— 2 —
Populations and predictions

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

Source: The Emerging Science of Machine Learning Benchmarks. M. Hardt, 2025. URL: https://mlbenchmarks.org. Compiled on 2025-05-01.

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

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

What was radical about the idea when it came up is that human populations clearly are not probability distributions. In a sense, they couldn't be further. The world we live in is a complex, interconnected dynamical system of people, institutions, infrastructure, and nature. What we observe changes over time and in response to our actions. What we see even changes in response to the models that we deploy in the world. It's important to keep this in mind to understand why sometimes things fail. Datasets collected at different times or different locations generally don't follow the same distributions. Machine learners call this *distribution shift*, but the expression starts from a wrong premise. There is no distribution to begin with. The dynamic nature of society contradicts the stationarity of the astronomical conception. When things go wrong in machine learning, it's often fundamentally this contradiction in disguise.

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

What makes the astronomical conception so useful is that it gives us a simple way to make predictions. A prediction is an educated guess about something we don't know for sure given information we do have. Bernoulli

Working draft available at https://mlbenchmarks.org

called it the *art of conjecturing*. In this sense, prediction is not just about the future. It also applies to things that have already happened, but we're uncertain about them. A camera took an image, but we're unsure from the pixels whether we're looking at a mouse or a hamster. A company filed their quarterly report with the Securities and Exchange Commission, but we're unsure from the numbers if financial fraud has occurred. In both cases, there's a definitive truth value about which we're unsure.

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

The example makes it clear that your best guess is specific to a population. It depends on whether you consider the population of all citizens of the United States of America or the population of Tokyo residents. It also depends on time. Your best guess in 2025 may not be your best guess in 1998.

2.1 Prediction

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

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

Working draft available at https://mlbenchmarks.org

There are two common settings of prediction:

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

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

- Given an English sentence, classify the sentiment of the sentence as *positive* or *negative*.
- Given an array of pixels, classify if the pixels make up an image of a *hamster* or the image of a *mouse*.
- Given a time series of meteorological measurements, predict the atmospheric pressure one hour from the last reading.
- Given features of a user and a video, predict the time the user is going to spend watching the video.

Predicted watch time is an interesting example. Ultimately, the goal is not just to predict watch time for the prediction's sake. There's always some other purpose with prediction. In this case, a video streaming platform might recommend content to a user in descending order of predicted watch time. The video of highest predicted watch time shows up first, the one with second highest predicted watch time comes next, and so forth. Naturally, the user is most likely to click on the item that shows up first on screen and much less likely to click on a lower ranked video. The result is a kind of *self-fulfilling prophecy*: The platform predicts high watch time and therefore displays the video prominently to the user, who is in turn more likely to watch the video. Formally, the prediction influences the target variable. This kind of dynamic is hard to avoid when applying prediction in real world systems.

Our formal setup, however, excludes any such dynamic. The reason is that we assume a joint distribution over labeled data points (x, y). This means

Working draft available at https://mlbenchmarks.org

that the target y is determined at the same time as the features x. Nothing we do based on x, like making a prediction, could possibly influence y in this formal setup. This is a fundamental problem with the astronomical conception. Predictions in the social world typically influence the outcome that they try to predict, a phenomenon called *performativity*. But for now, we continue with the astronomical conception.

If you think through real world applications of prediction, you'll likely find that almost none of them are perfectly clean examples of the formalism.

Optimal predictors

A good question to start with is what kind of predictor we would ideally like to have if we had full knowledge about the population. This requires that we formally pin down a mathematical objective. In classification, where the set $\mathcal{Y}$ is discrete, a natural objective is to maximize the probability of a correct guess. This objective corresponds to maximizing the *accuracy* of the predictor.

Definition 1. *The* accuracy *of a predictor f on a population (X, Y) is the probability $\mathbb{P}\{f(X) = Y\}$. The* classification error *is $\mathbb{P}\{f(X) \neq Y\}$.*

Formally, the goal is to find the accuracy maximizing predictor on the population (X, Y). That is, we want to solve the optimization problem:

$$\max_{f\colon \mathcal{X} \to \mathcal{Y}} \quad \mathbb{P}\{f(X) = Y\}$$

The maximum runs over all possible functions with no constraints on the kind of function whatsoever. The lack of any constraints is what gives this problem a clean and intuitive solution mathematically. On input x, the optimal predictor picks the label that's most likely to match the value of Y conditional on $X = x$.

Proposition 1. *Given a discrete population (X, Y), the accuracy maximizing predictor is given by any function f^* that satisfies*

$$f^*(x) = \arg\max_{y \in \mathcal{Y}} \mathbb{P}\{Y = y \mid X = x\}$$

for all inputs x with $\mathbb{P}(X = x) > 0$.

Proof. For any function $f\colon \mathcal{X} \to \mathcal{Y}$, rewrite accuracy as

$$\mathbb{P}\{Y = f(X)\} = \mathbb{E}\left[\mathbb{P}\{Y = f(x) \mid X = x\}\right],$$

Working draft available at https://mlbenchmarks.org

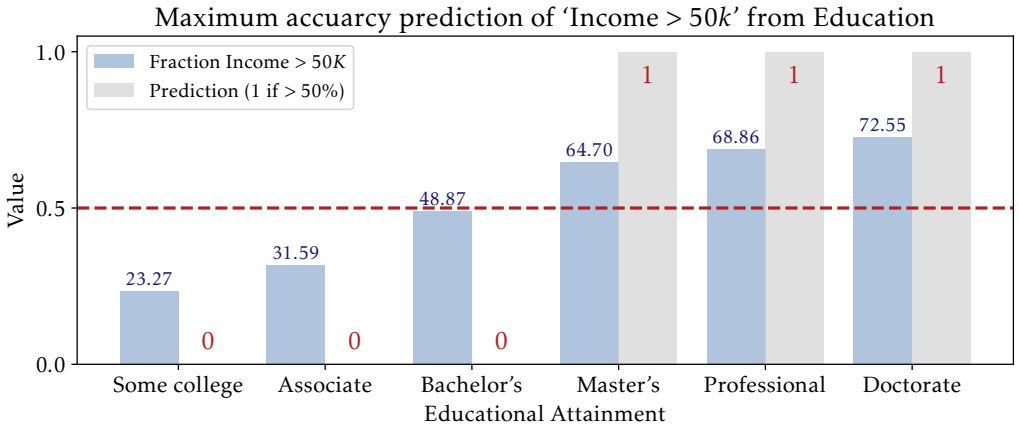

Figure 2.1: Accuracy maximizing prediction of (Income >50k) on the California U.S. Census population from 2018

where the expectation is taken over a random draw $x \sim X$ from the marginal X. Since any such x is in the support of X and X is discrete, we must have $\mathbb{P}\{X = x\} > 0$. Therefore, the conditional probability $\mathbb{P}\{Y = f(x) \mid X = x\}$ is well-defined.

Note that the expectation is a sum of non-negative terms. Since f is unconstrained, we can maximize the expectation pointwise for each $x \in \mathcal{X}$. For a fixed $x \in \mathcal{X}$ in the support of X, by assumption, $f^*(x)$ maximizes the expression $\mathbb{P}\{Y = f(x) \mid X = x\}$.

□

This matches our earlier intuition: The accuracy maximizing predictor outputs the most likely label given the available information. The optimal predictor isn't unique, since we get to break ties arbitrarily and the function value on inputs outside the support of X is arbitrary.

The optimal predictor generalizes to continuous populations and the intuition is the same. When the population (X, Y) has a joint probability density function $p(x, y)$, standard mathematical arguments give us a conditional density $p(y|x)$ that's defined almost everywhere in x. Formally, this requires some tedious measure-theoretic maneuvers that researchers in machine learning typically skip.

Stated this way, accuracy maximizing predictors pointwise maximize the

Working draft available at https://mlbenchmarks.org

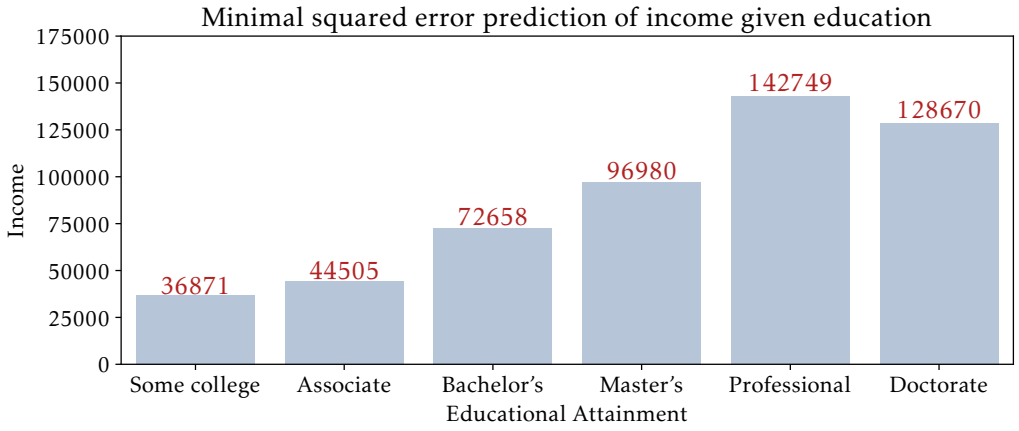

Figure 2.2: Mean squared error minimizing prediction

conditional density:

$$f^*(x) = \arg\max_y p(y|x).$$

Some call the expression $p(y|x)$ the *posterior* probability of the class y given the data x.

The equation characterizes the accuracy maximizing predictor at the population level. It does not give us an efficient algorithm for finding the best predictor. It's a characterization of optimality with full knowledge of the population. Much of learning theory is about the algorithmic question of finding good predictors when we don't have full knowledge of the population. Understanding what the best is that we can hope for, however, is an important first step. For now, we stay at the population level.

Regression

In regression, the label set $\mathcal{Y} = \mathbb{R}$ is the real line. The goal is to approximate the real-valued label rather than to match a discrete label exactly. A natural objective minimizes the mean squared difference between our prediction and the label.

Definition 2. *The* mean squared error *of a function $f : \mathcal{X} \to \mathcal{Y}$ on the population (X, Y) is the expected squared difference $\frac{1}{2}\mathbb{E}\left[(f(X) - Y)^2\right]$.*

We can again ask what predictor minimizes mean squared error. This

Working draft available at https://mlbenchmarks.org

turns out to be the *regression function*

$$r^*(x) = \mathbb{E}[Y \mid X = x]$$

that outputs the mean value of Y conditional on $X = x$.

Proposition 2. *The regression function* $r^*(x) = \mathbb{E}[Y \mid X = x]$ *minimizes mean squared error on the population* (X, Y).

Proof. The proof follows along the same lines as the argument for the accuracy maximizing predictor. We can minimize mean squared error for a fixed setting $X = x$ of the features:

$$\min_y \mathbb{E}\Big[(y - Y)^2 \mid X = x\Big]$$

Taking the derivative with respect to y and setting it to 0,

$$y = \mathbb{E}[Y \mid X = x],$$

after simplifying the expression using the linearity of expectation.

□

Note that in binary prediction, where $\mathcal{Y} = \{0, 1\}$, we can write the accuracy maximizing predictor as

$$f^*(x) = \mathbf{1}\Big\{\mathbb{E}[Y \mid X = x] > 1/2\Big\}, = \mathbf{1}\{r(x) > 1/2\},$$

where $\mathbf{1}\{\cdot\}$ is the indicator function. What this means is that the accuracy maximizing predictor is a rounding of the regression function.

Calibration

Calibration is an important property of regression functions. Restricting our attention to binary outcome variables,

Definition 3. *Say that a function* $f : \mathcal{X} \to [0, 1]$ *is* calibrated *with respect to a target variable* $Y \in \{0, 1\}$ *if for all values* $p \in [0, 1]$ *with* $\mathbb{P}\{f(X) = p\} > 0$ *we have*

$$\mathbb{P}\{Y = 1 \mid f(X) = p\} = p\,.$$

This condition means that the set of all instances assigned the value p has a p fraction of positive instances in it. Calibration expresses uncertainty

Working draft available at https://mlbenchmarks.org

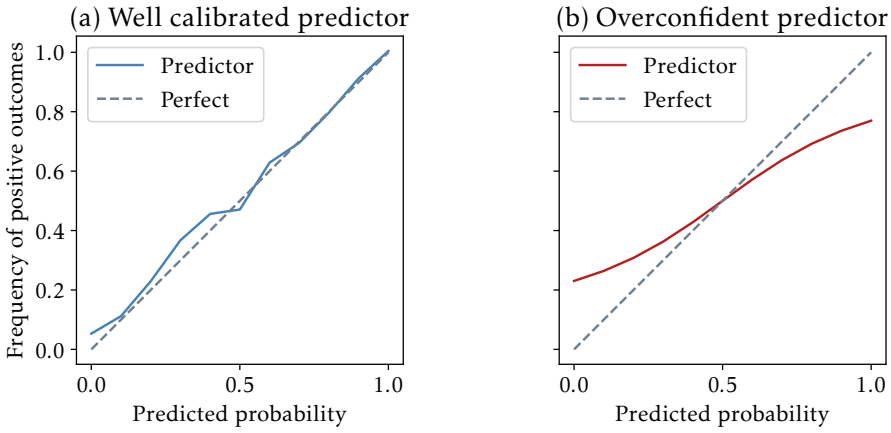

Figure 2.3: Calibration plots showing a well calibrated predictor (left) and a poorly calibrated predictor (right)

in a basic way. Given that we observe the prediction p, we know that on average over all instances with this prediction, there is a p fraction of positive outcomes. If we see a prediction of 0.54, we learn that there is significant uncertainty about the outcome. It's essentially a coin toss. In a sense, we can interpret a calibrated prediction as a probability. It's the probability of a positive outcome given the prediction.

Calibration has nothing to do with accuracy. To see this, note that the constant predictor $f(x) = p_1$ with $p_1 = \mathbb{P}\{Y = 1\}$ is always calibrated. In this example, all instances get the same score value p_1 and this score value by definition corresponds to the fraction of positive outcomes in the entire population. This shows that low accuracy predictors can be calibrated. Conversely, the accuracy maximizing predictor violates calibration whenever $p(y|x)$ is neither 0 nor 1. Remember that the accuracy maximizer f^* rounds everything to 0 or 1 so that typically $\mathbb{P}\{Y = 1 \mid f^*(X) = 1\} < 1$ and $\mathbb{P}\{Y = 1 \mid f^*(X) = 0\} > 0$. In other words, accuracy maximization tends to be overconfident.

In particular, don't confuse a calibrated prediction with any such notion as an *individual probability*. Calibration, in general, says nothing about the any specific instance x. It only talks about the set of instances with the same prediction.

Calibration is a fairly weak guarantee. After all, there's always a constant predictor that satisfies the condition. We can ask for a stronger condition by requiring calibration in subgroups. To designate groups, introduce a

Working draft available at https://mlbenchmarks.org

discrete random variable A that partitions the domain into strata $\{A = a\}$ of the population. Say that a function f satisfies *group calibration* with respect to A if it satisfies

$$\mathbb{P}\{Y = 1 \mid f(X) = p, A = a\} = p\,,$$

for all score values $p \in [0, 1]$ and groups a. Calibration is the same condition when the random variable A only takes on one value.

Fact 1. *The regression function $r^*(x) = \mathbb{E}[Y \mid X = x]$ satisfies group calibration with respect to any discrete random variable A fully determined by X, i.e., $A = h(X)$ for some measurable function h.*

In fact, the regression function is the only function that satisfies calibration with respect to all possible groups. This is because it satisfies calibration with respect to every possible atom $\{X = x\}$ in the population. We can't hope for more.

2.2 Risk minimization

We can generalize our discussion of optimal prediction further by introducing the idea of a loss function. A *loss function* takes two inputs, $\hat{y}$ and y, and returns a real number $\ell(\hat{y}, y)$ that we interpret as a quantified loss for predicting $\hat{y}$ when the target is y. A loss could be negative in which case it may be helpful to think of it as a reward.

We already encountered two loss functions, classification error and mean squared error. Classification error is the expected value of the loss function

$$\ell(\hat{y}, y) = \mathbb{1}\{\hat{y} \neq y\}\,,$$

known as *zero-one loss*. Indeed, writing the probability of an event as the expectation over the indicator of the event, we have

$$\mathbb{P}\{f(X) \neq Y\} = \mathbb{E}\left[\mathbb{1}\{\hat{y} \neq y\}\right].$$

Similarly, mean squared error is the expected value of the *squared loss*

$$\ell(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2\,.$$

Given a loss function, we can again minimize the expected loss over the population.

Working draft available at https://mlbenchmarks.org

Definition 4. *Define the* risk *of a function* $f : \mathcal{X} \to \mathcal{Y}$ *as*

$$R(f) := \mathbb{E}[\ell(f(X), Y)].$$

Here, the expectation is taken jointly over X *and* Y*.*

Risk minimization is the objective of minimizing risk:

$$\min_{f\,:\,\mathcal{X}\to\mathcal{Y}} \quad R(f)$$

We can solve the risk minimization problem with the same trick that applied to classification error and mean squared error: Solve the problem for each individual $x \in X$. Since the predictor f is unconstrained, nothing prevents us from doing so. For simplicity, I'll only include the binary case here.

Proposition 3. *The risk minimizing binary predictor for a binary loss function* ℓ *is the function*

$$f^*(x) = \mathbb{1}\left\{\frac{\mathbb{P}\{Y = 1 \mid X = x\}}{\mathbb{P}\{Y = 0 \mid X = x\}} \geq \frac{\ell(1,0) - \ell(0,0)}{\ell(0,1) - \ell(1,1)}\right\}.$$

Proof. To see why this predictor is optimal, we make use of the law of iterated expectation:

$$\mathbb{E}[\ell(f(X), Y)] = \mathbb{E}\left[\mathbb{E}\left[\ell(f(x), Y) \mid X = x\right]\right]$$

Here, the outer expectation is over a random draw of $x \sim X$ and the inner expectation samples Y conditional on $X = x$. Since there are no constraints on the predictor f, we can minimize the expression by minimizing the inner expectation independently for each possible setting that X can assume.

Indeed, for a fixed value x, we can expand the expected loss for each of the two possible predictions:

$$\begin{aligned}
\mathbb{E}[\ell(0, Y) \mid X = x] &= \ell(0,0)\,\mathbb{P}\{Y = 0 \mid X = x\} + \ell(0,1)\,\mathbb{P}\{Y = 1 \mid X = x\} \\
\mathbb{E}[\ell(1, Y) \mid X = x] &= \ell(1,0)\,\mathbb{P}\{Y = 0 \mid X = x\} + \ell(1,1)\,\mathbb{P}\{Y = 1 \mid X = x\}
\end{aligned}$$

The optimal assignment for this x is to set $f(x) = 1$ whenever the second expression is smaller than the first. Writing out this inequality and rearranging gives us the predictor specified in the lemma.

□

There's another way to state the optimal predictor. Using Bayes' rule, we have

$$p(y|x) = p(x|y) \cdot \frac{p_y}{p(x)}.$$

Working draft available at https://mlbenchmarks.org

Here, $p_y = \mathbb{P}\{Y = y\}$ is marginal probability of seeing label y, also called *base rate* of y. The expression $p(x|y)$ is the *likelihood* function. We can equivalently state the optimal predictor in terms of a ratio of likelihood functions:

$$f^*(x) = \mathbb{1}\left\{\frac{p(x|1)}{p(x|0)} \geq \frac{p_0}{p_1} \cdot \frac{\ell(1,0) - \ell(0,0)}{\ell(0,1) - \ell(1,1)}\right\}$$

This gives another intuitive interpretation of optimal prediction. The best predictor chooses the class that makes the data x more likely. But the threshold depends on the base rate of each class. If class 1 is highly unlikely in the data, the predictor requires a higher threshold to output 1. Ignoring the base rate when making predictions is the so-called *base rate fallacy*. If the symptoms of a patient match those of a rare disease somewhat better than those of the common flu, the doctor's best guess is probably still the common flu.

The threshold in optimal prediction also depends on what's at stake, the loss values. Four values fully specify a binary loss function. If, for example, the loss $\ell(1,0)$ of predicting 1 when the truth is 0 is high, the optimal predictor is again more conservative about positive predictions. It's worth taking a closer look at the four possible cases of binary prediction.

2.3 Errors and metrics

In a binary prediction problem, four cases can occur depending on the value of the prediction (positive/negative) and the true value (positive/negative):

- A *true positive* occurs when our prediction is positive ($\hat{Y} = 1$) and the true value is also positive ($Y = 1$).
- A false positive occurs when our prediction is positive ($\hat{Y} = 1$) and the true value is negative ($Y = 0$).
- A *true negative* happens when our prediction is negative ($\hat{Y} = 1$) and so is the true value ($Y = 0$).
- A *false negative* happens when our prediction is negative ($\hat{Y} = 0$) but the true value is positive ($Y = 1$).

The *confusion matrix* summarizes the four cases:

	$\hat{Y} = 1$	$\hat{Y} = 0$
$Y = 1$	True Positive (TP)	False Negative (FN)
$Y = 0$	False Positive (FP)	True Negative (TN)

Working draft available at https://mlbenchmarks.org

$\hat{Y} = 1$	$\hat{Y} = 0$

Corresponding to these four cases, there's additional terminology:

- The *true positive rate* is the probability $\mathbb{P}\{\hat{Y} = 1 \mid Y = 1\}$, also called *recall*.
- The *false positive rate* is $\mathbb{P}\{\hat{Y} = 1 \mid Y = 0\}$.
- The *true negative rate* is $\mathbb{P}\{\hat{Y} = 0 \mid Y = 0\}$.
- The *false negative rate* is $\mathbb{P}\{\hat{Y} = 0 \mid Y = 1\}$.

Note that true positive rate equals one minus false negative rate. The same relationship holds for true negative rate and false positive rate.

We can also turn the conditional probabilities around and get other meaningful concepts. One particularly useful concept is *precision* or *positive predictive value* $\mathbb{P}\{Y = 1 \mid \hat{Y} = 1\}$. Bayes' rule relates precision and recall:

$$\mathbb{P}\{Y = 1 \mid \hat{Y} = 1\} = \mathbb{P}\{\hat{Y} = 1 \mid Y = 1\} \cdot \frac{\mathbb{P}\{Y = 1\}}{\mathbb{P}\{\hat{Y} = 1\}}$$

What this suggests is that making more positive predictions, i.e., increasing $\mathbb{P}\{\hat{Y} = 1\}$, gives us high recall and low precision. Making fewer positive predictions increases precision at the cost of lowering recall. This tension is known as *precision-recall trade-off*. Another way to see this is by recalling that the optimal predictor has the form

$$f^*(x) = \mathbb{1}\left\{\frac{p(x|1)}{p(x|0)} \geq \tau\right\}$$

for some threshold value τ. By lowering τ we eventually achieve perfect recall. By increasing τ we eventually achieve perfect precision. The optimal predictor strikes some balance between the two.

Statisticians often express precision and recall in one metric called F_1-score, defined as the harmonic mean of precision and recall:

$$F_1 = \frac{2}{\text{precision}^{-1} + \text{recall}^{-1}} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

Sliding the threshold in optimal prediction also gives us a trade-off between true positive rate and false positive rate. A sufficiently small threshold gives us true and false positive rate 1. A sufficiently high threshold gives us

Working draft available at https://mlbenchmarks.org

true and false positive rate 0. Varying the threshold strikes some trade-off between the two. The entire curve of possible trade-offs is called *ROC curve*.

Many other such metrics derive from the confusion matrix. The terminology can be dense and overwhelming, but it suffices to understand the few underlying concepts.

2.4 Model training

Model training is the part of machine learning that this book is *not* about. Model training describes the set of heuristics practitioners apply to find good predictors. Training is the optimization process that takes a set of data points and produces a model. People use the term *model* freely to describe any trained system that maps inputs to outputs. Model training is an important core part of machine learning. Supervised learning refers to training methods that use labeled data. There's also unsupervised learning, where we only have x but not y in the data.

Throughout this book we'll always assume that someone else has done the model training for us. I call them the model builders. They are highly skilled, resourceful, and ambitious researchers, engineers, and practitioners who keep pushing the boundaries of model training. There's also a lot of beautiful theory about the optimization methods that model training requires. It's an entire subject in its own right. I think of the community of model builders as a computational resource. Once someone has put forward an interesting machine learning problem and the incentives point toward solving it, model builders will routinely find ways of doing so.

This focus of this text is on evaluating, comparing, ranking, and understanding trained models. It will nevertheless be helpful, especially later on, to know a little bit about how training works. There's no perfectly clean abstraction boundary between model training and model evaluation.

Training objective

At the core of model building is the training objective called *empirical risk minimization*. Rather than having knowledge of the full population, we assume that we only have a collection S consisting of n data points, where n is much smaller than the size of the population.

Definition 5. *Given a set* $S = \{(x_1, y_1), \ldots, (x_n, y_n)\} \subseteq \mathcal{X} \times \mathcal{Y}$ *of* n *labeled data points, the* empirical risk *of a function* $f : \mathcal{X} \to \mathcal{Y}$ *with respect to a loss function* ℓ

Working draft available at https://mlbenchmarks.org

is the sample average of the loss function

$$R_S(f) = \frac{1}{n}\sum_{i=1}^{n} \ell(f(x_i), y_i).$$

Analogous to risk minimization, define empirical risk minimization as the optimization problem:

$$\min_{f \in \mathcal{F}} \quad R_S(f).$$

One immediate difference is that we constrain $f \in \mathcal{F}$ to be from some *model family* $\mathcal{F}$. This model family could be the family of linear models, some architecture of deep convolutional networks, a transformer-based language model, and so forth. I won't go into detail about these different model families.

There's another important difference. In model training, you almost never directly minimize a loss function like classification error. The main reason is that this objective doesn't go well with the kind of gradient-based optimization methods people use in practice. These methods minimize empirical risk one small gradient step at a time and therefore require differentiable losses.

The most common training loss in machine learning today is the *cross entropy* loss. For binary prediction, where $y, \hat{y} \in \{0, 1\}$, the loss is

$$\ell(\hat{y}, y) = -\left[y \log \hat{y} + (1 - y) \log(1 - \hat{y})\right].$$

In the binary case, *logistic loss* is another name for it. For a multiclass problem with C classes, the cross entropy loss is:

$$\ell(\hat{y}, y) = -\sum_{c=1}^{C} y_c \log \hat{y}_c$$

Here, y_c is a *one-hot encoded* ground truth label, 1 for the correct class, 0 otherwise. We can therefore think of y as a distribution that is a point mass on the correct class. The prediction $\hat{y}$ is also a distribution where $\hat{y}_c$ the model's predicted probability for class c. Since $\hat{y}$ and y are both distributions, the cross entropy loss is closely related to the Kullback-Leibler divergence:

$$\mathrm{KL}(y, \hat{y}) = \ell(\hat{y}, y) - \mathrm{Entropy}(y)$$

where $\mathrm{Entropy}(y)$ is the entropy of the ground truth distribution y. In the case where y is the one-hot encoding of a single label, the entropy is 0,

Working draft available at https://mlbenchmarks.org

so $\mathrm{KL}(y\|\hat{y}) = \ell(\hat{y}, y)$. As long as the ground truth is fixed during training, minimizing cross entropy is equivalent to minimizing the KL-divergence from the true distribution to the predicted one.

A typical model won't directly spit out probabilities, but rather a vector $z \in \mathbb{R}^C$ consisting of C real numbers. It's customary to convert these into a probability distribution using the *softmax* operation:

$$\hat{y}_c = \frac{e^{z_c}}{\sum_{c=1}^{C} e^{z_c}}$$

Composing softmax, cross entropy loss, and one-hot encoding, things simplify to

$$\ell(\hat{y}, y) = -z_{\text{true}} + \log \sum_{c=1}^{C} e^{z_c},$$

where z_{true} is the coordinate of the vector z corresponding to the true class. This loss function is differentiable, so it goes well with gradient-based optimizers. Taking the derivative of the loss with respect to any of the predicted probabilities z_c, things simplify even further:

$$\frac{\partial \ell(\hat{y}, y)}{\partial z_c} = \hat{y}_c - y_c$$

Remember that y_c is the one hot encoding of the true class. This means that a small improvement in cross entropy loss on a single example nudges the model toward the correct class. The gradient vanishes when the model outputs the correct class. This is still true if y represents any distribution, not just a one-hot encoding. This property is what makes cross entropy a *proper scoring rule*: Minimizing the loss function recovers the true class probabilities.

Proposition 4. *The risk minimizer of the cross entropy loss is the condition probability model* $p(y|x)$.

At the population level, cross entropy minimization recovers the conditional probability function $p(y|x)$ of the data-generating distribution. When minimizing cross entropy on a sample S, we hope that we end up with a model that's not too far off. Understanding how good a trained model is, put simply, is the problem of model evaluation: Given a trained model, figure out how good it is, especially relative to other models.

Working draft available at https://mlbenchmarks.org

2.5 Notes

Hacking coined the phrase *astronomical conception of society* in *The Taming of Chance*[1], the sequel to his classic *The Emergence of Probability*[2]. *Emergence* explores the origins of probability in the 16th and 17th centuries. *Taming* takes on the rise of statistical thinking and population statistics in science and government in the 19th century. Daston covers the history of probability in the time period between *Emergence* and *Taming*.[3] Stigler is the reference for the history of statistics in the late 19th century.[4] I highly recommend these books to anyone interested in understanding how statistics came to be.

Much of the technical material I covered is standard and can be found in many textbooks. The chapter is similar in scope and content to Chapter 3 of *Fairness and Machine Learning*[5], and Chapter 2 of *Patterns, Predictions, and Actions*,[6] which is in turn similar to Chapter 2 in *Pattern Classification and Scene Analysis*[7] and Chapter 2 in *A Probabilistic Theory of Pattern Recognition*.[8]

For an advanced technical treatment of learning theory, consider Bach's *Learning Theory from First Principles*[9].

Working draft available at https://mlbenchmarks.org

Bibliography

[1] Ian Hacking. *The taming of chance*. Cambridge University Press, 1990.

[2] Ian Hacking. *The emergence of probability: A philosophical study of early ideas about probability, induction and statistical inference*. Cambridge University Press, 2006.

[3] Lorraine Daston. *Classical probability in the Enlightenment*. Princeton University Press, 2021.

[4] Stephen M Stigler. *The history of statistics: The measurement of uncertainty before 1900*. Harvard University Press, 1990.

[5] Solon Barocas, Moritz Hardt, and Arvind Narayanan. *Fairness and Machine Learning: Limitations and Opportunities*. MIT Press, 2023.

[6] Moritz Hardt and Benjamin Recht. *Patterns, predictions, and actions: Foundations of machine learning*. Princeton University Press, 2022.

[7] Richard O. Duda and Peter E. Hart. *Pattern Classification and Scene Analysis*. Wiley New York, 1973.

[8] Luc Devroye, László Györfi, and Gábor Lugosi. *A probabilistic theory of pattern recognition*, volume 31. Springer Science & Business Media, 2013.

[9] Francis Bach. *Learning theory from first principles*. MIT press, 2024.

Working draft available at https://mlbenchmarks.org