for a particular system depends on what the abstraction leaves out. Some differences change the conclusion; others have little practical effect. Learning to judge which differences matter is a central aim of this course. We begin with a simple learning setting in which we can determine exactly what permits generalization and what prevents it.

2.3 What's in a box? A simple model of learning

There are N boxes numbered $1, \ldots, N$. Each box contains one hidden bit. The complete collection of bits is given by a fixed but unknown function

$$f : [N] \longrightarrow \{0,1\}, \qquad [N] := \{1, \ldots, N\}.$$

Thus $f(x)$ is the bit in box x. The notation $\{0,1\}^{[N]}$ denotes the set of all such functions. Listing the values $(f(1), \ldots, f(N))$ identifies the same object with a vector in $\{0,1\}^N = \underbrace{\{0,1\} \times \cdots \times \{0,1\}}_{N \text{ times}}$.

A randomized sampling procedure selects *training* indices $X_1, \ldots, X_n$ and one *test* index X. The complete tuple

$$(X_1, \ldots, X_n, X)$$

produced by the randomized procedure may have *any* joint distribution on $[N]^{n+1}$. In particular, the indices may be dependent and may repeat. The procedure for selecting the indices can be made known to the learner. The procedure for selecting the indices does not use the hidden labeling function f, so the selected indices cannot themselves communicate information about the target.

The learner is shown the contents of the boxes with indices $X_1, \ldots, X_n$. No other information is revealed. Based on this information, the learner must be prepared to choose a bit for each of the boxes. After this choice is made, this choice is evaluated on the test index X. The evaluation simply compares the result with $f(X)$. In case of a match, the learner succeeds, otherwise it suffers a unit loss. Formally, if $\hat{Y} \in \{0, 1\}$ is the bit that is to be compared to $f(X)$, the learner's loss is 0 when $\hat{Y} = f(x)$ and is 1, otherwise. Formally, we can write that the learner's loss is

$$\mathbb{I}(\hat{Y} \neq f(X)) \,. \tag{2.1}$$

Here, $\mathbb{I}$ is the so-called *indicator function*. Its argument should be an event E. In particular, for any event E, we have

$$\mathbb{I}\{E\} = \begin{cases} 1, & \text{if } E \text{ is true}\,; \\ 0, & \text{if } E \text{ is false}\,. \end{cases}$$

Strictly speaking, in Eq. (2.1) we should therefore have used $\{\hat{Y} \neq f(X)\}$ as the argument of the indicator function; that is, we should have written $\mathbb{I}(\{\hat{Y} \neq f(X)\})$. This is visually cumbersome, so we abuse notation by dropping the opening and closing braces. We also do this when we write arguments of the probability function $\mathbb{P}$. These are standard notational abuses in probability, and we will continue to use them. One should nevertheless recognize them as abuses of notation. The loss in Eq. (2.1) is called the 0–1 *loss* of predicting $\hat{Y}$ when the truth is $f(X)$. This terminology will make more sense as we introduce other losses later.

Since X and $\hat{Y}$ are random, the loss in Eq. (2.1) is also random. To obtain a single numerical measure of the learner's performance, we take the expectation of this random loss. This gives the learner's expected loss, $\mathbb{E}[\mathbb{I}(\hat{Y} \neq f(X))]$. A basic property of expectations is that $\mathbb{E}[\mathbb{I}(E)] = \mathbb{P}(E)$. In words, the expected value of the indicator of an event is the probability of that event.

In learning terminology, a box index is an *input*, and its hidden bit is its *label*. Each revealed index–bit pair is a *labeled example*. The ordered collection of n examples is the *training sample*, also called the *training data*; repeated inputs are allowed. The *test input* is the index at which the learner is asked to predict a label.

Combinatorial explosions and asymptopia In this example, each box represents a possible input to some AI system. If we consider a chatbot that needs to provide yes/no answers to some questions and we use a transformer with a maximum context window length L of only $1,000$, with a (typical) token-vocabulary size of 10^5, we find that the number of possible inputs N is more than 10^{5000}. This is far more than the number of atoms in the observable universe.[2] The rapid (exponential) growth of the number of possible inputs by scaling some parameter such as L is called a *combinatorial explosion*. Observing every possible input is out of reach. Even with internet scale data, the number of training data points n is a vanishing fraction of all possible inputs. As such, to get a successful AI system, learning must transfer information from a vanishingly small fraction of the inputs to the rest of the input space.

Asymptotic analysis studies how quantities behave as parameters such as N grow without bound. It can reveal trends that are hard to see in a detailed finite calculation. For example, exponential and polynomial growth in runtime have very different consequences as the input size increases. Applying this guidance requires checking which parameter grows, what is held fixed, and whether

the limiting behavior describes the regime of interest. Constants and lower-order terms can still matter there. Even an enormous N does not by itself settle these questions: here, the amount of data and the structure of the labels will also matter. Explicit finite bounds help us make this judgment.

2.3.1 Randomized learners and risk

We will now formalize what a learner can do. First, consider learners that do not randomize. Given a labeled sample consisting of n index–bit pairs, such a learner chooses which bit to assign to each future index. Formally, we can view such a learner as mapping an element of $([N] \times \{0,1\})^n$ to $\{0,1\}^{[N]}$ (equivalently, to $\{0,1\}^N = \underbrace{\{0,1\} \times \cdots \times \{0,1\}}_{N \text{ times}}$). In addition to an n-tuple of index–bit pairs, a *randomized learner* takes a *random seed* W that is uniformly distributed over $[0,1]$. In Exercise 1, you will show that this choice places no restriction on the learner's output distribution. Formally, such a learner can be identified with a measurable deterministic map

$$\mathcal{A} : \big([N] \times \{0,1\}\big)^n \times [0,1] \longrightarrow \{0,1\}^{[N]}. \tag{2.2}$$

For $(s,w) \in ([N] \times \{0,1\})^n \times [0,1]$, $\mathcal{A}(s,w)$ is thus a function from $[N]$ to $\{0,1\}$. The prediction at the *test index* X is $\hat{Y} = \mathcal{A}(S_f, W)(X)$, where

$$S_f = \big((X_i, f(X_i))\big)_{i=1}^n \tag{2.3}$$

is the labeled sample the learner receives before making a decision.[3]

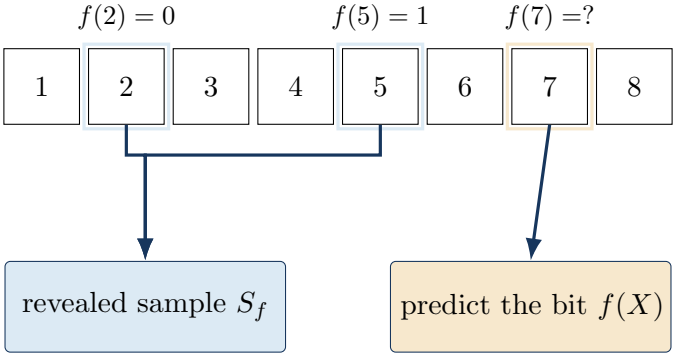


Figure 2: An example with $N = 8$: boxes 2 and 5 were revealed, and box 7 is the test box. The theorem allows any joint procedure for choosing the revealed and test indices.

The expected loss, also known as the *risk*, of the learner that uses $\mathcal{A}$, evaluated on a fixed hidden function f, is

$$R(\mathcal{A}, f) = \mathbb{E}[\mathbb{I}\{\mathcal{A}(S_f, W)(X) \neq f(X)\}]\,. \tag{2.4}$$

Here, the randomness comes from $(X_1, \ldots, X_n, X)$ and W. Importantly, W is independent of $(X_1, \ldots, X_n, X)$. This expresses that the algorithm's randomization is private. We suppress the dependence of the risk on the distribution of the random indices in the notation $R(\mathcal{A}, f)$ to minimize clutter.

2.3.2 Towards robust learners

Our goal is to make the learner robust. Given $\mathcal{A}$, we evaluate its performance under the worst-case choice of f. Thus, $\max_{f\in\{0,1\}^{[N]}} R(\mathcal{A}, f)$, the *worst-case risk*, is the final performance metric assigned to a learner that uses $\mathcal{A}$. Worst-case analysis is common in computer science: we want protection against the worst possible case. This is a form of risk aversion or conservatism. View this as a starting point. This robust criterion is sometimes viewed as a game: the learner chooses a procedure, and nature may choose any fixed arrangement of bits. The question is which method has the smallest worst-case risk.

The critical point is that in Eq. (2.4), the test index X either matches one of the indices $X_1, \ldots, X_n$ or matches none of them. If X matches one of $X_1, \ldots, X_n$, the learner should simply use the label associated with the matching index. That is, if the learner's data are $((X_1, Y_1), \ldots, (X_n, Y_n))$ and $X = X_J$ for some $J \in [n]$, then the learner should set $\hat{Y} = Y_J$. Using Y_J is valid because *we are promised* that for the unknown secret function f, $Y_J = f(X_J)$. Thus, choosing Y_J, we have $\hat{Y} = Y_J = f(X_J) = f(X)$. We are therefore guaranteed to be correct, which gives zero loss, the minimum possible loss—we are winning! The challenge is deciding what to do when X matches none of $X_1, \ldots, X_n$. One may then try to "figure out some pattern" in the data and follow that pattern. If we had more structure in the problem, this could work (more about this soon). Alternatively, one could simply always return zero or always return one. One optimal solution is to flip an unbiased coin and return 0 for heads and 1 for tails. To express this algorithm using the random seed W, set $\hat{Y} = \mathbb{I}(W \le 1/2)$ when X is not found in $\{X_1, \ldots, X_n\}$. We will call this algorithm $\mathcal{A}_{\text{mem}}$.

2.4 Exact solution of the unstructured boxes problem

Define the event

$$U = \{X \notin \{X_1, \ldots, X_n\}\}. \tag{2.5}$$

This event occurs when the test index does not appear among the indices for which we have information about the secret function f. This is the difficult case for the learner. On U^c, the learner has been shown the requested bit and can remember it. On U, no revealed label gives direct information about $f(X)$.

The smallest worst-case risk over the allowed learners is the *minimax risk*.

Theorem 2.4.1 (Arbitrary-labeling minimax risk)**.** *Consider any index-selection procedure allowed in the boxes problem, and let the learner's random seed W be independent of the selected indices. Minimizing over the randomized learners defined in Eq. (2.2),*

$$\min_{\mathcal{A}} \max_{f\in\{0,1\}^{[N]}} R(\mathcal{A}, f) = \frac{1}{2}\mathbb{P}(U). \tag{2.6}$$

Furthermore, $\mathcal{A}_{\text{mem}}$ is optimal, that is, it achieves the above minimax risk.

The theorem should be intuitive: Conditional on seeing the contents of the box opened at test time, the best loss is zero. Conditional on not seeing it, the exact minimax loss is $1/2$. The minimax risk is therefore entirely controlled by the probability of an unseen box.[4]

Equation (2.6) says two things. There is a learner whose risk is at most $\frac{1}{2}\mathbb{P}(U)$, however f is chosen. Conversely, for every learner, there is a labeling function f for which its risk is at least $\frac{1}{2}\mathbb{P}(U)$.

2.4.1 The same boxes, with one shared label

It is instructive to consider the following promise: all boxes contain the same bit. Opening any one box now determines every label. For $n \geq 1$, the learner that returns the first observed label at every input has zero risk, even if the test box has never been opened and N is enormous.

The observation supplies the value of the shared bit. The promise supplies the relation that makes this value informative about every other box. Thus the learner receives information both from the observed label and from the specified target class. The arbitrary-labeling theorem permits 2^N targets; the common-label promise permits only two. Section 2.7 develops a richer promise for the same boxes question.

Under a structural promise, which boxes are observed can matter more than how many are observed; see Exercise 9. This exercise thus illustrates the utility of having the right type of data.

2.4.2 Private randomization and risk

The order of choices is part of the theorem. Nature may know the algorithm, but it chooses a fixed f before the learner's seed W is drawn. If nature could instead choose the unseen test bit after seeing the prediction, it could force an error by choosing the opposite bit.

Remark 2.4.2 (Risk and repeated experiments). Random prediction does not promise half an error on one trial. If E_t is the error indicator in repetition t and the repetitions are independent under the same f and the same procedure, then the following convergence holds (*why?*; see Exercise 3).

$$\frac{1}{T}\sum_{t=1}^{T} E_t \longrightarrow R(\mathcal{A}, f) \qquad \text{almost surely.}$$

This is the operational meaning of risk as a long-run error rate.

2.5 Proof of Theorem 2.4.1

We give the main argument here in a compact form. A line-by-line version is given in Appendix A. That appendix also explains why learning to supply the omitted steps is important. The elementary probability facts used in both versions are reviewed separately in Appendix B. Supply the missing justifications in Exercises 6 and 7; the expanded proof in Appendix A provides intermediate steps.

We prove the equality by bounding its left-hand side (LHS) both above and below by its right-hand side (RHS). For the upper bound, we analyze the particular learner $\mathcal{A}_{\text{mem}}$. For the lower bound, we fix an arbitrary learner and show that some fixed hidden function must be hard for it.

Upper bound. We first show that the LHS is upper bounded by the RHS. First notice that

$$\min_{\mathcal{A}} \max_{f} R(\mathcal{A}, f) \leq \max_{f} R(\mathcal{A}_{\text{mem}}, f)\,.$$

Now fix some $f \in \{0,1\}^{[N]}$. We will show that $R(\mathcal{A}_{\text{mem}}, f) = \mathbb{P}(U)/2$. Once this has been shown for every fixed f, the upper bound follows.

For this fixed target f, let $E_f = \{\mathcal{A}_{\text{mem}}(S_f, W)(X) \neq f(X)\}$ be the event that $\mathcal{A}_{\text{mem}}$ makes an error. From the definition of R in Eq. (2.4),

$$R(\mathcal{A}_{\text{mem}}, f) = \mathbb{P}(E_f) = \mathbb{P}(E_f, U^c) + \mathbb{P}(E_f, U)\,. \tag{2.7}$$

On U^c, the test index agrees with at least one training index: $X = X_J$ for some $J \in [n]$. The learner returns the revealed label $f(X_J) = f(X)$, so E_f cannot occur. Thus,

$$E_f \cap U^c = \emptyset, \qquad \mathbb{P}(E_f, U^c) = 0\,.$$

On U, the learner predicts with the fair bit $B = \mathbb{I}\{W \le 1/2\}$. Write $Z = (X, X_1, \dots, X_n)$. Conditional on any fixed index tuple $z = (x, x_1, \dots, x_n)$ that has positive probability and for which U occurs, $f(x)$ is a fixed bit, whereas B is an independent fair bit. Therefore

$$\mathbb{P}(B \neq f(x) \mid Z = z) = \frac{1}{2}.$$

Averaging over the index tuples for which U occurs gives $\mathbb{P}(E_f, U) = \mathbb{P}(U)/2$. Substituting the two terms into Eq. (2.7) gives

$$R(\mathcal{A}_{\mathrm{mem}}, f) = \frac{1}{2}\mathbb{P}(U), \tag{2.8}$$

for every fixed f, which proves the upper bound.

Lower bound. We now turn to showing that the LHS is lower bounded by the RHS. Fix an arbitrary randomized learner $\mathcal{A}$. It suffices to prove that $\max_{f \in \{0,1\}^{[N]}} R(\mathcal{A}, f) \ge \frac{1}{2}\mathbb{P}(U)$. Write $Z = (X, X_1, \dots, X_n)$. Introduce an auxiliary random function

$$F : [N] \longrightarrow \{0, 1\}$$

by choosing the bits $F(1), \dots, F(N)$ independently, with

$$\mathbb{P}(F(j) = 0) = \mathbb{P}(F(j) = 1) = \frac{1}{2}, \qquad j \in [N].$$

Choose F independently of the index tuple $(X, X_1, \dots, X_n)$ and the learner's seed W. Set

$$\widehat{Y} = \mathcal{A}(S_F, W)(X).$$

We then have

$$\max_f R(\mathcal{A}, f) \ge \mathbb{E}\big[R(\mathcal{A}, F)\big] = \mathbb{P}\big(\widehat{Y} \neq F(X)\big) \ge \mathbb{P}\big(\widehat{Y} \neq F(X), U\big) \tag{2.9}$$

To continue from the last term in Eq. (2.9), let

$$\mathcal{Z} = \big\{z = (x, x_1, \dots, x_n) \in [N]^{n+1} : x \notin \{x_1, \dots, x_n\}\big\}\,.$$

Fix $z = (x, x_1, \dots, x_n) \in \mathcal{Z}$, a possible sample s, and $a, b \in \{0, 1\}$. On $\{Z = z, S_F = s\}$, the prediction is $\mathcal{A}(s, W)(x)$. The event $\{Z = z, S_F = s\}$ is determined by Z and coordinates $F(j)$ with $j \neq x$. Hence the independent fair bit $F(x)$ is independent of this event and of $\mathcal{A}(s, W)(x)$. Consequently,

$$\begin{aligned}\mathbb{P}\big(\widehat{Y} = a, F(x) = b, Z = z, S_F = s\big) \\ = \frac{1}{2}\mathbb{P}\big(\widehat{Y} = a, Z = z, S_F = s\big).\end{aligned} \tag{2.10}$$

This identity also holds when $\mathbb{P}(Z = z, S_F = s) = 0$. Summing the two disjoint mismatch cases and then all possible (z, s) gives

$$\begin{aligned}\mathbb{P}\big(\widehat{Y} \neq F(X), U\big) &= \frac{1}{2}\sum_{z\in\mathcal{Z}}\sum_{s}\Big[\mathbb{P}(\widehat{Y} = 1, Z = z, S_F = s) + \mathbb{P}(\widehat{Y} = 0, Z = z, S_F = s)\Big] \\ &= \frac{1}{2}\sum_{z\in\mathcal{Z}}\sum_{s}\mathbb{P}(Z = z, S_F = s) = \frac{1}{2}\sum_{z\in\mathcal{Z}}\mathbb{P}(Z = z) = \frac{1}{2}\mathbb{P}(U).\end{aligned}$$

Combining this with Eq. (2.9), we get $\max_f R(\mathcal{A}, f) \geq \mathbb{P}(U)/2$. Since $\mathcal{A}$ was arbitrary, this proves the lower bound. □

Remark 2.5.1 (Random targets in the lower-bound proof). The random function F is only a device for proving that some fixed target f must be hard. Section 2.8 returns to this device and uses it to compare Bayesian and minimax decision making.

2.6 Calculating the unseen probability in an i.i.d. example

The theorem allows *dependent* data (meaning, the joint distribution of $(X_1, \ldots, X_n, X)$ is arbitrary; the random variables in it can "depend" on each other). We now add a special assumption to obtain a closed form: $X_1, \ldots, X_n, X$ are independent draws from the same distribution P on $[N]$. Write $P(x)$ for the probability assigned to index x. Besides this giving us a closed form, the assumption models a scenario which is often not a bad approximation of reality: The data collection and the testing follows a random process, and the data points are chosen independently of each other. For example, if physical experiments are used to obtain a data point, these experiments are somehow isolated from each other in time and space, so the randomness that governs the outcome of obtaining the first datapoint does not influence that of the second.

In a laboratory experiment, independence can fail when the equipment is not reset or when conditions carry over between trials. In a survey, independent sampling requires selecting respondents independently from the specified population. Stopping people at a shopping mall and talking to them will not produce an independent sample (couples often shop together, people go to a shop at a certain time may have similar jobs), let alone one that is uniform. The characters of a text are also dependent. The *i.i.d.* (*independent, identically distributed*) assumption is common as it allows some crude quantification of how information scales with data, as we will illustrate now. We will return to going beyond this assumption later. An important part of this assumption is that the test index is drawn from the same distribution as the training indices. This equality of sampling laws models deployment data as being obtained in the same way as training data. The equality often fails; when only the input law changes, we have *covariate shift*, since inputs are also called *covariates* in statistics. TODO: Add a simple exercise about covariate shift; probability ratios govern how much the error shifts.

Conditioning on $X = x$ gives (see Exercise 4)

$$\mathbb{P}(U) = \sum_{x=1}^{N} P(x)(1 - P(x))^n. \tag{2.11}$$

If P is uniform, then

$$\mathbb{P}(U) = \left(1 - \frac{1}{N}\right)^n, \qquad \min_{\mathcal{A}}\max_{f} R(\mathcal{A}, f) = \frac{1}{2}\left(1 - \frac{1}{N}\right)^n. \tag{2.12}$$

For fixed N, the risk tends to zero as n grows. But when n/N is small, it remains close to $1/2$: eventual coverage does not help in this regime.

For example, a sufficient condition for worst-case risk at most 0.1 is

$$n \geq N \log 5, \tag{2.13}$$

because $(1 - 1/N)^n \leq e^{-n/N}$. Conversely, Bernoulli's inequality gives $(1 - 1/N)^n \geq 1 - n/N$, so risk at most 0.1 requires $n \geq 0.8N$. Exercise 5 asks you to prove both inequalities and derive these sample-size bounds. Thus an unstructured problem with independent uniform training and test indices requires a number of observations proportional to the number of boxes. When N is enormous relative to n, almost every test input remains unseen, and achieving, say, $n \geq 0.8N$ is hopeless.

2.7 What's in a box? Part II: a threshold promise

We keep the boxes, the observations, and the prediction question. As in the common-label example, we change the promise about their contents. In that example, we assumed that the labeling functions $f : [N] \to \{0, 1\}$ that we may ever see are constant. Here, we assume that these functions are nondecreasing as a function of the box indices: Going from smaller to larger indices, once a label is 1, all subsequent labels are 1. Equivalently, $f = f_k$ for some unknown *threshold* $k \in [N + 1]$, where

$$f_k(j) = \mathbb{I}\{j \geq k\}, \qquad j \in [N]. \tag{2.14}$$

(The case $k = N + 1$ is the all-zero function.) One observed positive label now constrains every box to its right, and one observed negative label constrains every box to its left. Information transfers between different inputs.

Just like in the previous section, for the quantitative bound below, assume that the training indices and the test index are independent draws from the same probability distribution P on $[N]$. Again, we will write $P(x)$ for the probability assigned to index x. This assumption ensures that a region with substantial test probability also has a substantial chance of being represented in the training data. Exercises 10 to 12 explore what can be proved under more general sampling assumptions.

Let K be the smallest index with an observed positive label, and set $K = N + 1$ if no positive label was observed. The learner returns f_K. Assume that the true labeling function (unknown to the learner) is f_k. The chance that the learner is missing the correct label on the test index X given the training indices $(X_1, \ldots, X_n)$ is

$$M = \mathbb{P}(f_K(X) \neq f_k(X) | X_1, \ldots, X_n)\,. \tag{2.15}$$

We will call this the *error mass* on a fresh sample from P. It follows from the definition that

$$M = \sum_{x=1}^{N} P(x)\mathbb{I}\{f_K(x) \neq f_k(x)\}.$$

The error mass is therefore a random variable through K, whose randomness comes from that of $X_1, \ldots, X_n$. The next proposition shows that the chance of this random error exceeding some level ϵ decreases exponentially fast in n (to be precise, in $n\epsilon$):

Proposition 2.7.1 (Threshold generalization)**.** *For every P, every threshold f_k, and every $\epsilon \in (0, 1)$,*

$$\mathbb{P}\big(M > \epsilon\big) \leq (1 - \epsilon)^n \leq e^{-n\epsilon}. \tag{2.16}$$

Consequently, $n \geq \epsilon^{-1} \log(1/\delta)$ observations suffice to guarantee that the error mass stays below ϵ with probability exceeding $1 - \delta$.

Note how the threshold to guarantee a small error mass is independent of N.

Proof. If $k = N + 1$, every label is zero, the learner is exact, and the claim is immediate. Assume therefore in the rest of the proof that $k \le N$. The learner chooses K to be the smallest index with a positive label if such an index exists, otherwise it chooses $K = N + 1$. Because of this, we claim that the learner makes a mistake exactly when tested at an index in the interval $[k, K) := \{k, k+1, \dots, K-1\}$ (in this proof intervals are restricted to integers in the interval).

First, notice that $K \ge k$ always holds, so this interval is well-defined (it could be empty when $K = k$). Indeed, if there was no positive label, $K = N + 1 \ge k$, while if there is a positive label, $f_k(K) = 1$, which is only possible if $K \ge k$ by the definition of f_k.

Now, since $K \ge k$ always holds, that is, the learner is overestimating the index k, the learner predicts all 0 labels correctly and it also predicts all 1 labels correctly for indices at least K. For any of the indices x in $[k, K)$, $f_k(x) = 1 \ne 0 = f_K(x)$, establishing the claim.

Hence, $M = \sum_x P(x)\mathbb{I}(f_K(x) \ne f_k(x)) = P([k, K)) = q(K-1)$ where for $k - 1 \le j \le N$, we define $q(j) = P([k, j])$, with $[k, k-1] = \emptyset$, so $q(k-1) = 0$. Clearly, this function is nondecreasing on $[k-1, N]$.

If $q(N) \le \epsilon$, $\mathbb{P}(M > \epsilon) = 0$ and the result holds. Therefore assume that $q(N) > \epsilon$ and let j_ϵ be the smallest index $k \le j \le N$ such that $q(j) > \epsilon$. Since q is nondecreasing, $q(j) \le \epsilon$ for every $j < j_\epsilon$ and $q(j) > \epsilon$ for every $j \ge j_\epsilon$. Therefore,

$$\{M > \epsilon\} = \{q(K-1) > \epsilon\} = \{K > j_\epsilon\}.$$

Now, $K > j_\epsilon$ can only hold if the sample $\{X_1, \dots, X_n\}$ had no point in $[k, j_\epsilon]$ (otherwise, at the point in this interval the label would be one, forcing K to be at most j_ϵ). Thus,

$$\{K > j_\epsilon\} \subset \{X_1 \notin [k, j_\epsilon]\} \cap \cdots \cap \{X_n \notin [k, j_\epsilon]\}. \tag{2.17}$$

Taking probabilities and using that $X_1, \dots, X_n$ are independent and identically distributed, we get

$$\mathbb{P}(M > \epsilon) = \mathbb{P}(K > j_\epsilon) \le \mathbb{P}(X_1 \notin [k, j_\epsilon])^n.$$

Now, $\mathbb{P}(X_1 \notin [k, j_\epsilon]) = 1 - \mathbb{P}(X_1 \in [k, j_\epsilon])$ and since $\mathbb{P}(X_1 \in [k, j_\epsilon]) = P([k, j_\epsilon]) = q(j_\epsilon) > \epsilon$, $\mathbb{P}(X_1 \notin [k, j_\epsilon]) < 1 - \epsilon$. Chaining the inequalities gives the first inequality, while the second follows by the elementary inequality that $\log(1 + x) \le x$ which holds for all $x > -1$.[5] □

2.7.1 Comparing expected error in the same boxes problem

The boxes theorem bounds risk averaged over the sample, test input, and learner's randomization. The threshold proposition gives a stronger kind of control: it bounds the probability that the learned predictor's error mass exceeds a specified level. This also yields an expected-risk bound.

Corollary 2.7.2 (Expected risk of the threshold learner)**.** *For every distribution P on $[N]$, every threshold target f_k with $k \in [N+1]$, and every sample size $n \ge 0$, let the training and test indices be independent draws from P. The risk of the learner returning f_K satisfies*

$$\mathbb{E}[M] \le \frac{1}{n+1}. \tag{2.18}$$

Exercise 8 asks you to derive this corollary from Proposition 2.7.1 by integrating the tail probability in Eq. (2.16).

For a direct comparison, use independent uniform training and test indices in both problems. Allowing arbitrary labels gives exact minimax risk $\frac{1}{2}(1 - 1/N)^n$, so expected error at most 0.1 requires $n \ge 0.8N$. With the threshold promise, $n = 9$ already suffices for expected error at most 0.1, for every N. This upper bound even holds for every input distribution P. With the common-label promise, one observation suffices for zero error.

2.7.2 Hypothesis classes and beyond

All three boxes problems ask what can be predicted about closed boxes from open ones. Their different answers come from different promises about the target: it may be any function, a constant function, or a threshold. We can express such a promise by specifying a set of possible targets.

Definition 2.7.3 (Realizability and consistency)**.** Let $\mathcal{F} \subseteq \{0,1\}^{[N]}$ be a nonempty *target class*. The assumption that the true labeling function f belongs to $\mathcal{F}$ is called *realizability*. A predictor g is *consistent* with a labeled sample $((x_i, y_i))_{i=1}^n$ if $g(x_i) = y_i$ for every i.

Such a set of predictors is commonly called a *hypothesis class*. The assumption that the target belongs to $\mathcal{F}$, called *realizability*, does not imply that the learner returns a member of $\mathcal{F}$. A learner that always returns a member of $\mathcal{F}$ is called *proper*; otherwise it is called *improper*. This is peculiar terminology: there is nothing improper about improper learning.

The next theorem assumes realizability and applies to learners that return a consistent member of $\mathcal{F}$. Such a member exists because the target f belongs to $\mathcal{F}$ and agrees with every training label.

For a predictor g and target f, a distribution P over the inputs, write

$$L_P(g, f) = \mathbb{P}(g(X) \neq f(X))$$

where the test point X has distribution P (with symbols, $X \sim P$). For a learned predictor $\widehat{f}$, this is the same as its error mass: $M = L_P(\widehat{f}, f)$, extending the notation from the threshold problem.

Theorem 2.7.4 (Generalization by consistency in a finite class)**.** *Let $\mathcal{F} \subseteq \{0,1\}^{[N]}$ be nonempty, $f \in \mathcal{F}$, and let $X_1, \ldots, X_n$ be independent draws from a distribution P on $[N]$. Any learner returning a member $\widehat{f} \in \mathcal{F}$ consistent with $((X_i, f(X_i)))_{i=1}^n$ satisfies, for every $\epsilon \in (0,1)$,*

$$\mathbb{P}\big(L_P(\widehat{f}, f) > \epsilon\big) \leq |\mathcal{F}| e^{-n\epsilon}. \tag{2.19}$$

In particular, for $\delta \in (0,1)$, $n \geq \epsilon^{-1}(\log|\mathcal{F}| + \log(1/\delta))$ suffices for error mass at most ϵ with probability at least $1 - \delta$.

Exercise 13 asks you to prove this theorem. It applies to any choice among consistent members, including a randomized choice. The restriction to $\mathcal{F}$ matters: matching observed labels alone gave no useful prediction on unseen boxes in the unstructured problem. The theorem also leaves a computational question: how do we find a consistent member efficiently?

For thresholds, $|\mathcal{F}| = N + 1$, so the cardinality bound introduces a $\log(N+1)$ term absent from Proposition 2.7.1. This is a limitation of counting hypotheses. General theory based on *VC dimension* recovers the optimal sample-complexity rate for thresholds without dependence on N. Our direct proof gives a simple argument and explicit constants.

Remark 2.7.5 (Consistency and optimal sample complexity)*.* An *empirical risk minimizer* (ERM) minimizes (average) loss on the training sample over the hypothesis class. Under realizability and 0–1 loss, this means choosing a consistent member. An arbitrary such choice need not achieve the optimal sample-complexity rate: some ERM learners require an extra logarithmic factor. Thus consistency alone does not settle how to learn with the fewest observations. See the bibliographical notes for optimal rates and limitations of ERM.

When the promise fails. Realizability requires justification, and for many applications we cannot justify it. *Label noise* and *misspecification* are different issues: noise concerns the observed labels, while misspecification means that the assumed model does not contain the data-generating

mechanism. Allowing label noise does not ensure that the underlying target belongs to a chosen class.

The *agnostic* approach compares the learner's performance with that of the best predictor in a chosen class, without assuming the class contains a perfect predictor. This comparison is useful, but a small gap to the best predictor in the class need not mean good absolute performance. Every member of the class may perform poorly.

Algorithm design beyond a class. Choosing a class is one design choice. It generally does not determine which predictor to prefer, how to find it, what data to acquire, or which performance criterion to use. Architecture, optimization, regularization, and data collection can all affect the result. Later in the course we will study *algorithmic bias*: the preferences built into a learning procedure, including its choices among predictors that fit the same observations.

Remark 2.7.6 (Bayesian modeling and design choices). A *prior* supplies weights on possible targets and can express preferences within a class. Choosing the prior, the loss, and the observation model remains part of designing the system. Section 2.8 develops the Bayesian decision criterion.

It helps to distinguish two questions. What can a method guarantee across a specified family of problems? What method should we use for this application, with this information, objective, and computational budget? General theory informs the second question; applying it requires judging whether its assumptions and criterion fit. Even the goal of building a broadly capable AI system requires choices about what capabilities to seek and how to evaluate them. Hypothesis classes help us study generalization, but choosing a class does not settle the broader problem of designing a learning system.

2.8 Optional reading: priors, best responses, and robustness

The lower-bound proof introduced a random function F to average risks over possible targets. A distribution over targets can also be used to define a decision criterion. This leads to two different questions: what minimizes average risk under a specified prior, and what minimizes worst-case risk?

2.8.1 Priors and Bayes rules

Let $\mathcal{F} = \{0,1\}^{[N]}$ and let π be a probability distribution on $\mathcal{F}$. Draw F from π independently of the indices and the learner's seed. A *Bayes rule* minimizes

$$\mathbb{E}[R(\mathcal{A}, F)]. \tag{2.20}$$

This expectation averages over F; $R(\mathcal{A}, f)$ already averages over the indices and the learner's seed. For a sample s of positive probability, the *posterior probability* of label 1 at index x is

$$p(s,x) = \mathbb{P}(F(x) = 1 \mid S_F = s).$$

A Bayes rule chooses a prediction to minimize its conditional expected loss. Exercise 17 asks you to derive that choice and compare it with the worst-case choice for an unknown bit.

Remark 2.8.1 (Information supplied by a prior). The prior supplies weights on possible targets. These weights may encode knowledge, judgment, or a modeling assumption; they are not selected by the Bayes optimization itself. Priors can also supply relations between labels: Exercise 29 compares independent labels with a shared label.

2.8.2 The finite minimax theorem

Let $\mathcal{D}$ be the finite set of deterministic learners, and write $\Delta(\mathcal{D})$ and $\Delta(\mathcal{F})$ for distributions on learners and targets. A distribution $\mu \in \Delta(\mathcal{D})$ describes a randomized learner. For independently drawn $A \sim \mu$ and $F \sim \pi$, define

$$L(\mu, \pi) = \mathbb{E}[R(A, F)]. \tag{2.21}$$

Here A is a random element of $\mathcal{D}$. Let δ_f put probability one on target f. The finite minimax theorem gives

$$\min_{\mu \in \Delta(\mathcal{D})} \max_{f \in \mathcal{F}} L(\mu, \delta_f) = \max_{\pi \in \Delta(\mathcal{F})} \min_{\mu \in \Delta(\mathcal{D})} L(\mu, \pi). \tag{2.22}$$

For each π, the inner minimum on the right is its *Bayes risk*. A prior attaining the outer maximum is called *least favorable*. Thus the optimal worst-case risk equals the largest Bayes risk over priors.

Remark 2.8.2 (Bayes optimality and minimax optimality). The uniform prior used in the boxes proof is least favorable. The equality of values does not imply that every Bayes rule for that prior is minimax; Exercise 28 asks you to establish both claims.

2.8.3 Performance across sampling laws

Let $\mathcal{Q}$ be the set of joint distributions of $(X_1, \ldots, X_n, X)$. Each $Q \in \mathcal{Q}$ is the same for every hidden target. Write $R_Q(\mathcal{A}, f)$ when the sampling law must be explicit, and define

$$V(Q) = \min_{\mathcal{A}} \max_{f \in \mathcal{F}} R_Q(\mathcal{A}, f) = \frac{1}{2} Q(U). \tag{2.23}$$

One can seek a learner minimizing the worst risk over both Q and f, or require a learner to attain $V(Q)$ separately for every Q. These are different requirements. Exercises 20 and 21 compare the corresponding benchmarks.

2.8.4 Choosing a criterion

A Bayesian model requires a prior; a minimax model requires a class of possible environments. Both require a loss, an observation protocol, and a class of allowed decisions. A broad environment class can make a worst-case guarantee conservative, while a prior can omit important possibilities. The choice should reflect the question and the information available.

Prefer minimax risk when an absolute performance guarantee is the governing objective; environment-dependent costs and common fallbacks can then affect the choice. Prefer *minimax regret* when the objective is to limit avoidable shortfall relative to available opportunities; the comparator set must then be specified carefully. If both considerations matter, examine both: small regret can coexist with large risk, and minimizing worst-case risk can tolerate avoidable losses in easier environments. Exercise 23 explores this choice through common fallbacks and changes to the available decisions.

Remark 2.8.3 (Which invariance properties should a criterion satisfy?). Should mixing every decision with the same fallback preserve their ranking? Should adding a decision that is not chosen preserve the choice among the original decisions? The calculations in Exercise 23 show that minimax risk and minimax regret respond differently. Which properties to insist on depends on the application: whether absolute losses or avoidable shortfalls matter, and what information the comparator should use.

2.9 Take-home messages

1. Machine learning moves part of the behavioral specification problem into data and feedback. Architecture, objectives, optimization, evaluation, and the surrounding program remain designed components.

2. Pretraining, post-training, and repeated evaluation supply different information. A capability should be attributed to the components that made it possible.

3. In the boxes problem, the training and test indices may have any joint distribution that is the same for every hidden function, with the learner's seed independent of the indices. The exact minimax risk is one half of the probability that the test index was not revealed.

4. Private randomization protects an unseen prediction against a fixed worst-case label. It creates no information about that label.

5. Generalization requires alignment: the learner and its observations must exploit relations that the target actually satisfies.

6. Finite input spaces can be astronomically large. Interpret asymptotic guarantees in the relevant regime, and inspect their explicit dependence on sample size, domain size, and the promised structure.

Endnotes

1. The prompts were selected to illustrate these behaviours. For a longer continuation, see GPT-2's fictional report about English-speaking unicorns, with invented scientists and quotations; that example was selected from ten generated samples. Sources for both examples are given in the bibliographical notes.

2. Only a tiny fraction of these sequences will take the form of sensible text. Even questions of the form "Is the integer with binary representation b divisible by three?" give $2^{300} > 10^{90}$ distinct meaningful inputs when b ranges over 300-bit strings. With an encoding using at most one token per binary digit, these questions fit comfortably within the context window.

3. **Function application.** We could write $(\mathcal{A}(S_f, W))(X)$ instead of $\mathcal{A}(S_f, W)(X)$ to emphasize that $\mathcal{A}(S_f, W)$ is evaluated first and the resulting function is then evaluated at X. We will follow the standard convention of omitting the extra parentheses. This is analogous to curried function application in programming languages.

4. **Deterministic minimax learners.** A fair coin attains the minimax value for every allowed sampling law. Some fixed sampling laws also admit deterministic minimax learners; Exercise 16 gives an example.

5. **The exact threshold tail.** The inclusion in Eq. (2.17) is actually an equality. Indeed, if the sample misses $[k, j_\epsilon]$, every observed positive index must exceed j_ϵ. Thus, when $q(N) > \epsilon$, the exact error-tail probability is
$$\mathbb{P}(M > \epsilon) = (1 - q(j_\epsilon))^n.$$
When $q(N) \le \epsilon$, this probability is zero.

Bibliographical notes

Pretraining examples. The base GPT-3 continuations discussed in the text appear in Figure 8 of Ouyang et al. (2022). The prompts were selected to illustrate those behaviours. GPT-2's fictional report about English-speaking unicorns appears in Table 13 of Radford et al. (2019); that example was selected from ten generated samples.

Counting possible inputs. For the comparison with the number of atoms in the observable universe, see Peter Norvig, *On the (Small) Number of Atoms in the Universe* (2016).

Probability. The conventions for random variables, vectors, and elements follow Lattimore and Szepesvári, *Bandit Algorithms*, Chapter 2. That chapter develops probability spaces, measurability, conditioning, and independence in greater generality. Appendix B collects the tools needed here.

Randomized algorithms. Randomized algorithms are a major subject in theoretical computer science. Their mathematical analysis typically assumes access to independent random bits. Software commonly uses pseudorandom generators, which deterministically expand a seed into a longer sequence; whether this adequately replaces ideal randomness depends on the application and assumptions. See Motwani and Raghavan, *Randomized Algorithms*, and Vadhan, *Pseudorandomness*.

Hypothesis classes, realizability, and ERM. Shalev-Shwartz and Ben-David's *Understanding Machine Learning: From Theory to Algorithms* (2014), Chapters 2–4, develops realizability, consistency, ERM, and finite-class guarantees. Remark 3.2 discusses proper versus improper learning; the terminology "hypothesis class" does not by itself restrict the learner's output. Section 5.2 separates approximation error from estimation error, clarifying why approaching the best predictor in a class need not give small absolute risk. Chapter 6 introduces VC dimension and the classical learnability characterization. The optimal-rate results below postdate that textbook.

VC dimension and optimal sample complexity. For a class of VC dimension d, Hanneke's *The Optimal Sample Complexity of PAC Learning* (2016) gives a realizable PAC learner using $O((d + \log(1/\delta))/\epsilon)$ observations. This removes the extra $\log(1/\epsilon)$ factor in the classical guarantee for arbitrary ERM learners. The algorithm combines consistent predictors by majority vote and may return a predictor outside the original class.

When improper learning helps. The distinction can affect the best achievable sample complexity, even without computational restrictions. Bousquet, Hanneke, Moran, and Zhivotovskiy's *Proper Learning, Helly Number, and an Optimal SVM Bound* (2020), Theorem 11, exhibits classes for which *every* proper learner needs an additional logarithmic factor. This is stronger than showing that some ERM choices are inefficient in their use of data.

The finite singleton class in its proof gives a boxes example: exactly one of N boxes is positive. A proper learner must designate one box as positive. If sampling is uniform over the other $N - 1$ boxes, reliable identification of the missing box requires order $N \log N$ observations. An improper learner can return the all-zero predictor until a positive label is observed, then return the identified singleton. For any P, this rule's error exceeds ϵ only if the positive box has mass greater than ϵ and is missed in every draw, an event of probability at most $e^{-n\epsilon}$. At $\epsilon = 1/(2(N-1))$ and fixed small failure probability, it therefore needs only order N observations. The gain comes from allowing a prediction that need not assert an unobserved positive label.

Transduction. Transduction can save labels when the unlabeled test inputs reveal structure that constrains their labels. Examples include graph-based label propagation, which encourages similar inputs to share labels, and transductive SVMs, which seek a large margin on labeled and unlabeled examples. In learning theory, the one-inclusion graph algorithm is central; its transductive analysis also yields inductive expected-risk guarantees. Label savings are not universal: for distribution-free, realizable binary classification with i.i.d. training and test samples and at least as many test as training examples, the optimal labeled-sample complexity has the same order in transduction and induction. See Tolstikhin and Lopez-Paz (2016).

Decision criteria. For axiomatic treatments, see Stoye's *Axioms for Minimax Regret Choice Correspondences* (2011), and Section 3.2 and Table 1 of his survey *New Perspectives on Statistical Decisions under Ambiguity*. These characterizations make explicit the tradeoffs between consistency requirements; they do not single out one criterion as universally preferable.

Glossary

This glossary collects learning and decision-theory terminology used in the notes. Basic probability terminology is defined in Appendix B and is omitted here.

Input; label; labeled example
An input x is an object on which a prediction is requested; its label y is the associated answer. A labeled example is an observed pair (x, y). In the boxes problem, x is a box index and $y = f(x)$ is its hidden bit.

Training sample
The observations supplied to the learner. Here $S_f = ((X_i, f(X_i)))_{i=1}^n$ is a tuple of labeled examples; indices may repeat, and independence is an additional assumption.

Test input
An input X at which a prediction is evaluated against its label $f(X)$. Its label is withheld for evaluation, but the same input and label may already have appeared in the training sample.

Sampling law
The joint distribution Q of $(X_1, \ldots, X_n, X)$. In the original boxes problem it may have dependence and repeated indices, but it does not depend on the hidden target f. Independent draws from a common distribution P are a special case; see Sections 2.3 and 2.6.

Target
The fixed, unknown function f that supplies the true labels. The noiseless boxes problem has $f : [N] \to \{0, 1\}$.

Predictor
A function g that assigns a predicted label to an input. A learned predictor $\widehat{f}$ depends on the training sample and possibly on the learner's random seed.

Learner
A procedure $\mathcal{A}$ that constructs a predictor from a training sample and private seed: $\widehat{f} = \mathcal{A}(S_f, W)$. The subsequent prediction at X is $\widehat{f}(X)$; see Eq. (2.2).

Generalization
Using training information to achieve low prediction error on test examples under a specified evaluation protocol. Test inputs need not be distinct from training inputs; the guarantees concern performance at test time.

Structural promise
An assumption restricting the possible targets, such as requiring all labels to agree or requiring them to form a threshold. Such restrictions can relate labels at different inputs.

Hypothesis class
A specified set $\mathcal{F}$ of candidate prediction functions. The class may be used to restrict the learner's search or to define a comparison benchmark; naming it alone does not restrict every learner's output to it.

Target class
The set of functions allowed as possible targets. The same set $\mathcal{F}$ may serve as both a target class and a hypothesis class, but these are different roles.

Realizability
In the noiseless setting here, the assumption $f \in \mathcal{F}$: the specified class contains the true labeling function. This assumption concerns the target, not the learner's choice of output.

Consistency
A predictor g is consistent with a sample $((x_i, y_i))_{i=1}^n$ if $g(x_i) = y_i$ for every i. This is a property of its predictions on the observed sample.

Empirical risk
Average loss on the observed training sample. Under 0–1 loss, for $s = ((x_i, y_i))_{i=1}^n$ with $n \geq 1$, it is $\widehat{L}_s(g) = n^{-1} \sum_{i=1}^n \mathbb{I}\{g(x_i) \neq y_i\}$.

Empirical risk minimization (ERM)
Choosing $g \in \arg\min_{h \in \mathcal{F}} \widehat{L}_s(h)$. Under realizability and noiseless labels, the minimum 0–1 empirical risk is zero, so every ERM output is consistent. Different choices among minimizers can have different generalization guarantees.

Proper; improper learning
A proper learner always returns a member of the class being learned; an improper learner may return a predictor outside it.

0–1 **loss**
The loss $\mathbb{I}\{\widehat{y} \neq y\}$ for predicting $\widehat{y}$ when the true label is y: zero for a correct prediction and one for an error.

Risk
Expected loss under the specified evaluation protocol. Here $R(\mathcal{A}, f) = \mathbb{E}[\mathbb{I}\{\mathcal{A}(S_f, W)(X) \neq f(X)\}]$ averages over the jointly sampled training and test indices and the learner's seed, with f fixed. Writing R_Q makes the sampling law explicit; see Eq. (2.4).

Error mass
For fixed g, f, and input distribution P, the probability assigned to prediction errors is $L_P(g, f) = \sum_{x=1}^N P(x)\mathbb{I}\{g(x) \neq f(x)\}$. For a learned $\widehat{f}$, the quantity $M = L_P(\widehat{f}, f)$ is random through the training data and any private randomization. If the test input has law P and is independent of both, then $R(\mathcal{A}, f) = \mathbb{E}[M]$, with the expectation over the training data and the learner's seed.

Unseen input
A test input whose index is absent from the training sample. In the boxes problem this is the event $U = \{X \notin \{X_1, \ldots, X_n\}\}$.

Threshold target
On the ordered inputs $[N]$, the function $f_k(x) = \mathbb{I}\{x \geq k\}$ for $k \in [N+1]$. The cases $k = 1$ and $k = N + 1$ give the all-one and all-zero functions, respectively; see Eq. (2.14).

Worst-case risk
For a fixed learner and sampling law Q, the value $\max_{f \in \mathcal{F}} R_Q(\mathcal{A}, f)$. The target is chosen to maximize expected loss, without seeing the learner's private random seed.

Minimax risk
The smallest worst-case risk among allowed learners, here $\min_{\mathcal{A}} \max_{f \in \mathcal{F}} R_Q(\mathcal{A}, f)$ for fixed Q. The target class and sampling assumptions are part of the criterion; maximizing over Q as well defines a different optimization problem.

Sample complexity
The smallest number of training observations for which an allowed learner can meet a specified guarantee uniformly over the stated targets and sampling distributions. For example, a high-probability guarantee requires $\mathbb{P}(M \leq \epsilon) \geq 1 - \delta$; an expected-risk guarantee instead bounds $\mathbb{E}[M]$ under the independent-test setup.

Private randomization
The learner's seed W is independent of the sampled indices, and the target is chosen without access to it. The learner may use W together with its observations to randomize its predictions.

Label noise
Randomness or corruption in observed labels, so that they need not equal the value of a fixed labeling function at the observed input.

Misspecification
Failure of the assumed model class to contain the data-generating mechanism. In the noiseless function setting, this means $f \notin \mathcal{F}$.

Agnostic learning
Learning with a guarantee relative to the best predictor in a specified class, without assuming that any member predicts perfectly. The comparison controls excess risk over that class's best risk, which may itself be large.

Algorithmic bias
Preferences built into a learning procedure, including its choices among predictors that fit the same observations.

Inductive learning
Constructing a predictor from training data before receiving the particular inputs on which it will be evaluated.

Transductive learning
Predicting labels for particular unlabeled test inputs that are supplied to the learner along with its training data. Which inputs are revealed, and how they are selected, are part of the protocol; see Exercise 14.

Prior
A specified distribution π on possible targets. In the Bayesian boxes model, F is drawn from π independently of the sampled indices and the learner's seed.

Posterior
Given jointly distributed random elements Y and V, if the distribution of Y is regarded as its prior distribution, then the conditional distribution of Y given $V = v$ is its posterior distribution given $V = v$. For the boxes model and a sample s of positive probability, its mass at f is $\mathbb{P}(F = f \mid S_F = s)$.

Bayes rule
A learner minimizing average risk under a specified prior and sampling law. It chooses predictions to minimize posterior expected loss at observations of positive probability. More than one learner may attain the minimum.

Bayes risk
The minimum prior-averaged risk, $\min_{\mathcal{A}} \mathbb{E}[R(\mathcal{A}, F)]$ with F drawn from the prior π. This expectation averages over targets; $R(\mathcal{A}, f)$ already averages over the sampled indices and private randomization.

Least-favorable prior
A prior maximizing Bayes risk among the allowed priors. In the finite decision problem here, its Bayes risk equals the minimax risk by Eq. (2.22). A Bayes rule for such a prior need not itself be minimax; see Remark 2.8.2.

Comparator; benchmark
A comparator is a reference learner; a benchmark is the reference performance value used to evaluate another learner. Two benchmarks here are $\min_{\mathcal{A}'} R_Q(\mathcal{A}', f)$ for a fixed (Q, f) and $V(Q) = \min_{\mathcal{A}'} \max_f R_Q(\mathcal{A}', f)$ for a fixed Q. The first permits choosing a comparator for the particular target; the second requires it to handle all targets. See Exercise 21.

Regret
Excess loss over an environment-specific benchmark: if the loss of decision $\mathcal{A}$ in environment θ is $\ell(\mathcal{A}, \theta)$ and the benchmark is $b(\theta)$, its regret is $\ell(\mathcal{A}, \theta) - b(\theta)$. The environment and benchmark must be specified as part of the criterion.

Minimax regret
The smallest worst-case regret among allowed decisions, $\min_{\mathcal{A}} \max_{\theta} \{\ell(\mathcal{A}, \theta) - b(\theta)\}$. It controls excess loss relative to the benchmark; a small regret can coexist with a large absolute risk. See Exercise 23.

2.10 Exercises

General instructions. Justify every answer, including yes/no answers, even when the question does not explicitly ask for a justification.

Exercise profiles. Filled mountains show difficulty on a four-mountain scale: one is a direct application, two uses a familiar method, three combines ideas, and four is a substantial challenge. Filled cogs show technical load on a three-cog scale: one is light, two is sustained, and three is dense. Filled hourglasses show estimated time: one is under 20 minutes, two is 20–45 minutes, three is 45–90 minutes, and four is more than 90 minutes. Pale pictograms show the unused part of each scale. The work row uses the canonical order proof, calculation, construction, computation, and interpretation; pale icons mark work types not used. ◎ marks the intended learning outcome. Time estimates cover the non-optional parts. Ratings are not prerequisites or grading weights. A superscript asterisk marks an exercise based on supplementary lecture material; it is not a difficulty rating and does not make the exercise optional when assigned.

Exercise 1: One random seed is enough.

◎ Construct a deterministic map driven by one uniform seed that realizes any prescribed finite output distribution.

For each training sample $s \in \mathcal{S} := ([N] \times \{0,1\})^m$, let Q_s be a probability distribution on $\mathcal{F} := \{0,1\}^{[N]}$.

(a) Let W be uniformly distributed on $[0,1]$. Construct a map $\mathcal{A} : \mathcal{S} \times [0,1] \to \mathcal{F}$ such that for every fixed $s \in \mathcal{S}$, $\mathcal{A}(s, W)$ has distribution Q_s.

(b) *Optional technical check.* Give the finite sets $\mathcal{S}$ and $\mathcal{F}$ their discrete σ-algebras and give $[0,1]$ its Borel σ-algebra. Show that $\mathcal{A}$ is measurable.

Exercise 2: A randomized learner as a deterministic map.

Represent an informal randomized prediction rule as a deterministic function of its data, input, and random seed.

Write the memorization-and-coin-flip learner explicitly in the form (2.2). Specify which values of W produce predictions 0 and 1 on an unseen input. On contradictory samples, define the map arbitrarily.

Exercise 3: Repeated experiments and risk.

Connect risk to the almost-sure long-run error frequency in repeated independent experiments.

Suppose the boxes experiment is repeated independently, using the same learner $\mathcal{A}$, target f, and sampling law each time. Let E_t be the indicator of an error on repetition t. State a law of large numbers for $T^{-1}\sum_{t=1}^{T} E_t$ as $T \to \infty$, specifying the mode of convergence and identifying the limit in terms of the learner's risk.

Exercise 4: The probability of an unseen input.

Derive the unseen-input probability and the exact sample-size threshold under uniform sampling.

Let $X_1, \ldots, X_m, X$ be independent draws from P on $[N]$.

(a) For $P(x) > 0$, compute $\mathbb{P}(U \mid X = x)$, writing out the factorization that uses independence.

(b) Apply the law of total probability to show that

$$\mathbb{P}(U) = \sum_{x=1}^{N} P(x)(1 - P(x))^m.$$

(c) For uniform sampling:

(i) Specialize to uniform P and show that the minimax risk is

$$\frac{1}{2}\left(1 - \frac{1}{N}\right)^m.$$

(ii) For $N \geq 2$, find the smallest integer m for which this risk is at most 0.1.

Exercise 5: Elementary sample-size bounds.

Use elementary exponential and product inequalities to obtain matching-order sample-size bounds for the uniform-boxes problem.

(a) Prove $1 - u \leq e^{-u}$ for $u \geq 0$. *Hint: differentiate* $e^{-u} - (1 - u)$.

(b) Prove *Bernoulli's inequality*, $(1-u)^m \geq 1 - mu$, for $u \in [0,1]$ and integers $m \geq 0$, by induction.

(c) Generalize part (b): for $p_1, \ldots, p_m \in [0,1]$, prove

$$\prod_{i=1}^{m}(1 - p_i) \geq 1 - \sum_{i=1}^{m} p_i.$$

Explain how this inequality follows from the union bound when p_i is the probability of an event and the events are independent.

(d) Use these inequalities to show that $m \geq N \log 5$ is sufficient, and $m \geq 0.8N$ is necessary, for the uniform-boxes minimax risk to be at most 0.1.

Completing the boxes proof

Exercise 6: The memorization upper bound.

Prove that randomized memorization attains error $\mathbb{P}(U)/2$ for every fixed target under an arbitrary joint sampling law.

Show that, for every fixed target f, randomized memorization satisfies

$$R(\mathcal{A}_{\text{mem}}, f) = \frac{1}{2}\mathbb{P}(U).$$

Use the corresponding intermediate steps and "why?" questions in Appendix A as a guide, and the probability tools in Appendix B as needed.

Exercise 7: The random-target lower bound.

Use an auxiliary random target to prove that every learner has worst-case risk at least $\mathbb{P}(U)/2$.

Show that every learner $\mathcal{A}$ satisfies

$$\max_f R(\mathcal{A}, f) \geq \frac{1}{2}\mathbb{P}(U).$$

Include the hidden-bit independence calculation on the event that the test input is unseen. Use the corresponding intermediate steps and "why?" questions in Appendix A as a guide, and the probability tools in Appendix B as needed.

Exercise 8: From a tail bound to expected error.

Convert a tail-probability bound into an expected-error bound using the tail-integral identity.

(a) Let Y take finitely many values in $[0, 1]$. Prove

$$Y = \int_0^1 \mathbb{I}(Y > t)\, dt, \qquad \mathbb{E}[Y] = \int_0^1 \mathbb{P}(Y > t)\, dt.$$

Justify exchanging the expectation and integral by writing the expectation as a finite sum.

(b) Suppose that $M \in [0, 1]$ satisfies $\mathbb{P}(M > t) \leq (1 - t)^m$ for every $t \in (0, 1)$. Apply the identity in part (a) to show that

$$\mathbb{E}[M] \leq \frac{1}{m + 1}.$$

(c) *Optional extension.*

(i) For any nonnegative random variable Y, prove $\mathbb{E}[Y] = \int_0^\infty \mathbb{P}(Y > t)\, dt$, allowing both sides to be infinite.

(ii) Then, for a real-valued Y with $\mathbb{E}[|Y|] < \infty$, prove

$$\mathbb{E}[Y] = \int_0^\infty \mathbb{P}(Y > t)\, dt - \int_0^\infty \mathbb{P}(Y < -t)\, dt.$$

Exercise 9: Which boxes should the data cover?

Determine how coverage of label-homogeneous groups, rather than repeated coverage within one group, controls minimax risk.

Let $N = 2n$, with $n \geq 2$. The boxes form two known groups of n boxes each. Each box has a label in $\{0, 1\}$. Labels are constant within each group, but the two groups may have different labels. The test index is uniform over all N boxes.

(a) Compute the minimax risk for each of two training samples: two distinct boxes from the first group, and one box from each group. Give an optimal learner in each case and prove that its worst-case risk cannot be improved. Allow randomized learners.

(b) Suppose the training sample reveals every box in the first group. Compute the minimax risk and compare it with the risks from observing just one box from that group and from observing one box from each group.

Sampling assumptions for threshold learning

Exercise 10: Error under arbitrary sampling.

Express threshold prediction error exactly as failure to observe the interval between the target and test indices.

Fix a threshold target f_k. Let $\mathcal{A}$ be the learner that returns f_K, where K is the smallest observed positive index, or $N + 1$ if none is observed. Allow any joint distribution of the training and test indices, as in the original boxes problem. Prove

$$R(\mathcal{A}, f_k) = \mathbb{P}\big(X \geq k,\ X_i \notin \{k, \dots, X\} \text{ for every } i\big).$$

Prove this by showing equality of the corresponding events, before taking probabilities.

Exercise 11: Conditional coverage and threshold error.

Derive a threshold-error tail bound under dependent sampling with conditional coverage.

Fix the target and learner from Exercise 10. Suppose the test index is an independent draw from a distribution P. The training indices may be dependent. Assume that, for some $\alpha \in (0, 1]$, every interval $I \subseteq [N]$ satisfies

$$\mathbb{P}(X_i \in I \mid X_1, \dots, X_{i-1}) \geq \alpha P(I) \qquad \text{almost surely}$$

for each i, with unconditional probability when $i = 1$. Let $L = P(\{k, \dots, K - 1\})$ be the learned predictor's error mass, where the set is empty if $K = k$.

(a) Adapt the proof of Proposition 2.7.1 to show, for $\epsilon \in (0, 1)$,

$$\mathbb{P}(L > \epsilon) \leq (1 - \alpha\epsilon)^m \leq e^{-\alpha m \epsilon}.$$

(b) Verify the conditional-coverage inequality for independent draws from P with $\alpha = 1$.

(c) What sample size suffices to ensure $L \leq \epsilon$ with probability at least $1 - \delta$, for $\delta \in (0, 1)$?

Hints For (a), bound the probability of missing a fixed interval at every step by conditioning successively on the previous indices.

Exercise 12: Dependent samples and matching marginals.

◎ Construct dependent samples that satisfy conditional coverage and show why matching marginals alone do not ensure improvement with sample size.

(a) Construct a sampling procedure whose training indices are not independent and verify the conditional-coverage inequality in Exercise 11 for some $\alpha \in (0, 1)$.

(b) Let $N = 2$, $f = f_1$, and $X_1 = \cdots = X_m = Z$ for $m \geq 1$, where Z is uniform on $\{1, 2\}$. Let the test index be independently uniform. Compute the learner's risk for every $m \geq 1$. Use this expression to determine whether increasing m reduces the risk in this example.

Explain why the first construction satisfies conditional coverage while the second has the correct marginal distribution for every training index but does not.

Exercise 13: Generalization by consistency.

◎ Prove the finite-class consistency bound and distinguish a sufficient counting bound from the sample size a particular learner actually needs.

Use the assumptions and notation of Theorem 2.7.4.

(a) For a predictor $g \in \mathcal{F}$, compute the probability that g is consistent with all m observations in terms of $L_P(g, f)$. Write the factorization that uses the sampling assumptions.

(b) Prove that every learner returning a consistent $\widehat{f} \in \mathcal{F}$ satisfies

$$\mathbb{P}\big(L_P(\widehat{f}, f) > \epsilon\big) \leq |\mathcal{F}| e^{-m\epsilon}, \qquad \epsilon \in (0, 1).$$

You may use the union bound: $\mathbb{P}(\bigcup_{j=1}^{r} E_j) \leq \sum_{j=1}^{r} \mathbb{P}(E_j)$ for events $E_1, \ldots, E_r$. Your proof should apply to every choice among consistent members, including randomized choices.

(c) Compare the guarantees:

(i) Derive the stated sample-size guarantee.

(ii) Specialize it to the class of thresholds and compare it with Proposition 2.7.1. Does this prove that the threshold learner needs more observations than the sufficient sample size in Proposition 2.7.1? Support your answer with the two bounds.

Induction and transduction

Exercise 14: One test input gives no transductive advantage.

◎ Convert a single-input transductive rule into an inductive predictor with identical risk.

An *inductive* learner uses the training sample to produce a prediction function for future inputs. A *transductive* learner also receives the particular unlabeled test inputs whose labels it must predict.

Consider the original boxes protocol with one test input X, and allow any known promise $f \in \mathcal{H} \subseteq \{0, 1\}^{[N]}$. A transductive learner $\mathcal{B}$ receives (s, x, w) and returns a bit. Show that there is an inductive learner $\mathcal{A}$ of the form Eq. (2.2) with exactly the same risk as $\mathcal{B}$ for every $f \in \mathcal{H}$. The training sample and its selection procedure are unchanged, and the learner's output function need not belong to $\mathcal{H}$.

Exercise 15: When unlabeled test inputs carry information.

Show how a label-dependent test-set promise can make unlabeled test indices informative.

Three boxes have nondecreasing binary labels. The training data reveal $f(1) = 0$ and $f(3) = 1$. The test set consists of box 2 and one other box, with the promise that their labels differ. Loss is the fraction of the two test labels predicted incorrectly. Allow randomized learners in both cases; the hidden labels are chosen before the learner's private random seed is drawn.

(a) Find the smallest worst-case expected loss when the learner must produce its prediction function before seeing the test set.

(b) Find the smallest worst-case expected loss when the learner receives both test indices before predicting their labels.

(c) Explain why this does not contradict Exercise 14.

Exercise 16: Nonuniqueness for a fixed sampling distribution.

Show by direct risk calculation that a deterministic minimax learner can differ from randomized memorization under a fixed sampling law.

Use the notation Q and $V(Q)$ from Section 2.8.3. Let $N = 2$ and $m = 1$, and let X_1 and X be independent and uniform on $\{1, 2\}$. Consider the following deterministic learner. Given a sample $((1, y))$, it returns the function g with $g(1) = y$ and $g(2) = 1 - y$. Given a sample $((2, y))$, it returns the function g with $g(2) = y$ and $g(1) = y$.

(a) Compute $Q(U)$ and $V(Q)$.

(b) Show that this learner has risk $1/4$ for every hidden function f.

(c) Give a sample and test index on which this learner and $\mathcal{A}_{\text{mem}}$ have different prediction distributions. Use this example to determine whether minimax learners are unique for this Q.

Decisions for one bit

Exercise 17: Bayes decisions for one bit.

Derive the Bayes-optimal randomized prediction for one observed or uncertain bit.

Fix a training sample s consistent with at least one hidden function, and a test index x. Let q be the learner's probability of predicting 1.

(a) If x appears in s, show that a zero-error rule must return its observed label with probability one.

(b) Suppose x is unseen and its conditional probability of label 1 is p under a specified prior. Derive the conditional expected loss as a function of p and q, then characterize its minimizing values of q for every p.

Exercise 18: Worst-case decisions for one bit.

Compare minimax and least-favorable-prior responses and determine how an adaptive adversary changes the value.

For an unknown label $b \in \{0, 1\}$, let q be the probability of predicting 1 and let $r(q, b)$ be the expected zero-one loss.

(a) Write the two-by-two loss matrix and solve

$$\min_{q\in[0,1]} \max_{b} r(q,b), \qquad \max_{p\in[0,1]} \min_{q\in[0,1]} \big((1-p)r(q,0) + p\, r(q,1)\big).$$

Identify all optimal q in the first problem and all optimal q for the least-favorable p in the second. Are these sets equal?

(b) Suppose an adversary chooses b after seeing the learner's predicted bit. Give an optimal strategy for this adversary, compute the resulting minimax value, and compare it with the first value in part (a).

Exercise 19: Two meanings of "the learner does not know Q."

Distinguish a distribution-free minimax guarantee from simultaneous minimax optimality for every sampling law.

Assume $N \geq 2$ and $m \geq 1$, and compare

$$\min_{\mathcal{A}} \max_{Q\in\mathcal{Q}} \max_{f} R_Q(\mathcal{A}, f) \tag{2.24}$$

with the requirement

$$\mathcal{A} \in \bigcap_{Q\in\mathcal{Q}} \arg\min_{\mathcal{B}} \max_{f} R_Q(\mathcal{B}, f). \tag{2.25}$$

(a) Show that the value of Eq. (2.24) is $1/2$.

(b) Show that a learner that always predicts with a fair coin is optimal for Eq. (2.24), even though it ignores revealed labels.

(c) Show that Eq. (2.25) forces the learner to return the revealed label at every seen test index and to make an unbiased prediction at every unseen test index, for every consistent labeled sample.

(d) Does the learner in part (b) satisfy Eq. (2.25)? Give a sampling law witnessing your answer.

Exercise 20: Expressing simultaneous optimality by minimax regret.

Express simultaneous law-specific minimax optimality as zero worst-case regret relative to $V(Q)$.

Consider the excess risk relative to the Q-specific minimax value:

$$\min_{\mathcal{A}} \max_{Q\in\mathcal{Q}} \left\{ \max_{f} R_Q(\mathcal{A}, f) - V(Q) \right\}. \tag{2.26}$$

(a) Show that the value of Eq. (2.26) is zero.

(b) Prove that a learner attains this value exactly when it is minimax optimal simultaneously for every Q.

(c) Using Eq. (2.23), characterize its prediction at every seen and unseen test index for every consistent labeled sample.

(d) Compute the worst-case regret of the learner that always predicts with a fair coin and determine whether it minimizes Eq. (2.26). Compare its optimality here with its optimality for Eq. (2.24).

Comparing regret criteria

Exercise 21: Oracle benchmarks and comparator information.

◎ Determine how the comparator's allowed information changes the meaning of minimax regret.

Assume $N \geq 2$ and $m \geq 1$. Compare Eq. (2.26) with

$$\min_{\mathcal{A}} \max_{Q \in \mathcal{Q},\, f \in \{0,1\}^{[N]}} \left\{ R_Q(\mathcal{A}, f) - \min_{\mathcal{A}'} R_Q(\mathcal{A}', f) \right\}.$$

(a) Compute the benchmark $\min_{\mathcal{A}'} R_Q(\mathcal{A}', f)$ for each (Q, f), and hence the value of this criterion. Give a comparator $\mathcal{A}'$ attaining the benchmark, specifying it as a map from a sample and seed to a predictor. The target f is not an input to that map.

(b) Both criteria have the form

$$\min_{\mathcal{A}} \max_{\theta} \left\{ \ell(\mathcal{A}, \theta) - \min_{\mathcal{A}'} \ell(\mathcal{A}', \theta) \right\}.$$

Identify the environment θ and loss ℓ in each case. What information can the choice of comparator depend on? Which unknown quantities must that comparator still handle uniformly?

Exercise 22: Fixed-law and distribution-free minimizers.

◎ Show why objectives with the same fixed-law minimizers can select different learners after maximizing over sampling laws.

Assume $N \geq 2$ and $m \geq 1$. Compare Eq. (2.26) with

$$\min_{\mathcal{A}} \max_{Q \in \mathcal{Q},\, f \in \{0,1\}^{[N]}} \left\{ R_Q(\mathcal{A}, f) - \min_{\mathcal{A}'} R_Q(\mathcal{A}', f) \right\}.$$

(a) Compare the minimizing learners:

(i) Show that, for a fixed Q, the two criteria have the same minimizing learners.

(ii) Does this remain true after maximizing over Q? For the learner that always predicts with a fair coin and for $\mathcal{A}_{\text{mem}}$, compute worst-case risk and both regrets under each of the following laws: $X_1 = \cdots = X_m = 1$ with $X = 1$, and with $X = 2$. Use these calculations to establish whether each learner minimizes each criterion.

(b) Which criterion would you use to measure performance relative to an oracle that knows the target? Which would you use to require an optimal worst-case guarantee for every sampling law? Identify the matching formula in each case.

Minimax risk and minimax regret

Exercise 23: Minimax decisions and mixtures.

◎ Compare the decisions and mixtures selected by minimax risk and minimax regret.

Consider two decisions with the following risks:

Decision	Environment 1	Environment 2
A	0.1	1.0
B	0.3	0.9

Here regret is measured relative to the smallest risk among the available decisions in each environment.

(a) In each case below, give the worst risk and worst regret at each optimizer and determine whether the criteria select the same decision.

(i) Find the minimax-risk and minimax-regret decisions when choosing between A and B.

(ii) Repeat when allowing randomization between them.

Exercise 24: A common fallback.

Prove that a common fallback preserves regret rankings but can reverse worst-risk rankings.

Consider two decisions with risks

$$R(A,1) = 0.1, \quad R(A,2) = 1.0, \qquad R(B,1) = 0.3, \quad R(B,2) = 0.9.$$

Suppose each available decision is implemented with probability $\lambda \in (0,1)$; otherwise the same fallback H is implemented. This random choice is independent of the environment. Thus

$$R_\lambda(A,\theta) = \lambda R(A,\theta) + (1-\lambda)R(H,\theta).$$

Determine how this transformation affects rankings by worst-case risk and by worst-case regret.

(a) Show that when the *whole menu* undergoes this transformation, regret rankings are unchanged.

(b) Give λ and the two risks of H for which the minimax-risk ranking of A and B reverses, and verify the reversal.

Exercise 25: Changing the decision menu.

Explain why adding an unselected decision can change minimax regret but not existing worst risks.

Consider two decisions with risks

$$R(A,1) = 0.1, \quad R(A,2) = 1.0, \qquad R(B,1) = 0.3, \quad R(B,2) = 0.9.$$

Here regret is measured relative to the smallest risk among the available decisions in each environment. Determine whether adding an available decision can change which of A and B is preferred under each criterion, even if the new decision is not chosen.

(a) Show that adding a decision cannot reverse the minimax-risk ranking of A and B.

(b) Compute the worst-case regret of each of A, B, and C after adding C, with risks 1.0 and 0.05. Identify the minimax-regret decision and compare it with the choice in Exercise 23(a)(i).

(c) Prove that adding a decision that improves neither environment's best achievable risk leaves all existing regrets unchanged.

Exercise 26: Probability rules and their notation.

Reconstruct the probability identities used throughout the course from the axioms, including zero-probability cases.

Use the definitions in Appendix B.

(a) Derive the following from the probability axioms:

(i) $\mathbb{P}(\emptyset) = 0$.

(ii) $\mathbb{P}(E^c) = 1 - \mathbb{P}(E)$ for every event E.

(iii) Monotonicity of probability.

(b) Events and indicators:
  (i) Write $\mathbb{P}(X = x, Y \in D)$ and $\mathbb{I}(X \neq Y)$ with their events expressed as subsets of Ω.
  (ii) Show that $\mathbb{I}(E)$ is a measurable real-valued function whenever E is an event.

(c) Expectations:
  (i) For finite-valued random variables with an arbitrary joint distribution, prove linearity of expectation directly from the finite-sum definition.
  (ii) Also prove $\mathbb{E}[\mathbb{I}(E)] = \mathbb{P}(E)$.

(d) For a finite partition, prove each rule below, accounting for zero-probability parts of the partition:
  (i) The multiplication rule.
  (ii) Total probability.
  (iii) Total expectation.

(e) Conditioning and independence:
  (i) Suppose T is independent of (V, W), C is determined by V, and $\mathbb{P}(C) > 0$. Prove, by writing conditional probabilities as ratios, that

$$\mathbb{P}(T \in D, W \in B \mid C) = \mathbb{P}(T \in D)\mathbb{P}(W \in B \mid C).$$

  (ii) Give an example where two independent random variables become dependent after conditioning on an event involving both of them.

Exercise 27: Products of expectations and probability bounds.

Derive expectation factorization, Markov's inequality, and the union bound directly from finite sums and indicator inequalities.

Use the probability and expectation rules in Appendix B. All random variables below take finitely many values.

(a) Let $X_1, \ldots, X_n$ be mutually independent, and let $g_1, \ldots, g_n$ be real-valued functions on their respective ranges. Prove directly from the finite-sum definition that

$$\mathbb{E}\left[\prod_{i=1}^{n} g_i(X_i)\right] = \prod_{i=1}^{n} \mathbb{E}[g_i(X_i)].$$

(b) Prove *Markov's inequality*: for $Y \geq 0$ and $a > 0$,

$$\mathbb{P}(Y \geq a) \leq \frac{\mathbb{E}[Y]}{a}.$$

(c) Prove the *union bound*: for events $E_1, \ldots, E_m$ on the same probability space,

$$\mathbb{P}\left(\bigcup_{j=1}^{m} E_j\right) \leq \sum_{j=1}^{m} \mathbb{P}(E_j).$$

Does this require independence of the events?

Hints For (b), compare $a\mathbb{I}\{Y \geq a\}$ with Y. For (c), compare the indicator of the union with the sum of the indicators.

Exercise 28: Bayes-optimal need not mean minimax.

◎ Distinguish Bayes optimality under a least-favorable prior from minimax optimality by constructing and analyzing a counterexample.

(a) Using Section 2.8, show that the uniform prior on hidden functions is least favorable in the boxes problem.

(b) When $\mathbb{P}(U) > 0$, construct a Bayes rule for this prior that is not minimax, and identify a target witnessing its failure.

(c) Show that $\mathcal{A}_{\text{mem}}$ is both Bayes-optimal and minimax.

(d) Does every Bayes rule for a least-favorable prior have to be minimax?

Exercise 29: Priors encode different relations.

◎ Show through conditional probabilities how a prior can encode either independent labels or a shared relation among labels.

Compare independent fair labels with a uniform prior on the two constant functions. For $x \neq j$ and $y \in \{0, 1\}$, compute $\mathbb{P}(F(x) = 1 \mid F(j) = y)$ under each prior. Under which prior does observing one label determine all the others?

A A detailed version of the proof of Theorem 2.4.1

For a result of this caliber, no one would write a proof as detailed as the one shown below in a research paper or an advanced textbook. Instead, one would see a few short hints from which the proof can be constructed. The compact proof in Section 2.5 has this form. The compact form is useful for remembering what is important about the proof.

This appendix writes out a detailed version of the proof. Being able to produce the details shown here from a "proof sketch" should be routine. However, when people do not have much experience with probability calculations, this is challenging. The amount of detail needed to convince oneself that a result holds depends on one's level of expertise. People with more expertise skip many steps. People with less expertise should not skip the steps. Skipping steps is dangerous: when doing that, one runs the risk of convincing oneself of something that is not even true. Yours truly is definitely not an exception to this. Hence, even if my final proof is short, I very often write out things to the level of detail shown here (or at least do this in my head).

The proof is written as a forward calculation. Each parenthetical "(why?)" marks a step that the reader should justify. The probability facts needed to answer these questions are developed separately in Appendix B; the answers are deliberately left out of the proof itself.

We prove the equality by bounding its left-hand side (LHS) both above and below by its right-hand side (RHS). For the upper bound, we analyze the particular learner $\mathcal{A}_{\mathrm{mem}}$. For the lower bound, we fix an arbitrary learner and show that some fixed hidden function must be hard for it.

Upper bound. We first show that the LHS is upper bounded by the RHS. First notice that

$$\min_{\mathcal{A}} \max_{f} R(\mathcal{A}, f) \le \max_{f} R(\mathcal{A}_{\mathrm{mem}}, f) \qquad \text{(why?)}.$$

Now fix some $f \in \{0,1\}^{[N]}$. We will show that $R(\mathcal{A}_{\mathrm{mem}}, f) = \mathbb{P}(U)/2$. Once this has been shown for every fixed f, the upper bound follows (why?).

Let

$$E_f = \{\mathcal{A}_{\mathrm{mem}}(S_f, W)(X) \neq f(X)\}$$

be the event that $\mathcal{A}_{\mathrm{mem}}$ makes an error. From the definition of R in Eq. (2.4),

$$R(\mathcal{A}_{\mathrm{mem}}, f) = \mathbb{P}(E_f) = \mathbb{P}(E_f, U^c) + \mathbb{P}(E_f, U) \qquad \text{(why?)}.$$

On U^c, the test index agrees with at least one training index: $X = X_J$ for some $J \in [n]$. The learner returns the revealed label $f(X_J) = f(X)$, so E_f cannot occur. Thus,

$$E_f \cap U^c = \emptyset, \qquad \mathbb{P}(E_f, U^c) = \mathbb{P}(\emptyset) = 0 \qquad \text{(why?)}.$$

On U, the learner predicts with the fair bit

$$B = \mathbb{I}\{W \le 1/2\}.$$

The error event on U is

$$\begin{aligned} E_f \cap U = &\big(\{B = 1\} \cap \{f(X) = 0\} \cap U\big) \\ &\cup \big(\{B = 0\} \cap \{f(X) = 1\} \cap U\big) \qquad \text{(why?)}. \end{aligned}$$

Therefore,

$$\begin{aligned}\mathbb{P}(E_f, U) &= \mathbb{P}(B=1, f(X)=0, U) + \mathbb{P}(B=0, f(X)=1, U) \\ &= \mathbb{P}(B=1)\mathbb{P}(f(X)=0, U) \\ &\quad + \mathbb{P}(B=0)\mathbb{P}(f(X)=1, U) && \text{(why?)} \\ &= \frac{1}{2}\mathbb{P}(f(X)=0, U) + \frac{1}{2}\mathbb{P}(f(X)=1, U) \\ &= \frac{1}{2}\mathbb{P}(U) && \text{(why?).}\end{aligned}$$

Combining the two terms gives

$$R(\mathcal{A}_{\mathrm{mem}}, f) = \frac{1}{2}\mathbb{P}(U),$$

finishing the upper-bound proof.

Lower bound. We now turn to showing that the LHS is lower bounded by the RHS. Fix an arbitrary randomized learner $\mathcal{A}$. It suffices to prove that $\max_{f\in\{0,1\}^{[N]}} R(\mathcal{A}, f) \geq \mathbb{P}(U)/2$ (why?). Introduce an auxiliary random function

$$F : [N] \longrightarrow \{0,1\}$$

by choosing the bits $F(1), \ldots, F(N)$ independently, with

$$\mathbb{P}(F(j)=0) = \mathbb{P}(F(j)=1) = \frac{1}{2}, \qquad j \in [N].$$

Choose F independently of the index tuple $(X, X_1, \ldots, X_n)$ and the learner's seed W. Set

$$\widehat{Y} = \mathcal{A}(S_F, W)(X), \qquad Z = (X, X_1, \ldots, X_n).$$

We then have

$$\begin{aligned}\max_f R(\mathcal{A}, f) &\geq \mathbb{E}\big[R(\mathcal{A}, F)\big] && \text{(why?)} \\ &= \mathbb{P}\big(\widehat{Y} \neq F(X)\big) && \text{(why?)} \\ &\geq \mathbb{P}\big(\widehat{Y} \neq F(X), U\big) && \text{(why?).}\end{aligned}$$

To continue from the last term, let

$$\mathcal{Z} = \big\{z = (x, x_1, \ldots, x_n) \in [N]^{n+1} : x \notin \{x_1, \ldots, x_n\}\big\}.$$

Fix $z = (x, x_1, \ldots, x_n) \in \mathcal{Z}$, a possible sample $s \in ([N] \times \{0,1\})^n$, and $a, b \in \{0,1\}$. On the event $\{Z = z, S_F = s\}$, we have $\widehat{Y} = \mathcal{A}(s, W)(x)$. Moreover, this event is determined by Z and coordinates $F(j)$ with $j \neq x$ (why?). Since $F(x)$ is a fair bit independent of those quantities and of W,

$$\begin{aligned}&\mathbb{P}\big(\widehat{Y} = a, F(x) = b, Z = z, S_F = s\big) \\ &\quad = \frac{1}{2}\mathbb{P}\big(\widehat{Y} = a, Z = z, S_F = s\big) \qquad \text{(why?).}\end{aligned} \tag{2.27}$$

The error event on $\{Z = z, S_F = s\}$ is the disjoint union of the cases $(\widehat{Y}, F(x)) = (1, 0)$ and $(0, 1)$. Therefore,

$$\begin{aligned}\mathbb{P}\big(\widehat{Y} \neq F(X), U\big) &= \frac{1}{2}\sum_{z\in\mathcal{Z}}\sum_{s}\Big[\mathbb{P}(\widehat{Y} = 1, Z = z, S_F = s) + \mathbb{P}(\widehat{Y} = 0, Z = z, S_F = s)\Big] \\ &= \frac{1}{2}\sum_{z\in\mathcal{Z}}\sum_{s}\mathbb{P}(S_F = s, Z = z) \\ &= \frac{1}{2}\sum_{z\in\mathcal{Z}}\mathbb{P}(Z = z) \qquad \text{(why?)} \\ &= \frac{1}{2}\mathbb{P}(U) \qquad \text{(why?)}.\end{aligned}$$

Combining this calculation with the preceding forward calculation gives $\max_f R(\mathcal{A}, f) \geq \mathbb{P}(U)/2$, which, by our earlier remarks, suffices to establish the lower bound.

B Probability tools

This appendix states the definitions and general rules used in these notes. Their applications to the boxes proof are left to Exercises 6 and 7; elementary derivations are collected in Exercise 26. Appendix A supplies intermediate steps for readers who need them.

B.1 Probability spaces, events, and notation

A *probability space* is a triple $(\Omega, \mathcal{G}, \mathbb{P})$. The set Ω contains the possible outcomes ω. Its *events* are the members of $\mathcal{G}$, a collection containing Ω and closed under complements and countable unions (a *σ-algebra*). The *probability measure* $\mathbb{P} : \mathcal{G} \to [0, 1]$ satisfies $\mathbb{P}(\Omega) = 1$ and

$$\mathbb{P}\left(\bigcup_{i=1}^{\infty} E_i\right) = \sum_{i=1}^{\infty}\mathbb{P}(E_i) \quad \text{for pairwise disjoint events } E_i.$$

In particular, $\mathbb{P}(\emptyset) = 0$, $\mathbb{P}(E^c) = 1 - \mathbb{P}(E)$, and $E \subseteq G$ implies $\mathbb{P}(E) \leq \mathbb{P}(G)$. On a finite outcome space, we can take every subset to be an event. For a continuous seed, measurability specifies which subsets have assigned probabilities.

Several abbreviations suppress events or outcomes. In particular,

$$\begin{aligned}\mathbb{P}(X = x) &= \mathbb{P}(\{\omega \in \Omega : X(\omega) = x\}), \\ \mathbb{P}(X \in D, Y \in B) &= \mathbb{P}(\{\omega : X(\omega) \in D,\ Y(\omega) \in B\}), \\ \mathbb{P}(E, G) &= \mathbb{P}(E \cap G).\end{aligned}$$

The braces around an event and the outcome argument ω are often omitted. These are abbreviations with precise meanings. If an expression seems unclear, expand it and check the domains of its functions; the shorthand does not permit arbitrary operations.

All random quantities in an argument are jointly defined on the stated probability space unless a different probability measure is explicitly introduced. Thus every occurrence of $\mathbb{P}$ already has a single meaning. A sentence saying that a probability is "over" one source of randomness cannot supply missing semantics and should not be needed. When an argument changes the probability measure, takes a conditional probability, or fixes a formerly random quantity, that change must appear in the mathematical setup or notation before the probability is used.

B.2 Random variables, random elements, and indicators

A *random variable* is a measurable function $X : \Omega \to \mathbb{R}$: for each Borel set $D \subseteq \mathbb{R}$, the inverse image $\{\omega : X(\omega) \in D\}$ is an event. *Borel sets* form the σ-algebra generated by open intervals. A *random vector* takes values in $\mathbb{R}^d$. A *random element* can take values in a more general measurable space, with the same inverse-image requirement. In this note, the random function F is an element of a finite set of functions; its coordinate values $F(j)$ are real-valued random variables.

We generally use uppercase letters for random variables and lowercase letters for fixed quantities. The predictor f_K is random because K depends on the training sample. The expression $X = j$ defines an event; it does not replace the random variable X by a constant.

For an event E, its *indicator* $\mathbb{I}(E)$ is the function $\Omega \to \{0, 1\}$ given by

$$\big(\mathbb{I}(E)\big)(\omega) = \begin{cases} 1, & \omega \in E, \\ 0, & \omega \notin E. \end{cases}$$

Both $\mathbb{I}(E)$ and $\mathbb{I}_E$ are used for this function; we use the event as an argument. We also write $\mathbb{I}\{E\}$, and omit the outcome argument when working with the indicator as a random variable. For example, $\mathbb{I}(X \neq Y)$ denotes the indicator of $\{\omega : X(\omega) \neq Y(\omega)\}$. Its measurability follows from E being an event.

For a coin toss, let $\Omega = \{H, T\}$ and $E = \{H\}$. Then $\mathbb{I}(E)$ assigns 1 to outcome H and 0 to outcome T. It is a function on the outcome space, not the probability $\mathbb{P}(E)$.

For a measurable function g, the notation $Y = g(Z)$ means $Y(\omega) = g(Z(\omega))$. For any measurable set D,

$$\{Y \in D\} = \{Z \in g^{-1}(D)\}.$$

Thus the events involving Y are determined by Z. This dependence on Z is how applying a function to a random element produces another random element.

B.3 Distributions and expectation

The *law* of a random element Z assigns probability $P_Z(D) = \mathbb{P}(Z \in D)$ to each measurable set D. It is a measure on the space of values of Z, whereas $\mathbb{P}$ is a measure on Ω. Here, P denotes an input distribution on $[N]$ and Q a joint law of the indices. For a finite space we abbreviate $P(\{x\})$ as $P(x)$. We often specify a joint law without explicitly constructing Ω.

For a finite-valued real random variable Y, its *expectation* is

$$\mathbb{E}[Y] = \sum_{y \in \mathrm{supp}(Y)} y\mathbb{P}(Y = y), \qquad \mathrm{supp}(Y) = \{y : \mathbb{P}(Y = y) > 0\}.$$

For a finite-valued random element Z and a real-valued function h,

$$\mathbb{E}[h(Z)] = \sum_z h(z)\mathbb{P}(Z = z).$$

The same notation $\mathbb{E}$ denotes expectation for integrable random variables that are not finite-valued; its general definition is an integral. All expectations below involve bounded random variables.

The rules used in these notes include

$$\mathbb{E}[aY + bV] = a\mathbb{E}[Y] + b\mathbb{E}[V],$$
$$Y \leq V \text{ almost surely } \implies \mathbb{E}[Y] \leq \mathbb{E}[V],$$
$$\mathbb{E}[\mathbb{I}(E)] = \mathbb{P}(E).$$

Here a, b are constants. Linearity requires no independence. An inequality holding *almost surely* can fail only on an event of probability zero. For finitely many real numbers a_j and any random index J among them,

$$\mathbb{E}[a_J] \le \max_j a_j.$$

This follows by applying monotonicity to $a_J \le \max_j a_j$.

B.4 Conditioning and averaging

For events E, C with $\mathbb{P}(C) > 0$, define the *conditional probability*

$$\mathbb{P}(E \mid C) = \frac{\mathbb{P}(E \cap C)}{\mathbb{P}(C)}.$$

It follows that $\mathbb{P}(E \cap C) = \mathbb{P}(C)\mathbb{P}(E \mid C)$. The notation $\mathbb{P}(E \mid Z = z, S = s)$ conditions on the event $\{Z = z\} \cap \{S = s\}$. If that event has probability zero, the ratio is undefined. In sums, we can omit its term. The main proof instead assigns an arbitrary conditional probability measure on such events; its value has no effect because the corresponding weight is zero.

Events $C_1, \dots, C_r$ form a *partition* if they are pairwise disjoint and their union is Ω. *Total probability* gives

$$\mathbb{P}(E) = \sum_{j=1}^{r} \mathbb{P}(E \cap C_j) = \sum_{j:\mathbb{P}(C_j)>0} \mathbb{P}(C_j)\mathbb{P}(E \mid C_j).$$

In particular, for finite-valued Z, S,

$$\mathbb{P}(E) = \sum_{z:\mathbb{P}(Z=z)>0} \mathbb{P}(Z = z)\mathbb{P}(E \mid Z = z),$$

$$\mathbb{P}(E) = \sum_{(z,s):\mathbb{P}(Z=z,S=s)>0} \mathbb{P}(Z = z, S = s)\mathbb{P}(E \mid Z = z, S = s).$$

Applying the multiplication rule successively also yields

$$\mathbb{P}\left(\bigcap_{i=1}^{n} E_i\right) = \prod_{i=1}^{n} \mathbb{P}\left(E_i \,\middle|\, \bigcap_{j<i} E_j\right),$$

when the conditioning events have positive probability. If an earlier intersection has probability zero, the full intersection also has probability zero and the calculation can stop there.

For bounded Y and $\mathbb{P}(C) > 0$, define the *conditional expectation*

$$\mathbb{E}[Y \mid C] = \frac{\mathbb{E}[Y\mathbb{I}(C)]}{\mathbb{P}(C)}.$$

If Y is finite-valued this equals $\sum_y y\mathbb{P}(Y = y \mid C)$. For a finite-valued Z, the expression $\mathbb{E}[Y \mid Z]$ is a random variable: it takes value $\mathbb{E}[Y \mid Z = z]$ on each event $\{Z = z\}$ of positive probability; its values on zero-probability events can be assigned arbitrarily. Similarly, $\mathbb{P}(E \mid Z) = \mathbb{E}[\mathbb{I}(E) \mid Z]$. *Total expectation* and its iterated form are

$$\mathbb{E}[Y] = \mathbb{E}[\mathbb{E}[Y \mid Z]], \qquad \mathbb{E}[\mathbb{E}[Y \mid Z, S] \mid Z] = \mathbb{E}[Y \mid Z] \quad \text{almost surely.}$$

If $h(Z)$ is bounded, then

$$\mathbb{E}[h(Z)Y \mid Z] = h(Z)\mathbb{E}[Y \mid Z] \quad \text{almost surely.}$$

These identities permit successive conditioning on the training history.

B.5 Independence and functions of observations

Random elements Y, Z are *independent* if, for every pair of measurable sets,

$$\mathbb{P}(Y \in D, Z \in B) = \mathbb{P}(Y \in D)\mathbb{P}(Z \in B).$$

For finite-valued elements it suffices to check all pairs of individual values. *Mutual independence* of several elements requires the corresponding factorization for joint events; pairwise independence alone is insufficient. Measurable functions of independent elements remain independent. In particular, if Y is independent of Z, it is independent of every event determined by Z.

Exercise 27 derives the corresponding factorization of expectations of products, along with Markov's inequality and the union bound.

Conditioning can destroy independence. A useful sufficient condition for preserving it is the following: suppose T is independent of the pair (V, W), and C is an event determined by V, with $\mathbb{P}(C) > 0$. Then

$$\mathbb{P}(T \in D, W \in B \mid C) = \mathbb{P}(T \in D)\mathbb{P}(W \in B \mid C). \tag{2.28}$$

Thus T keeps its law and is independent of W under this conditioning. For mutually independent coordinates, each coordinate is independent of the tuple of all the others. These general facts are the tools for the unseen-bit calculation in Exercise 7.

B.6 Tails and repeated experiments

For $0 \le Y \le 1$, the *tail-integral identity* is

$$\mathbb{E}[Y] = \int_0^1 \mathbb{P}(Y > t)\, dt. \tag{2.29}$$

Exercise 8 derives the identity for finite-valued Y and applies it to the threshold learner.

If $E_1, E_2, \ldots$ are independent Bernoulli random variables with the same mean p, the *strong law of large numbers* states

$$\frac{1}{T}\sum_{t=1}^{T} E_t \longrightarrow p \qquad \text{almost surely.}$$

This means the set of infinite outcome sequences on which convergence holds has probability one. Exercise 3 connects the statement to the risk of a learner.

References

[Aden-Ali et al.(2022)] Ishaq Aden-Ali, Yeshwanth Cherapanamjeri, Abhishek Shetty, and Nikita Zhivotovskiy. The one-inclusion graph algorithm is not always optimal. arXiv:2212.09270, 2022. https://arxiv.org/abs/2212.09270.

[Bousquet et al.(2020)] Olivier Bousquet, Steve Hanneke, Shay Moran, and Nikita Zhivotovskiy. Proper learning, Helly number, and an optimal SVM bound. In *Proceedings of the 33rd Conference on Learning Theory*, volume 125 of PMLR, pages 582–609, 2020. https://proceedings.mlr.press/v125/bousquet20a.html.

[Hanneke(2016)] Steve Hanneke. The optimal sample complexity of PAC learning. *Journal of Machine Learning Research*, 17(38):1–15, 2016. https://jmlr.org/papers/v17/15-389.html.

[Joachims(1999)] Thorsten Joachims. Transductive inference for text classification using support vector machines. In *Proceedings of the 16th International Conference on Machine Learning*, pages 200–209, 1999. https://www.cs.cornell.edu/people/tj/publications/joachims_99c.pdf.

[Lattimore and Szepesvári(2020)] Tor Lattimore and Csaba Szepesvári. *Bandit Algorithms*. Cambridge University Press, 2020. https://tor-lattimore.com/downloads/book/book.pdf.

[Motwani and Raghavan(1995)] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[Norvig(2016)] Peter Norvig. On the (small) number of atoms in the universe. Web essay, 2016. https://www.norvig.com/atoms.html.

[Ouyang et al.(2022)] Long Ouyang et al. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems 35*, pages 27730–27744, 2022. https://arxiv.org/abs/2203.02155.

[Radford et al.(2019)] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. OpenAI technical report, 2019. https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

[Shalev-Shwartz and Ben-David(2014)] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.

[Stoye(2011)] Jörg Stoye. Axioms for minimax regret choice correspondences. *Journal of Economic Theory*, 146(6):2226–2251, 2011. https://doi.org/10.1016/j.jet.2011.10.004.

[Stoye(2012)] Jörg Stoye. New perspectives on statistical decisions under ambiguity. *Annual Review of Economics*, 4:257–282, 2012. https://doi.org/10.1146/annurev-economics-080511-110959.

[Tolstikhin and Lopez-Paz(2016)] Ilya Tolstikhin and David Lopez-Paz. Minimax lower bounds for realizable transductive classification. arXiv:1602.03027, 2016. https://arxiv.org/abs/1602.03027.

[Vadhan(2012)] Salil P. Vadhan. Pseudorandomness. *Foundations and Trends in Theoretical Computer Science*, 7(1–3):1–336, 2012.

[Zhu et al.(2003)] Xiaojin Zhu, Zoubin Ghahramani, and John Lafferty. Semi-supervised learning using Gaussian fields and harmonic functions. In *Proceedings of the 20th International Conference on Machine Learning*, pages 912–919, 2003. https://www.cs.cmu.edu/~zhuxj/pub/zgl.pdf.