2 Background and Notation

In this chapter we establish the mathematical notation used throughout this book and introduce the basic foundation of machine learning that this text builds upon. Readers generally familiar with this field can cursorily read this chapter to become familiar with our notation. For a more thorough treatment of machine learning, the reader should refer to a text such as (Hastie, Tibshirani, & Friedman 2003) or (Vapnik 1995).

2.1 Basic Notation

Here we give a brief overview of the formal notation we use throughout this text. For more, along with foundations in basic logic, set theory, linear algebra, mathematical optimization, and probability we refer the reader to Appendix A.

We use $=$ to denote *equality* and $\triangleq$ to denote *defined as*. The typeface style of a character is used to differentiate between elements of a set, sets, and spaces as follows. Individual objects such as scalars are denoted with italic font (e.g., x) and multidimensional vectors are denoted with bold font (e.g., $\mathbf{x}$). A set is denoted using blackboard bold characters (e.g., $\mathbb{X}$). However, when referring to the *entire* set or universe that spans a particular kind of object (i.e., a space), we use calligraphic script such as in $\mathcal{X}$ to distinguish it from subsets $\mathbb{X}$ contained within this space.

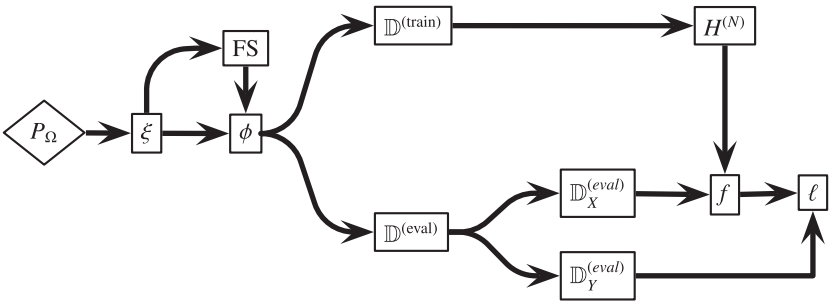
2.2 Statistical Machine Learning

Machine learning encompasses a vast field of techniques that extract information from data as well as the theory and analysis relating to these algorithms. In describing the task of machine learning, Mitchell (1997) wrote,

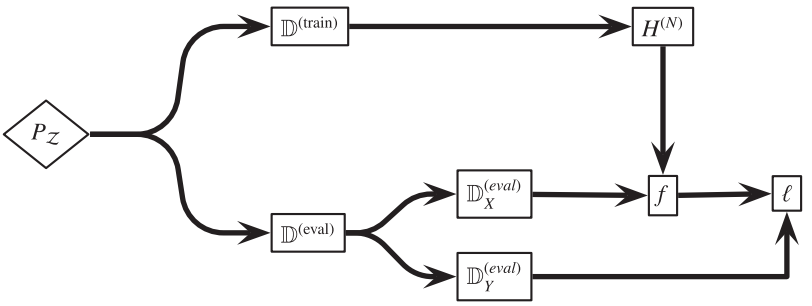
> A computer program is said to *learn* from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

This definition encompasses a broad class of methods. We present an overview of the terminology and mechanisms for a particular notion of learning that is often referred to as statistical machine learning. In particular, the notion of experience is cast as data, the task is to choose an action (or make a prediction/decision) from an action or

https://doi.org/10.1017/9781107338548.002 Published online by Cambridge University Press



(a) The complete learning framework.



(b) The learning framework with implicit data collection.

Figure 2.1 Diagrams depicting the flow of information through different phases of learning. **(a)** All major phases of the learning algorithm except for model selection. Here objects from the space Ω are drawn from the distribution P_Ω and parsed into measurements that then are used in the feature selector FS. It selects a feature mapping ϕ that is used to create training and evaluation datasets, $\mathbb{D}^{(\text{train})}$ and $\mathbb{D}^{(\text{eval})}$. The learning algorithm $H^{(N)}$ selects a hypothesis f based on the training data, and its predictions are assessed on $\mathbb{D}^{(\text{eval})}$ according to the loss function ℓ. **(b)** The training and prediction phases of learning with implicit data collection phases. Here the data are assumed to be drawn directly from $P_\mathcal{Z}$ instead of being drawn from P_Ω and subsequently mapped into the space $\mathcal{Z}$ by the measurement process.

decision space, and the performance metric is a loss function that measures the cost that the learner incurs for a particular prediction/action compared to the best or correct one. Figure 2.1 illustrates the data flow for learning in this setting. The data lies in a product space $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ composed of an input space $\mathcal{X}$ and an output space $\mathcal{Y}$, which are discussed later. The training dataset $\mathbb{D}^{(\text{train})}$ (consisting of N coupled training examples from the product space) is drawn from the distribution $P_\mathcal{Z}$ and is used by the learning procedure $H^{(N)}$ to produce a hypothesis (or classifier) f. This classifier is a function that makes predictions on a new set of data $\mathbb{D}^{(\text{eval})}$ (assumed to be drawn from the same distribution) and is assessed according to the loss function ℓ. Figure 2.1(a) additionally

https://doi.org/10.1017/9781107338548.002 Published online by Cambridge University Press

depicts the data collection phase of learning discussed in Section 2.2.1. While measurement and initial feature extraction are important aspects for the security of a learning algorithm, our focus in this book is on the security of the learning algorithm.

Throughout this book, we only consider inductive learning methods for which learning takes the form of generalizing from prior experiences. The method of induction requires an inductive bias: a set of (implicit) assumptions used to create generalizations from a set of observations. An example of an inductive bias is Ockham's Razor—the preference for the simplest hypothesis that is consistent with the observations. Naive Bayes methods use *maximal conditional independence* as their inductive bias, while the inductive bias of a support vector machine is *maximal margin*. The inductive bias of most methods is an implicit bias built into the learning procedure.

In this book, we focus on techniques from statistical machine learning that can be described as empirical risk minimization procedures. Later, we summarize the components of these procedures and provide notation to describe them, but first, their overall goal is to minimize the expected loss (risk) incurred on predictions made for the unseen evaluation data, $\mathbb{D}^{(\text{eval})}$. Minimizing the average loss (or risk) on the training data is often used as a surrogate for minimizing the expected loss on unseen random evaluation data, and under the appropriate conditions, the error on the training data can be used to bound generalization error (cf. Vapnik 1995, Chapter 1). Underlying such results is the stationarity assumption that the training data and evaluation data are both drawn from the same distribution $P_{\mathcal{Z}}$ as depicted in Figure 2.1. Subsequently, we examine scenarios that violate this stationarity assumption and evaluate the impact these violations have on the performance of learning methods. However, while we study the impact on performance of empirical risk minimizers, these violations would have similar effects on any learner based on stationarity, which is often required to guarantee generalization. Further we demonstrate that these violations have less impact on alternative empirical risk minimizers that were designed to be robust against distributional deviations. Vulnerabilities are neither unique to empirical risk minimization procedures nor are they inherent to them, but rather guarding against these exploits requires learners designed to be resilient against violations of their assumptions. There is also a tradeoff between the robustness and the effectiveness of the procedure, which we emphasize in each chapter.

2.2.1 Data

Real-world objects such as emails or network packets are contained in a space Ω of all such objects. Usually, applying a learning algorithm directly to real-world objects is difficult because the learner cannot parse the objects' structures or the objects may have extraneous elements that are irrelevant to the learner's task. These objects are transformed into a more amenable representation by a mapping from real-world abstractions (e.g., objects or events) into a space of representative observations—the process of measurement. In this process, each real-world abstraction, $\omega \in \Omega$, is measured and represented to the learning algorithm as a composite object $x \in \mathcal{X}$. Often there are D simple measurements of ω; the i^{th} measurement (or feature) x_i is from a space $\mathcal{X}_i$, and the

https://doi.org/10.1017/9781107338548.002 Published online by Cambridge University Press

composite representation (or data point) $\mathbf{x} \in \mathcal{X}$ is represented as a tuple $(x_1, x_2, \ldots, x_D)$. The space of all such data points is $\mathcal{X} \triangleq \mathcal{X}_1 \times \mathcal{X}_2 \times \ldots \times \mathcal{X}_D$. Each feature is usually real-valued $\mathcal{X}_i = \Re$, integer-valued $\mathcal{X}_i = \mathfrak{Z}$, boolean $\mathcal{X}_i = \{\text{true}, \text{false}\}$, or categorical $\mathcal{X}_i = \{A_1, A_2, \ldots, A_k\}$. Figure 2.1(a) formally represents the measurement process with the measurement map $\xi : \Omega \mapsto \mathcal{X}$, which represents the learner's perspective of the world.

Data collection is the application of a measurement map ξ to a sequence of N objects $\omega^{(1)}, \omega^{(2)}, \ldots, \omega^{(N)}$ resulting in an indexed set of N data points $\left\{x^{(i)}\right\}_{i=1}^{N} \subseteq \mathcal{X}^N$, which we refer to as a dataset and denote it by $\mathbb{D}$. The dataset represents a sequence of observations of the environment and serves as the basis for the learner's ability to generalize past experience to future events or observations. Various assumptions are made about the structure of the dataset, but most commonly, the learner assumes the data points are independent and identically distributed. All the learning algorithms we investigate assume that the data is independently sampled from an unknown but stationary distribution, although with various degrees of dependence on this assumption.

Labels

In many learning problems, the learner is tasked with learning to predict the unobserved state of the world based on its observed state. Observations are partitioned into two sets. Those that are observed are the explanatory variables (also referred to as the covariates, input, predictor, or controlled variables) and the unobserved quantities to be predicted comprise the response variables (also referred to as the output or outcome variables). In the context of this book and our focus on classification, we refer to the observed independent quantities as the data point (as discussed earlier) and to the dependent categorical quantity as its label. The learner is expected to be able to predict the label for a data point having seen past instances of data points coupled with their labels. In this form, each datum consists of two paired components: a data point x from an input space $\mathcal{X}$ and a label y from a response space $\mathcal{Y}$. The components of the datum are also referred to as the predictor (input) variable x and the response (output) variable y. These paired objects belong to the Cartesian product: $\mathcal{Z} \triangleq \mathcal{X} \times \mathcal{Y}$. We also assume these instances are randomly drawn from a joint distribution $P_{\mathcal{Z}}$ over this paired space that may also be denoted by $P_{\mathcal{X}\times\mathcal{Y}}$ when convenient.

In learning problems that include labels (e.g., supervised or semi-supervised learning), the learner trains on a set of paired data from $\mathcal{Z}$. In particular, a labeled dataset is an indexed set of N instances from $\mathcal{Z}$: $\mathbb{D} \triangleq \left\{z^{(1)}, z^{(2)}, \ldots, z^{(N)}\right\}$ where $z^{(i)} \in \mathcal{Z}$ is drawn from $P_{\mathcal{Z}}$. The indexed set of only the data points is $\mathbb{D}_X \triangleq \left\{x^{(1)}, x^{(2)}, \ldots, x^{(N)}\right\}$ and the indexed set of only the labels is $\mathbb{D}_Y \triangleq \left\{y^{(1)}, y^{(2)}, \ldots, y^{(N)}\right\}$. In the case that $\mathcal{X} = \mathcal{A}^D$ for some numeric set $\mathcal{A}$, the i^{th} data point can be expressed as a D-dimensional vector $\mathbf{x}^{(i)}$ and the data can be expressed as an $N \times D$ matrix $\mathbf{X}$ defined by $\mathbf{X}_{i,\bullet} = \mathbf{x}^{(i)}$. Similarly, when $\mathcal{Y}$ is a scalar set (e.g., booleans, reals), $y^{(i)}$ is a scalar, and the labels can be expressed as a simple N-dimensional vector $\mathbf{y}$. In the remainder of this book, N will refer to the size of $\mathbb{D}$, and where applicable, D will refer to the dimension of its data points.

https://doi.org/10.1017/9781107338548.002 Published online by Cambridge University Press

Feature Selection

Typically, measurement is only the first phase in the overall process of data extraction. After a dataset is collected, it is often altered through a process of feature selection. Feature selection is a mapping $\phi_{\mathbb{D}}$ of the original measurements into a space $\hat{\mathcal{X}}$ of features[1]: $\phi_{\mathbb{D}} : \mathcal{X} \to \hat{\mathcal{X}}$. Unlike the data-independent measurement mapping ξ, the feature selection map often is selected in a data-dependent fashion according to the entire dataset $\mathbb{D}$, to extract aspects of the data most relevant to the learning task. Further, measurement often is an irreversible physical process, whereas feature selection usually can be redone by reprocessing the original measurements. In many settings, one can retroactively alter the feature selection process by redefining the feature selection map and reapplying it to the measured data, whereas it is impossible to make retroactive measurements on the original objects unless the information required to reconstruct this raw data is still available. For the purposes of this book, we do not distinguish between the feature selection and measurement phases because the attacks we study target downstream aspects of learning. We merge them together into a single step and disregard $\hat{\mathcal{X}}$ except explicitly in reference to feature selection. We revisit potential roles for feature selection in security-sensitive settings in Section 9.1.1.2.

2.2.2 Hypothesis Space

A learning algorithm is tasked with selecting a hypothesis that best supports the data. Here we consider the hypothesis to be a function f mapping from the data space $\mathcal{X}$ to the response space $\mathcal{Y}$; i.e., $f : \mathcal{X} \to \mathcal{Y}$. Of course, there are many such hypotheses, which together form the family $\mathcal{F}$ of all possible hypotheses or the hypothesis space. This space is most generally the set of all functions that map $\mathcal{X}$ onto $\mathcal{Y}$ as we discussed in Appendix A.1: $\mathcal{F} \triangleq \{f \mid f : \mathcal{X} \to \mathcal{Y}\}$. The hypothesis space $\mathcal{F}$ may be constrained by assumptions made about the form of the hypotheses. For instance, when $\mathcal{X} = \Re^D$, the family may be restricted to generalized linear functions of the form $f^{\beta}_{\mathbf{a},b}(\mathbf{x}) \triangleq \beta\left(\mathbf{a}^\top \mathbf{x} + b\right)$ where $\beta : \Re \to \mathcal{Y}$ is some mapping from the reals to the response space. In the case that $\mathcal{Y} = \{0, 1\}$, the function $\beta(x) = \mathrm{I}[x > 0]$ yields the family of all halfspaces on $\Re^D$ parameterized by $(\mathbf{a}, b)$. In the case that $\mathcal{Y} = \Re$ the identity function $\beta(x) = x$ defines the family of linear functions on $\Re^D$ also parameterized by $(\mathbf{a}, b)$.

2.2.3 The Learning Model

We describe the learner as a model and a training procedure. The model captures assumptions made about the observed data—this model provides limitations on the space of hypotheses and also available prior knowledge or inductive bias on these hypotheses (e.g., a prior distribution in a Bayesian setting or a regularizer in a risk

[1] In the literature, feature selection chooses a subset of the measurements ($\hat{\mathcal{X}} \subseteq \mathcal{X}$), and feature extraction creates composite features from the original measurements. We do not differentiate between these two processes and will refer to both as feature selection.

https://doi.org/10.1017/9781107338548.002 Published online by Cambridge University Press

minimization setting). That is, the model is a set of assumptions about the relationship between the observed data and the hypothesis space, but the model does not specify how hypotheses are selected—that is done by a training procedure as discussed later. For example, consider a simple location estimation procedure for normally distributed data. The data model specifies that the data points are independently drawn from a unit-variance Gaussian distribution centered at an unknown parameter θ; i.e., $X \sim \mathrm{N}(\theta, 1)$. The mean and the median are procedures for estimating the location parameter θ. By distinguishing between the model and the training procedure, we can study two different aspects of a learner's vulnerabilities.

2.2.4 Supervised Learning

The primary focus of this work will be analyzing the task of prediction in a supervised learning setting. In the supervised learning framework, the observed data is a paired dataset $\mathbb{D} = \left\{\left(x^{(i)}, y^{(i)}\right)\right\}$, which we assume to be drawn from an unknown distribution $P_{\mathcal{X}\times\mathcal{Y}}$. The objective of prediction is to select the best hypothesis or function $\hat{f}$ for predicting the response variable based on the observed predictor variable. More precisely, given a hypothesis space $\mathcal{F}$ of functions mapping from the input space $\mathcal{X}$ to the output space $\mathcal{Y}$, the prediction task is to select a classification hypothesis (classifier) $\hat{f} \in \mathcal{F}$ with the smallest expected prediction cost; i.e., the cost incurred in predicting the label y of a new random instance (x, y) when only the predictor variable x is revealed to the classifier. To accomplish this task, the learner is given a labeled training dataset $\mathbb{D}$ and a cost or loss function ℓ. This cost function assigns a numeric cost to each combination of data instance, true label, and classifier label (most commonly, however, the cost function used is identical for every data instance).

To accomplish the prediction task, a learning algorithm $H^{(N)}$ (also called a training algorithm) is used to select the best hypothesis from $\mathcal{F}$ based on the training data and the cost function. This learner is a mapping from a dataset $\mathbb{D} \in \mathcal{Z}^N$ to a hypothesis f in the hypothesis space: $H^{(N)} : \mathcal{Z}^N \to \mathcal{F}$; that is, a mapping from N training examples in $\mathcal{Z}$ to some hypothesis f in the hypothesis space $\mathcal{F}$. If the algorithm has a randomized element we use the notation $H^{(N)} : \mathcal{Z}^N \times \mathfrak{R} \to \mathcal{F}$ to capture that fact that the hypothesis depends on a random element $R \sim P_{\mathfrak{R}}$.

Training

The process we describe here is batch learning—the learner trains on a training set $\mathbb{D}^{(\text{train})}$ and is evaluated on an evaluation set $\mathbb{D}^{(\text{eval})}$. This setting can be generalized to a repeated process of online learning in which the learner continually retrains on evaluation data after obtaining evaluation labels (we return to this setting in Section 3.6). In a pure online setting, prediction and retraining occur every time a new data point is received. In the batch learning setting (or in a single epoch of online learning), the learner $H^{(N)}$ forms a hypothesis $\hat{f}$ based on the collected data $\mathbb{D}^{(\text{train})}$—the process known as training. A plethora of different training procedures have been used in the supervised learning setting for (regularized) empirical risk minimization under a wide

https://doi.org/10.1017/9781107338548.002 Published online by Cambridge University Press

variety of settings. We will not detail these methods further, but instead introduce the basic setting for classification.

In a classification problem the response space is a finite set of labels each of which corresponds to some subset of input space (although these subsets need not be disjoint). The learning task is to construct a classifier that can best assign these labels to new data points based on labeled training examples from each class. In a binary classification setting there are only two labels, "−" and "+"; i.e., the response space is $\mathcal{Y} = \{"-", "+"\}$. Where mathematically convenient, we will use 0 and 1 in place of the labels "−" and "+" respectively; i.e., we will implicitly redefine the label y to be $\mathrm{I}[y = "+"]$. In binary classification, we refer to the two classes as the negative class ($y = "-"$) and the positive class ($y = "+"$). The training set $\mathbb{D}^{(\text{train})}$ consists of labeled instances from both classes. We primarily focus on binary classification for security applications in which a defender attempts to separate instances (i.e., data points), some or all of which come from a malicious attacker, into harmful and benign classes. This setting covers many important security applications, such as host and network intrusion detection, virus and worm detection, and spam filtering. In detecting malicious activity, the positive class (with label "+") indicates malicious intrusion instances while the negative class (with label "−") indicates benign or innocuous normal instances. In Chapter 6, we also consider the anomaly detection setting, in which the training set only contains normal instances from the negative class.

Risk Minimization

The goal of the learner is to find the *best* hypothesis $f^\star$ from the hypothesis space $\mathcal{F}$ that best predicts the target concept (according to some measure of correctness) on instances drawn according to the unknown distribution $P_{\mathcal{Z}}$. Ideally the learner is able to distinguish $f^\star$ from any other hypothesis $f \in \mathcal{F}$, with high probability, based on the observed data $\mathbb{D}$ of data points drawn from $P_{\mathcal{Z}}$, but this is seldom realistic or even possible. Instead, the learner should choose the *best* hypothesis in the space according to some criteria for preferring one hypothesis over another—this is the performance measure. The measure can be any assessment of a hypothesis; in statistical machine learning, the most common goal is *risk minimization*, which is based on a loss function $\ell : \mathcal{Y} \times \mathcal{Y} \mapsto \Re_{0+}$ the non-negative reals. The learner selects a hypothesis $\hat{f} \in \mathcal{F}$ that minimizes the expected loss, or risk, over all hypotheses ($\hat{f} \in \operatorname{argmin}_{f \in \mathcal{F}} R(P_{\mathcal{Z}}, f)$) where the risk is given by

$$R(P_{\mathcal{Z}}, f) \triangleq \int_{(x,y)\in\mathcal{Z}} \ell(y, f(x)) \; dP_{\mathcal{Z}}(x, y)$$

and $P_{\mathcal{Z}}(x, y)$ is a probability measure for the distribution $P_{\mathcal{Z}}$. Unfortunately, this minimization is infeasible since this distribution is unknown. Instead, in the empirical risk minimization framework (cf. Vapnik 1995) the learner selects $\hat{f}$ to minimize the empirical risk on the dataset $\mathbb{D} \sim P_{\mathcal{Z}}$ defined as

$$\tilde{R}_N(f) = \frac{1}{N} \sum_{(x,y)\in\mathbb{D}} \ell(y, f(x))$$

with $N = |\mathbb{D}|$.

https://doi.org/10.1017/9781107338548.002 Published online by Cambridge University Press

Regularization

If the space of hypotheses $\mathcal{F}$ is too expressive, there will be a hypothesis that fits the empirical observations exactly, but it may not be able to make accurate predictions about unseen instances; e.g., consider constructing a lookup table mapping from observed data points to their responses: this classifier can perfectly predict observed instances, but does not generalize to unobserved instances. This phenomenon is known as overfitting the training data. One possibility to avoid overfitting is to only consider a small or restricted space of hypotheses; e.g., the space of linear functions. Alternatively, one could allow for a large space of hypotheses, but penalize the complexity of a hypothesis—a practice known as regularization. The learner selects a hypothesis $\hat{f}$ that minimizes the modified objective

$$\tilde{R}_N(f) + \lambda \cdot \rho(f) \tag{2.1}$$

where the function $\rho : \mathcal{F} \to \Re$ is a measure of the complexity of a hypothesis and $\lambda \in \Re_+$ the positive reals, is a regularization parameter that controls the tradeoff between risk minimization and hypothesis complexity.

Prediction and Evaluation

Once trained on a dataset, the learned hypothesis is subsequently used to predict the response variables for a set of unlabeled data. We call this the *evaluation phase* although it may also be referred to as the test or prediction phase. Initially, only a new data point x is presented to the predictor (or rather the predictor is queried with the data point). The learned hypothesis $\hat{f}$ predicts a value $\hat{y} = \hat{f}(x)$ in the space $\mathcal{Y}$ of all possible responses.[2] Finally, the actual label y is revealed, and the agent receives a loss $\ell(\hat{y}, y)$ as an assessment of its performance. In the binary classification setting, there are generally two types of classification mistakes: a false positive (FP) is a normal instance classified as positive, and a false negative (FN) is a malicious instance classified as negative. Selecting an appropriate tradeoff between false positives and false negatives is an application-specific task.

The performance of a learner is typically assessed on a labeled evaluation dataset, $\mathbb{D}^{(\text{eval})}$. Predictions are generated by $\hat{f}$ for each data point $x^{(i)} \in \mathbb{D}_X^{(\text{eval})}$ in the evaluation dataset, and the losses incurred are aggregated into various performance measures. In the binary classification setting, the typical performance measures are the false-positive rate (FPR), the fraction of negative instances classified as positive, and the false-negative rate (FNR), the fraction of positive instances classified as negatives. Often a classifier is tuned to have a particular (empirical) false-positive rate based on held-out training data (validation dataset), and its resulting false-negative rate is assessed at that FP level.

[2] The space of allowed predictions or actions $\mathcal{A}$ need not be the same as the space of allowed responses, $\mathcal{Y}$. This allows the learner to choose from a larger range of responses (hedging bets) or to restrict the learner to some desired subset. However, unless explicitly stated, we will assume $\mathcal{A} = \mathcal{Y}$.

https://doi.org/10.1017/9781107338548.002 Published online by Cambridge University Press

2.2.5 Other Learning Paradigms

It is also important to consider cases where a classifier has more than two classes or a real-valued output. Indeed, the spam filter SpamBayes, which we study in Chapter 5, uses a third label, *unsure*, to allow the end-user to examine these potential spam more closely. However, generalizing the analysis of errors to more than two classes is not straightforward, and furthermore most systems in practice make a single fundamental distinction (for example, regardless of the label applied by the spam filter, the end-user will ultimately decide to treat each class as either junk messages or legitimate mail). For these reasons, and in keeping with common practice in the literature, we limit our analysis to binary classification and leave extensions to the multi-class or real-valued prediction as a future discussion.

In Chapter 6, we also study an anomaly detection setting. Like binary classification, anomaly detection consists of making one of two predictions: the data is normal ("−") or the data is anomalous ("+"). Unlike the classification setting, the training data usually only consist of examples from the negative class. Because of this, it is common practice to calibrate the detector to achieve a desired false-positive rate on held-out training data.

There are other interesting learning paradigms to consider such as semi-supervised, unsupervised, and reinforcement learning. However, as they do not directly affect our discussion in this text, we will not discuss these frameworks. For a thorough discussion of different learning settings refer to (Hastie et al. 2003) or (Mitchell 1997).

https://doi.org/10.1017/9781107338548.002 Published online by Cambridge University Press