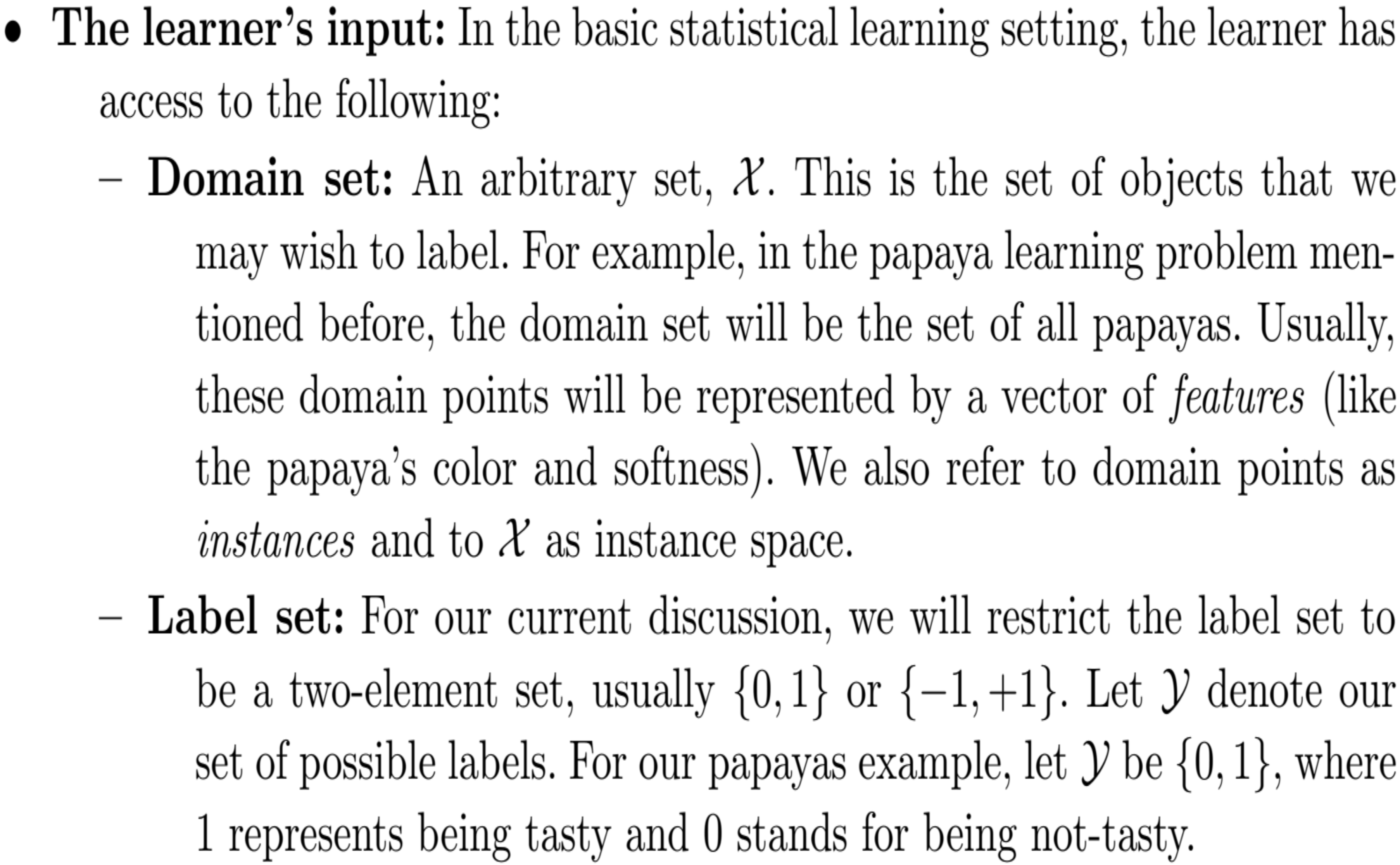
A Gentle Start

Let us begin our mathematical analysis by showing how successful learning can be achieved in a relatively simplified setting.

- Imagine you have just arrived in some small Pacific island. You soon find out that papayas are a significant ingredient in the local diet. However, you have never before tasted papayas.
- ***You have to learn how to predict whether a papaya you see in the market is tasty or not.***
- you need to decide which features of a papaya your prediction should be based on.
- On the basis of your previous experience with other fruits, you decide to use two features:
  - ***the papaya's color, ranging from dark green, through orange***
  - ***red to dark brown, and the papaya's softness, ranging from rock hard to mushy.***

A Formal Model: The Statistical Learning Framework

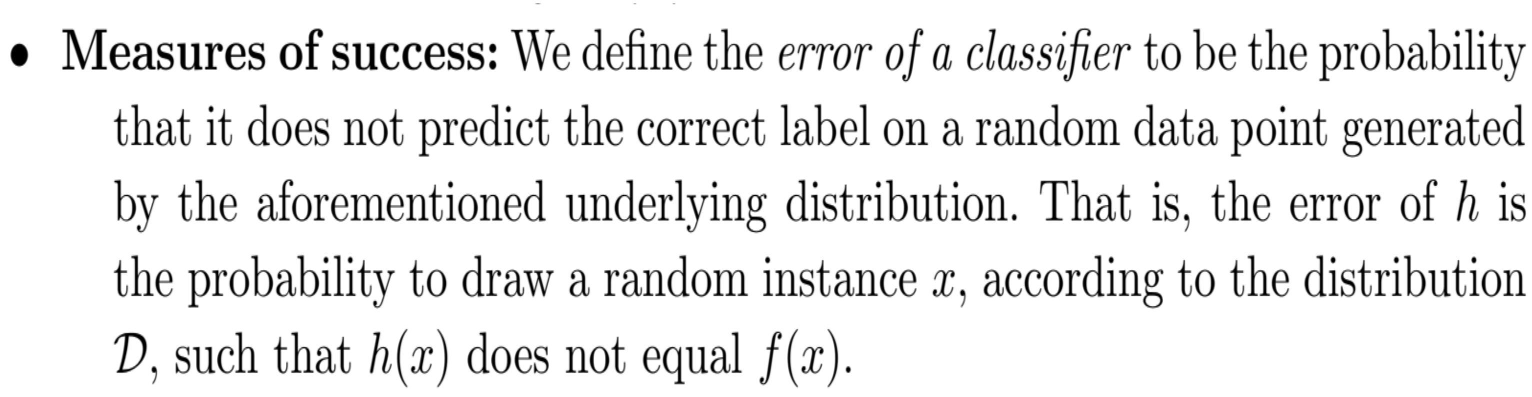


- **The learner's input:** In the basic statistical learning setting, the learner has access to the following:
  - **Domain set:** An arbitrary set, $\mathcal{X}$. This is the set of objects that we may wish to label. For example, in the papaya learning problem mentioned before, the domain set will be the set of all papayas. Usually, these domain points will be represented by a vector of *features* (like the papaya's color and softness). We also refer to domain points as *instances* and to $\mathcal{X}$ as instance space.
  - **Label set:** For our current discussion, we will restrict the label set to be a two-element set, usually $\{0, 1\}$ or $\{-1, +1\}$. Let $\mathcal{Y}$ denote our set of possible labels. For our papayas example, let $\mathcal{Y}$ be $\{0, 1\}$, where 1 represents being tasty and 0 stands for being not-tasty.

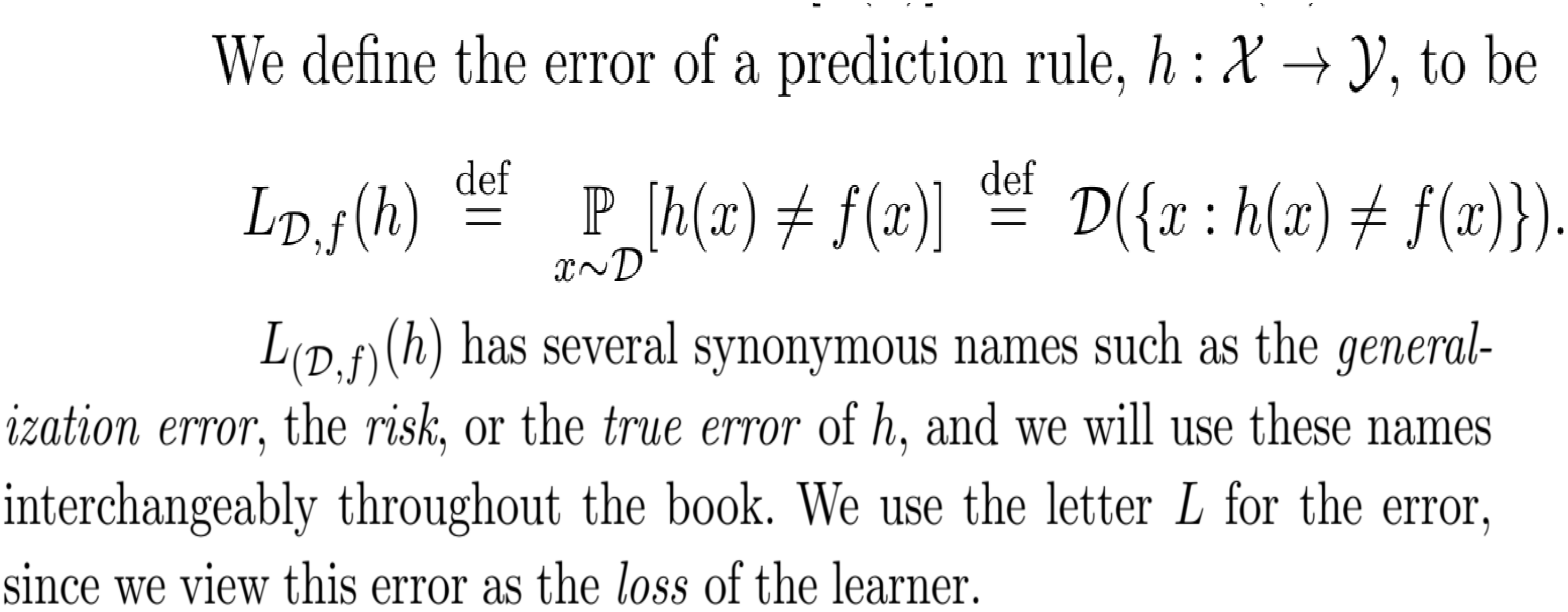
  - **Training data:** $S = ((x_1, y_1) \dots (x_m, y_m))$ is a finite sequence of pairs in $\mathcal{X} \times \mathcal{Y}$: that is, a sequence of labeled domain points. This is the input that the learner has access to (like a set of papayas that have been tasted and their color, softness, and tastiness). Such labeled examples are often called *training examples*. We sometimes also refer to S as a *training set*.[1]

- **The learner's output:** The learner is requested to output a *prediction rule*, $h : \mathcal{X} \to \mathcal{Y}$. This function is also called a *predictor*, a *hypothesis*, or a *classifier*. The predictor can be used to predict the label of new domain points.

- **A simple data-generation model** We now explain how the training data is generated. First, we assume that the instances (the papayas we encounter) are generated by some probability distribution (in this case, representing the environment). Let us denote that probability distribution over $\mathcal{X}$ by $\mathcal{D}$. It is important to note that we do not assume that the learner knows anything about this distribution. For the type of learning tasks we discuss, this could be any arbitrary probability distribution. As to the labels, in the current discussion we assume that there is some "correct" labeling function, $f : \mathcal{X} \to \mathcal{Y}$, and that $y_i = f(x_i)$ for all i. This assumption will be relaxed in the next chapter. The labeling function is unknown to the learner. In fact, this is just what the learner is trying to figure out. In summary, each pair in the training data S is generated by first sampling a point x_i according to $\mathcal{D}$ and then labeling it by f.



- **Measures of success:** We define the *error of a classifier* to be the probability that it does not predict the correct label on a random data point generated by the aforementioned underlying distribution. That is, the error of h is the probability to draw a random instance x, according to the distribution $\mathcal{D}$, such that $h(x)$ does not equal $f(x)$.



We define the error of a prediction rule, $h : \mathcal{X} \to \mathcal{Y}$, to be

$$L_{\mathcal{D},f}(h) \stackrel{\text{def}}{=} \underset{x\sim\mathcal{D}}{\mathbb{P}}[h(x) \neq f(x)] \stackrel{\text{def}}{=} \mathcal{D}(\{x : h(x) \neq f(x)\}).$$

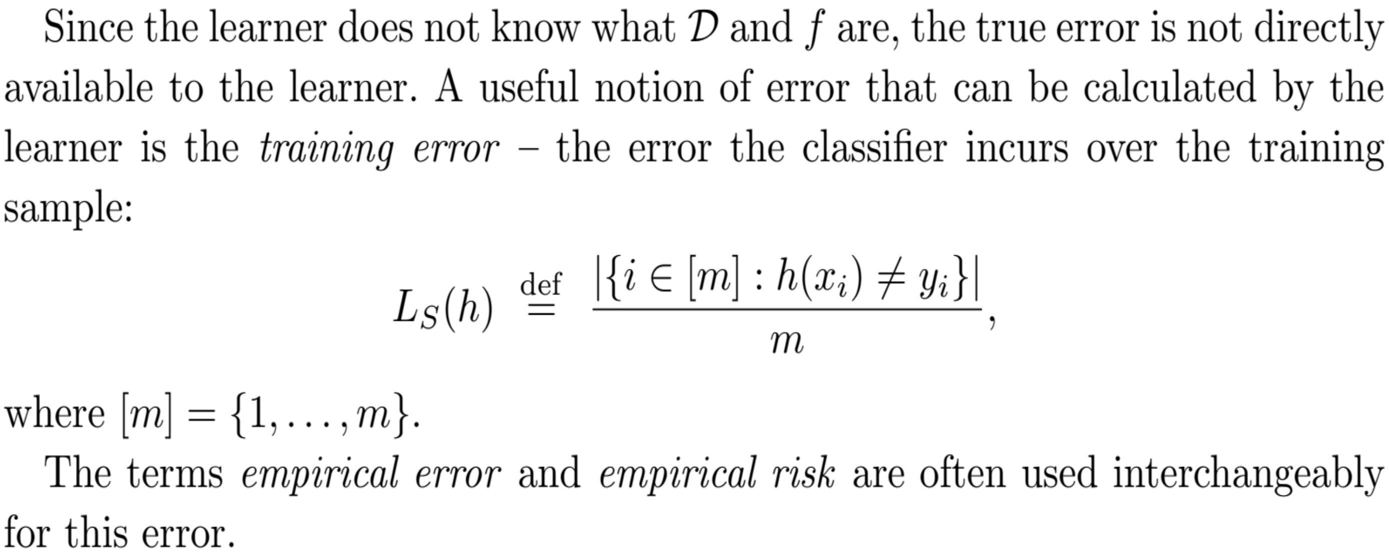
$L_{(\mathcal{D},f)}(h)$ has several synonymous names such as the *generalization error*, the *risk*, or the *true error* of h, and we will use these names interchangeably throughout the book. We use the letter L for the error, since we view this error as the *loss* of the learner.

- **A note about the information available to the learner** The learner is blind to the underlying distribution $\mathcal{D}$ over the world and to the labeling function f.

In our papayas example, we have just arrived in a new island and we have no clue as to how papayas are distributed and how to predict their tastiness. The only way the learner can interact with the environment is through observing the training set.

Empirical Risk Minimization

As mentioned earlier, a learning algorithm receives as input a training set S, sampled from an unknown distribution $\mathcal{D}$ and labeled by some target function f, and should output a predictor $h_S : \mathcal{X} \to \mathcal{Y}$ (the subscript S emphasizes the fact that the output predictor depends on S). The goal of the algorithm is to find h_S that minimizes the error with respect to the unknown $\mathcal{D}$ and f.



Since the learner does not know what $\mathcal{D}$ and f are, the true error is not directly available to the learner. A useful notion of error that can be calculated by the learner is the *training error* – the error the classifier incurs over the training sample:

$$L_S(h) \stackrel{\text{def}}{=} \frac{|\{i \in [m] : h(x_i) \neq y_i\}|}{m},$$

where $[m] = \{1, \ldots, m\}$.

The terms *empirical error* and *empirical risk* are often used interchangeably for this error.

Since the training sample is the snapshot of the world that is available to the learner, it makes sense to search for a solution that works well on that data. This learning paradigm – coming up with a predictor h that minimizes $L_S(h)$ – is called *Empirical Risk Minimization* or ERM for short.

Although the ERM rule seems very natural, without being careful, this approach may fail miserably.

Something May Go Wrong – Overfitting

Two Types of Error

True Error (aka. **expected risk**)

$$R(h) = P_{\mathbf{x} \sim p^*(\mathbf{x})}(c^*(\mathbf{x}) \neq h(\mathbf{x}))$$

This quantity is always **unknown**

Train Error (aka. **empirical risk**)

$$\begin{aligned}\hat{R}(h) &= P_{\mathbf{x} \sim \mathcal{S}}(c^*(\mathbf{x}) \neq h(\mathbf{x})) \\ &= \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}(c^*(\mathbf{x}^{(i)}) \neq h(\mathbf{x}^{(i)})) \\ &= \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}(y^{(i)} \neq h(\mathbf{x}^{(i)}))\end{aligned}$$

We can **measure** this on the training data

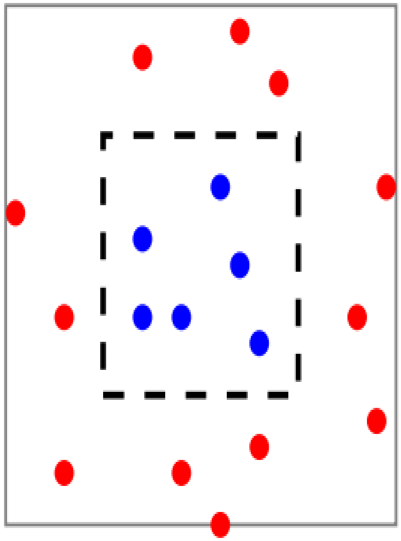
where $\mathcal{S} = \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)})\}_{i=1}^{N}$ is the training data set, and $\mathbf{x} \sim \mathcal{S}$ denotes that $\mathbf{x}$ is sampled from the empirical distribution.

predict the taste of a papaya on the basis of its softness and color.

Assume that the probability distribution $\mathcal{D}$ is such that instances are distributed uniformly within the gray square and the labeling function, f, determines the label to be 1 if the instance is within the inner blue square, and 0 otherwise.

area of the gray square in the picture is 2 and the area of the blue square is 1.

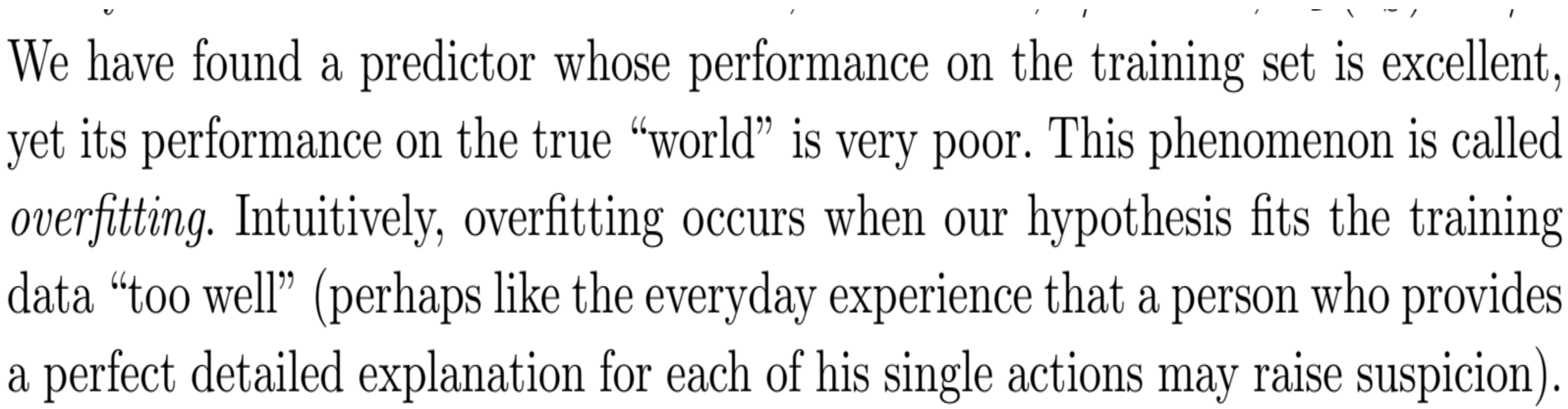
$$h_S(x) = \begin{cases} y_i & \text{if } \exists i \in [m] \text{ s.t. } x_i = x \\ 0 & \text{otherwise.} \end{cases}$$



$L_S(h_S) = 0$ empirical-minimum-cost hypotheses

$L_{\mathcal{D}}(h_S) = 1/2.$ true error

Overfitting



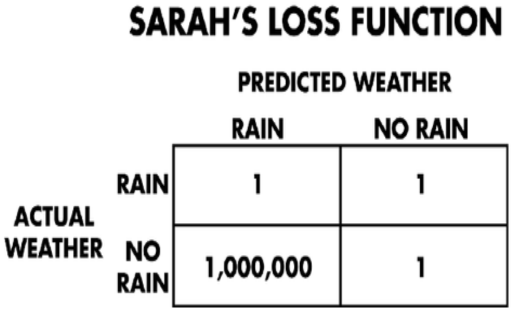
We have found a predictor whose performance on the training set is excellent, yet its performance on the true "world" is very poor. This phenomenon is called *overfitting*. Intuitively, overfitting occurs when our hypothesis fits the training data "too well" (perhaps like the everyday experience that a person who provides a perfect detailed explanation for each of his single actions may raise suspicion).

- Underfitting: neither model the training data nor generalize the new data.

Empirical Risk Minimization Example

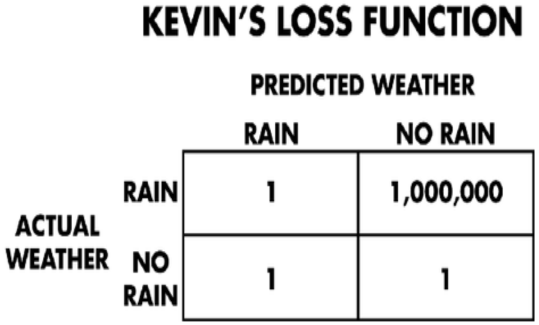
f: Sky → Weather

- Kevin hates getting rained on.
- Sarah hates to carry an umbrella.



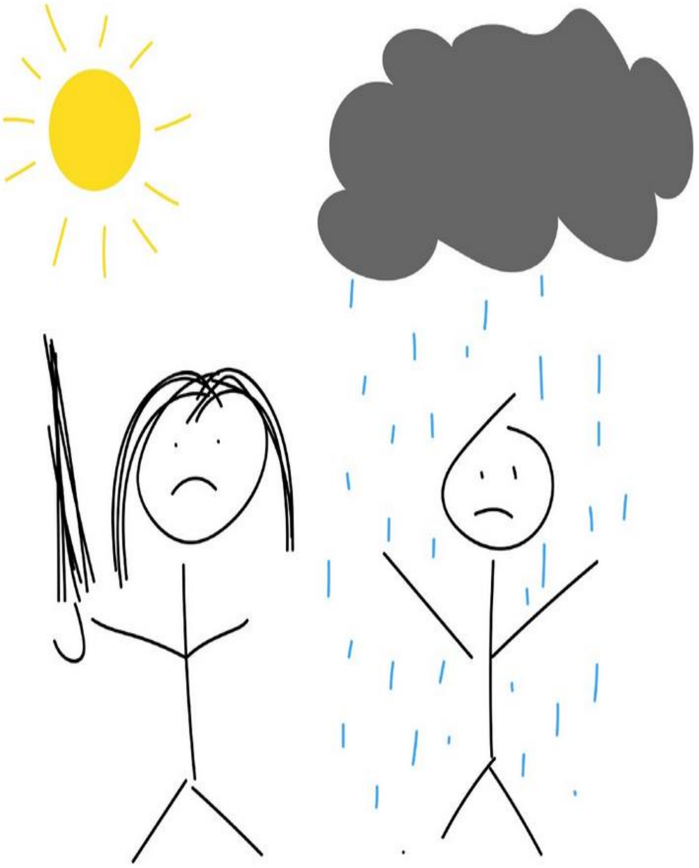
SARAH'S LOSS FUNCTION

		PREDICTED WEATHER	
		RAIN	NO RAIN
ACTUAL WEATHER	RAIN	1	1
	NO RAIN	1,000,000	1



KEVIN'S LOSS FUNCTION

		PREDICTED WEATHER	
		RAIN	NO RAIN
ACTUAL WEATHER	RAIN	1	1,000,000
	NO RAIN	1	1



Empirical Risk Minimization with Inductive Bias

We have just demonstrated that the ERM rule might lead to overfitting. Rather than giving up on the ERM paradigm, we will look for ways to rectify it. We will search for conditions under which there is a guarantee that ERM does not overfit, namely, conditions under which when the ERM predictor has good performance with respect to the training data, it is also highly likely to perform well over the underlying data distribution.

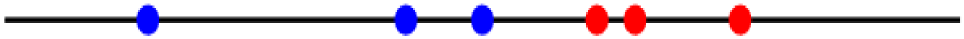
A common solution is to apply the ERM learning rule over a restricted search space. Formally, the learner should choose in advance (before seeing the data) a set of predictors. This set is called a *hypothesis class* and is denoted by $\mathcal{H}$. Each $h \in \mathcal{H}$ is a function mapping from $\mathcal{X}$ to $\mathcal{Y}$. For a given class $\mathcal{H}$, and a training sample, S, the $\mathrm{ERM}_{\mathcal{H}}$ learner uses the ERM rule to choose a predictor $h \in \mathcal{H}$, with the lowest possible error over S. Formally,

$$\mathrm{ERM}_{\mathcal{H}}(S) \in \operatorname*{argmin}_{h \in \mathcal{H}} L_S(h),$$

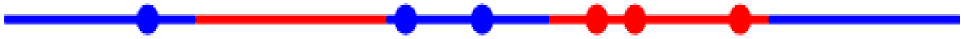
where argmin stands for the set of hypotheses in $\mathcal{H}$ that achieve the minimum value of $L_S(h)$ over $\mathcal{H}$. By restricting the learner to choosing a predictor from $\mathcal{H}$, we *bias* it toward a particular set of predictors. Such restrictions are often called an *inductive bias*. Since the choice of such a restriction is determined before the learner sees the training data, it should ideally be based on some prior knowledge about the problem to be learned.

Example: Learning dosage safety

- Learning problem: which medicine dosages are safe?
- $\mathcal{X} = [0, 100]$ (dosage), $\mathcal{Y} = \{0, 1\}$ (causes side effects?)
- A possible training sample (blue: label is 0, red: label is 1):



- ERM without inductive bias might return the following rule:

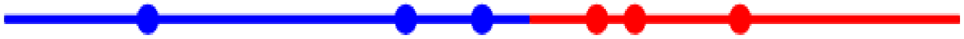


Here $\operatorname{err}(\hat{h}_S, S) = 0$. What do you think is $\operatorname{err}(\hat{h}_S, \mathcal{D})$?

- Inductive bias: Limit the ERM algorithm to return only functions that describe **thresholds** on the line:

$$\forall x \in \mathcal{X}, f_a(x) := \mathbb{I}[x \geq a].$$

- Now the algorithm will return something like this:



Again, $\operatorname{err}(\hat{h}_S, S) = 0$. What do you think is $\operatorname{err}(\hat{h}_S, \mathcal{D})$ this time?

Empirical Risk Minimization is a part of _________________

a) Supervised learning
b) Unsupervised Learning
c) Reinforcement Learning
d) All of the mentioned

- Is it possible to train a hypothesis on the each possible instance of a class.

- Yes
- No

Observer has no idea about the ________

a) Distribution of example
b) Actual target function
c) A and b
d) None of the mentioned

Which of the following Quantity is always unknown?

A) True Error
B) Train Error

Thank You !!!