52 NEAREST-NEIGHBOR RULE

We encountered one instance of Bayesian inference in Chapter 50 based on the quadratic loss in the context of mean-square-error estimation. We explained there that the optimal solution, for inferring a hidden zero-mean random variable $\boldsymbol{x}$ from observations of another zero-mean random variable $\boldsymbol{y}$, is given by the conditional estimator, $\mathbb{E}\,(\boldsymbol{x}|\boldsymbol{y})$, whose computation requires knowledge of the conditional distribution, $f_{\boldsymbol{x}|\boldsymbol{y}}(x|y)$. Even when the estimator $\widehat{\boldsymbol{x}} = c(\boldsymbol{y})$ is restricted to affine functions of $\boldsymbol{y}$, the solution continues to require knowledge of some statistical moments of $\{\boldsymbol{x},\boldsymbol{y}\}$ in the form of their variances or covariances, $\{\sigma_x^2, r_{xy}, R_y\}$. We addressed this challenge in the last two chapters by using a collection of training data measurements $\{x(n), y_n\}$ arising from the joint distribution $f_{\boldsymbol{x},\boldsymbol{y}}(x,y)$ to replace the stochastic risk, $\mathbb{E}\,(\boldsymbol{x}-\boldsymbol{y}^{\mathsf T}w)^2$, by an empirical least-squares risk, with and without regularization such as:

$$w^\star \;\stackrel{\Delta}{=}\; \underset{w\in\mathbb{R}^M}{\operatorname{argmin}} \left\{ \alpha\|w\|_1 \;+\; \rho\|w\|^2 \;+\; \frac{1}{N}\sum_{n=0}^{N-1}\left(x(n)-y_n^{\mathsf T}w\right)^2 \right\} \tag{52.1}$$

where α and ρ are nonnegative regularization factors.

Moving forward, we will consider more general Bayesian inference problems involving other types of loss functions $Q(\boldsymbol{x},\widehat{\boldsymbol{x}})$, besides the quadratic loss:

$$\widehat{\boldsymbol{x}}_Q \;\stackrel{\Delta}{=}\; \underset{\widehat{\boldsymbol{x}}=c(\boldsymbol{y})}{\operatorname{argmin}} \left\{ \mathbb{E}\,Q(\boldsymbol{x},\widehat{\boldsymbol{x}}) \right\} \tag{52.2}$$

We already know from result (28.5) that here too the optimal solution $\widehat{\boldsymbol{x}}_Q$ requires knowledge of the conditional pdf, $f_{\boldsymbol{x}|\boldsymbol{y}}(x|y)$, since

$$\widehat{x}_Q \;\stackrel{\Delta}{=}\; \underset{\widehat{x}=c(y)}{\operatorname{argmin}} \left\{ \mathbb{E}_{\boldsymbol{x}|\boldsymbol{y}}\left(Q(\boldsymbol{x},\widehat{x}|\boldsymbol{y}=y) \right) \right\} \tag{52.3}$$

where the expectation of the loss function is evaluated relative to $f_{\boldsymbol{x}|\boldsymbol{y}}(x|y)$. We will follow two paths. One path is similar to what we did in the previous two chapters for mean-square-error estimation. We will replace the stochastic risk in (52.2) by an empirical risk, add regularization, use an affine model for $c(\boldsymbol{y})$, and then apply some stochastic approximation algorithm to learn the solution. This construction would amount to solving problems of the form:

$$w^\star \;\stackrel{\Delta}{=}\; \underset{w\in\mathbb{R}^M}{\operatorname{argmin}} \left\{ q(w) \;+\; \frac{1}{N}\sum_{n=0}^{N-1} Q(x(n),\widehat{x}(n)) \right\}, \quad \widehat{x}(n) = y_n^{\mathsf T}w \tag{52.4}$$

where $q(w)$ denotes the regularization factor. This first approach will be studied at great length in later chapters in the context of the Perceptron algorithm, support vector machines, kernel methods, neural networks, and other related methods. The main difference between these methods will be the choice of the loss function $Q(\cdot,\cdot)$ and the way by which the predictor $\widehat{\boldsymbol{x}}$ is constructed from $\boldsymbol{y}$. While most methods will employ affine constructions, kernel methods and the neural network structure will allow for some nonlinear mappings from $\boldsymbol{y}$ to $\boldsymbol{x}$.

In the current and next few chapters, however, we will follow a second more direct path to solving (52.2). We will introduce *data-based* methods that infer either the conditional pdf $f_{\boldsymbol{x}|\boldsymbol{y}}(x|y)$ or the joint pdf $f_{\boldsymbol{x},\boldsymbol{y}}(x,y)$ directly from the data, rather than minimize an empirical risk. In these investigations, we will focus on the important case of predicting the label of a random variable $\boldsymbol{x}$ from observations $\boldsymbol{y}$. Specifically, we will focus on the classification problem where $\boldsymbol{x}$ is *discrete* and assumes one of two binary values, $+1$ or -1. We will also consider multi-class problems where $\boldsymbol{x}$ can assume one of a multitude of discrete levels.

NOTATION: Regression vs. Classification

Before proceeding, we motivate a change in notation. Form this point onwards in our presentation, we will be dealing mainly with classification problems where the unknown $\boldsymbol{x}$ assumes discrete values. The variable $\boldsymbol{x}$ can be either binary-valued such as $\boldsymbol{x} \in \{-1,+1\}$ or $\boldsymbol{x} \in \{0,1\}$, or multi-valued such as assuming integer values $\boldsymbol{x} \in \{1,2,\ldots,R\}$. In order to emphasize the fact that the hidden variable is *discrete*, we will henceforth use the Greek symbol $\boldsymbol{\gamma}$ to refer to a binary discrete variable and the normal symbol $\boldsymbol{r}$ to refer to a multi-level discrete variable:

$$\begin{aligned}
&(\textit{notation for discrete hidden variables}) \\
&\gamma \in \{-1,1\} \;\text{ or }\; \gamma \in \{0,1\}, \qquad (\textbf{binary values}) \qquad (52.5a)\\
&\boldsymbol{r} \in \{1,2,3,\ldots,R\}, \qquad (\textbf{integer values}) \qquad (52.5b)
\end{aligned}$$

Both $\boldsymbol{\gamma}$ and $\boldsymbol{r}$ are random variables, just like the notation $\boldsymbol{x}$. By introducing these symbols, it becomes easier for the reader to recognize whether a statement is dealing with discrete or continuous variables, a classification or regression problem, and whether the discrete variable itself is binary or multi-level. We will refer to $\{\boldsymbol{\gamma},\boldsymbol{r}\}$ as the *class* or *label* variable. For similar reasons, we will replace the observation variable $\boldsymbol{y}$ by the letter $\boldsymbol{h}$ and refer to it as the *feature* vector. In this way, regression problems deal with variables $(\boldsymbol{x},\boldsymbol{y})$ while classification problems deal with variables $(\boldsymbol{\gamma},\boldsymbol{h})$ or $(\boldsymbol{r},\boldsymbol{h})$:

$$\left\{\begin{array}{l}\text{notation } (\boldsymbol{x},\boldsymbol{y}) \text{ reserved for regression/estimation problems}\\ \text{notation } (\boldsymbol{\gamma},\boldsymbol{h}) \text{ or } (\boldsymbol{r},\boldsymbol{h}) \text{ reserved for classification problems}\end{array}\right. \qquad (52.6)$$

In the context of classification, each entry of $\boldsymbol{h}$ is called an *attribute*. These entries will generally assume numerical values, but they can also be categorical, such as when an attribute refers to the color of an object (say, red, blue, or yellow) or its size (say, small, medium, or large). It is customary to transform categorical entries into numerical values, as explained in a later chapter when we discuss decision trees, so that it is sufficient for our purposes to treat $\boldsymbol{h}$ as a vector with numerical entries..

52.1 BAYES CLASSIFIER

We review briefly the Bayes classifier solution from Sec. 28.3 in view of the new notation for classification problems. Given a feature vector $\boldsymbol{h} \in \mathbb{R}^M$, we are interested in deducing its label $\boldsymbol{r} \in \{1, 2, \ldots, R\}$ by seeking a mapping $c(\boldsymbol{h}) : \mathbb{R}^M \rightarrow \{1, 2, \ldots, R\}$ that minimizes the probability of error, namely,

$$\widehat{\boldsymbol{r}}_{\text{bayes}} = \underset{c(\boldsymbol{h})}{\text{argmin}}\; \mathbb{P}\left(c(\boldsymbol{h}) \neq \boldsymbol{r}\right) \tag{52.7}$$

We know from (28.67) that the optimal solution is given by the MAP estimator:

$$\widehat{\boldsymbol{r}}_{\text{bayes}} \;=\; \underset{r \in \{1,2,\ldots,R\}}{\text{argmax}} \;\; \mathbb{P}(\boldsymbol{r} = r | \boldsymbol{h} = h) \tag{52.8}$$

We denote the optimal mapping that corresponds to this construction by $c^{\bullet}(h)$ using the bullet superscript:

$$\widehat{\boldsymbol{r}}_{\text{bayes}} = c^{\bullet}(h), \qquad (\textbf{Bayes classifier}) \tag{52.9}$$

In our notation, the $\bullet$ superscript will refer to the *ideal* solution that we are aiming to achieve. As seen from (52.8), this solution requires knowledge of the conditional probability distribution $\mathbb{P}(\boldsymbol{r} = r|\boldsymbol{h} = h)$, which is generally unavailable. The $\star$ superscript, such as writing $c^{\star}(h)$, will refer to approximations obtained by solving more tractable formulations:

$$\widehat{r} = c^{\star}(h), \qquad (\textbf{approximate classifier}) \tag{52.10}$$

In this and the next few chapters, we will describe *data-based* methods that infer either the conditional pdf $\mathbb{P}(\boldsymbol{r} = r|\boldsymbol{h} = h)$ or the joint probability distribution $f_{\boldsymbol{r},\boldsymbol{h}}(r, h)$ from the data and lead to approximate classifiers $c^{\star}(h)$. Among these methods we list the nearest-neighbor (NN) rule of this chapter, the naïve Bayes classifier, the linear and Fisher discriminant analysis methods (LDA, FDA), and the logistic regression method. Methods that approximate the conditional probabilities $\mathbb{P}(\boldsymbol{r} = r|\boldsymbol{h} = h)$ are referred to as *discriminative*, whereas methods that approximate the joint pdf $f_{\boldsymbol{r},\boldsymbol{h}}(r, h)$, or its components $\mathbb{P}(\boldsymbol{r} = r)$ and the reverse conditional $f_{\boldsymbol{h}|\boldsymbol{r}}(h|r)$, are referred to as *generative*. This is because discriminative techniques allow us to discriminate between the classes, while generative techniques allow us to generate additional data $\{r, h\}$ that mimic the distribution of the training data.

Before explaining the steps involved in the nearest-neighbor construction, it is useful to comment on how performance is assessed in general for classification algorithms that rely on training data. These comments are valid for all learning methods described here and in future chapters.

Classification errors

In classification problems, we make a distinction between training data and test data. The data $\{r(n), h_n\}$ used to train the classifier are referred to as *training data*. Once a classifier $\widehat{r} = c^{\star}(h)$ is learned, we can evaluate its performance on the training data by counting the number of erroneous decisions that would result if the classifier were applied to that data. This measure results in the *training error*, also called the *empirical error rate* and is evaluated as follows:

$$R_{\text{emp}}(c^{\star}) \stackrel{\Delta}{=} \frac{1}{N}\sum_{n=0}^{N-1} \mathbb{I}\left[c^{\star}(h_n) \neq r(n)\right], \quad (\textbf{empirical error on training data}) \tag{52.11}$$

where the notation $\mathbb{I}[x]$ denotes the indicator function defined by:

$$\mathbb{I}[x] \stackrel{\Delta}{=} \begin{cases} 1, & \text{when argument } x \text{ is true} \\ 0, & \text{otherwise} \end{cases} \tag{52.12}$$

The argument of the indicator function in (52.11) is comparing the predicted label $\widehat{r}(n) = c^{\star}(h_n)$ to the true label $r(n)$ for the sample of index n. Note that $R_{\text{emp}}(c^{\star})$ is a number in the range $[0, 1]$ and it measures the empirical probability of error on the training data for the classifier $c^{\star}(h)$. In general, the empirical error will be small because $c^{\star}(h)$ will be determined with the aim of minimizing it.

In most classification applications, however, the main purpose for learning $c^{\star}(h)$ is to employ it to perform inference on future data that were not part of the training phase. For this reason, it is customary to assess performance on a separate collection of T test data points denoted by $\{r(t), h_t\}$, and which were *not* part of the training phase but are assumed to arise from the same underlying distribution $f_{\boldsymbol{h},\boldsymbol{r}}(h, r)$ as the training data. The empirical error rate on the test data is given by

$$R_{\text{emp}}(c^{\star}) \stackrel{\Delta}{=} \frac{1}{T}\sum_{t=0}^{T-1} \mathbb{I}\left[c^{\star}(h_t) \neq r(t)\right], \quad (\textbf{empirical error on test data}) \tag{52.13}$$

where $\widehat{r}(t) = c^{\star}(h_t)$ denotes the prediction for each test label $r(t)$. We use the same symbol $R_{\text{emp}}(\cdot)$ to refer to empirical errors (whether measured on training or test data); it will be clear from the context whether we are referring to one case or the other. In general, the above empirical error on test data will be larger than the empirical error on training data but we desire the gap to be small. Learning algorithms that lead to small error gaps are said to *generalize* well, namely, they are able to extend their good performance on training data to arbitrary test data as well. It is important to emphasize that the ultimate objective of a learning algorithm is not to attain a small empirical error on a *particular* test data. More critically, classifiers should be able to generalize, i.e., they should be able to classify well arbitrary *future* feature vectors from the same data distribution that were *not* part of the original training and test datasets. This is the *crux* of the learning problem. In future Chapter 64, we will develop conditions under which classifiers that perform well on a sufficient amount of test data can be expected to generalize well for other data.

We formally measure the generalization ability of a classifier $c^\star(h)$ by defining its *generalization* error as the following expected value (which we also denote by P_e since, as explained by (52.15), it amounts to the probability of error by the classifier):

$$R(c^\star) \;\stackrel{\Delta}{=}\; \mathbb{E}\,\mathbb{I}\left[c^\star(\boldsymbol{h}) \neq \boldsymbol{r}\right], \qquad (\textbf{generalization error}) \tag{52.14}$$

where the expectation is over the joint probability distribution of the random data $\{\boldsymbol{r}, \boldsymbol{h}\}$. This risk can be expressed in an equivalent form involving the probability of erroneous decisions since

$$\begin{aligned} R(c^\star) &= 1 \times \mathbb{P}\left(c^\star(\boldsymbol{h}) \neq \boldsymbol{r}\right) \;+\; 0 \times \mathbb{P}\left(c^\star(\boldsymbol{h}) = \boldsymbol{r}\right) \\ &= \mathbb{P}(c^\star(\boldsymbol{h}) \neq \boldsymbol{r}) \\ &\stackrel{\Delta}{=} P_e \end{aligned} \tag{52.15}$$

That is,

$$\textbf{probability of error} = \textbf{generalization error} \tag{52.16}$$

We will establish in future Chapter 64 that, under some reasonable conditions on the structure of a classifier (namely, not too simple and not too complex) and on the amount of training data available (which needs to be large enough), the empirical error on a test dataset provides a good approximation for the generalization error, i.e., $R_{\text{emp}}(c^\star) \approx R(c^\star)$. This means that classifiers that perform well on test data are expected to perform well more broadly on the entire population.

Example 52.1 (**The need to generalize**) Consider a collection of $N-$feature vectors $\{h_n \in \mathbb{R}^M\}$ and the corresponding labels $r(n) \in \{1, 2, \ldots, R\}$, for $n = 0, 1, \ldots, N-1$. We can construct a classifier that memorizes the behavior of the training samples perfectly as follows:

$$c(h) = \begin{cases} r(h), & \text{if } h \in \{h_0, h_1, \ldots, h_{N-1}\} \\ r, & \text{selected randomly from } \{1, 2, \ldots, R\} \text{ for any other } h \end{cases} \tag{52.17}$$

That is, the classifier assigns the label $r(n)$ to each vector h coinciding with one h_n from the training set, and assigns a random label r to any other feature vector. Then, the empirical error on the training data for this classifier will be zero (i.e., the smallest it can be), while its empirical error on arbitrary test data can be unacceptably large. We therefore have an example of a classifier that performs exceedingly well on the training data but delivers poor performance on test data. This is an example of *overfitting*.

52.2 K-NN CLASSIFIER

We consider first binary classification problems with label $\gamma \in \{\pm 1\}$. Assume we have access to N pairs of data points $\{\gamma(n), h_n\}$ where $h_n \in \mathbb{R}^M$ is the $n-$th feature vector and $\gamma(n)$ the corresponding label. This collection plays the role of the training dataset. Now, given a new feature h, the objective is to determine its most likely label. The nearest neighbor rule predicts γ as follows.

Neighborhoods

The vectors $\{h, h_n\}$ are points in $M-$dimensional space. We define a neighborhood around h consisting of the $k-$nearest feature vectors from the training set to h, where closeness is measured in terms of the Euclidean distance (or some other distance metric, if desired). Let the notation $\mathcal{N}_k(h)$ refer to the indices of the $k-$closest neighbors to h from within the training set $\{h_n\}$:

$$\mathcal{N}_k(h) \stackrel{\Delta}{=} \Big\{\text{index set of } k-\text{closest neighbors to } h \text{ from training set } \{h_n\}\Big\} \tag{52.18}$$

If we envision a *hypersphere* centered at location h and engulfing the neighborhood of h, then the radius of the sphere should be large enough to include only k neighboring points within $\mathcal{N}_k(h)$. Figure 52.1 illustrates this construction in the plane for $M = 2$ and $k = 5$. In the figure, training features from class $\boldsymbol{\gamma} = +1$ are represented by circles, while training features from class $\boldsymbol{\gamma} = -1$ are represented by squares. The location of h is represented by a triangle. The figure draws a circle around h that encompasses its 5 closest neighbors from the training set. The class of h is then declared to be the one corresponding to the *majority class* among its neighbors. In this case, the feature h is declared to belong to class $+1$ since four of its neighbors belong to this class. In the case of a tie, one can select the class randomly between $+1$ and -1 or set it, by convention, to $+1$.

The $k-$nearest neighbor decision rule can be expressed analytically as follows. We first use the neighborhood around h to count the number of neighbors of h that belong to class $+1$:

$$p(h) \stackrel{\Delta}{=} \frac{1}{k} \sum_{n \in \mathcal{N}_k(h)} \mathbb{I}[\gamma(n) = +1] \tag{52.19}$$

where $\mathbb{I}[x]$ is the indicator function: its value is one when its argument is true and zero otherwise. The division by k transforms $p(h)$ into a measure of the fraction of $+1$ neighbors that exist within $\mathcal{N}_k(h)$. The majority vote then translates into applying the following rule to determine the label of h:

$$\gamma^\star(h) = \begin{cases} +1, & \text{if } p(h) \geq 1/2 \\ -1, & \text{otherwise} \end{cases} \tag{52.20}$$

We refer to this mapping from h to its predicted label $\gamma^\star(h)$ by writing $c^\star(h)$. Note that in effect $p(h)$ is approximating the conditional probability $\mathbb{P}(\boldsymbol{\gamma} =$

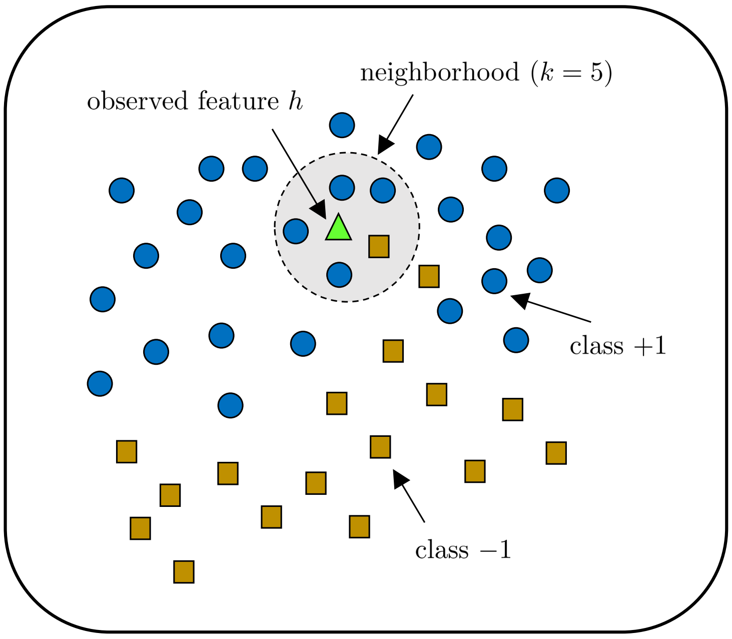

Figure 52.1 The set of 5−nearest neighbors around the feature vector h (represented by the triangle) consists of four circular features (belonging to class $+1$) and one square feature (belonging to class -1). Accordingly, based on a majority vote, the feature h is declared to belong to class $+1$.

$+1|\boldsymbol{h} = h)$ that is needed in the implementation of the Bayes classifier:

$$p(h) = \widehat{\mathbb{P}}(\boldsymbol{\gamma} = +1|\boldsymbol{h} = h) \tag{52.21}$$

In other words, the $k-$nearest neighbor rule uses the training data and the $k-$size neighborhoods to estimate the conditional probabilities $\mathbb{P}(\boldsymbol{\gamma} = +1|\boldsymbol{h} = h)$ locally in order to carry out the classification task.

Example 52.2 **(Weighted $k-$NN)** The traditional $k-$NN rule assigns equal weights to all neighbors of a feature vector h before deciding on its label. It is sometimes natural to expect that neighbors that are closer to h are more likely to belong to the same class as h, than neighbors that are farther away from it. In *weighted* $k-$NN, each neighbor $\ell \in \mathcal{N}_h$ is assigned a nonnegative weight w_ℓ; for convenience, we normalize the weights to add up to one. For example, one (but not the only) way to compute the weights is to determine the distances between h and each of its neighbors in $\mathcal{N}_h$ and to normalize the distances by their sum:

$$d_\ell \stackrel{\Delta}{=} \|h - h_\ell\|, \quad h_\ell \in \mathcal{N}_h \tag{52.22a}$$

$$w_\ell = \frac{d_\ell}{\sum_{\ell' \in \mathcal{N}_h} d_{\ell'}} \tag{52.22b}$$

The resulting decision rule is expressed analytically as follows. We count the *weighted* number of neighbors of h that belong to class $+1$:

$$p(h) \stackrel{\Delta}{=} \sum_{\ell \in \mathbb{N}_k(h)} w_\ell \, \mathbb{I}[\gamma(\ell) = +1] \tag{52.23}$$

and use a majority vote to decide on the label for h:

$$\gamma^\star(h) = \begin{cases} +1, & \text{if } p(h) \geq 1/2 \\ -1, & \text{otherwise} \end{cases} \tag{52.24}$$

Multiclass classification

The nearest neighbor rule can be extended to multiclass classification problems with R classes. In this case, we declare h to belong to the class r that receives the majority of votes within its neighborhood.

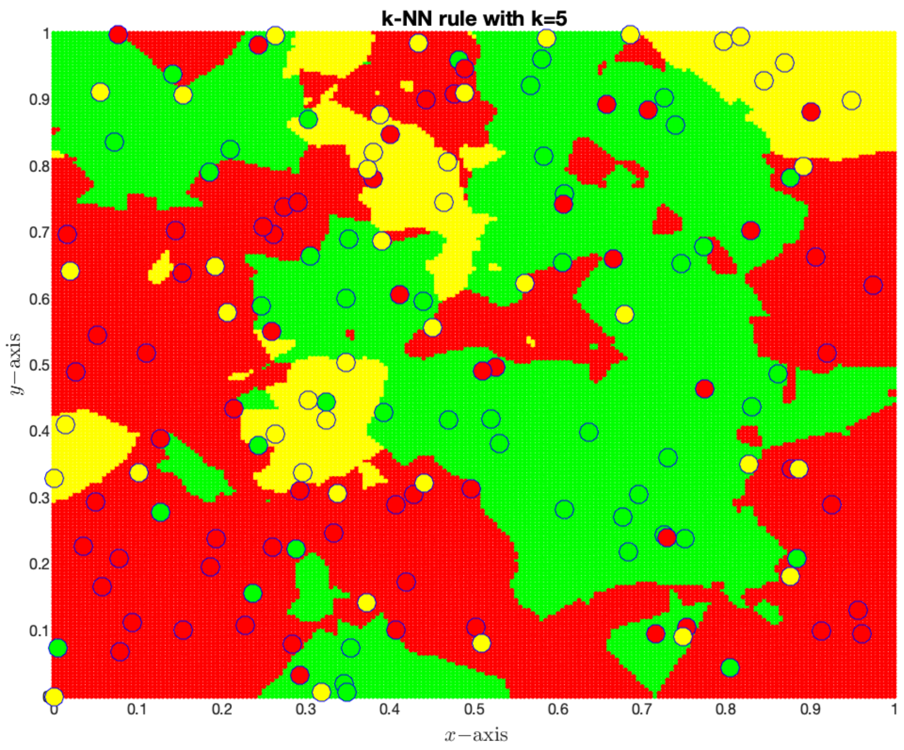

Figure 52.2 The separation regions generated by applying a $5-$NN rule over 150 randomly generated feature vectors $h_n \in \mathbb{R}^2$ arising from 3 classes: green (class $r = 1$), red (class $r = 2$), and yellow (class $r = 3$).

Figure 52.2 illustrates the separating regions in the plane that would result for $k = 5$ neighbors and $R = 3$ classes. The training data are represented by the colored circles (green for $r = 1$, red for $r = 2$, and yellow for $r = 3$). The colored regions represent the class that would be assigned to any feature vector falling into the region. For example, if a location in the plane is colored in red, the color indicates that the majority of the 5 neighbors to this location will belong to class $r = 2$. Therefore, any feature h falling into the red region will be assigned to class $r = 2$, and similarly for the two other colored regions. This figure was generated using a total of $N = 150$ random training points within the region $[0,1] \times [0,1]$.

The nearest-neighbor (NN) rule is a discriminative method that approximates $\mathbb{P}(\boldsymbol{r} = r|\boldsymbol{h} = h)$ directly from the training data, and does not make any assumption about the form of these probabilities. It is an example of a *nonparametric* learning method, which does not involve learning the parameters of separating surfaces as will happen with other learning methods discussed in future chapters such as the Perceptron, support vector machines, and neural networks. The NN rule operates directly on the available data and does not even involve a training phase. While the NN construction is straightforward, we will find that it suffers from several important challenges.

Voronoi diagrams

When $k = 1$ (i.e., when classification is decided by considering only the label of the closest neighbor), we can partition the feature space into a Voronoi diagram consisting of cells. Each cell n is characterized by a seed point h_n, which is one of the points from the training set. The boundaries of each cell n define a region in space consisting of all $M-$dimensional vectors, h, that are closest to h_n, than to any other seed. These boundaries can be determined as follows. If we draw line segments connecting any particular seed point, h_n, to its neighboring seeds, then the boundaries of cell n would be determined from the bisecting lines that cut these segments in half. Figure 52.3 illustrates this construction for $M = 2$. A total of $N = 100$ random feature vectors are generated in the region $[0,1] \times [0,1]$ and the resulting Voronoi diagram is shown. The lines specify the equidistant boundaries between adjacent feature points; points from class $+1$ are denoted in green while points from class -1 are denoted in red. Once the Voronoi diagram is generated, classification by means of the $1-$NN rule is achieved automatically as follows. Given a new feature vector, h, we determine the cell that it falls into. Then, the class of h is selected to match the class of the seed for that cell.

52.3 PERFORMANCE GUARANTEE

There is a fundamental and reassuring result on the performance of the nearest-neighbor classifier. It is sufficient to describe the result for the $1-$NN rule; a similar conclusion applies to $k-$NN and is described in the comments at the end of the chapter. Let $c^{\bullet}(h)$ denote the optimal Bayes classifier (52.8). This classifier minimizes the probability of misclassification and delivers the label $\boldsymbol{r}^{\bullet}(h)$. We denote the smallest probability of error that is attained by the classifier by P_e^{bayes}.

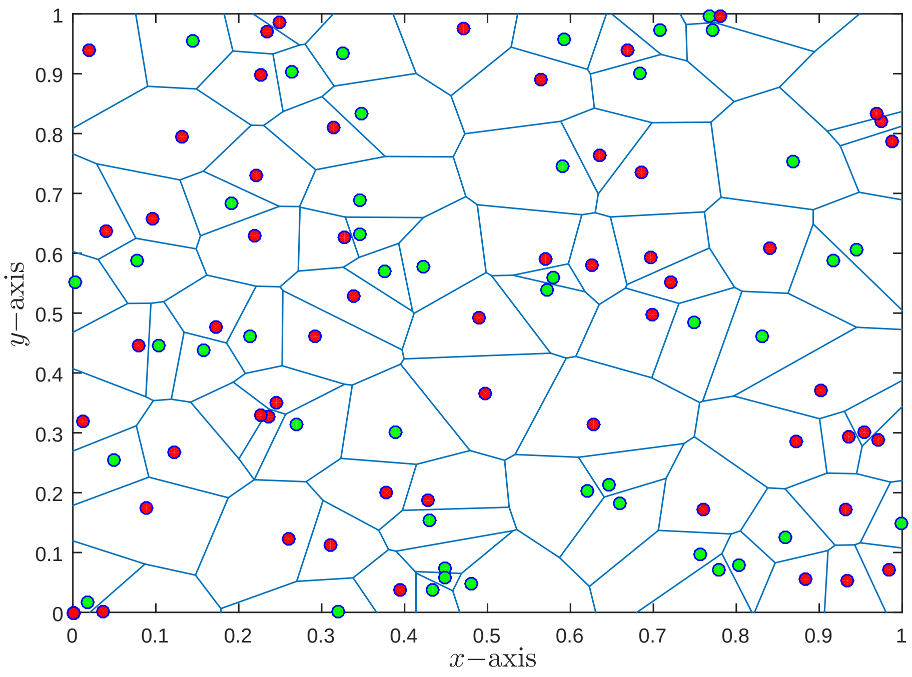

Figure 52.3 Voronoi diagram for 100 randomly generated feature vectors $h_n \in \mathbb{R}^2$. Points in green belong to class $+1$ while points in red belong to class -1.

Let $c^\star(h)$ denote the $1-$NN classifier that results from the training data $\{r(n), h_n\}$ of size N. This is the classifier that is defined by the Voronoi diagram corresponding to this data. The generalization error (or probability of misclassification) for this classifier over the *entire* distribution of the data is given by expression (52.15), which we denote by:

$$P_e \overset{\Delta}{=} \mathbb{P}(c^\star(\boldsymbol{h}) \neq \boldsymbol{r}) \;=\; R(c^\star) \tag{52.25}$$

The following classical result now holds; its proof appears in Appendix 52.A — see Prob. 52.1 for an alternative argument.

Theorem 52.1. (Generalization error of $1-$NN classifiers) *Consider a multi-class classification problem with R labels, $r \in \{1, 2, \ldots, R\}$. Let P_e^{bayes} denote the smallest probability of error attained by the Bayes classifier (52.8). Let P_e denote the probability of error attained by the $1-$NN classifier (i.e., its generalization error). Then, for independent realizations $\{r(n), h_n\}$ and for large sample sizes $N \to \infty$, it holds that*

$$P_e^{\text{bayes}} \;\leq\; P_e \;\leq\; P_e^{\text{bayes}}\left(2 - \frac{R}{R-1}P_e^{\text{bayes}}\right) \leq 2P_e^{\text{bayes}} \tag{52.26}$$

Result (52.26) means that the probability of error of the $1-$NN classifier is at most twice as bad as the best possible performance given by the optimal Bayes classifier. The result also means that any other classifier structure can at most reduce the probability of error of $1-$NN by one half.

Challenges

While the $k-$NN rule is appealing, it faces some important challenges that limit its performance. The classifier is sensitive to noise and outliers, and requires that the training data be stored and processed continuously. Moreover, the following properties are evident:

(C1) The classifier treats *equally* all entries (i.e., attributes) of the feature vector. If, for example, some attributes are more relevant to the classification task than the remaining attributes, this aspect is ignored by the $k-$NN implementation because all entries in the feature vector contribute similarly to the calculation of Euclidean distances and the determination of neighborhoods.

(C2) The $k-$NN classifier does not perform well in high-dimensional feature spaces when M is large for at least two reasons:

- **(a)** First, for each new feature h, the classifier needs to perform a search over the entire training set to determine the neighborhood of h. This step is demanding for large M and N.
- **(b)** Second, and more importantly, in high-dimensional spaces, the training samples $\{h_n\}$ only provide a *sparse* representation for the behavior of the data distribution $f_{\boldsymbol{r},\boldsymbol{h}},(r,h)$. The available training examples need not be enough for effective learning.

We comment on these issues in the sequel, and in the next chapter, and explain how clustering helps ameliorate some of these difficulties.

52.4 K-MEANS ALGORITHM

One way to address challenge (C2a) and reduce the complexity of the search step is to cluster the training data into a small number of clusters, and to base the classification decision on comparisons against the clusters rather than against the entirety of the training dataset. Clustering is a procedure that partitions the N feature vectors $\{h_n \in \mathbb{R}^M\}$ into a small collection of K groups (called *clusters*), with the expectation that vectors within the same group share similar properties. One popular method to perform clustering is Lyold algorithm, also known as the $k-$means algorithm, which operates as follows.

Algorithm

We select the desired number of clusters, K, and assign to each cluster k an initial mean vector denoted by $\mu_k \in \mathbb{R}^M$. There are several ways by which these initial vectors can be chosen (and their choice influences the performance of the clustering algorithm) — we describe three methods further ahead. Once the initial vectors have been chosen, then the $k-$means algorithm applies repeatedly the operations shown in listing (52.29) and continually updates the mean vectors $\{\mu_k\}$ as follows:

(1) Each feature vector h_n in the training set is assigned to the cluster whose mean μ_k is the closest to h_n (if there are multiple possible clusters, we select one of them at random):

$$\text{cluster for } h_n \stackrel{\Delta}{=} \underset{1\leq k\leq K}{\text{argmin}} \, \|h_n - \mu_k\| \tag{52.27}$$

Let the notation $\mathcal{C}_k$ represent a generic cluster k (i.e., the collection of the indexes of all feature vectors in it).

(2) Following the assignments of the $\{h_n\}$ to clusters, the mean vector μ_k for each cluster is updated by averaging the feature vectors that ended up within that cluster:

$$\mu_k = \frac{1}{|\mathcal{C}|_k} \sum_{n\in\mathcal{C}_k} h(n) \tag{52.28}$$

where $|\mathcal{C}_k|$ denotes the cardinality of the set (the number of its elements).

Observe that this algorithm performs clustering in an *unsupervised* manner; it acts directly on the feature vectors and does not require any class information. The performance of the algorithm is, however, sensitive to the choice of K, the presence of outliers, and the selection of the initial mean vectors.

$k-$means algorithm (also known as Lyold algorithm)

```
given $N$ feature vectors $\{h_n\}$, of size $M \times 1$ each;
given the number of clusters, $K$;
select $K$ initial mean vectors $\{\mu_k\}$, one for each cluster.
repeat until convergence :
    assign each $h_n$ to the cluster with the closest mean $\mu_k$;
    for each cluster $k$, replace $\mu_k$ by the average of all vectors in it;
end
return clusters $\{\mathcal{C}_k\}$ and their means $\{\mu_k\}$
```
(52.29)

Interpretation and derivation

We explain in the comments at the end of the chapter that the $k-$means clustering algorithm is related to the expectation-maximization (EM) algorithm described earlier in Chapter 32 for Gaussian mixture models. Both algorithms perform clustering and the main difference is that the $k-$means method performs *hard* assignments of samples h_n by assigning them to the cluster of the closest mean vector, whereas the EM method performs *soft* assignments by computing likelihood values and assigning a feature h_n to the most likely cluster — see listings (52.38) and (52.37).

One way to motivate the $k-$means algorithm is pursued in Prob. 52.7 and is based on the following argument. Consider a collection of N feature vectors $\{h_n\}$ that we wish to distribute among K non-overlapping clusters denoted by $\{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_K\}$. Each cluster $\mathcal{C}_k$ is characterized by a mean vector μ_k corresponding to the average of all features within it, as shown by (52.28). We can seek an optimal assignment of feature vectors by formulating the following optimization problem:

$$\min_{\mathcal{C}_1,\mathcal{C}_2,\ldots,\mathcal{C}_K} \left\{ \sum_{k=1}^{K} \sum_{n\in\mathcal{C}_k} \|h_n - \mu_k(\mathcal{C}_k)\|^2 \right\} \tag{52.30}$$

where the means $\{\mu_k\}$ are dependent on $\{\mathcal{C}_k\}$. The unknowns are the clusters $\{\mathcal{C}_k\}$, i.e., their constituent feature vectors. This is generally a hard non-convex problem to solve. An approximate solution can be pursued by employing an *alternating minimization approach*. For each cluster k and feature h_n, we introduce the scalar

$$a_{nk} = \begin{cases} 1, & \text{if } h_n \in \mathcal{C}_k \\ 0, & \text{otherwise} \end{cases} \tag{52.31}$$

which reveals whether h_n lies in $\mathcal{C}_k$. There are NK such scalars since the integer subscripts n and k run over $0 \le n \le N-1$ and $1 \le k \le K$. Using the binary-valued scalars $\{a_{nk}\}$, we can rewrite the optimization problem (52.30) in the equivalent form:

$$\min_{\{a_{nk}\}} \left\{ \sum_{k=1}^{K} \sum_{n=0}^{N-1} a_{nk} \|h_n - \mu_k(\mathcal{C}_k)\|^2 \right\} \tag{52.32}$$

If we now alternate between minimizing over the $\{a_{nk}\}$ for a fixed set of means $\{\mu_k\}$, and minimizing over the $\{\mu_k\}$ for a fixed set of assignments $\{a_{nk}\}$, we arrive at the $k-$means algorithm:

$$\left\{a_{nk}^{(\ell)}\right\} = \underset{\{a_{nk}\}}{\text{argmin}} \; \sum_{k=1}^{K} \sum_{n=0}^{N-1} a_{nk} \left\| h_n - \mu_k^{(\ell-1)} \right\|^2 \tag{52.33a}$$

$$\left\{\mu_k^{(\ell)}\right\} = \underset{\{\mu_k\}}{\text{argmin}} \; \sum_{k=1}^{K} \sum_{n=0}^{N-1} a_{nk}^{(\ell)} \|h_n - \mu_k\|^2 \tag{52.33b}$$

where ℓ is an iteration index. The reader is asked to carry out this derivation in Prob. 52.7.

Selection of initial means

Three popular methods for selecting the initial mean vectors $\{\mu_k\}$ are the following:

(1) (**Forgy initialization**) selects the K mean vectors by sampling randomly without replacement from the N training vectors $\{h_n\}$.

(2) (**Random partitioning**) assigns the N feature vectors $\{h_n\}$ at random to K clusters and then computes the means of these clusters and uses them as the initial mean vectors.

(3) (**$k-$means++ initialization**) spreads out the selection of the mean vectors as follows. It starts by selecting one mean vector uniformly at random from the N training feature vectors $\{h_n\}$. We denote this first selection by μ_1. Subsequently, the squared distances from μ_1 to all feature vectors are computed and denoted by

$$d(n) \triangleq \|\mu_1 - h_n\|^2, \quad n = 0, 1, \dots, N-1 \tag{52.34}$$

These squared distances are normalized to add up to one and used to define a probability measure:

$$p(n) \triangleq \frac{d(n)}{\sum_{n=0}^{N-1} d(n)}, \quad n = 0, 1, \dots, N-1 \tag{52.35}$$

In this way, feature vectors that are farthest away from μ_1 receive higher probability values. The method subsequently selects a second mean vector, μ_2, randomly from the data according to this probability distribution. By construction, feature vectors $\{h_n\}$ that are father away from μ_1 will have a higher likelihood of being selected. We end up with two mean vectors $\{\mu_1, \mu_2\}$. The process continues as follows:

(a) For each feature vector, h_n, compute the squared distance, $d(n)$, from h_n to the *closest* mean vector.

(b) Normalize the distances, $d(n)$, according to (52.35) and use the normalized values as probability measures.

(c) Select randomly a new mean vector from the $\{h_n\}$ according to this probability distribution and add it to the collection of previously selected means. Repeat steps (a)-(c) until all K means have been selected.

Use for clustering

The two plots in the first row of Fig. 52.4 show $N = 250$ random feature vectors $h_n \in \mathbb{R}^2$ belonging to five different classes; the classes are colored in the plot on the left in order to identify them to the reader. The $k-$means algorithm is blind to the class information and operates on the unlabeled data in the plot on the right. The three plots in the bottom row show the result of applying the $k-$means algorithm for each of the three initialization procedures (Forgy, random, and $k-$means++). The location of the mean vector for each cluster is marked by a large $\times$ symbol. It is seen in this simulation that the location of the mean vectors is largely unaffected by the type of the initialization. The plots in the bottom also show the Voronoi diagrams (separation regions) that result from using the mean vectors. Figure 52.5 illustrates a second situation where the Voronoi regions are sensitive to the initialization procedure. The figure shows the result of applying the same $k-$means clustering algorithm to a second collection of $N = 250$ randomly generated feature vectors in the square region $[0,1] \times [0,1]$.

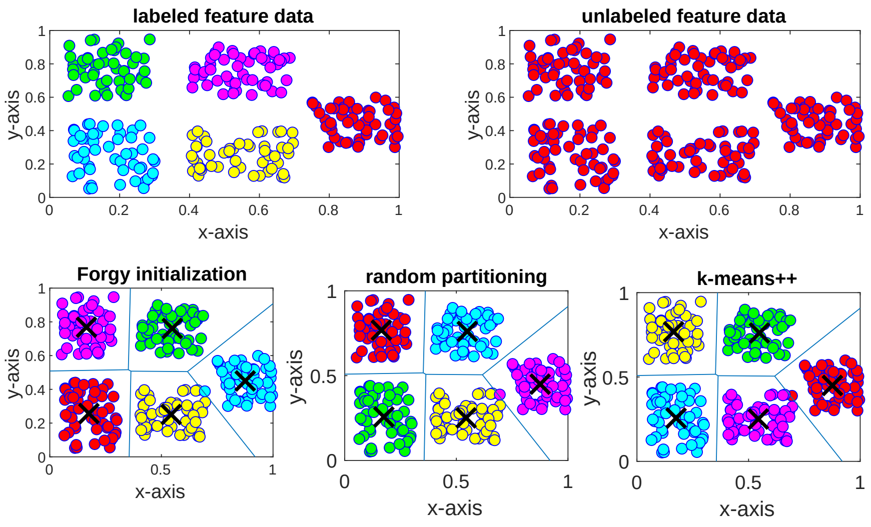

Figure 52.4 The plots in the top row show $N = 250$ feature vectors $h_n \in \mathbb{R}^2$ belonging to five different classes; in the plot on the left, the classes are colored. The plots in the bottom row show the result of applying $k-$means clustering to the data using the three initialization methods Forgy, random, and $k-$means$++$. In this case, all methods perform similarly. The $\times$ marks show the location of the mean vectors for the clusters.

Use for classification

We can exploit the result of the clustering operation to perform classification. We first associate a class $r(k)$ with each cluster k. The class value is determined by considering a majority vote among the members of the cluster. For example, if the majority of the members in the cluster belong to class $r = 1$, then the cluster will be assigned this label. In this way, we end up associating a label $r(k)$ with each cluster mean μ_k. During classification, when a new feature vector h arrives, we determine the closest mean vector to it and declare the class of h to be that of this mean vector. In other words, we carry out a $1-$NN classification scheme by relying solely on the K cluster means. Since $K \ll N$, we end up with a computationally more efficient implementation than the traditional $1-$NN solution that relies on comparing against the entire dataset.

Another useful feature of the $k-$means algorithm is that it can also be used to perform classification in a *semi-supervised* setting when we have available labels for only a subset of the feature vectors $\{h_n\}$ but not for all of them. In this case, we start by clustering the N feature vectors $\{h_n\}$ using the $k-$means construction; this amounts to an *unsupervised* step since labels are not necessary to carry out this step. We then label the clusters by using the limited labels that are available. We only consider those feature vectors within each cluster for which the label information is known, and associate the majority label to

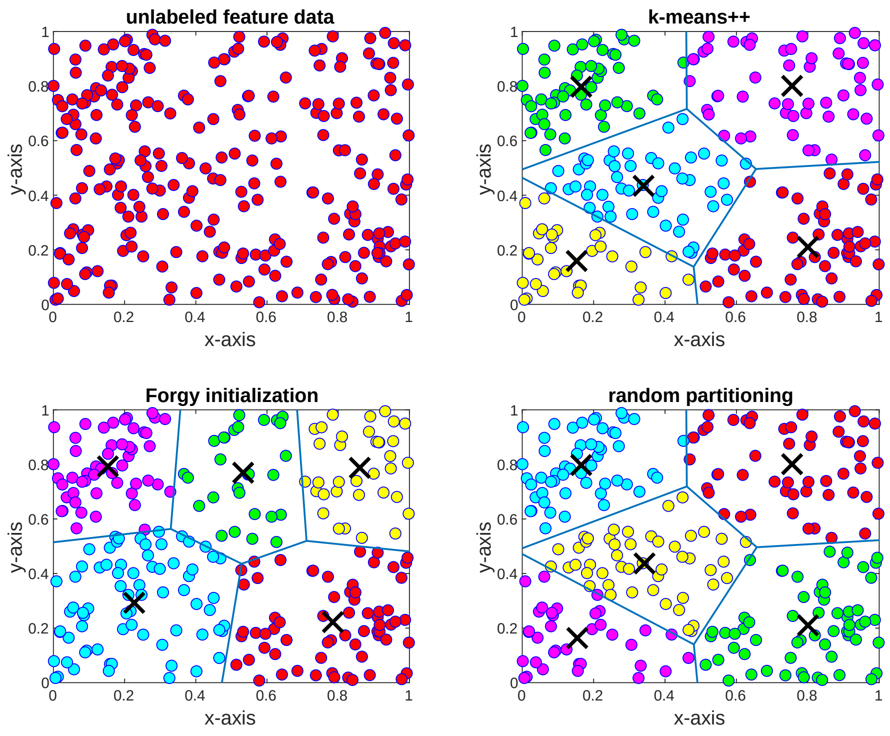

Figure 52.5 The leftmost top plot shows $N = 250$ feature vectors $h_n \in \mathbb{R}^2$ randomly generated in the square region $[0, 1] \times [0, 1]$. The other three plots show the result of applying $k-$means clustering on this data using the three initialization methods Forgy, random, and $k-$means$++$. In this case, the clustering results differ. The $\times$ symbol marks show the location of the mean vectors for the clusters.

the cluster; this amounts to a *semi-supervised* step. Subsequently, when a new feature vector h arrives, we determine the closest cluster mean to it and declare the class of h to be of that cluster.

Example 52.3 **(Clustering MNIST dataset)** We apply the $k-$means clustering algorithm to the MNIST dataset. The classification results obtained here will not be as reliable as the ones we will obtain by using other more elaborate classification schemes in future chapters. The example here is only meant to illustrate the operation of the clustering algorithm.

The MNIST dataset is useful for classifying handwritten digits. It contains 60,000 labeled training examples and 10,000 labeled test examples. Each entry in the dataset is a 28×28 grayscale image, which we transform into an $M = 784-$long feature vector, h_n. Each pixel in the image and, therefore, each entry in h_n, assumes integer values in the range $[0, 255]$. Every feature vector (or image) is assigned an integer label in the range 0 to 9 depending on which digit the image corresponds to. Figure 52.6 shows randomly selected images from the training dataset.

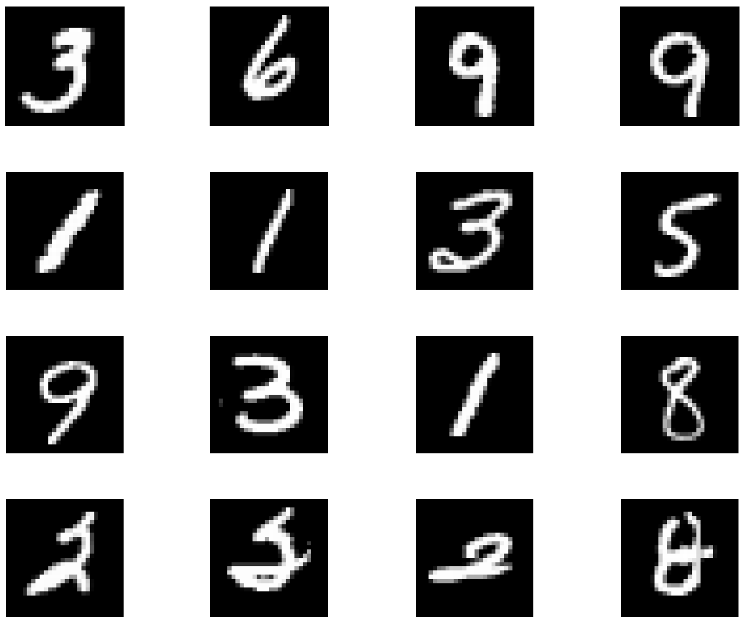

Figure 52.6 Randomly selected images from the MNIST dataset for handwritten digits. Each image is 28×28 grayscale with pixels assuming integer values in the range $[0, 255]$. The dataset can be downloaded from `http://yann.lecun.com/exdb/mnist/` or `https://github.com/daniel-e/mnist_octave`.

We pre-process the images $\{h_n\}$ by scaling their entries by 255 (so that they assume values in the range $[0, 1]$). We subsequently compute the mean feature vectors for the training and test sets. We center the scaled feature vectors around their respective means in both sets. Figure 52.7 shows randomly selected images for the digits 0 and 1 before and after processing.

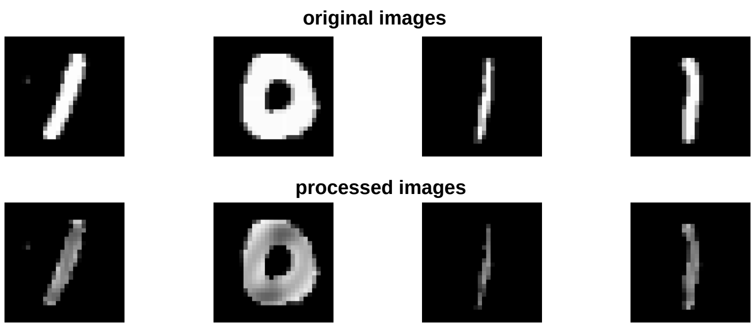

Figure 52.7 Randomly selected images for the digits 0 and 1 from the MNIST dataset for handwritten digits. The top row shows original images and the bottom row shows the processed images, whose pixels are scaled down to the interval $[0, 1]$ and centered around the mean feature vectors for training and testing.

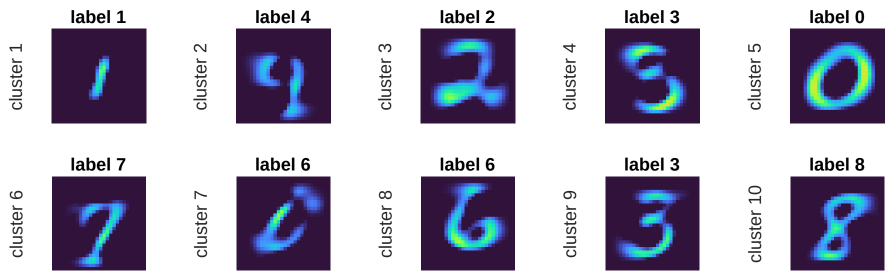

Figure 52.8 The mean image for each cluster, obtained by averaging the images assigned to the cluster. The images are shown using a color scale for emphasis. On top of each image, we assign a class label to the cluster. This label is obtained by a majority vote, namely, by determining the digit that is most repeated within the images in the cluster.

We apply the $k-$means++ algorithm to identify $K = 10$ clusters in the normalized training samples. We run the algorithm for 1000 iterations. At the end of these iterations, we obtain the mean vectors (centroids) for each of the clusters and plot them in Fig. 52.8. The figure shows $K = 10$ clusters labeled $k = 1$ through $k = 10$; note that the number we are assigning to refer to each cluster is different from the actual digit numbering from 0 to 9. We further assign a class label to each cluster using a majority vote. The digit that is most repeated within a cluster determines its label. Table 52.1 lists some statistics about the clusters: it shows the number of images that end up in each cluster, and the number of times that the most frequent digit appeared within the cluster. For example, a total of 9420 training images are assigned to cluster 1 and 6593 of these images happen to correspond to digit 1. This class label is assigned to the first cluster and it is written on top of the mean image corresponding to the cluster. Likewise, among the 8891 images in cluster 2, the most represented digit is 4 and it occurs 3180 times. We therefore assign the label 4 to cluster 2, and so forth. In the table, the first column lists the cluster number and the second column lists the class label that is assigned to the cluster. The last column shows the relative frequency of the most represented digit within each cluster.

Table 52.1 The table lists the clusters, their assigned labels, the total number of images in each cluster, the number of occurrences of the most frequent digit in the cluster, and its relative frequency within that cluster.

cluster number	cluster label	total images	occurrences of most frequent digit	percentage
1	1	9420	6593	70.0%
2	4	8891	3180	35.8%
3	2	4455	4105	92.1%
4	3	5076	2117	41.7%
5	0	4540	4289	94.5%
6	7	8488	3840	45.2%
7	**6**	5329	1737	32.6%
8	**6**	4291	3766	87.8%
9	3	4832	2833	58.6%
10	8	4678	3373	72.1%

Observe from Fig. 52.8 and also from the data in Table 52.1 that clusters 7 and 8 are labeled as corresponding to the same digit 6. There is no label corresponding to digit 5 in the figure and table. We can examine more closely the frequency of digit occurrences within each cluster, as shown in the following listing:

$$\left[\begin{array}{r|rrrrrrrrrr}
 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ \hline
1 & 5 & \boxed{6593} & 676 & 301 & 220 & 282 & 173 & 442 & 508 & 220 \\
2 & 31 & 16 & 167 & 171 & \boxed{3180} & 356 & 53 & 1831 & 149 & 2937 \\
3 & 6 & 30 & \boxed{4105} & 125 & 33 & 4 & 56 & 45 & 33 & 18 \\
4 & 233 & 4 & 243 & \boxed{2117} & 6 & 1314 & 29 & 8 & 1058 & 64 \\
5 & \boxed{4289} & 0 & 44 & 22 & 8 & 44 & 55 & 19 & 20 & 39 \\
6 & 7 & 10 & 65 & 38 & 1776 & 145 & 3 & \boxed{3840} & 126 & 2478 \\
7 & 888 & 10 & 253 & 154 & 455 & 1479 & \boxed{1737} & 22 & 260 & 71 \\
8 & 147 & 7 & 116 & 27 & 136 & 53 & \boxed{3766} & 4 & 26 & 9 \\
9 & 295 & 14 & 117 & \boxed{2833} & 1 & 1178 & 37 & 1 & 298 & 58 \\
10 & 22 & 58 & 172 & 343 & 27 & 566 & 9 & 53 & \boxed{3373} & 55
\end{array}\right]$$

The top row contains the digits 0 through 9. The first column contains the cluster numbers 1 through 10. Each row in the listing relates to one cluster. The numbers in the row show how many images corresponding to each digit appear within the cluster. We place a box around the most repeated digit. For example, for cluster 7, the most repeated digit is 6 with 1737 images; the second repeated digit is 5 with 1479 images. Compare these frequencies with the occurrences of digits 5 and 6 within cluster 8: there are 3766 images for digit 6 and only 53 images for digit 5. These results suggest that, if desired, we may label cluster 7 as corresponding to digit 5. Actually, during the testing/classification phase discussed next, we will find out that the algorithm will end up assigning images for digit 5 to cluster 7.

Table 52.2 Number of occurrences for each digit in the test data, along with the cluster it is assigned to and the number of images for that digit that were assigned to this cluster.

digit	occurrences in test data	assignments to same cluster	percentage	assigned to cluster
0	980	718	73.3%	5
1	1135	1105	97.4%	1
2	1032	700	67.8%	3
3	1010	523	51.8%	9
4	982	556	56.6%	2
5	892	233	26.2%	7
6	958	656	68.5%	8
7	1028	629	61.2%	6
8	974	555	60.0%	10
9	1009	538	53.3%	2
TOTAL	10,000	6213	62.1%	

Once clustering is completed, and a label is assigned to each cluster, we can now use the cluster structure to perform classification. For this purpose, we assign each of the $10,000$ testing samples to the cluster with the closest centroid to it, and set the label for this test sample to that of its closest cluster. We assess performance as follows. Table 52.2 lists the number of occurrences of each digit in the test data. For example, there are

980 images corresponding to digit 0, 1135 images corresponding to digit 1, and so forth. During testing, we find that 718 of the images corresponding to digit 0 are found to be closest to the centroid of cluster 5, whose label is "digit 0." We therefore say that 718 test images corresponding to digit 0 are correctly classified, which amounts to a 73.3% success rate for digit 0. These numbers are listed in the columns of Table 52.2. We also place on top of cluster 1 in Fig. 52.9 the label "digit 0" to indicate that, during testing, this cluster accounts for the largest proportion of classifications in favor of "digit 0."

Consider next digit 5. There are 892 occurrences of test images corresponding to digit 5 in the test data. Of these, 233 of them are assigned to cluster 7; this is the highest number of images for digit 5 that are assigned to a single cluster (the numbers in the third column of the table show the largest number of same-cluster assignments for each digit). We therefore find that the success rate for digit 5 is 26.2% under this construction. We place on top of cluster 7 in Fig. 52.9 the label "digit 5" to indicate that, during testing, this cluster accounts for the largest proportion of classifications in favor of "digit 5." It follows from the numbers in the table that the misclassification rate over the MNIST test data is close to 38%. We will be able to attain significantly better performance in later chapters by using other classification methods.

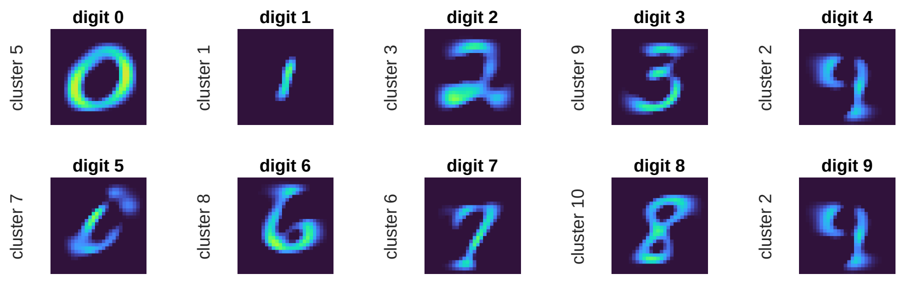

Figure 52.9 The label on top of each cluster shows the digit label from the testing set that is most often assigned to that cluster. The images are shown using a color scale for emphasis.

52.5 COMMENTARIES AND DISCUSSION

Nearest-neighbor rule. The earliest formulation of the nearest-neighbor (NN) rule appears to be the work by Fix and Hodges (1951) in an unpublished report from the 1951 *USAF School of Aviation Medicine*. Some of the earliest applications in the context of pattern classification appear in the publications by Johns (1961), Sebestyen (1962), Kanal (1962), Kanal *et al.* (1962), Harley *et al.* (1963), and Nilsson (1965). One fundamental and surprising result on the performance of the 1−NN classifier is expression (52.26), due to Cover and Hart (1967). The result states that for large sample sizes, the probability of error of the classifier is at most twice as bad as the best possible performance by the optimal Bayes classifier. The result also means, as stated in the aforementioned reference, that *"any other decision rule based on the infinite data set* *can cut the probability of error by at most one half."* An extension to the $k-$NN rule was given by Devroye (1981) in the following form for binary classifiers ($R = 2$):

$$P_e \leq (1+a)P_e^{\text{bayes}} \tag{52.36a}$$

$$a \overset{\Delta}{=} \frac{\alpha\sqrt{k}}{k-3.25}\left(1+\frac{\beta}{\sqrt{k-3}}\right),\ k \text{ odd},\ k \geq 5 \tag{52.36b}$$

$$\alpha \approx 0.3340, \quad \beta \approx 0.9750 \tag{52.36c}$$

Note that the factor a converges to zero at the rate $O(1/\sqrt{k})$. While these statements are reassuring, unfortunately, the conclusion only holds in the limit of large data sizes with $N \to \infty$. Since the seminal result by Cover and Hart (1967), there have been many other studies on nearest-neighbor rules and variations. Representative examples of these efforts include the works by Cover (1968), Peterson (1970), Hellman (1970), Wilson (1972), Fukunaga and Hostetler (1975), Dudani (1976), and Altman (1992) — see also the texts by Tukey (1977), Devroye, Gyorfi, and Lugosi (1996), Duda, Hart, and Stork (2000), Chávez *et al.* (2001), Shakhnarovich, Darrell, and Indyk (2006), Chaudhuri and Dasgupta (2014), Biau and Devroye (2015), and Chen and Shah (2018).

Voronoi diagrams. We illustrated in Fig. 52.3 the use of Voronoi diagrams in the context of nearest-neighbor rules. These diagrams divide the plane into a collection of convex regions consisting of one seed point each, along with all points that are closest to the seed. The diagrams are also referred to as *tessellations* since they tessellate the space and divide it into polygons without gaps. Such diagrams have found applications in many other areas including in the arts, geometry, geography, sciences, and engineering. One early notable application of Voronoi diagrams was by the English physician **John Snow (1813-1858)** who used them to locate the source of the 1854 cholera outbreak in the Soho area in central London. He concluded that most of the individuals infected by the disease lived closer to the Broad Street public water pump than any other water pump in the area. His investigation was reported in the publication by Snow (1854); today, he is considered the father of modern epidemiology. Although the designation "Voronoi diagram" is after the Russian mathematician **Georgy Voronoi (1868-1908)**, who formally defined the concept in Voronoi (1908), there have been informal instances of such diagrams as far back as three centuries earlier by the German astronomer **Johannes Kepler (1571–1630)** and the French mathematician **René Descartes (1596–1650)**; Kepler used tessellations in his studies of snowflakes and the sphere packing problem in Kepler (1611) while Descartes used them to identify clusters of stars in Descartes (1644) — see the accounts by Aurenhammer and Klein (2000), Okabe, Boots, and Sugihara (2000), and Liebling and Purnin (2012). Prior to Voronoi (1908), the diagrams were also used by Snow in 1854 and more formally by the German mathematician **Gustav Dirichlet (1805-1859)** in the work by Dirichlet (1850) on quadratic forms. Useful overviews on Voronoi diagrams appear in the article by Aurenhammer and Klein (2000) and the text by Okabe, Boots, and Sugihara (2000).

$k-$means clustering. There are several variations of the clustering problem in statistical analysis, i.e., the problem of partitioning data into clusters. Some of the earliest formulations appear in the works by Dalenius (1950), Dalenius and Gurney (1951), Marschak (1954), Cox (1957), Fisher (1958), and Ward (1963). For example, Fisher (1958) motivates the article by posing the following question in the abstract: *"Given a set of arbitrary numbers, what is a practical procedure for grouping them so that the variance within groups is minimized?"* Fisher focused on the one-dimensional case, $M = 1$. Since solving the clustering formulation (52.30) in its generality is an NP-hard problem, it is necessary to resort to approximate solutions. One of the most popular algorithms is the $k-$means procedure described in the body of the chapter. The original idea for the $k-$means algorithm appears to be the works by Steinhaus (1957), Lloyd (1957), and Sebestyen (1962), although Lyold published his work only 25 years later in 1982. The designation "$k-$means" was proposed by MacQueen (1965,1967);

for example, the author states in the abstract of MacQueen (1967) that the objective is *"to describe a process for partitioning an $N-$dimensional population into k sets on the basis of a sample. The process, which is called "k-means," appears to give partitions which are reasonably efficient in the sense of within–class variance."* The same algorithm was independently developed by Forgy (1965). The $k-$means++ variant for selecting the initial mean (seed) vectors is more recent and was proposed independently by Ostrovsky *et al.* (2006) and Arthur and Vassilvitskii (2007); the latter reference contains several results on the behavior of the $k-$means++ procedure. Useful studies on the convergence properties of the $k-$means algorithm (also called Lloyd algorithm) appear in Abaya and Wise (1984), Sabin and Gray (1986), Har-Peled and Sadri (2005), Arthur and Vassilvitskii (2006,2007), and Du, Emelianenko, and Ju (2006). Accessible overviews on clustering algorithms in classification and data quantization/compression are given by Hartigan (1975), Gray and Neuhoff (1998), Du, Faber, and Gunzburger (1999), MacKay (2003), Tan, Steinbach, and Kumar (2005), and Witten, Frank, and Hall (2011). To facilitate comparison with the EM algorithm described next, we list the $k-$means clustering method in the form shown in (52.37).

$k-$means clustering algorithm

```
given feature vectors $\{h_n \in \mathbb{R}^M\}$, for $n = 0, 1, \ldots, N-1$;
given number of clusters, $K$;
given initial mean vectors conditions : $\pi_k^{(0)},\ k = 1, 2, \ldots, K$;
repeat until convergence over $m \geq 1$ :
    (determine clusters): for each $n = 0, 1, \ldots, N-1$ and $k = 1, \ldots, K$ :
        $r^{(m)}(k, h_n) = \begin{cases} 1, & \text{if } h_n \text{ is closest to } \mu_k^{(m-1)} \\ 0, & \text{otherwise} \end{cases}$
        $N_k^{(m)} = \sum_{n=0}^{N-1} r^{(m)}(k, h_n)$
    (update means): for each $k = 1, \ldots, K$
        $\mu_k^{(m)} = \frac{1}{N_k^{(m)}} \sum_{n=0}^{N-1} r^{(m)}(k, h_n) h_n$
end
return $\{\widehat{\mu}_k\} \leftarrow \{\mu_k^{(m)}\}$
```
(52.37)

Connection to the EM algorithm. There is a useful connection between the $k-$means clustering algorithm (52.29) and the expectation-maximization algorithm (32.67) for Gaussian mixture models studied in an earlier chapter. If we assume the covariance matrices of the Gaussian components are preset to the identity matrix (i.e., if we assume spherical clusters), and focus exclusively on estimating the mean vectors, then the EM algorithm (32.67) reduces to listing (52.38), where m denotes the iteration index and h_n denotes the $n-$th feature. For comparison purposes, we have rewritten the $k-$means clustering algorithm in the form shown in (52.37). Observe that there is a *hard* assignment of the sample h_n to one of the clusters (the one determined by the closest mean vector to h_n). In contrast, the EM implementation performs a *soft* assignment of h_n based on the responsibility factor $r^{(m)}(k, h_n)$: it measures the likelihood that sample h_n belongs to cluster k. The $k-$means algorithm sets these factors to one or zero, depending on whether h_n is closest to μ_k or not.

Special case of the EM algorithm (32.67) for K clusters

```
given feature vectors $\{h_n \in \mathbb{R}^M\}$, for $n = 0, 1, \ldots, N-1$;
assumed $K$ Gaussian mixture components;
given initial conditions : $\pi_k^{(0)}, \mu_k^{(0)}$, $k = 1, 2, \ldots, K$;
repeat until convergence over $m \geq 1$ :
    (E-step): for each $n = 0, 1, \ldots, N-1$ and $k = 1, \ldots, K$ :
        $r^{(m)}(k, h_n) = \dfrac{\pi_k^{(m-1)} \exp\left\{-\frac{1}{2}\left\|h_n - \mu_k^{(m-1)}\right\|^2\right\}}{\sum_{j=1}^{K} \pi_j^{(m-1)} \exp\left\{-\frac{1}{2}\left\|h_n - \mu_j^{(m-1)}\right\|^2\right\}}$        (52.38)
        $N_k^{(m)} = \sum_{n=0}^{N-1} r^{(m)}(k, h_n)$
    (M-step): for each $k = 1, \ldots, K$
        $\mu_k^{(m)} = \frac{1}{N_k^{(m)}} \sum_{n=0}^{N-1} r^{(m)}(k, h_n) h_n$
        $\pi_k^{(m)} = N_k^{(m)}/N$
end
return $\{\widehat{\pi}_k, \widehat{\mu}_k\} \leftarrow \{\pi_k^{(m)}, \mu_k^{(m)}\}$
```

MNIST dataset. Example 52.3 applies the $k-$means clustering algorithm to the MNIST dataset. It contains 60,000 labeled training examples and 10,000 labeled test examples. This popular dataset was used by LeCun *et al.* (1998) to perform classification of handwritten digits. It can be downloaded from `http://yann.lecun.com/exdb/mnist/` and also `https://github.com/daniel-e/mnist_octave`.

PROBLEMS

52.1 Consider the 1-NN decision rule applied to a binary classification problem and introduce the random variable $t(\boldsymbol{h}) = \mathbb{P}(\boldsymbol{\gamma} = +1|\boldsymbol{h})$. Assume $N \to \infty$, where N denotes the sample size. Let P_e^∞ denote the asymptotic misclassification error as $N \to \infty$ for the 1−NN classifier.

(a) Show that $P_e^\infty = \mathbb{E}\left\{2t(\boldsymbol{h})(1-t(\boldsymbol{h})\right\}$.

(b) Conclude the validity of property (52.26) for the 1−NN classification rule, namely, that the asymptotic probability of error is bounded by twice the probability of error by the Bayes classifier regardless of the underlying distribution.

52.2 Refer to the bias-variance relation of Prob. 27.16. We use the result here to examine the bias-variance tradeoff for the $k-$NN strategy. Consider scalar and real-valued variables $\{\boldsymbol{\gamma}, \boldsymbol{h}, \boldsymbol{v}\}$ satisfying a model of the form $\boldsymbol{\gamma} = f(\boldsymbol{h}) + \boldsymbol{v}$, for some known function $f(\cdot)$. The variable $\boldsymbol{v}$ is zero-mean noise with variance σ_v^2 and is independent of $\boldsymbol{h}$. Consider a collection of independent data realizations $\{\gamma(n), h_n\}$. Upon the arrival of a new feature h, we estimate the corresponding γ as follows:

$$\widehat{\gamma} = \frac{1}{k} \sum_{\ell \in \mathcal{N}_h} \gamma(\ell)$$

where the average is computed over the $k-$nearest neighbors to h, denoted by the set

$\mathcal{N}_h$. Show that, conditioned on the feature data $\{h_n\}$:

$$\mathbb{E}\left((\boldsymbol{\gamma}-\widehat{\boldsymbol{\gamma}})^2|\boldsymbol{h}=h\right) = \sigma_v^2 + \frac{\sigma_v^2}{k} + \left(f(h) - \frac{1}{k}\sum_{\ell\in\mathcal{N}_h} f(h_\ell)\right)^2$$

where the second term on the right-hand side denotes the variance factor (it decays with k), and the last term denotes the squared bias factor.

52.3 We continue with Prob. 52.2. Let $\boldsymbol{\gamma}^\bullet$ denote the optimal mean-square-error estimator for $\boldsymbol{\gamma}$ given $\boldsymbol{h}$. Show that $\boldsymbol{\gamma}^\bullet = f(\boldsymbol{h})$ with estimation error variance equal to $\mathbb{E}\,(\widetilde{\boldsymbol{\gamma}}^\bullet)^2 = \sigma_v^2$. Let $\widetilde{\boldsymbol{\gamma}} = \boldsymbol{\gamma} - \widehat{\boldsymbol{\gamma}}$ for the $k-$NN estimator from Prob. 52.2. Use the result of that problem to conclude that

$$\mathbb{E}\,\widetilde{\boldsymbol{\gamma}}^2 - \mathbb{E}\,(\widetilde{\boldsymbol{\gamma}}^\bullet)^2 = \frac{\sigma_v^2}{k} + \mathbb{E}\Big(f(\boldsymbol{h}) - \frac{1}{k}\sum_{\ell\in\mathcal{N}_{\boldsymbol{h}}} \gamma(\ell)\Big)^2$$

52.4 Consider two distinct points $a, b \in \mathbb{R}^M$. Show that the bisector of the segment joining them is a hyperplane in $\mathbb{R}^M$.

52.5 Consider the collection of $m-$dimensional points $\mathcal{F} = \{h_1, h_2, \ldots, h_N\}$. For any h_a from this set, we define its Voronoi cell as the set of all points h that satisfy

$$\text{Voronoi}(h_a) = \left\{h \in \mathbb{R}^M \;\Big|\; \|h-h_a\| \leq \|h-h_n\|^2,\;\; \forall\, h_n \in \mathcal{F}\right\}$$

Show that the Voronoi cell is a convex set.

52.6 Consider a Voronoi diagram similar to the one shown in Fig. 52.3. Let N be the number of seed points $\{h_n\}$. Let N_e denote the total number of edges in the diagram, and let N_v denote the total number of vertices. Verify that $N_v - N_e + N = 1$.

52.7 Explain that the $k-$means algorithm solves problem (52.32) by alternating between minimizing over the $\{a_{nk}\}$ for a fixed set of means $\{\mu_k\}$, and minimizing over the $\{\mu_k\}$ for a fixed set of assignments $\{a_{nk}\}$.

52.8 Consider a cluster $\mathcal{C}$ consisting of a collection of $M-$dimensional feature vectors, denoted generically by $h \in \mathcal{C}$. Let μ denote the mean of the cluster, i.e., the mean of the vectors in $\mathcal{C}$. For any vector $x \in \mathbb{R}^M$, show that

$$\sum_{h\in\mathcal{C}} \|h-x\|^2 = \sum_{h\in\mathcal{C}} \|h-\mu\|^2 + |\mathcal{C}|\,\|\mu-x\|^2$$

where $|\mathcal{C}|$ denotes the cardinality of $\mathcal{C}$.

52.9 Argue that problem (52.30) is equivalent to solving:

$$\min_{\mathcal{C}_1,\mathcal{C}_2,\ldots,\mathcal{C}_K} \sum_{k=1}^{K}\left\{\frac{1}{|\mathcal{C}_k|}\sum_{n,m\in\mathcal{C}_k} \|h_n - h_m\|^2\right\}$$

52.10 We assumed in (52.42) that $\mathbb{P}_\epsilon > 0$ for any ϵ. That is, we assumed that all features h are well-behaved in the sense that if we encircle each one of them by a small sphere of radius ϵ, then there is a positive probability that other feature vectors will be present inside the sphere. Let us assume, to the contrary, that there exist some subset of feature vectors, denoted by $\bar{h} \in \bar{\mathcal{H}}$, that is not well-behaved, meaning for any $\bar{h}$ in this set, there will exist some $\bar{\epsilon} > 0$ such that $\mathbb{P}_\epsilon = 0$ for any $\epsilon < \bar{\epsilon}$. In other words, no feature vectors will exist in spheres surrounding $\bar{h}$ of radius smaller than $\bar{\epsilon}$. Prove that this is an impossibility, i.e., that $\bar{\mathcal{H}}$ is a set of probability zero.

52.11 Let $\{\boldsymbol{h}_1, \ldots, \boldsymbol{h}_N\}$ denote independent and identically distributed random variables selected according to a distribution $\boldsymbol{h} \sim f_{\boldsymbol{h}}(h)$ with compact support $\mathcal{H}$ in $\mathbb{R}^M$. For each $\boldsymbol{h}_n$, let $\boldsymbol{h}_n'$ denote its nearest neighbor from among the remaining vectors and define the expected squared $\ell_\infty-$distance:

$$d^2 \triangleq \frac{1}{N}\sum_{n=1}^{N} \|\boldsymbol{h}_n - \boldsymbol{h}_n'\|_\infty^2$$

Let D denote the diameter of the set $\mathcal{H}$, meaning that the $\ell_\infty-$distance between any two points in $\mathcal{H}$ cannot exceed D. Show that

$$d^2 \leq \begin{cases} 16D^2/N^{2/M}, & M \geq 2 \\ 4D^2/N, & M = 1 \end{cases}$$

Remark. See the book by Biau and Devroye (2015, Ch. 2) for a related discussion.

52.A PERFORMANCE OF THE NN-CLASSIFIER

In this appendix we establish Theorem 52.1 on the generalization error of the $1-$NN classifier. The proof follows arguments similar to Cover and Hart (1967). The lower bound in (52.26) is obvious since the Bayes classifier minimizes the probability of error by construction. Let us focus on the upper bound.

Let h denote an arbitrary feature vector arising from the probability distribution $f_{\boldsymbol{h}}(h)$. We denote its actual class by $r(h)$. Let $x_{h,N}$ denote the nearest neighbor to h from among the N given feature vectors $\{h_n\}$:

$$x_{h,N} \;=\; \underset{x\in\{h_n\}_{n=0}^{N-1}}{\text{argmin}} \;\|h-x\|^2 \tag{52.39}$$

We denote the class of $x_{h,N}$ by $r(x)$. Note that the location of $x_{h,N}$ depends on both h and the data size N. For simplicity, we will drop the subscripts h and N from $x_{h,N}$ and refer to the variable by x. Let $f_{\boldsymbol{x}|\boldsymbol{h}}(x|h)$ denote the conditional pdf of the closest neighbor variable $\boldsymbol{x}$ given $\boldsymbol{h} = h$. This pdf is also dependent on N since $\boldsymbol{x}$ is dependent on N. It is reasonable to assume that, as the sample size increases to $N \to \infty$, the pdf $f_{\boldsymbol{x}|\boldsymbol{h}}(x|h)$ tends to a Dirac impulse function concentrated at h, i.e.,

$$\lim_{N\to\infty} f_{\boldsymbol{x}|\boldsymbol{h}}(x|h) \;=\; \delta(x-h) \tag{52.40}$$

which means that the pdf becomes concentrated at location h; recall that such impulse functions satisfy the sifting property

$$\int_{x\in\mathcal{X}} g(x)\delta(x-h)dx \;=\; g(h) \tag{52.41}$$

for any function $g(x)$ defined at location h, and where the integration is over the domain of x. Assumption (52.40) can be motivated as follows. Choose an arbitrary $\epsilon > 0$ and let $\mathcal{S}(\epsilon)$ denote a sphere of radius $\epsilon > 0$ centered at h. The probability that some feature vector h' falls within the sphere is given by — see Prob. 52.10:

$$\mathbb{P}_\epsilon \;=\; \int_{h\in\mathcal{S}(\epsilon)} f_{\boldsymbol{h}}(h)dh \;> 0 \tag{52.42}$$

The probability that the N feature vectors, which are assumed to be chosen independently of each other, fall outside the sphere is given by

$$\mathbb{P}\Big(N \text{ features outside } \mathcal{S}(\epsilon)\Big) \;=\; (1-\mathbb{P}_\epsilon)^N \;\overset{N\to\infty}{\longrightarrow}\; 0 \tag{52.43}$$

This result holds regardless of the radius of the sphere. Therefore, by shrinking the size of the sphere around h, and as the sample size N tends to infinity, we find that the nearest neighbor to h converges to h with probability one and assumption (52.40) is justified.

Now given a feature vector h, whose closest neighbor is x, the $1-$NN classifier assigns to h the same label as x. Therefore, the probability of error by this classifier is given by

$$\begin{aligned}\mathbb{P}(\text{error}|h,x) &= \mathbb{P}\Big(\boldsymbol{r}(x) \neq \boldsymbol{r}(h)|h,x\Big)\\ &= 1-\mathbb{P}\Big(\boldsymbol{r}(x) = \boldsymbol{r}(h)|h,x\Big)\\ &\overset{(a)}{=} 1-\sum_{\boldsymbol{r}=1}^{R}\mathbb{P}\Big(\boldsymbol{r}=r(h)|h\Big)\,\mathbb{P}\Big(\boldsymbol{r}=r(x)|x\Big)\end{aligned} \tag{52.44}$$

where the rightmost term in (a) is a sum over the probabilities of the classes for h and x being the same; this is because there are R possibilities for $\boldsymbol{r}(x)$ given by $r \in \{1,2,\ldots,R\}$. If we integrate the above error over the conditional pdf of $\boldsymbol{x}$ given h, and let $N\to\infty$, we obtain the average probability of error for a given h:

$$\begin{aligned}\mathbb{P}(\text{error}|h) &= \int_{x\in\mathcal{H}}\mathbb{P}(\text{error}|h,x)\,f_{\boldsymbol{x}|\boldsymbol{h}}(x|h)dx\\ &\overset{(52.40)}{=} \int_{x\in\mathcal{H}}\left\{1-\sum_{\boldsymbol{r}=1}^{R}\mathbb{P}\Big(\boldsymbol{r}=r(h)|h\Big)\,\mathbb{P}\Big(\boldsymbol{r}=r(x)|x\Big)\right\}\delta(x-h)dx,\;\; N\to\infty\\ &= 1-\sum_{\boldsymbol{r}=1}^{R}\mathbb{P}^2\Big(\boldsymbol{r}=r(h)|\boldsymbol{h}=h\Big)\end{aligned} \tag{52.45}$$

If we further integrate over the pdf of $\boldsymbol{h}$, we obtain the probability of error for the 1−NN classifier:

$$P_e = \int_{h\in\mathcal{H}}\left\{1-\sum_{\boldsymbol{r}=1}^{R}\mathbb{P}^2\Big(\boldsymbol{r}=r(h)|\boldsymbol{h}=h\Big)\right\}f_{\boldsymbol{h}}(h)dh \tag{52.46}$$

We want to compare this expression to P_e^{bayes}, which we know from (28.69) is given by

$$P_e^{\text{bayes}} = \int_{h\in\mathcal{H}}\Big\{1-\mathbb{P}\Big(\boldsymbol{r}^\bullet(h)=r(h)|\boldsymbol{h}=h\Big)\Big\}f_{\boldsymbol{h}}(h)dh \tag{52.47}$$

Let us examine the sum that appears inside (52.46). We split it into two terms:

$$\begin{aligned}&\sum_{\boldsymbol{r}=1}^{R}\mathbb{P}^2\Big(\boldsymbol{r}=r(h)|\boldsymbol{h}=h\Big)\\ &= \mathbb{P}^2\Big(\boldsymbol{r}^\bullet(h)=r(h)|\boldsymbol{h}=h\Big) + \underbrace{\sum_{\boldsymbol{r}\neq\boldsymbol{r}^\bullet(h)}^{R}\mathbb{P}^2\Big(\boldsymbol{r}=r(h)|\boldsymbol{h}=h\Big)}_{\triangleq\, A}\\ &\overset{(28.68)}{=} \Big(1-P^{\text{bayes}}(\text{error}|h)\Big)^2 + A\end{aligned} \tag{52.48}$$

where the first term depends on the probability of error of the Bayes classifier at h, and the second term is a sum we are denoting by the letter A. If we minimize A over its terms we can determine a lower bound for the sum of squared probabilities on the left. Hence, we formulate the optimization problem:

$$\begin{aligned}
&\min \sum_{r \neq r^\bullet(h)}^{R} \mathbb{P}^2\Big(\boldsymbol{r} = r(h)|h\Big) \\
&\qquad\qquad \text{subject to} \quad \mathbb{P}\Big(\boldsymbol{r} = r(h)|h\Big) \geq 0 \\
&\qquad\qquad\quad \text{and} \quad \sum_{r \neq r^\bullet(h)}^{R} \mathbb{P}\Big(\boldsymbol{r} = r(h)|h\Big) \;=\; P^{\text{bayes}}(\text{error}|h)
\end{aligned} \tag{52.49}$$

where the minimization is over the individual terms $\mathbb{P}^2(\boldsymbol{r} = r(h)|h)$. We are therefore minimizing a sum of nonnegative terms subject to a constraint on what their sum should be. A straightforward Lagrange multiplier argument will show that the solution is obtained when all probabilities are equal to each other, i.e., when

$$\mathbb{P}\Big(\boldsymbol{r} = r(h)|\boldsymbol{h} = h\Big) \;=\; \frac{P^{\text{bayes}}(\text{error}|h)}{R-1}, \;\; \text{for any } \boldsymbol{r} \neq \boldsymbol{r}^\bullet \tag{52.50}$$

Substituting into (52.48) we determine a lower bound as follows:

$$\begin{aligned}
&\sum_{r=1}^{R} \mathbb{P}^2\Big(\boldsymbol{r} = r(h)|h\Big) \\
&= \Big(1 - P^{\text{bayes}}(\text{error}|h)\Big)^2 + \sum_{r \neq r^\bullet(h)}^{R} \mathbb{P}^2\Big(\boldsymbol{r} = r(h)|\boldsymbol{h} = h\Big) \\
&\geq 1 - 2P^{\text{bayes}}(\text{error}|h) + \Big(P^{\text{bayes}}(\text{error}|h)\Big)^2 + \frac{R-1}{(R-1)^2}\Big(P^{\text{bayes}}(\text{error}|h)\Big)^2 \\
&\geq 1 - 2P^{\text{bayes}}(\text{error}|h) + \Big(P^{\text{bayes}}(\text{error}|h)\Big)^2 + \frac{1}{R-1}\Big(P^{\text{bayes}}(\text{error}|h)\Big)^2 \\
&\geq 1 - 2P^{\text{bayes}}(\text{error}|h) + \frac{R}{R-1}\Big(P^{\text{bayes}}(\text{error}|h)\Big)^2
\end{aligned} \tag{52.51}$$

which implies that

$$1 - \sum_{r=1}^{R} \mathbb{P}^2\Big(\boldsymbol{r} = r(h))|\boldsymbol{h} = h\Big) \;\leq\; 2P^{\text{bayes}}(\text{error}|h) - \frac{R}{R-1}\Big(P^{\text{bayes}}(\text{error}|h)\Big)^2 \tag{52.52}$$

Substituting this bound into (52.46) and integrating over the distribution of $\boldsymbol{h}$ we obtain

$$\begin{aligned}
P_e &\leq \int_{h\in\mathcal{H}} \left\{ 2P^{\text{bayes}}(\text{error}|h) - \frac{R}{R-1}\Big(P^{\text{bayes}}(\text{error}|h)\Big)^2 \right\} f_{\boldsymbol{h}}(h)dh \\
&= 2P_e^{\text{bayes}} - \frac{R}{R-1}\left\{ \int_{h\in\mathcal{H}} \Big(P(\text{error}|h)\Big)^2 f_{\boldsymbol{h}}(h)dh \right\} \\
&\leq 2P_e^{\text{bayes}} - \frac{R}{R-1}\Big(P_e^{\text{bayes}}\Big)^2
\end{aligned} \tag{52.53}$$

where in the last step we used the fact that for any scalar random variable $\boldsymbol{x}$, it holds that $(\mathbb{E}\,\boldsymbol{x})^2 \leq \mathbb{E}\,\boldsymbol{x}^2$ and, hence,

$$\begin{aligned}
\Big(P_e^{\text{bayes}}\Big)^2 \;&\triangleq\; \Big(\mathbb{E}\,P^{\text{bayes}}(\text{error}|h)\Big)^2 \\
&\leq\; \mathbb{E}\,\Big(P^{\text{bayes}}(\text{error}|h)\Big)^2 \\
&=\; \int_{h\in\mathcal{H}} \Big(P^{\text{bayes}}(\text{error}|h)\Big)^2 f_{\boldsymbol{h}}(h)dh
\end{aligned} \tag{52.54}$$

REFERENCES

Abaya, E. and F. Wise (1984), “Convergence of vector quantizers with applications to optimal quantization,” *SIAM J. Appl. Math.*, vol. 44, pp. 183–189.

Altman, N. S. (1992), “An introduction to kernel and nearest-neighbor nonparametric regression,” *The American Statistician*, vol. 46, no. 3, pp. 175–185.

Arthur, D. and S. Vassilvitskii (2006), “How slow is the k-means method?” *Proc. Annual Symposium on Computational Geometry* (SCG), pp. 144–153, Sedona, AZ, USA.

Arthur, D. and S. Vassilvitskii (2007), “k-means++: The advantages of careful seeding,” *Proc. Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 1027–1035, New Orleans.

Aurenhammer, F. and R. Klein (2000) “Voronoi diagrams,” in *Handbook of Computational Geometry*, J-R. Sack and J. Urrutia, *Eds.*, , pp. 201–290, North-Holland, The Netherlands.

Biau, G. and L. Devroye (2015), *Lectures on the Nearest Neighbor Method*, Springer, NY.

Chaudhuri, K. and S. Dasgupta (2014), “Rates of convergence for nearest neighbor classification,” *Proc. Neural Information Process. Systems* (NIPS), pp. 1–9, Montreal, Canada.

Chávez, E., G. Navarro, R. Baeza-Yates, and J. L. Marroquin (2001), “Searching in metric spaces,” *ACM Computing Surveys*, vo. 33, no. 3, pp. 273–321.

Chen, G. H. and D. Shah (2018), *Explaining the Success of Nearest Neighbor Methods in Prediction*, Now Publishers.

Cover, T. M. (1968), “Estimation by the nearest neighbor rule,” *IEEE Trans. Information Theory*, vol. 14, pp. 21–27.

Cover, T. M. and P. E. Hart (1967), “Nearest neighbor pattern classification,” *IEEE Trans. Information Theory*, vol. 13, no. 1, pp. 21–27.

Cox, D. R. (1957), “Note on grouping,” *J. Amer. Statist. Assoc.*, vol. 52, pp. 543–547.

Dalenius, T. (1950), “The problem of optimum stratification,” *Skandinavisk Aktuarietidskrift*, pp. 203–213.

Dalenius, T. and M. Gurney (1951), “The problem of optimum stratification II,” *Skandinavisk Aktuarietidskrift*, pp. 133–148.

Descartes, R. (1644), *Principia Philosophiae*, Amstelodami, apud Ludovicum Elzevirium.

Devroye, L. (1981), “On the asymptotic probability of error in nonparametric discrimination,” *The Annals of Statistics*, vol. 9, no. 6, pp. 1320–1327.

Devroye, L., L. Gyorfi, and G. Lugosi (1996), *A Probabilistic Theory of Pattern Recognition*, Springer, NY.

Dirichlet, G. L. (1850), “Über die Reduktion der positiven quadratischen Formen mit drei unbestimmten ganzen Zahlen,” *Journal für die reine und angewandte Mathematik*, vol. 40, pp. 209–227.

Du, Q., M. Emelianenko, and L. Ju (2006), “Convergence of the Lloyd algorithm for computing centroidal Voronoi tessellations,” *SIAM J. Numerical Analysis*, vol. 44, pp. 102–119.

Du, Q., V. Faber, and M. Gunzburger (1999), “Centroidal Voronoi tessellations: Applications and algorithms,” *SIAM Review*, vol. 41, no. 4, pp. 637–676.

Duda, R. O., P. E. Hart, and D. G. Stork (2000), *Pattern Classification*, 2nd edition, Wiley, NY.

Dudani, S. A. (1976), “The distance-weighted k-nearest-neighbor rule,” *IEEE Trans. Syst. Man Cybern.*, vol. 6, pp. 325–327.

Fisher, W. D. (1958), “On grouping for maximum homogeneity,” *J. Amer. Statist. Assoc.*, vol. 53, pp. 789–798.

Fix, E. and J. L. Hodges, Jr. (1951), “Discriminatory analysis, nonparametric discrimination,” *USAF School of Aviation Medicine*, Randolph Field, TX, Project 21-49-004, Report 4, Contract AF41(128)-31, Feb. 1951.

Forgy, E. W. (1965), "Cluster analysis of multivariate data: Efficiency versus interpretability of classifications," *Biometrics*, vol. 21, pp. 768–769.

Fukunaga, K. and L. Hostetler, (1975), "k-nearest-neighbor Bayes-risk estimation," *IEEE Trans. Information Theory*, vol. 21, no. 3, pp. 285–293.

Gray, R. M. and D. L. Neuhoff (1998), "Quantization," *IEEE Trans. Information Theory*, vol. 44, no. 6, pp. 2325–2373.

Har-Peled, S. and B. Sadri (2005), "How fast is the k-means method?" *Algorithmica*, vol. 41, pp. 185–202.

Harley, T., J. Bryan, L. Kanal, D. Taylor, and J. Grayum (1963), "Semi-automatic imagery screening research study and experimental investigation," *Philco Reports*, vol. I, report nos. 2-3.

Hartigan, J. A. (1975), *Clustering Algorithms*, Wiley, NY.

Hellman, M. E. (1970), "The nearest neighbor classification rule with a reject option," *IEEE Trans. Syst. Man Cybern.*, vol. 6, no. 3, pp. 179–185.

Johns, M. V. (1961), "An empirical Bayes approach to non-parametric two-way classification," in *Studies in Item Analysis and Prediction*, H. Solomon, *Ed.*, Stanford University Press.

Kanal, L. (1962), "Evaluation of a class of pattern recognition networks," *Biological Prototypes and Synthetic Systems*, vol. 1, pp. 261–269, Plenum Press, NY.

Kanal, L., F. Slymaker, D. Smith, and W. Walker (1962), "Basic principles of some pattern recognition systems," *Proc. National Electronics Conference*, vol. 18, pp. 279–295, Chicago, IL.

Kepler, J. (1611),*The Six-Sided Snowflake*, Oxford University Press, 2014 edition.

LeCun, Y., L. Bottou, Y. Bengio, and P. Haffner (1998), "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324.

Liebling, T. M. and L. Pournin (2012), "Voronoi diagrams and delaunay triangulations: Ubiquitous Siamese twins," *Documenta Mathematica*, extra volume ISMP, pp. 419–431.

Lloyd, S. P. (1957), "Least square quantization in PCM," internal *Bell Tel. Labs.* report. The material was presented at the *Institute of Mathematical Statistics Meeting*, Atlantic City, NJ, Sep. 10–13, 1957. Published 25 years later as Lloyd, S. P. (1982), "Least squares quantization in PCM," *IEEE Trans. Information Theory*, vol. 28, no. 2, pp. 129–137.

MacKay, D. J. C. (2003), *Information Theory, Inference, and Learning Algorithms*, Cambridge University Press.

MacQueen, J. B. (1965), "On convergence of k-means and partitions with minimum average variance," abstract, *Ann. Math. Statist.*, vol. 36, p. 1084.

MacQueen, J. B. (1967), "Some methods for classification and analysis of multivariate observations," *Proc. 5th Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1, pp. 281–297, University of California Press.

Marschak, J. (1954), "Towards an economic theory of organization and information," *Decision Processes*, R. M. Thrall, C. H. Coombs, and R. C. Davis, *Eds.*, Wiley, NY.

Nilsson, N. (1965), *Learning Machines*, McGraw-Hill, NY.

Okabe, A., B. Boots, and K. Sugihara (2000), *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*, 2nd edition, Wiley, NY.

Ostrovsky, R., Y. Rabani, L. J. Schulman, and C. Swamy, C. (2006), "The effectiveness of Lloyd-type methods for the k-means problem," *Proc. 47th Annual IEEE Symposium on Foundations of Computer Science* (FOCS), pp. 165–174, Berkeley, CA.

Peterson, D. W. (1970), "Some convergence properties of a nearest neighbor decision rule," *IEEE Trans. Information Theory*, vol. 16, pp. 26–31.

Sabin, M. J. and R. M. Gray (1986), "Global convergence and empirical consistency of the generalized Lloyd algorithm," *IEEE Trans. Information Theory*, vol. 32, no. 2, pp. 148–155.

Sebestyen, G. (1962), *Decision Making Processes in Pattern Recognition*, MacMillan, NY.

Shakhnarovich, G., T. Darrell, and P. Indyk (2006), *Nearest-Neighbor Methods in Learning and Vision*, MIT Press, Cambridge, MA.

Snow, J. (1854), *On the Mode of Communication of Cholera,* 2nd edition, John Churchill, London.

Steinhaus, H. (1957), "Sur la division des corps matériels en parties," *Bull. Acad. Polon. Sci.*, vol. 4, no. 12, pp. 801–804.

Tan, P.-N., M. Steinbach, and V. Kumar (2005), *An Introduction to Data Mining*, Addison-Wesley.

Tukey, J. (1977), *Exploratory Data Analysis*, Addison-Wesley, Reading, MA.

Voronoi, G. F. (1908), "Nouvelles applications des paramètres continus à la théorie de formes quadratiques," *Journal für die reine und angewandte Mathematik*, vol. 134, pp. 198–287.

Ward, J. (1963), "Hierarchical grouping to optimize an objective function," *J. Amer. Statist. Assoc.*, vol. 58, pp. 236–244.

Wilson, D. L. (1972), "Asymptotic properties of nearest neighbor rules using edited data," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 2, no. 3, pp. 408–421.

Witten, I. H., E. Frank, and M. A. Hall (2011), *Data Mining: Practical Machine Learning Tools and Techniques*, 3rd edition, Morgan Kaufmann.