Chapter 4

Introduction to Prediction Problems

Machine learning addresses many problem settings, which can sometimes feel overwhelming. As a non-exhaustive list, these include supervised learning (with classification and regression); semi-supervised learning; unsupervised learning; completion under missing features; structured prediction; learning to rank; statistical relational learning; active learning; and temporal prediction (with time series prediction and policy evaluation in reinforcement learning and online learning). For some of these settings, such as active learning and reinforcement learning, the data collection is a central part of the algorithm and can significantly determine the quality of the learned predictive models. Most other settings assume that data has been collected—without our ability to influence that collection—and now we simply need to analyze that data and learn the best predictors that we can. In this passive setting, we can either assume that the data is i.i.d.—which is the most common—or that there are dependencies between data points—such as in time series prediction or statistical relational learning. There are also settings where the data is incomplete, say because a user did not fill in their age.

One ontology, therefore, could consider the following dimensions to categorize machine learning problems: passive vs. active; i.i.d. vs. non-i.i.d.; complete vs. incomplete. As with all ontologies, each problem will not perfectly fit into these categories. Further, it is likely that most data collection is not completely passive (even if only because the human modeler influences collection of data), is likely not i.i.d. (even if we intended it to be), and likely has some missing components. Nonetheless, algorithms will make these assumptions, to varying degrees, even if the data does not satisfy those assumptions. For the majority of these notes, we will focus on the simplest setting: passive, i.i.d. and complete. Further, because so many techniques arise from the basic classification and regression models within supervised learning, we will focus mostly on these.

In this chapter, we will first introduce classification and regression and then discuss criteria for selecting functions for classification and regression, to motivate the algorithms developed in later chapters.

4.1 Supervised learning problems

We start by defining a data set $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, where $x_i \in \mathcal{X}$ is the $i$-th object and $y_i \in \mathcal{Y}$ is the corresponding target designation. We usually assume that $\mathcal{X} = \mathbb{R}^d$, in which case $x_i = (x_{i1}, x_{i2}, \ldots, x_{id})$ is a $d$-dimensional vector called a data point or instance. Each dimension of $x_j$ is typically called a feature or an attribute. We will often organize the dataset into a matrix $X \in \mathbb{R}^{n \times d}$ where each row corresponds to a sample and each column corresponds to a feature (see Figure 4.1).

We generally distinguish between two related but different types of prediction problems:
classification and regression. Generally speaking, we have a regression problem when $\mathcal{Y}$ is continuous and a classification problem if $\mathcal{Y}$ is discrete. Additionally, we might specify this distinction in the solution approach, i.e., using a regression approach (logistic regression) or a classification approach (naive Bayes). In both prediction scenarios, we assume that the features are easy to collect for each object (e.g., by measuring the height of a person or the square footage of a house), while the target variable is difficult to observe or expensive to collect (e.g., final selling price of a house). Such situations usually benefit from the construction of a computational model that predicts targets from a set of input values. The model is trained using a set of input objects for which target values have already been collected.

**In regression** we construct a function to predict continuous targets, such as $\mathcal{Y} = \mathbb{R}$ or $\mathcal{Y} = [0, \infty)$. For example, linear regression produces predictions in $\mathbb{R}$, and Poisson regression produces only positive predictions, i.e., in $[0, \infty)$. An example of a regression problem is shown in Table 4.1.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>1250</td>
<td>5</td>
<td>2.85</td>
<td>56,650</td>
<td>12.5</td>
<td>2.35</td>
</tr>
<tr>
<td>$x_2$</td>
<td>3200</td>
<td>9</td>
<td>8.21</td>
<td>245,800</td>
<td>3.1</td>
<td>3.95</td>
</tr>
<tr>
<td>$x_3$</td>
<td>825</td>
<td>12</td>
<td>0.34</td>
<td>61,050</td>
<td>112.5</td>
<td>5.10</td>
</tr>
</tbody>
</table>

Table 4.1: An example of a regression problem: prediction of the price of a house in a particular region. Here, features indicate the size of the house (size) in square feet, the age of the house (age) in years, the distance from the city center (dist) in miles, the average income in a one square mile radius (inc), and the population density in the same area (dens). The target indicates the price a house is sold at, e.g., in hundreds of thousands of dollars.
In classification we construct a function that predicts discrete class labels; this function is typically called a classifier. The cardinality of $\mathcal{Y}$ in classification problems is usually small, e.g. $\mathcal{Y} = \{\text{healthy, diseased}\}$. An example of a data set for classification with $n = 3$ data points and $d = 5$ features is shown in Table 4.2.

Classification problems can be further subdivided into multi-class and multi-label problems. A multi-class problem consists of providing the single label for an input. For example, for (simple) blood-type with $\mathcal{Y} = \{A, B, AB, O\}$, a patient can only be labeled with one of these labels. Within multi-class problems, if there are only two classes, it is called binary classification, such as the example in Table 4.2. In multi-label, an input can be associated with more than one label. An example of a multi-label problem is the classification of text documents into categories such as \{sports, medicine, travel, politics\}. Here, a single document may be related to more than one value in the set; e.g. an article on sports medicine. The learned function can now return multiple outputs.

Typically, to make the outputs more consistent between these two settings, the output for both multi-class and multi-label is an indicator vector. For $m = |\mathcal{Y}|$, the prediction for blood types might be $[0 \ 1 \ 0 \ 0]$ to indicate blood-type B and the prediction for four article labels could be $[1 \ 1 \ 0 \ 0]$ if it is both an article pertaining to sports and medicine.

Structured-output prediction $\mathcal{Y}$ can be a set of structured outputs, e.g. strings, trees, or graphs. This classification scenario is usually referred to as structured-output learning. The cardinality of the output space in structured-output learning problems is often very high. For example, when predicting the functions of a protein, an entire ontology tree needs to be predicted, as certain functionality is a subset of other functionality.

Deciding how to formalize the problem Observe that there does not exist a strict distinction between classification and regression. For example, consider the output space $\mathcal{Y} = \{0, 1, 2\}$. We can treat this as a multi-class classification problem, or we could presume $\mathcal{Y} = [0, 2]$ and learning a regression model. We can then threshold the predictions returned by the regression model, by rounding them to the closest integer.

How do you decide which problem formulation to use? Though the mathematical procedures in machine learning are precise, deciding how to formulate real-world problem is subtle, and so inherently less clear-cut. The selection of a particular way of modeling depends on the analyst and their knowledge of the domain as well as technical aspects of learning. In this example, you could ask: is there inherently an ordering to the outputs $\{0, 1, 2\}$? If not, say they correspond to Prefers apples, Prefers oranges, Prefers bananas, then it may be a poor choice to model the output as an interval, which often implies ordering. On the other hand, regression functions can be easier to learn and often produce surprisingly good classification predictions. Further, if there is an ordering to these class, say Good, Better, Best, then most classification models—which do not assume an ordering on the outputs—would not be able to take advantage of this ordering to improve prediction performance.

Selecting the function class, and objective, just like selecting distributions and priors, is an important step in using machine learning effectively. Fortunately, there is a wealth of knowledge, especially empirically, that can guide this selection. As you learn more about the methods, combined with some information about structure in your domain, you will become better at specifying the model class.
Table 4.2: An example of a binary classification problem: prediction of a disease state for a patient. Here, features indicate weight (wt), height (ht), temperature (T), systolic blood pressure (sbp), and diastolic blood pressure (dbp). The class labels indicate presence of a particular disease, e.g., diabetes. This data set contains one positive data point ($x_2$) and two negative data points ($x_1$, $x_3$). The class label shows a disease state, i.e., $y_i = +1$ indicates the presence while $y_i = -1$ indicates absence of disease.

### 4.2 Unsupervised learning and semi-supervised learning

Datasets are not always complete: in some cases, we can only get labels for a small subset of instances, or we cannot get any labels at all. For example, when predicting whether a cat is in an image or not, we would need a human to take each image and label it with a 0 or 1. This labeling can be expensive, and so we can only expect that a small number of all pictures with cats have such an associated label. Using supervised learning only on this labeled subset is likely to produce a poor predictor, because of limited data. Semi-supervised learning deals with taking advantage of all the unlabeled data, to supplement the small labeled dataset, by finding structure in the features. For example, the features may lie on a lower-dimensional manifold; this structure could be inferred from the unlabeled data, and potentially more effectively restrict the function class to learn on the labeled dataset. Unsupervised learning is focused only on obtaining this structure, without the goal of learning a function to predict targets, because no targets are provided. Unsupervised learning will be discussed in Section 9.2.2, as a part of representation learning.

These two problem settings can be seen as an instance of a larger setting of learning under missing data. In general, it may not only be difficult to gather the outputs, but also some of the features. For example, when collecting patient data, it is likely that some patients will omit some information. Even though it is difficult to gather the information “has disease”, it can also be difficult to ensure that other more straightforward data like “age” or “weight” is gathered. Further, one might even ask why there is a distinction between features and targets: they are all associated information about one item, like a patient. Given that a patient does have a disease, you may want to use this feature and their age to predict their weight—which they so blithely chose not to disclose. This more general way of approaching the problem can be useful when data is missing and leads to the general problem of completion. Different techniques are often used in such a setting, and we will not address it further until Section 9.2.2. For now, we keep the focus on supervised learning, which we will still be able to use even when some features are missing, using some simple heuristics for dealing with this missing data.
4.3 Optimal classification and regression models

Our goal now is to establish the performance criteria that will be used to evaluate predictors $f : \mathcal{X} \rightarrow \mathcal{Y}$ and subsequently define optimal classification and regression models. We start with classification and consider a situation where the joint probability distribution $p(x, y)$ is either known or can be learned from data. Also, we will consider that we are given a cost function $c : \mathcal{Y} \times \mathcal{Y} \rightarrow [0, \infty)$, where for each prediction $\hat{y}$ and true target value $y$ the classification cost can be expressed as a constant $c(\hat{y}, y)$, regardless of the input $x \in \mathcal{X}$ given to the classifier. This cost function can simply be stored as a $|\mathcal{Y}| \times |\mathcal{Y}|$ cost matrix.

The criterion for optimality of a classifier will be probabilistic. In particular, we are interested in minimizing the expected cost

$$E[C] = \int_{\mathcal{X}} \sum_{y} c(\hat{y}, y)p(x, y)dx$$

$$= \int_{\mathcal{X}} p(x) \sum_{y} c(\hat{y}, y)p(y|x)dx,$$

where the integration is over the entire input space $\mathcal{X} = \mathbb{R}^d$. From this equation, we can see that the optimal classifier can be expressed as

$$f_{BR}(x) = \arg\min_{\hat{y} \in \mathcal{Y}} \left\{ \sum_{y} c(\hat{y}, y)p(y|x) \right\},$$

for any $x \in \mathcal{X}$. We will refer to this classifier as the Bayes risk classifier. One example where the Bayes risk classifier can be useful is the medical domain. Suppose our goal is to decide whether a patient with a particular set of symptoms $(x)$ should be sent for an additional lab test ($y = 1$ if yes and $y = -1$ if not), with cost $c_{\text{lab}}$, in order to improve diagnosis. However, if we do not perform a lab test and the patient is later found to have needed the test for proper treatment, we may incur a significant penalty, say $c_{\text{lawsuit}}$. If $c_{\text{lawsuit}} \gg c_{\text{lab}}$, as it is expected to be, then the classifier needs to appropriately adjust its outputs to account for the cost disparity in different forms of incorrect prediction.

In many practical situations, however, it may not be possible to define a meaningful cost matrix and, thus, a reasonable criterion would be to minimize the probability of a classifier’s error $P(f(x) \neq y)$. This corresponds to the situation where the cost function is defined as

$$c(\hat{y}, y) = \begin{cases} 0 & \text{when } y = \hat{y} \\ 1 & \text{when } y \neq \hat{y} \end{cases}$$

After plugging these values in the definition for $f_{BR}(x)$, the Bayes risk classifier simply becomes the maximum a posteriori (MAP) classifier. That is,

$$f_{\text{MAP}}(x) = \arg\max_{y \in \mathcal{Y}} \{p(y|x)\}.$$  

Therefore, if $p(y|x)$ is known or can be accurately learned, we are fully equipped to make the prediction that minimizes the total cost. In other words, we have converted the problem
of minimizing the expected classification cost or probability of error, into the problem of
learning functions, more specifically learning probability distributions.

The analysis for regression is a natural extension of that for classification. Here too,
we are interested in minimizing the expected cost of prediction of the true target
\( y \) when a predictor \( f(x) \) is used. The expected cost can be expressed as

\[
E[C] = \int_{\mathcal{X}} \int_{\mathcal{Y}} c(f(x), y) p(x, y) dy dx,
\]

where \( c : \mathbb{R} \times \mathbb{R} \to [0, \infty) \) is again some cost function between the predicted value \( f(x) \) and
the true value \( y \). For simplicity, we will consider

\[
c(f(x), y) = (f(x) - y)^2,
\]

which results in

\[
E[C] = \int_{\mathcal{X}} \int_{\mathcal{Y}} (f(x) - y)^2 p(x, y) dy dx
\]

\[
= \int_{\mathcal{X}} p(x) \int_{\mathcal{Y}} (f(x) - y)^2 p(y|x) dy dx.
\]

Assuming \( f(x) \) is flexible enough to be separately optimized for each unit volume \( dx \), we see that minimizing \( E[C] \) leads us to the problem of minimizing

\[
g(u) = \int_{\mathcal{Y}} (u - y)^2 p(y|x) dy,
\]

where we used a substitution \( u = f(x) \). We can now differentiate \( g \) with respect to \( u \) as

\[
\frac{\partial g(u)}{\partial u} = 2 \int_{\mathcal{Y}} (u - y) p(y|x) dy = 0
\]

\[
\implies u \int_{\mathcal{Y}} p(y|x) dy = \int_{\mathcal{Y}} y p(y|x) dy = 1
\]

which results in the optimal solution

\[
f^*(x) = \int_{\mathcal{Y}} y p(y|x) dy
\]

\[
= \mathbb{E}[Y|x].
\]

Therefore, the optimal regression model in the sense of minimizing the square error between
the prediction and the true target is the conditional expectation \( \mathbb{E}[Y|x] \). It may appear that
in the above equations, setting \( f(x) = y \) would always lead to \( E[C] = 0 \). Unfortunately, this
would be an invalid operation because for a single input \( x \) there may be multiple possible
outputs \( y \) and they can certainly appear in the same data set. To be a well-defined function,
\( f(x) \) must always have the same output for the same input. \( E[C] = 0 \) can only be achieved
if \( p(y|x) \) is a delta function for every \( x \).

Having found the optimal regression model, we can now write the expected cost in the
cases of both optimal and suboptimal models \( f(x) \). That is, we are interested in expressing
\( E[C] \) when
1. \( f(x) = \mathbb{E}[Y|x] \)

2. \( f(x) \neq \mathbb{E}[Y|x] \).

We have already found \( \mathbb{E}[C] \) when \( f(x) = \mathbb{E}[Y|x] \). The expected cost can be simply expressed as

\[
\mathbb{E}[C] = \int_{\mathcal{X}} \int_{\mathcal{Y}} (\mathbb{E}[Y|x] - y)^2 p(x, y) dy dx,
\]

which corresponds to the (weighted) squared error between the optimal prediction and the target everywhere in the feature space. This is the best scenario in regression; we cannot achieve a lower cost on average for this squared cost.

The next situation is when \( f(x) \neq \mathbb{E}[Y|x] \). Here, we will proceed by decomposing the squared error as

\[
(f(x) - y)^2 = (f(x) - \mathbb{E}[Y|x] + \mathbb{E}[Y|x] - y)^2
= (f(x) - \mathbb{E}[Y|x])^2 + 2(f(x) - \mathbb{E}[Y|x])(\mathbb{E}[Y|x] - y) + (\mathbb{E}[Y|x] - y)^2
\]

We will now take a more detailed look at \( g(x, y) \) when placed under the expectation over \( p(x, y) \)

\[
\mathbb{E}[g(X, y)] = \int_{\mathcal{X}} \int_{\mathcal{Y}} (f(x) - \mathbb{E}[Y|x])(\mathbb{E}[Y|x] - y)p(x, y) dy dx
= \int_{\mathcal{X}} \int_{\mathcal{Y}} f(x)\mathbb{E}[Y|x]p(x, y) dy dx - \int_{\mathcal{X}} \int_{\mathcal{Y}} f(x)y p(x, y) dy dx
+ \int_{\mathcal{X}} \int_{\mathcal{Y}} \mathbb{E}[Y|x]yp(x, y) dy dx - \int_{\mathcal{X}} \int_{\mathcal{Y}} \mathbb{E}[Y|x]\mathbb{E}[Y|x]p(x, y) dy dx
= \int_{\mathcal{X}} f(x)\mathbb{E}[Y|x]p(x) dx - \int_{\mathcal{X}} f(x)p(x) \int_{\mathcal{Y}} yp(y|x) dy dx
- \int_{\mathcal{X}} \mathbb{E}[Y|x]\mathbb{E}[Y|x]p(x) dx
= 0.
\]

Therefore, we can now express the expected cost as

\[
\mathbb{E}[C] = \int_{\mathcal{X}} \int_{\mathcal{Y}} (f(x) - y)^2 p(x, y) dy dx
\]

\[
= \int_{\mathcal{X}} (f(x) - \mathbb{E}[Y|x])^2 p(x) dx + \int_{\mathcal{X}} \int_{\mathcal{Y}} (\mathbb{E}[Y|x] - y)^2 p(x, y) dy dx,
\]

where the first term is the “distance” between the trained model \( f(x) \) and the optimal model \( \mathbb{E}[Y|x] \) and the second term is the “distance” between the optimal model \( \mathbb{E}[Y|x] \) and the correct target value \( y \). These terms are also often called the reducible and irreducible errors. If we extend the class of functions \( f \) to predict \( \mathbb{E}[Y|x] \), we can reduce the first expected error. However, the second distance is an inherent error, where for any inputs \( x \), there is a distribution over possible values that we can observe. This relates to the problem of
partial observability, where there is always some stochasticity due to a lack of information. This irreducible distance could potentially be further reduced by providing more feature information (i.e., extending the information in $x$). However, for a given dataset, with the given features, this error is irreducible.

To sum up, we argued here that optimal classification and regression models critically depend on knowing or accurately learning the posterior distribution $p(y|x)$. This task can be solved in different ways, but a straightforward approach is to assume a functional form for $p(y|x)$, say $p(y|x, \theta)$, where $\theta$ is a set of weights or parameters that are to be learned from the data. Alternatively, we can learn the class-conditional and prior distributions, $p(x|y)$ and $p(y)$, respectively. Using

$$p(y|x) = \frac{p(x, y)}{p(x)} = \frac{p(x|y)p(y)}{\sum_y p(x, y)} = \frac{p(x|y)p(y)}{\sum_y p(x|y)p(y)}$$

we can see that these two learning approaches are equivalent in theory. The choice depends on our prior knowledge and/or preferences.

Models obtained by directly estimating $p(y|x)$ are called discriminative models and models obtained by directly estimating $p(x|y)$ and $p(y)$ are called generative models. Direct estimation of the joint distribution $p(x, y)$ is relatively rare as it may be more difficult to hypothesize its parametric or non-parametric form from which the parameters are to be found. Finally, in some situations we can train a model without an explicit probabilistic approach in mind. In these cases, we typically aim to show a good performance of an algorithm in practice, say on a large number of data sets, according to a performance measure relevant to the problems at hand.

\section*{4.4 [Advanced] Bayes Optimal Models}

We saw earlier that optimal prediction models reduce to the learning of the posterior distribution $p(y|x)$ which is then used to minimize the expected cost (risk, loss). However, in practice, the probability distribution $p(y|x)$ must be modeled using a particular functional form and a set of tunable coefficients. When $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{Y} = \{0, 1\}$, one such example is used in logistic regression, where

$$p(1|x) = \frac{1}{1 + e^{-(w_0 + \sum_{j=1}^d w_j x_j)}}$$

and $p(0|x) = 1 - p(1|x)$. Here $(w_0, w_1, \ldots, w_d) \in \mathbb{R}^{d+1}$ is a set of weights that are to be inferred from a given data set $\mathcal{D}$ and $x \in \mathbb{R}^d$ is an input data point. A number of other types of functional relationships can be used as well, providing a vast set of possibilities for modeling distributions. We will adjust our notation to more precisely denote the posterior distribution as

$$p(y|x) = p(y|x, f),$$

70
where $f$ is a particular function from some function (hypothesis) space $\mathcal{F}$. We can think of $\mathcal{F}$ as a set of all functions from a specified class, say for all $(w_0, w_1, \ldots, w_d) \in \mathbb{R}^{k+1}$ in the example above, but we can also extend the functional class beyond simple parameter variation to incorporate non-linear decision surfaces.

In a typical learning problem, we are given a data set $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$ and are asked to model $p(y|x)$. For this purpose, we will think of $\mathcal{D}$ as a realization of a random variable $\mathcal{D}$ and will assume that $\mathcal{D}$ was drawn according to the true underlying distribution $p(x, y)$. Thus, our task is to express $p(y|x, \mathcal{D})$. Using the sum and product rules, will rewrite our original task of estimating $p(y|x)$ as

$$
p(y|x, \mathcal{D}) = \int_{\mathcal{F}} p(y|x, f, \mathcal{D})p(f|x, \mathcal{D})df
$$

Here we used conditional independence between output $Y$ and the data set $\mathcal{D}$ once a particular model $f$ was selected based on $\mathcal{D}$; thus $p(y|x, f, \mathcal{D}) = p(y|x, f)$. This equation, gives us a sense that the optimal decision can be made through a mixture of distributions $p(y|x, f)$, where the weights are given as posterior densities $p(f|x, \mathcal{D})$. In finite hypothesis spaces $\mathcal{F}$ we have that

$$
p(y|x, \mathcal{D}) = \sum_{f \in \mathcal{F}} p(y|x, f)p(f|x, \mathcal{D}),
$$

and $p(f|x, \mathcal{D})$ are posterior probabilities. We may further assume that $p(f|x, \mathcal{D}) = p(f|\mathcal{D})$, in which case the weights can be precomputed based on the given data set $\mathcal{D}$. This leads to more efficient calculations of the posterior probabilities.

In classification, we can rewrite our original MAP classifier as

$$
f_{\text{MAP}}(x, \mathcal{D}) = \arg \max_{y \in \mathcal{Y}} \{p(y|x, \mathcal{D})\},
$$

which readily leads to the following formulation

$$
f_{\text{MAP}}(x, \mathcal{D}) = \arg \max_{y \in \mathcal{Y}} \left\{ \int_{\mathcal{F}} p(y|x, f, \mathcal{D})p(f|\mathcal{D})df \right\}
$$

$$
= \arg \max_{y \in \mathcal{Y}} \left\{ \int_{\mathcal{F}} p(y|x, f)p(\mathcal{D}|f)p(f)df \right\}.
$$

It can be shown that no classifier can outperform the Bayes optimal classifier. Interestingly, the Bayes optimal model also hints that a better prediction performance can be achieved by combining multiple models and averaging their outputs. This provides theoretical support for ensemble learning and methods such as bagging and boosting.

One problem in Bayes optimal classification is efficient calculation of $f_{\text{MAP}}(x, \mathcal{D})$, given that the function (hypothesis) space $\mathcal{F}$ is generally uncountable. One approach to this is sampling of functions from $\mathcal{F}$ according to $p(f)$ and then calculating $p(f|\mathcal{D})$ or $p(\mathcal{D}|f)$. This can be computed until $p(y|x, \mathcal{D})$ converges.