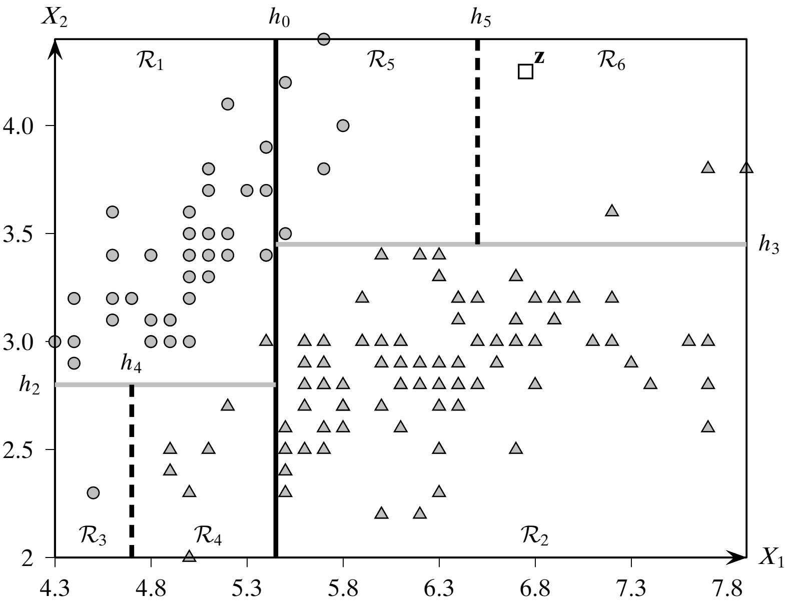
CHAPTER 19 Decision Tree Classifier

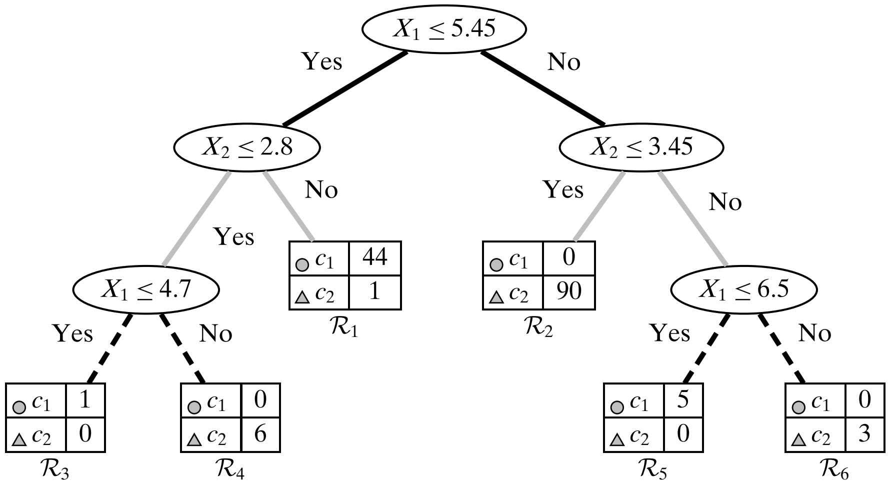
Let the training dataset $\mathbf{D} = \{\mathbf{x}_i, y_i\}_{i=1}^n$ consist of n points in a d-dimensional space, with y_i being the class label for point $\mathbf{x}_i$. We assume that the dimensions or the attributes X_j are numeric or categorical, and that there are k distinct classes, so that $y_i \in \{c_1, c_2, \ldots, c_k\}$. A decision tree classifier is a recursive, partition-based tree model that predicts the class $\hat{y}_i$ for each point $\mathbf{x}_i$. Let $\mathcal{R}$ denote the data space that encompasses the set of input points $\mathbf{D}$. A decision tree uses an axis-parallel hyperplane to split the data space $\mathcal{R}$ into two resulting half-spaces or regions, say $\mathcal{R}_1$ and $\mathcal{R}_2$, which also induces a partition of the input points into $\mathbf{D}_1$ and $\mathbf{D}_2$, respectively. Each of these regions is recursively split via axis-parallel hyperplanes until the points within an induced partition are relatively pure in terms of their class labels, that is, most of the points belong to the same class. The resulting hierarchy of split decisions constitutes the decision tree model, with the leaf nodes labeled with the majority class among points in those regions. To classify a new *test* point we have to recursively evaluate which half-space it belongs to until we reach a leaf node in the decision tree, at which point we predict its class as the label of the leaf.

Example 19.1. Consider the Iris dataset shown in Figure 19.1a, which plots the attributes `sepal length` (X_1) and `sepal width` (X_2). The classification task is to discriminate between c_1, corresponding to `iris-setosa` (in circles), and c_2, corresponding to the other two types of Irises (in triangles). The input dataset $\mathbf{D}$ has $n = 150$ points that lie in the data space which is given as the rectangle, $\mathcal{R} = range(X_1) \times range(X_2) = [4.3, 7.9] \times [2.0, 4.4]$.

The recursive partitioning of the space $\mathcal{R}$ via axis-parallel hyperplanes is illustrated in Figure 19.1a. In two dimensions a hyperplane is simply a line. The first split corresponds to hyperplane h_0 shown as a black line. The resulting left and right half-spaces are further split via hyperplanes h_2 and h_3, respectively (shown as gray lines). The bottom half-space for h_2 is further split via h_4, and the top half-space for h_3 is split via h_5; these third level hyperplanes, h_4 and h_5, are shown as dashed lines. The set of hyperplanes and the set of six leaf regions, namely $\mathcal{R}_1, \ldots, \mathcal{R}_6$, constitute the decision tree model. Note also the induced partitioning of the input points into these six regions.



(a) Recursive Splits



(b) Decision Tree

Figure 19.1. Decision trees: recursive partitioning via axis-parallel hyperplanes.

Consider the test point $\mathbf{z} = (6.75, 4.25)^T$ (shown as a white square). To predict its class, the decision tree first checks which side of h_0 it lies in. Because the point lies in the right half-space, the decision tree next checks h_3 to determine that $\mathbf{z}$ is in the top half-space. Finally, we check and find that $\mathbf{z}$ is in the right half-space of h_5, and we reach the leaf region $\mathcal{R}_6$. The predicted class is c_2, as that leaf region has all points (three of them) with class c_2 (triangles).

19.1 DECISION TREES

A decision tree consists of internal nodes that represent the decisions corresponding to the hyperplanes or split points (i.e., which half-space a given point lies in), and leaf nodes that represent regions or partitions of the data space, which are labeled with the majority class. A region is characterized by the subset of data points that lie in that region.

Axis-Parallel Hyperplanes

A hyperplane $h(\mathbf{x})$ is defined as the set of all points $\mathbf{x}$ that satisfy the following equation

$$h(\mathbf{x}): \mathbf{w}^T\mathbf{x} + b = 0 \tag{19.1}$$

Here $\mathbf{w} \in \mathbb{R}^d$ is a *weight vector* that is normal to the hyperplane, and b is the offset of the hyperplane from the origin. A decision tree considers only *axis-parallel hyperplanes*, that is, the weight vector must be parallel to one of the original dimensions or axes X_j. Put differently, the weight vector $\mathbf{w}$ is restricted *a priori* to one of the standard basis vectors $\{\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_d\}$, where $\mathbf{e}_i \in \mathbb{R}^d$ has a 1 for the jth dimension, and 0 for all other dimensions. If $\mathbf{x} = (x_1, x_2, \dots, x_d)^T$ and assuming $\mathbf{w} = \mathbf{e}_j$, we can rewrite Eq. (19.1) as

$$\begin{aligned} &h(\mathbf{x}): \mathbf{e}_j^T\mathbf{x} + b = 0, \text{ which implies that} \\ &h(\mathbf{x}): x_j + b = 0 \end{aligned}$$

where the choice of the offset b yields different hyperplanes along dimension X_j.

Split Points

A hyperplane specifies a decision or *split point* because it splits the data space $\mathcal{R}$ into two half-spaces. All points $\mathbf{x}$ such that $h(\mathbf{x}) \le 0$ are on the hyperplane or to one side of the hyperplane, whereas all points such that $h(\mathbf{x}) > 0$ are on the other side. The split point associated with an axis-parallel hyperplane can be written as $h(\mathbf{x}) \le 0$, which implies that $x_i + b \le 0$, or $x_i \le -b$. Because x_i is some value from dimension X_j and the offset b can be chosen to be any value, the generic form of a split point for a numeric attribute X_j is given as

$$X_j \le v$$

where $v = -b$ is some value in the domain of attribute X_j. The decision or split point $X_j \le v$ thus splits the input data space $\mathcal{R}$ into two regions $\mathcal{R}_Y$ and $\mathcal{R}_N$, which denote the set of *all possible points* that satisfy the decision and those that do not.

Data Partition

Each split of $\mathcal{R}$ into $\mathcal{R}_Y$ and $\mathcal{R}_N$ also induces a binary partition of the corresponding input data points $\mathbf{D}$. That is, a split point of the form $X_j \le v$ induces the data partition

$$\begin{aligned} \mathbf{D}_Y &= \{\mathbf{x} \mid \mathbf{x} \in \mathbf{D}, x_j \le v\} \\ \mathbf{D}_N &= \{\mathbf{x} \mid \mathbf{x} \in \mathbf{D}, x_j > v\} \end{aligned}$$

where $\mathbf{D}_Y$ is the subset of data points that lie in region $\mathcal{R}_Y$ and $\mathbf{D}_N$ is the subset of input points that line in $\mathcal{R}_N$.

Purity

The purity of a region $\mathcal{R}_j$ is defined in terms of the mixture of classes for points in the corresponding data partition $\mathbf{D}_j$. Formally, purity is the fraction of points with the majority label in $\mathbf{D}_j$, that is,

$$purity(\mathbf{D}_j) = \max_i \left\{ \frac{n_{ji}}{n_j} \right\} \tag{19.2}$$

where $n_j = |\mathbf{D}_j|$ is the total number of data points in the region $\mathcal{R}_j$, and n_{ji} is the number of points in $\mathbf{D}_j$ with class label c_i.

Example 19.2. Figure 19.1b shows the resulting decision tree that corresponds to the recursive partitioning of the space via axis-parallel hyperplanes illustrated in Figure 19.1a. The recursive splitting terminates when appropriate stopping conditions are met, usually taking into account the size and purity of the regions. In this example, we use a size threshold of 5 and a purity threshold of 0.95. That is, a region will be split further only if the number of points is more than five and the purity is less than 0.95.

The very first hyperplane to be considered is $h_1(\mathbf{x}) : x_1 - 5.45 = 0$ which corresponds to the decision

$$X_1 \leq 5.45$$

at the root of the decision tree. The two resulting half-spaces are recursively split into smaller half-spaces.

For example, the region $X_1 \leq 5.45$ is further split using the hyperplane $h_2(\mathbf{x}) : x_2 - 2.8 = 0$ corresponding to the decision

$$X_2 \leq 2.8$$

which forms the left child of the root. Notice how this hyperplane is restricted only to the region $X_1 \leq 5.45$. This is because each region is considered independently after the split, as if it were a separate dataset. There are seven points that satisfy the condition $X_2 \leq 2.8$, out of which one is from class c_1 (circle) and six are from class c_2 (triangles). The purity of this region is therefore $6/7 = 0.857$. Because the region has more than five points, and its purity is less than 0.95, it is further split via the hyperplane $h_4(\mathbf{x}) : x_1 - 4.7 = 0$ yielding the left-most decision node

$$X_1 \leq 4.7$$

in the decision tree shown in Figure 19.1b.

Returning back to the right half-space corresponding to h_2, namely the region $X_2 > 2.8$, it has 45 points, of which only one is a triangle. The size of the region is 45, but the purity is $44/45 = 0.98$. Because the region exceeds the purity threshold it is not split further. Instead, it becomes a leaf node in the decision tree, and the entire region ($\mathcal{R}_1$) is labeled with the majority class c_1. The frequency for each class is also noted at a leaf node so that the potential error rate for that leaf can be computed. For example, we can expect that the probability of misclassification in region $\mathcal{R}_1$ is $1/45 = 0.022$, which is the error rate for that leaf.

Categorical Attributes

In addition to numeric attributes, a decision tree can also handle categorical data. For a categorical attribute X_j, the split points or decisions are of the $X_j \in V$, where $V \subset dom(X_j)$, and $dom(X_j)$ denotes the domain for X_j. Intuitively, this split can be considered to be the categorical analog of a hyperplane. It results in two "half-spaces," one region $\mathcal{R}_Y$ consisting of points $\mathbf{x}$ that satisfy the condition $x_i \in V$, and the other region $\mathcal{R}_N$ comprising points that satisfy the condition $x_i \notin V$.

Decision Rules

One of the advantages of decision trees is that they produce models that are relatively easy to interpret. In particular, a tree can be read as set of decision rules, with each rule's antecedent comprising the decisions on the internal nodes along a path to a leaf, and its consequent being the label of the leaf node. Further, because the regions are all disjoint and cover the entire space, the set of rules can be interpreted as a set of alternatives or disjunctions.

Example 19.3. Consider the decision tree in Figure 19.1b. It can be interpreted as the following set of disjunctive rules, one per leaf region $\mathcal{R}_i$

$$
\begin{aligned}
&\mathcal{R}_3: \text{If } X_1 \le 5.45 \text{ and } X_2 \le 2.8 \text{ and } X_1 \le 4.7, \text{ then class is } c_1, \text{ or}\\
&\mathcal{R}_4: \text{If } X_1 \le 5.45 \text{ and } X_2 \le 2.8 \text{ and } X_1 > 4.7, \text{ then class is } c_2, \text{ or}\\
&\mathcal{R}_1: \text{If } X_1 \le 5.45 \text{ and } X_2 > 2.8, \text{ then class is } c_1, \text{ or}\\
&\mathcal{R}_2: \text{If } X_1 > 5.45 \text{ and } X_2 \le 3.45, \text{ then class is } c_2, \text{ or}\\
&\mathcal{R}_5: \text{If } X_1 > 5.45 \text{ and } X_2 > 3.45 \text{ and } X_1 \le 6.5, \text{ then class is } c_1, \text{ or}\\
&\mathcal{R}_6: \text{If } X_1 > 5.45 \text{ and } X_2 > 3.45 \text{ and } X_1 > 6.5, \text{ then class is } c_2
\end{aligned}
$$

19.2 DECISION TREE ALGORITHM

The pseudo-code for decision tree model construction is shown in Algorithm 19.1. It takes as input a training dataset $\mathbf{D}$, and two parameters η and π, where η is the leaf size and π the leaf purity threshold. Different split points are evaluated for each attribute in $\mathbf{D}$. Numeric decisions are of the form $X_j \le v$ for some value v in the value range for attribute X_j, and categorical decisions are of the form $X_j \in V$ for some subset of values in the domain of X_j. The best split point is chosen to partition the data into two subsets, $\mathbf{D}_Y$ and $\mathbf{D}_N$, where $\mathbf{D}_Y$ corresponds to all points $\mathbf{x} \in \mathbf{D}$ that satisfy the split decision, and $\mathbf{D}_N$ corresponds to all points that do not satisfy the split decision. The decision tree method is then called recursively on $\mathbf{D}_Y$ and $\mathbf{D}_N$. A number of stopping conditions can be used to stop the recursive partitioning process. The simplest condition is based on the size of the partition $\mathbf{D}$. If the number of points n in $\mathbf{D}$ drops below the user-specified size threshold η, then we stop the partitioning process and make $\mathbf{D}$ a leaf. This condition prevents over-fitting the model to the training set, by avoiding to model very small subsets of the data. Size alone is not sufficient because if

ALGORITHM 19.1. Decision Tree Algorithm

```
DecisionTree ($\mathbf{D}, \eta, \pi$):
1  $n \leftarrow |\mathbf{D}|$ // partition size
2  $n_i \leftarrow |\{\mathbf{x}_j | \mathbf{x}_j \in \mathbf{D}, y_j = c_i\}|$ // size of class $c_i$
3  $purity(\mathbf{D}) \leftarrow \max_i \left\{\frac{n_i}{n}\right\}$
4  if $n \leq \eta$ or $purity(\mathbf{D}) \geq \pi$ then // stopping condition
5      $c^* \leftarrow \arg\max_{c_i} \left\{\frac{n_i}{n}\right\}$ // majority class
6      create leaf node, and label it with class $c^*$
7      return
8  $(split\ point^*, score^*) \leftarrow (\emptyset, 0)$ // initialize best split point
9  foreach (attribute $X_j$) do
10     if ($X_j$ is numeric) then
11         $(v, score) \leftarrow$ Evaluate-Numeric-Attribute($\mathbf{D}, X_j$)
12         if $score > score^*$ then $(split\ point^*, score^*) \leftarrow (X_j \leq v, score)$
13     else if ($X_j$ is categorical) then
14         $(V, score) \leftarrow$ Evaluate-Categorical-Attribute($\mathbf{D}, X_j$)
15         if $score > score^*$ then $(split\ point^*, score^*) \leftarrow (X_j \in V, score)$
   // partition $\mathbf{D}$ into $\mathbf{D}_Y$ and $\mathbf{D}_N$ using $split\ point^*$, and call
      recursively
16 $\mathbf{D}_Y \leftarrow \{\mathbf{x} \in \mathbf{D} \mid \mathbf{x}$ satisfies $split\ point^*\}$
17 $\mathbf{D}_N \leftarrow \{\mathbf{x} \in \mathbf{D} \mid \mathbf{x}$ does not satisfy $split\ point^*\}$
18 create internal node $split\ point^*$, with two child nodes, $\mathbf{D}_Y$ and $\mathbf{D}_N$
19 DecisionTree($\mathbf{D}_Y$); DecisionTree($\mathbf{D}_N$)
```

the partition is already pure then it does not make sense to split it further. Thus, the recursive partitioning is also terminated if the purity of **D** is above the purity threshold π. Details of how the split points are evaluated and chosen are given next.

19.2.1 Split Point Evaluation Measures

Given a split point of the form $X_j \leq v$ or $X_j \in V$ for a numeric or categorical attribute, respectively, we need an objective criterion for scoring the split point. Intuitively, we want to select a split point that gives the best separation or discrimination between the different class labels.

Entropy

Entropy, in general, measures the amount of disorder or uncertainty in a system. In the classification setting, a partition has lower entropy (or low disorder) if it is relatively pure, that is, if most of the points have the same label. On the other hand, a partition has higher entropy (or more disorder) if the class labels are mixed, and there is no majority class as such.

The entropy of a set of labeled points $\mathbf{D}$ is defined as follows:

$$H(\mathbf{D}) = -\sum_{i=1}^{k} P(c_i|\mathbf{D}) \log_2 P(c_i|\mathbf{D}) \tag{19.3}$$

where $P(c_i|\mathbf{D})$ is the probability of class c_i in $\mathbf{D}$, and k is the number of classes. If a region is pure, that is, has points from the same class, then the entropy is zero. On the other hand, if the classes are all mixed up, and each appears with equal probability $P(c_i|\mathbf{D}) = \frac{1}{k}$, then the entropy has the highest value, $H(\mathbf{D}) = \log_2 k$.

Assume that a split point partitions $\mathbf{D}$ into $\mathbf{D}_Y$ and $\mathbf{D}_N$. Define the *split entropy* as the weighted entropy of each of the resulting partitions, given as

$$H(\mathbf{D}_Y, \mathbf{D}_N) = \frac{n_Y}{n} H(\mathbf{D}_Y) + \frac{n_N}{n} H(\mathbf{D}_N) \tag{19.4}$$

where $n = |\mathbf{D}|$ is the number of points in $\mathbf{D}$, and $n_Y = |\mathbf{D}_Y|$ and $n_N = |\mathbf{D}_N|$ are the number of points in $\mathbf{D}_Y$ and $\mathbf{D}_N$.

To see if the split point results in a reduced overall entropy, we define the *information gain* for a given split point as follows:

$$Gain(\mathbf{D}, \mathbf{D}_Y, \mathbf{D}_N) = H(\mathbf{D}) - H(\mathbf{D}_Y, \mathbf{D}_N) \tag{19.5}$$

The higher the information gain, the more the reduction in entropy, and the better the split point. Thus, given split points and their corresponding partitions, we can score each split point and choose the one that gives the highest information gain.

Gini Index

Another common measure to gauge the purity of a split point is the *Gini index*, defined as follows:

$$G(\mathbf{D}) = 1 - \sum_{i=1}^{k} P(c_i|\mathbf{D})^2 \tag{19.6}$$

If the partition is pure, then the probability of the majority class is 1 and the probability of all other classes is 0, and thus, the Gini index is 0. On the other hand, when each class is equally represented, with probability $P(c_i|\mathbf{D}) = \frac{1}{k}$, then the Gini index has value $\frac{k-1}{k}$. Thus, higher values of the Gini index indicate more disorder, and lower values indicate more order in terms of the class labels.

We can compute the weighted Gini index of a split point as follows:

$$G(\mathbf{D}_Y, \mathbf{D}_N) = \frac{n_Y}{n} G(\mathbf{D}_Y) + \frac{n_N}{n} G(\mathbf{D}_N)$$

where n, n_Y, and n_N denote the number of points in regions $\mathbf{D}$, $\mathbf{D}_Y$, and $\mathbf{D}_N$, respectively. The lower the Gini index value, the better the split point.

Other measures can also be used instead of entropy and Gini index to evaluate the splits. For example, the Classification And Regression Trees (CART) measure is given as

$$CART(\mathbf{D}_Y, \mathbf{D}_N) = 2\frac{n_Y}{n}\frac{n_N}{n} \sum_{i=1}^{k} \Big| P(c_i|\mathbf{D}_Y) - P(c_i|\mathbf{D}_N) \Big| \tag{19.7}$$

This measure thus prefers a split point that maximizes the difference between the class probability mass function for the two partitions; the higher the CART measure, the better the split point.

19.2.2 Evaluating Split Points

All of the split point evaluation measures, such as entropy [Eq. (19.3)], Gini-index [Eq. (19.6)], and CART [Eq. (19.7)], considered in the preceding section depend on the class probability mass function (PMF) for $\mathbf{D}$, namely, $P(c_i|\mathbf{D})$, and the class PMFs for the resulting partitions $\mathbf{D}_Y$ and $\mathbf{D}_N$, namely $P(c_i|\mathbf{D}_Y)$ and $P(c_i|\mathbf{D}_N)$. Note that we have to compute the class PMFs for all possible split points; scoring each of them independently would result in significant computational overhead. Instead, one can incrementally compute the PMFs as described in the following paragraphs.

Numeric Attributes

If X is a numeric attribute, we have to evaluate split points of the form $X \le v$. Even if we restrict v to lie within the value range of attribute X, there are still an infinite number of choices for v. One reasonable approach is to consider only the midpoints between two successive distinct values for X in the sample $\mathbf{D}$. This is because split points of the form $X \le v$, for $v \in [x_a, x_b)$, where x_a and x_b are two successive distinct values of X in $\mathbf{D}$, produce the same partitioning of $\mathbf{D}$ into $\mathbf{D}_Y$ and $\mathbf{D}_N$, and thus yield the same scores. Because there can be at most n distinct values for X, there are at most $n-1$ midpoint values to consider.

Let $\{v_1, \ldots, v_m\}$ denote the set of all such midpoints, such that $v_1 < v_2 < \cdots < v_m$. For each split point $X \le v$, we have to estimate the class PMFs:

$$\hat{P}(c_i|\mathbf{D}_Y) = \hat{P}(c_i|X \le v) \tag{19.8}$$

$$\hat{P}(c_i|\mathbf{D}_N) = \hat{P}(c_i|X > v) \tag{19.9}$$

Let $I()$ be an indicator variable that takes on the value 1 only when its argument is true, and is 0 otherwise. Using the Bayes theorem, we have

$$\hat{P}(c_i|X \le v) = \frac{\hat{P}(X \le v|c_i)\hat{P}(c_i)}{\hat{P}(X \le v)} = \frac{\hat{P}(X \le v|c_i)\hat{P}(c_i)}{\sum_{j=1}^{k} \hat{P}(X \le v|c_j)\hat{P}(c_j)} \tag{19.10}$$

The prior probability for each class in $\mathbf{D}$ can be estimated as follows:

$$\hat{P}(c_i) = \frac{1}{n}\sum_{j=1}^{n} I(y_j = c_i) = \frac{n_i}{n} \tag{19.11}$$

where y_j is the class for point $\mathbf{x}_j$, $n = |\mathbf{D}|$ is the total number of points, and n_i is the number of points in $\mathbf{D}$ with class c_i. Define N_{vi} as the number of points $x_j \le v$ with class c_i, where x_j is the value of data point $\mathbf{x}_j$ for the attribute X, given as

$$N_{vi} = \sum_{j=1}^{n} I(x_j \le v \text{ and } y_j = c_i) \tag{19.12}$$

We can then estimate $P(X \leq v|c_i)$ as follows:

$$\hat{P}(X \leq v|c_i) = \frac{\hat{P}(X \leq v \text{ and } c_i)}{\hat{P}(c_i)} = \left(\frac{1}{n}\sum_{j=1}^{n} I(x_j \leq v \text{ and } y_j = c_i)\right) \Big/ (n_i/n)$$
$$= \frac{N_{vi}}{n_i} \tag{19.13}$$

Plugging Eqs. (19.11) and (19.13) into Eq. (19.10), and using Eq. (19.8), we have

$$\hat{P}(c_i|\mathbf{D}_Y) = \hat{P}(c_i|X \leq v) = \frac{N_{vi}}{\sum_{j=1}^{k} N_{vj}} \tag{19.14}$$

We can estimate $\hat{P}(X > v|c_i)$ as follows:

$$\hat{P}(X > v|c_i) = 1 - \hat{P}(X \leq v|c_i) = 1 - \frac{N_{vi}}{n_i} = \frac{n_i - N_{vi}}{n_i} \tag{19.15}$$

Using Eqs. (19.11) and (19.15), the class PMF $\hat{P}(c_i|\mathbf{D}_N)$ is given as

$$\hat{P}(c_i|\mathbf{D}_N) = \hat{P}(c_i|X > v) = \frac{\hat{P}(X > v|c_i)\hat{P}(c_i)}{\sum_{j=1}^{k} \hat{P}(X > v|c_j)\hat{P}(c_j)} = \frac{n_i - N_{vi}}{\sum_{j=1}^{k}(n_j - N_{vj})} \tag{19.16}$$

Algorithm 19.2 shows the split point evaluation method for numeric attributes. The for loop on line 4 iterates through all the points and computes the midpoint values v and the number of points N_{vi} from class c_i such that $x_j \leq v$. The for loop on line 12 enumerates all possible split points of the form $X \leq v$, one for each midpoint v, and scores them using the gain criterion [Eq. (19.5)]; the best split point and score are recorded and returned. Any of the other evaluation measures can also be used. However, for Gini index and CART a lower score is better unlike for gain where a higher score is better.

In terms of computational complexity, the initial sorting of values of X (line 1) takes time $O(n \log n)$. The cost of computing the midpoints and the class-specific counts N_{vi} takes time $O(nk)$ (for loop on line 4). The cost of computing the score is also bounded by $O(nk)$, because the total number of midpoints v can be at most n (for loop on line 12). The total cost of evaluating a numeric attribute is therefore $O(n \log n + nk)$. Ignoring k, because it is usually a small constant, the total cost of numeric split point evaluation is $O(n \log n)$.

Example 19.4 (Numeric Attributes). Consider the 2-dimensional Iris dataset shown in Figure 19.1a. In the initial invocation of Algorithm 19.1, the entire dataset $\mathbf{D}$ with $n = 150$ points is considered at the root of the decision tree. The task is to find the best split point considering both the attributes, X_1 (`sepal length`) and X_2 (`sepal width`). Because there are $n_1 = 50$ points labeled c_1 (`iris-setosa`), the other class c_2 has $n_2 = 100$ points. We thus have

$$\hat{P}(c_1) = 50/150 = 1/3$$
$$\hat{P}(c_2) = 100/150 = 2/3$$

ALGORITHM 19.2. Evaluate Numeric Attribute (Using Gain)

```
EVALUATE-NUMERIC-ATTRIBUTE (D, X):
1  sort $\mathbf{D}$ on attribute $X$, so that $x_j \le x_{j+1}, \forall j = 1, \ldots, n-1$
2  $\mathcal{M} \leftarrow \emptyset$ // set of midpoints
3  for $i = 1, \ldots, k$ do $n_i \leftarrow 0$
4  for $j = 1, \ldots, n-1$ do
5      if $y_j = c_i$ then $n_i \leftarrow n_i + 1$ // running count for class $c_i$
6      if $x_{j+1} \ne x_j$ then
7          $v \leftarrow \frac{x_{j+1}+x_j}{2}$; $\mathcal{M} \leftarrow \mathcal{M} \cup \{v\}$ // midpoints
8          for $i = 1, \ldots, k$ do
9              $N_{vi} \leftarrow n_i$ // Number of points such that $x_j \le v$ and $y_j = c_i$
10 if $y_n = c_i$ then $n_i \leftarrow n_i + 1$
   // evaluate split points of the form $X \le v$
11 $v^* \leftarrow \emptyset$; $score^* \leftarrow 0$ // initialize best split point
12 forall $v \in \mathcal{M}$ do
13     for $i = 1, \ldots, k$ do
14         $\hat{P}(c_i|\mathbf{D}_Y) \leftarrow \frac{N_{vi}}{\sum_{j=1}^{k} N_{vj}}$
15         $\hat{P}(c_i|\mathbf{D}_N) \leftarrow \frac{n_i - N_{vi}}{\sum_{j=1}^{k} n_j - N_{vj}}$
16     $score(X \le v) \leftarrow Gain(\mathbf{D}, \mathbf{D}_Y, \mathbf{D}_N)$ // use Eq. (19.5)
17     if $score(X \le v) > score^*$ then
18         $v^* \leftarrow v$; $score^* \leftarrow score(X \le v)$
19 return $(v^*, score^*)$
```

The entropy [Eq. (19.3)] of the dataset $\mathbf{D}$ is therefore

$$H(\mathbf{D}) = -\left(\frac{1}{3}\log_2\frac{1}{3} + \frac{2}{3}\log_2\frac{2}{3}\right) = 0.918$$

Consider split points for attribute X_1. To evaluate the splits we first compute the frequencies N_{vi} using Eq. (19.12), which are plotted in Figure 19.2 for both the classes. For example, consider the split point $X_1 \le 5.45$. From Figure 19.2, we see that

$$N_{v1} = 45 \qquad\qquad N_{v2} = 7$$

Plugging in these values into Eq. (19.14) we get

$$\hat{P}(c_1|\mathbf{D}_Y) = \frac{N_{v1}}{N_{v1}+N_{v2}} = \frac{45}{45+7} = 0.865$$

$$\hat{P}(c_2|\mathbf{D}_Y) = \frac{N_{v2}}{N_{v1}+N_{v2}} = \frac{7}{45+7} = 0.135$$

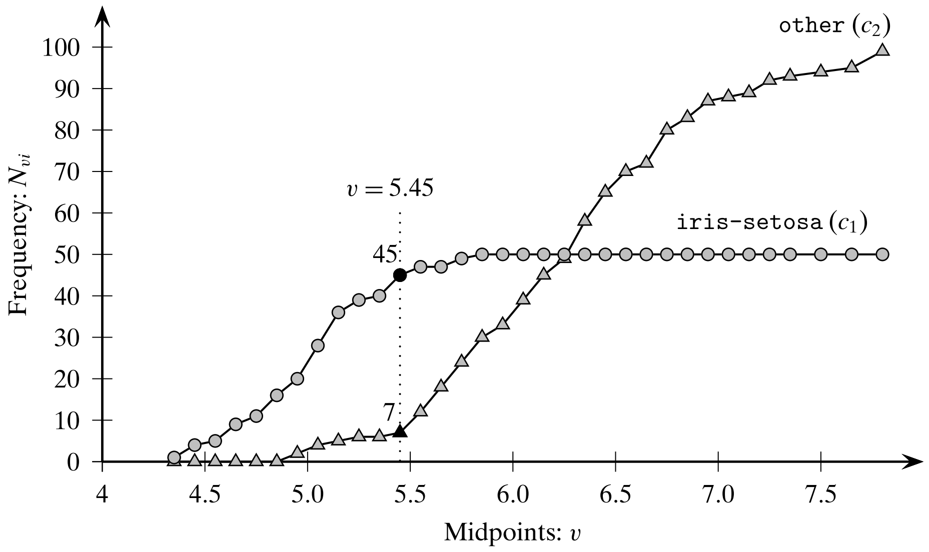


Figure 19.2. Iris: frequencies N_{vi} for classes c_1 and c_2 for attribute `sepal length`.

and using Eq. (19.16), we obtain

$$\hat{P}(c_1|\mathbf{D}_N) = \frac{n_1 - N_{v1}}{(n_1 - N_{v1}) + (n_2 - N_{v2})} = \frac{50-45}{(50-45)+(100-7)} = 0.051$$

$$\hat{P}(c_2|\mathbf{D}_N) = \frac{n_2 - N_{v2}}{(n_1 - N_{v1}) + (n_2 - N_{v2})} = \frac{(100-7)}{(50-45)+(100-7)} = 0.949$$

We can now compute the entropy of the partitions $\mathbf{D}_Y$ and $\mathbf{D}_N$ as follows:

$$H(\mathbf{D}_Y) = -(0.865 \log_2 0.865 + 0.135 \log_2 0.135) = 0.571$$

$$H(\mathbf{D}_N) = -(0.051 \log_2 0.051 + 0.949 \log_2 0.949) = 0.291$$

The entropy of the split point $X \leq 5.45$ is given via Eq. (19.4)

$$H(\mathbf{D}_Y, \mathbf{D}_N) = \frac{52}{150} H(\mathbf{D}_Y) + \frac{98}{150} H(\mathbf{D}_N) = 0.388$$

where $n_Y = |\mathbf{D}_Y| = 52$ and $n_N = |\mathbf{D}_N| = 98$. The information gain for the split point is therefore

$$Gain = H(\mathbf{D}) - H(\mathbf{D}_Y, \mathbf{D}_N) = 0.918 - 0.388 = 0.53$$

In a similar manner, we can evaluate all of the split points for both attributes X_1 and X_2. Figure 19.3 plots the gain values for the different split points for the two attributes. We can observe that $X \leq 5.45$ is the best split point and it is thus chosen as the root of the decision tree in Figure 19.1b.

The recursive tree growth process continues and yields the final decision tree and the split points as shown in Figure 19.1b. In this example, we use a leaf size threshold of 5 and a purity threshold of 0.95.

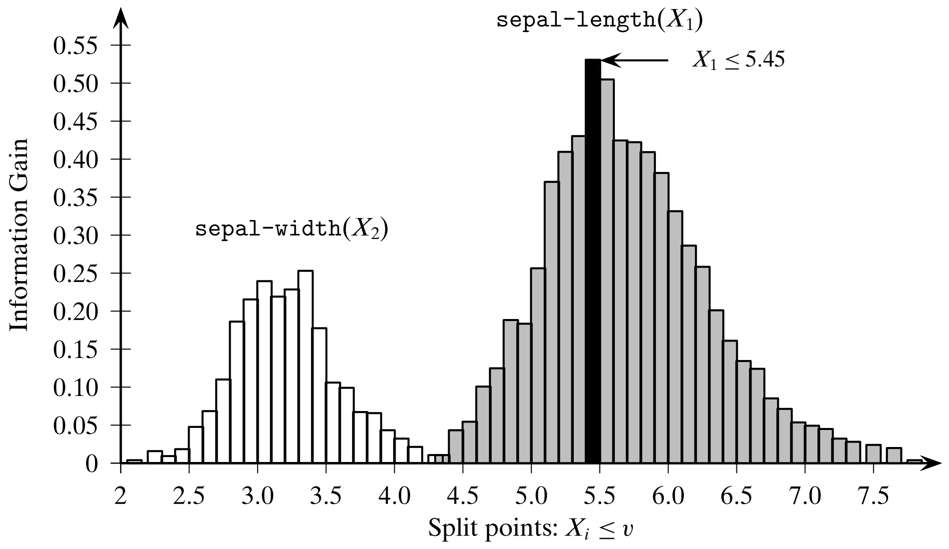


Figure 19.3. Iris: gain for different split points, for `sepal length` and `sepal width`.

Categorical Attributes

If X is a categorical attribute we evaluate split points of the form $X \in V$, where $V \subset dom(X)$ and $V \neq \emptyset$. In words, all distinct partitions of the set of values of X are considered. Because the split point $X \in V$ yields the same partition as $X \in \overline{V}$, where $\overline{V} = dom(X) \setminus V$ is the complement of V, the total number of distinct partitions is given as

$$\sum_{i=1}^{\lfloor m/2 \rfloor} \binom{m}{i} = O(2^{m-1}) \tag{19.17}$$

where m is the number of values in the domain of X, that is, $m = |dom(X)|$. The number of possible split points to consider is therefore exponential in m, which can pose problems if m is large. One simplification is to restrict V to be of size one, so that there are only m split points of the form $X_j \in \{v\}$, where $v \in dom(X_j)$.

To evaluate a given split point $X \in V$ we have to compute the following class probability mass functions:

$$P(c_i|\mathbf{D}_Y) = P(c_i|X \in V) \qquad\qquad P(c_i|\mathbf{D}_N) = P(c_i|X \notin V)$$

Making use of the Bayes theorem, we have

$$P(c_i|X \in V) = \frac{P(X \in V|c_i)P(c_i)}{P(X \in V)} = \frac{P(X \in V|c_i)P(c_i)}{\sum_{j=1}^{k} P(X \in V|c_j)P(c_j)}$$

However, note that a given point $\mathbf{x}$ can take on only one value in the domain of X, and thus the values $v \in dom(X)$ are mutually exclusive. Therefore, we have

$$P(X \in V|c_i) = \sum_{v \in V} P(X = v|c_i)$$

and we can rewrite $P(c_i|\mathbf{D}_Y)$ as

$$P(c_i|\mathbf{D}_Y) = \frac{\sum_{v \in V} P(X=v|c_i)P(c_i)}{\sum_{j=1}^{k} \sum_{v \in V} P(X=v|c_j)P(c_j)} \tag{19.18}$$

Define n_{vi} as the number of points $\mathbf{x}_j \in \mathbf{D}$, with value $x_j = v$ for attribute X and having class $y_j = c_i$:

$$n_{vi} = \sum_{j=1}^{n} I(x_j = v \text{ and } y_j = c_i) \tag{19.19}$$

The class conditional empirical PMF for X is then given as

$$\begin{aligned}\hat{P}(X=v|c_i) &= \frac{\hat{P}\big(X=v \text{ and } c_i\big)}{\hat{P}(c_i)} \\ &= \left(\frac{1}{n}\sum_{j=1}^{n} I(x_j = v \text{ and } y_j = c_i)\right) \Big/ \big(n_i/n\big) \\ &= \frac{n_{vi}}{n_i}\end{aligned} \tag{19.20}$$

Note that the class prior probabilities can be estimated using Eq. (19.11) as discussed earlier, that is, $\hat{P}(c_i) = n_i/n$. Thus, substituting Eq. (19.20) in Eq. (19.18), the class PMF for the partition $\mathbf{D}_Y$ for the split point $X \in V$ is given as

$$\hat{P}(c_i|\mathbf{D}_Y) = \frac{\sum_{v \in V} \hat{P}(X=v|c_i)\hat{P}(c_i)}{\sum_{j=1}^{k} \sum_{v \in V} \hat{P}(X=v|c_j)\hat{P}(c_j)} = \frac{\sum_{v \in V} n_{vi}}{\sum_{j=1}^{k} \sum_{v \in V} n_{vj}} \tag{19.21}$$

In a similar manner, the class PMF for the partition $\mathbf{D}_N$ is given as

$$\hat{P}(c_i|\mathbf{D}_N) = \hat{P}(c_i|X \notin V) = \frac{\sum_{v \notin V} n_{vi}}{\sum_{j=1}^{k} \sum_{v \notin V} n_{vj}} \tag{19.22}$$

Algorithm 19.3 shows the split point evaluation method for categorical attributes. The for loop on line 4 iterates through all the points and computes n_{vi}, that is, the number of points having value $v \in dom(X)$ and class c_i. The for loop on line 7 enumerates all possible split points of the form $X \in V$ for $V \subset dom(X)$, such that $|V| \le l$, where l is a user specified parameter denoting the maximum cardinality of V. For example, to control the number of split points, we can also restrict V to be a single item, that is, $l = 1$, so that splits are of the form $V \in \{v\}$, with $v \in dom(X)$. If $l = \lfloor m/2 \rfloor$, we have to consider all possible distinct partitions V. Given a split point $X \in V$, the method scores it using information gain [Eq. (19.5)], although any of the other scoring criteria can also be used. The best split point and score are recorded and returned.

In terms of computational complexity the class-specific counts for each value n_{vi} takes $O(n)$ time (for loop on line 4). With $m = |dom(X)|$, the maximum number of partitions V is $O(2^{m-1})$, and because each split point can be evaluated in time $O(mk)$, the for loop in line 7 takes time $O(mk2^{m-1})$. The total cost for categorical attributes is therefore $O(n + mk2^{m-1})$. If we make the assumption that $2^{m-1} = O(n)$, that is, if we bound the maximum size of V to $l = O(\log n)$, then the cost of categorical splits is bounded as $O(n \log n)$, ignoring k.

ALGORITHM 19.3. Evaluate Categorical Attribute (Using Gain)

```
EVALUATE-CATEGORICAL-ATTRIBUTE ($\mathbf{D}, X, l$):
1  for $i = 1, \ldots, k$ do
2      $n_i \leftarrow 0$
3      forall $v \in dom(X)$ do $n_{vi} \leftarrow 0$
4  for $j = 1, \ldots, n$ do
5      if $x_j = v$ and $y_j = c_i$ then $n_{vi} \leftarrow n_{vi} + 1$ // frequency statistics
   // evaluate split points of the form $X \in V$
6  $V^* \leftarrow \emptyset$; $score^* \leftarrow 0$ // initialize best split point
7  forall $V \subset dom(X)$, such that $1 \le |V| \le l$ do
8      for $i = 1, \ldots, k$ do
9          $\hat{P}(c_i|\mathbf{D}_Y) \leftarrow \frac{\sum_{v \in V} n_{vi}}{\sum_{j=1}^{k} \sum_{v \in V} n_{vj}}$
10         $\hat{P}(c_i|\mathbf{D}_N) \leftarrow \frac{\sum_{v \notin V} n_{vi}}{\sum_{j=1}^{k} \sum_{v \notin V} n_{vj}}$
11     $score(X \in V) \leftarrow Gain(\mathbf{D}, \mathbf{D}_Y, \mathbf{D}_N)$ // use Eq. (19.5)
12     if $score(X \in V) > score^*$ then
13         $V^* \leftarrow V$; $score^* \leftarrow score(X \in V)$
14 return $(V^*, score^*)$
```

Example 19.5 (Categorical Attributes). Consider the 2-dimensional Iris dataset comprising the `sepal length` and `sepal width` attributes. Let us assume that `sepal length` has been discretized as shown in Table 19.1. The class frequencies n_{vi} are also shown. For instance $n_{a_1 2} = 6$ denotes the fact that there are 6 points in $\mathbf{D}$ with value $v = a_1$ and class c_2.

Consider the split point $X_1 \in \{a_1, a_3\}$. From Table 19.1 we can compute the class PMF for partition $\mathbf{D}_Y$ using Eq. (19.21)

$$\hat{P}(c_1|\mathbf{D}_Y) = \frac{n_{a_1 1} + n_{a_3 1}}{(n_{a_1 1} + n_{a_3 1}) + (n_{a_1 2} + n_{a_3 2})} = \frac{39 + 0}{(39 + 0) + (6 + 43)} = 0.443$$

$$\hat{P}(c_2|\mathbf{D}_Y) = 1 - \hat{P}(c_1|\mathbf{D}_Y) = 0.557$$

with the entropy given as

$$H(\mathbf{D}_Y) = -(0.443 \log_2 0.443 + 0.557 \log_2 0.557) = 0.991$$

To compute the class PMF for $\mathbf{D}_N$ [Eq. (19.22)], we sum up the frequencies over values $v \notin V = \{a_1, a_3\}$, that is, we sum over $v = a_2$ and $v = a_4$, as follows:

$$\hat{P}(c_1|\mathbf{D}_N) = \frac{n_{a_2 1} + n_{a_4 1}}{(n_{a_2 1} + n_{a_4 1}) + (n_{a_2 2} + n_{a_4 2})} = \frac{11 + 0}{(11 + 0) + (39 + 12)} = 0.177$$

$$\hat{P}(c_2|\mathbf{D}_N) = 1 - \hat{P}(c_1|\mathbf{D}_N) = 0.823$$

Table 19.1. Discretized `sepal length` attribute: class frequencies

Bins	v: values	Class frequencies (n_{vi})	
		c_1:`iris-setosa`	c_2:`other`
$[4.3, 5.2]$	Very Short (a_1)	39	6
$(5.2, 6.1]$	Short (a_2)	11	39
$(6.1, 7.0]$	Long (a_3)	0	43
$(7.0, 7.9]$	Very Long (a_4)	0	12

Table 19.2. Categorical split points for `sepal length`

V	Split entropy	Info. gain
$\{a_1\}$	0.509	0.410
$\{a_2\}$	0.897	0.217
$\{a_3\}$	0.711	0.207
$\{a_4\}$	0.869	0.049
$\{a_1, a_2\}$	0.632	0.286
$\{a_1, a_3\}$	0.860	0.058
$\{a_1, a_4\}$	0.667	0.251
$\{a_2, a_3\}$	0.667	0.251
$\{a_2, a_4\}$	0.860	0.058
$\{a_3, a_4\}$	0.632	0.286

with the entropy given as

$$H(\mathbf{D}_N) = -(0.177 \log_2 0.177 + 0.823 \log_2 0.823) = 0.673$$

We can see from Table 19.1 that $V \in \{a_1, a_3\}$ splits the input data $\mathbf{D}$ into partitions of size $|\mathbf{D}_Y| = 39 + 6 + 43 = 88$, and $\mathbf{D}_N = 150 - 88 = 62$. The entropy of the split is therefore given as

$$H(\mathbf{D}_Y, \mathbf{D}_N) = \frac{88}{150} H(\mathbf{D}_Y) + \frac{62}{150} H(\mathbf{D}_N) = 0.86$$

As noted in Example 19.4, the entropy of the whole dataset $\mathbf{D}$ is $H(\mathbf{D}) = 0.918$. The gain is then given as

$$Gain = H(\mathbf{D}) - H(\mathbf{D}_Y, \mathbf{D}_N) = 0.918 - 0.86 = 0.058$$

The split entropy and gain values for all the categorical split points are given in Table 19.2. We can see that $X_1 \in \{a_1\}$ is the best split point on the discretized attribute X_1.

19.2.3 Computational Complexity

To analyze the computational complexity of the decision tree method in Algorithm 19.1, we assume that the cost of evaluating all the split points for a numeric or categorical attribute is $O(n\log n)$, where $n = |\mathbf{D}|$ is the size of the dataset. Given $\mathbf{D}$, the decision tree algorithm evaluates all d attributes, with cost $(dn\log n)$. The total cost depends on the depth of the decision tree. In the worst case, the tree can have depth n, and thus the total cost is $O(dn^2\log n)$.

19.3 FURTHER READING

Among the earliest works on decision trees are Hunt, Marin, and Stone (1966); Breiman et al. (1984); and Quinlan (1986). The description in this chapter is largely based on the C4.5 method described in Quinlan (1993), which is an excellent reference for further details, such as how to prune decision trees to prevent overfitting, how to handle missing attribute values, and other implementation issues. A survey of methods for simplifying decision trees appears in Breslow and Aha (1997). Scalable implementation techniques are described in Mehta, Agrawal, and Rissanen (1996) and Gehrke et al. (1999).

Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. (1984). *Classification and Regression Trees*. Boca Raton, FL: Chapman and Hall/CRC Press.

Breslow, L. A. and Aha, D. W. (1997). "Simplifying decision trees: A survey." *Knowledge Engineering Review*, 12 (1): 1–40.

Gehrke, J., Ganti, V., Ramakrishnan, R., and Loh, W.-Y. (1999). "BOAT-optimistic decision tree construction." *ACM SIGMOD Record*, 28 (2): 169–180.

Hunt, E. B., Marin, J., and Stone, P. J. (1966). *Experiments in Induction*. New York: Academic Press.

Mehta, M., Agrawal, R., and Rissanen, J. (1996). "SLIQ: A fast scalable classifier for data mining." *In Proceedings of the International Conference on Extending Database Technology* (pp. 18–32). New York: Springer-Verlag.

Quinlan, J. R. (1986). "Induction of decision trees." *Machine Learning*, 1 (1): 81–106.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. New York: Morgan Kaufmann.

19.4 EXERCISES

Q1. True or False:

- **(a)** High entropy means that the partitions in classification are "pure."
- **(b)** Multiway split of a categorical attribute generally results in more pure partitions than a binary split.

Q2. Given Table 19.3, construct a decision tree using a purity threshold of 100%. Use information gain as the split point evaluation measure. Next, classify the point (Age=27,Car=Vintage).

Q3. What is the maximum and minimum value of the CART measure [Eq. (19.7)] and under what conditions?

Table 19.3. Data for Q2: `Age` is numeric and `Car` is categorical. `Risk` gives the class label for each point: high (H) or low (L)

Point	Age	Car	**Risk**
$\mathbf{x}_1$	25	Sports	L
$\mathbf{x}_2$	20	Vintage	H
$\mathbf{x}_3$	25	Sports	L
$\mathbf{x}_4$	45	SUV	H
$\mathbf{x}_5$	20	Sports	H
$\mathbf{x}_6$	25	SUV	H

Q4. Given the dataset in Table 19.4. Answer the following questions:

Table 19.4. Data for Q4

Instance	a_1	a_2	a_3	Class
1	T	T	5.0	Y
2	T	T	7.0	Y
3	T	F	8.0	N
4	F	F	3.0	Y
5	F	T	7.0	N
6	F	T	4.0	N
7	F	F	5.0	N
8	T	F	6.0	Y
9	F	T	1.0	N

(a) Show which decision will be chosen at the root of the decision tree using information gain [Eq. (19.5)], Gini index [Eq. (19.6)], and CART [Eq. (19.7)] measures. Show all split points for all attributes.

(b) What happens to the purity if we use Instance as another attribute? Do you think this attribute should be used for a decision in the tree?

Q5. Consider Table 19.5. Let us make a nonlinear split instead of an axis parallel split, given as follows: $AB - B^2 \leq 0$. Compute the information gain of this split based on entropy (use $\log_2$, i.e., log to the base 2).

Table 19.5. Data for Q5

	A	B	Class
x_1	3.5	4	H
x_2	2	4	H
x_3	9.1	4.5	L
x_4	2	6	H
x_5	1.5	7	H
x_6	7	6.5	H
x_7	2.1	2.5	L
x_8	8	4	L

CHAPTER 20 Linear Discriminant Analysis

Given labeled data consisting of d-dimensional points $\mathbf{x}_i$ along with their classes y_i, the goal of linear discriminant analysis (LDA) is to find a vector $\mathbf{w}$ that maximizes the separation between the classes after projection onto $\mathbf{w}$. Recall from Chapter 7 that the first principal component is the vector that maximizes the projected variance of the points. The key difference between principal component analysis and LDA is that the former deals with unlabeled data and tries to maximize variance, whereas the latter deals with labeled data and tries to maximize the discrimination between the classes.

20.1 OPTIMAL LINEAR DISCRIMINANT

Let us assume that the dataset $\mathbf{D}$ consists of n labeled points $\{\mathbf{x}_i, y_i\}$, where $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \{c_1, c_2, \ldots, c_k\}$. Let $\mathbf{D}_i$ denote the subset of points labeled with class c_i, i.e., $\mathbf{D}_i = \{\mathbf{x}_j | y_j = c_i\}$, and let $|\mathbf{D}_i| = n_i$ denote the number of points with class c_i. We assume that there are only $k = 2$ classes. Thus, the dataset $\mathbf{D}$ can be partitioned into $\mathbf{D}_1$ and $\mathbf{D}_2$.

Let $\mathbf{w}$ be a unit vector, that is, $\mathbf{w}^T\mathbf{w} = 1$. By Eq. (1.7), the projection of any d-dimensional point $\mathbf{x}_i$ onto the vector $\mathbf{w}$ is given as

$$\mathbf{x}_i' = \left(\frac{\mathbf{w}^T\mathbf{x}_i}{\mathbf{w}^T\mathbf{w}}\right)\mathbf{w} = \left(\mathbf{w}^T\mathbf{x}_i\right)\mathbf{w} = a_i\mathbf{w}$$

where a_i specifies the offset or coordinate of $\mathbf{x}_i'$ along the line $\mathbf{w}$:

$$a_i = \mathbf{w}^T\mathbf{x}_i$$

Thus, the set of n scalars $\{a_1, a_2, \ldots, a_n\}$ represents the mapping from $\mathbb{R}^d$ to $\mathbb{R}$, that is, from the original d-dimensional space to a 1-dimensional space (along $\mathbf{w}$).

Example 20.1. Consider Figure 20.1, which shows the 2-dimensional Iris dataset with `sepal length` and `sepal width` as the attributes, and `iris-setosa` as class c_1 (circles), and the other two Iris types as class c_2 (triangles). There are $n_1 = 50$ points in c_1 and $n_2 = 100$ points in c_2. One possible vector $\mathbf{w}$ is shown, along with the projection

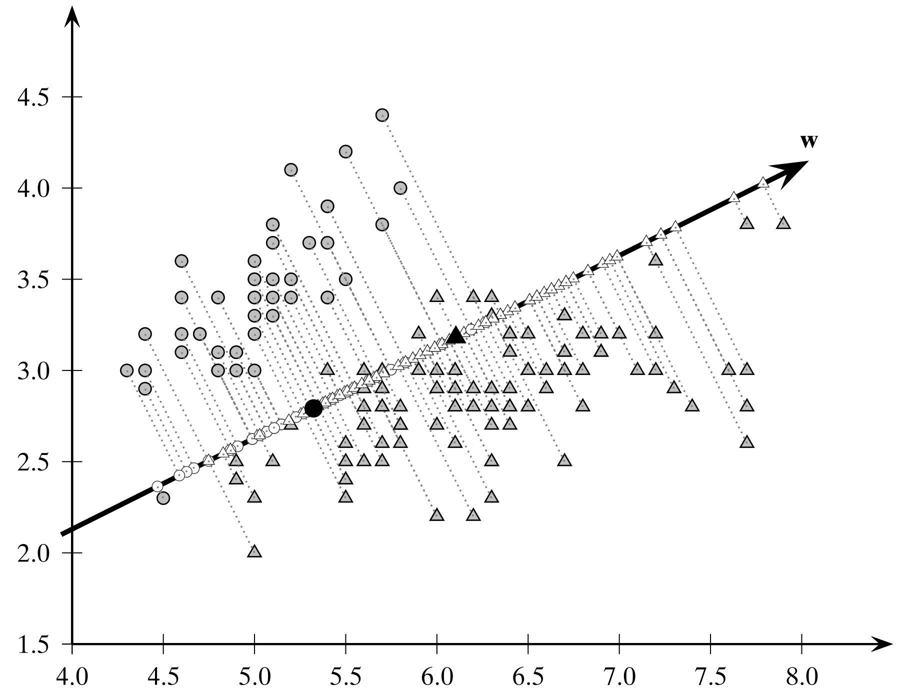


Figure 20.1. Projection onto **w**.

of all the points onto **w**. The projected means of the two classes are shown in black. Here **w** has been translated so that it passes through the mean of the entire data. One can observe that **w** is not very good in discriminating between the two classes because the projection of the points onto **w** are all mixed up in terms of their class labels. The optimal linear discriminant direction is shown in Figure 20.2.

Each point coordinate a_i has associated with it the original class label y_i, and thus we can compute, for each of the two classes, the mean of the projected points as follows:

$$\begin{aligned} m_1 &= \frac{1}{n_1} \sum_{\mathbf{x}_i \in \mathbf{D}_1} a_i \\ &= \frac{1}{n_1} \sum_{\mathbf{x}_i \in \mathbf{D}_1} \mathbf{w}^T \mathbf{x}_i \\ &= \mathbf{w}^T \left(\frac{1}{n_1} \sum_{\mathbf{x}_i \in \mathbf{D}_1} \mathbf{x}_i \right) \\ &= \mathbf{w}^T \boldsymbol{\mu}_1 \end{aligned}$$

where $\boldsymbol{\mu}_1$ is the mean of all point in $\mathbf{D}_1$. Likewise, we can obtain

$$m_2 = \mathbf{w}^T \boldsymbol{\mu}_2$$

In other words, the mean of the projected points is the same as the projection of the mean.

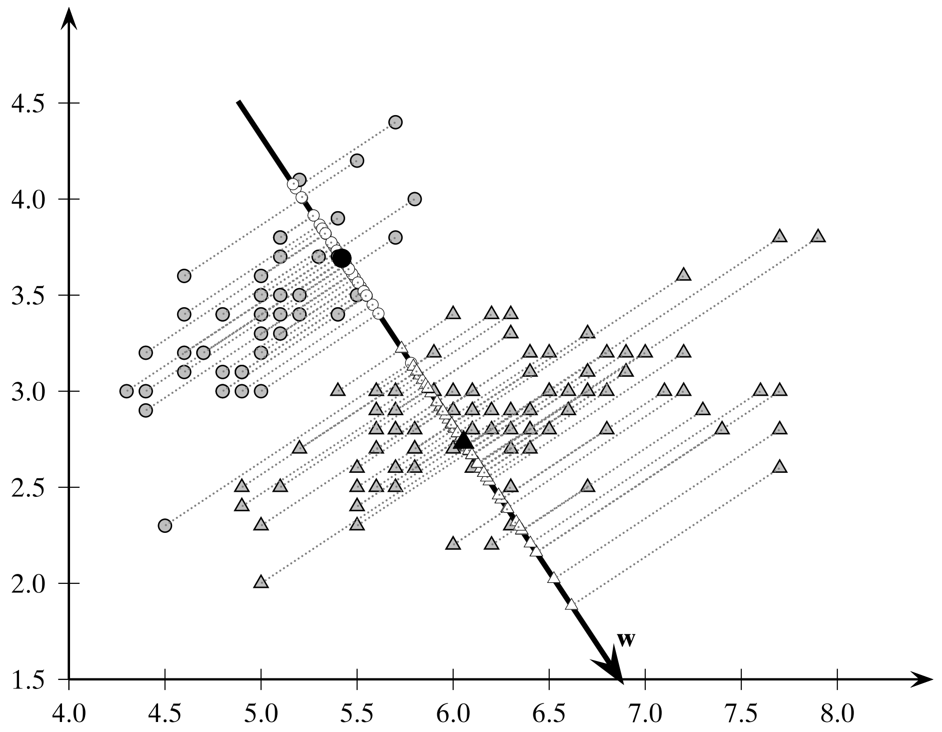


Figure 20.2. Linear discriminant direction $\mathbf{w}$.

To maximize the separation between the classes, it seems reasonable to maximize the difference between the projected means, $|m_1 - m_2|$. However, this is not enough. For good separation, the variance of the projected points for each class should also not be too large. A large variance would lead to possible overlaps among the points of the two classes due to the large spread of the points, and thus we may fail to have a good separation. LDA maximizes the separation by ensuring that the *scatter* s_i^2 for the projected points within each class is small, where scatter is defined as

$$s_i^2 = \sum_{\mathbf{x}_j \in \mathbf{D}_i} (a_j - m_i)^2$$

Scatter is the total squared deviation from the mean, as opposed to the variance, which is the average deviation from mean. In other words

$$s_i^2 = n_i \sigma_i^2$$

where $n_i = |\mathbf{D}_i|$ is the size, and σ_i^2 is the variance, for class c_i.

We can incorporate the two LDA criteria, namely, maximizing the distance between projected means and minimizing the sum of projected scatter, into a single maximization criterion called the *Fisher LDA objective*:

$$\max_{\mathbf{w}} J(\mathbf{w}) = \frac{(m_1 - m_2)^2}{s_1^2 + s_2^2} \tag{20.1}$$

The goal of LDA is to find the vector $\mathbf{w}$ that maximizes $J(\mathbf{w})$, that is, the direction that maximizes the separation between the two means m_1 and m_2, and minimizes the total scatter $s_1^2 + s_2^2$ of the two classes. The vector $\mathbf{w}$ is also called the *optimal linear discriminant (LD)*. The optimization objective [Eq. (20.1)] is in the projected space. To solve it, we have to rewrite it in terms of the input data, as described next.

Note that we can rewrite $(m_1 - m_2)^2$ as follows:

$$\begin{aligned}(m_1 - m_2)^2 &= \left(\mathbf{w}^T(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)\right)^2\\ &= \mathbf{w}^T\left((\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T\right)\mathbf{w}\\ &= \mathbf{w}^T\mathbf{B}\mathbf{w} \end{aligned} \tag{20.2}$$

where $\mathbf{B} = (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T$ is a $d \times d$ rank-one matrix called the *between-class scatter matrix*.

As for the projected scatter for class c_1, we can compute it as follows:

$$\begin{aligned} s_1^2 &= \sum_{\mathbf{x}_i \in \mathbf{D}_1} (a_i - m_1)^2\\ &= \sum_{\mathbf{x}_i \in \mathbf{D}_1} (\mathbf{w}^T\mathbf{x}_i - \mathbf{w}^T\boldsymbol{\mu}_1)^2\\ &= \sum_{\mathbf{x}_i \in \mathbf{D}_1} \left(\mathbf{w}^T(\mathbf{x}_i - \boldsymbol{\mu}_1)\right)^2\\ &= \mathbf{w}^T\left(\sum_{\mathbf{x}_i \in \mathbf{D}_1} (\mathbf{x}_i - \boldsymbol{\mu}_1)(\mathbf{x}_i - \boldsymbol{\mu}_1)^T\right)\mathbf{w}\\ &= \mathbf{w}^T\mathbf{S}_1\mathbf{w} \end{aligned} \tag{20.3}$$

where $\mathbf{S}_1$ is the *scatter matrix* for $\mathbf{D}_1$. Likewise, we can obtain

$$s_2^2 = \mathbf{w}^T\mathbf{S}_2\mathbf{w} \tag{20.4}$$

Notice again that the scatter matrix is essentially the same as the covariance matrix, but instead of recording the average deviation from the mean, it records the total deviation, that is,

$$\mathbf{S}_i = n_i \boldsymbol{\Sigma}_i \tag{20.5}$$

Combining Eqs. (20.3) and (20.4), the denominator in Eq. (20.1) can be rewritten as

$$s_1^2 + s_2^2 = \mathbf{w}^T\mathbf{S}_1\mathbf{w} + \mathbf{w}^T\mathbf{S}_2\mathbf{w} = \mathbf{w}^T(\mathbf{S}_1 + \mathbf{S}_2)\mathbf{w} = \mathbf{w}^T\mathbf{S}\mathbf{w} \tag{20.6}$$

where $\mathbf{S} = \mathbf{S}_1 + \mathbf{S}_2$ denotes the *within-class scatter matrix* for the pooled data. Because both $\mathbf{S}_1$ and $\mathbf{S}_2$ are $d \times d$ symmetric positive semidefinite matrices, $\mathbf{S}$ has the same properties.

Using Eqs. (20.2) and (20.6), we write the LDA objective function [Eq. (20.1)] as follows:

$$\max_{\mathbf{w}} \; J(\mathbf{w}) = \frac{\mathbf{w}^T\mathbf{B}\mathbf{w}}{\mathbf{w}^T\mathbf{S}\mathbf{w}} \tag{20.7}$$

To solve for the best direction $\mathbf{w}$, we differentiate the objective function with respect to $\mathbf{w}$, and set the result to zero. We do not explicitly have to deal with the constraint that $\mathbf{w}^T\mathbf{w} = 1$ because in Eq. (20.7) the terms related to the magnitude of $\mathbf{w}$ cancel out in the numerator and the denominator.

Recall that if $f(x)$ and $g(x)$ are two functions then we have

$$\frac{d}{dx}\left(\frac{f(x)}{g(x)}\right) = \frac{f'(x)g(x) - g'(x)f(x)}{g(x)^2}$$

where $f'(x)$ denotes the derivative of $f(x)$. Taking the derivative of Eq. (20.7) with respect to the vector $\mathbf{w}$, and setting the result to the zero vector, gives us

$$\frac{d}{d\mathbf{w}}J(\mathbf{w}) = \frac{2\mathbf{Bw}(\mathbf{w}^T\mathbf{Sw}) - 2\mathbf{Sw}(\mathbf{w}^T\mathbf{Bw})}{(\mathbf{w}^T\mathbf{Sw})^2} = \mathbf{0}$$

which yields

$$\begin{aligned}
\mathbf{B\,w}(\mathbf{w}^T\mathbf{Sw}) &= \mathbf{S\,w}(\mathbf{w}^T\mathbf{Bw})\\
\mathbf{B\,w} &= \mathbf{S\,w}\left(\frac{\mathbf{w}^T\mathbf{Bw}}{\mathbf{w}^T\mathbf{Sw}}\right)\\
\mathbf{B\,w} &= J(\mathbf{w})\mathbf{Sw}\\
\mathbf{Bw} &= \lambda\mathbf{Sw} \qquad (20.8)
\end{aligned}$$

where $\lambda = J(\mathbf{w})$. Eq. (20.8) represents a *generalized eigenvalue problem* where λ is a generalized eigenvalue of $\mathbf{B}$ and $\mathbf{S}$; the eigenvalue λ satisfies the equation $\det(\mathbf{B} - \lambda\mathbf{S}) = 0$. Because the goal is to maximize the objective [Eq. (20.7)], $J(\mathbf{w}) = \lambda$ should be chosen to be the largest generalized eigenvalue, and $\mathbf{w}$ to be the corresponding eigenvector. If $\mathbf{S}$ is *nonsingular*, that is, if $\mathbf{S}^{-1}$ exists, then Eq. (20.8) leads to the regular eigenvalue–eigenvector equation, as

$$\begin{aligned}
\mathbf{Bw} &= \lambda\mathbf{Sw}\\
\mathbf{S}^{-1}\mathbf{Bw} &= \lambda\mathbf{S}^{-1}\mathbf{Sw}\\
(\mathbf{S}^{-1}\mathbf{B})\mathbf{w} &= \lambda\mathbf{w} \qquad (20.9)
\end{aligned}$$

Thus, if $\mathbf{S}^{-1}$ exists, then $\lambda = J(\mathbf{w})$ is an eigenvalue, and $\mathbf{w}$ is an eigenvector of the matrix $\mathbf{S}^{-1}\mathbf{B}$. To maximize $J(\mathbf{w})$ we look for the largest eigenvalue λ, and the corresponding dominant eigenvector $\mathbf{w}$ specifies the best linear discriminant vector.

Algorithm 20.1 shows the pseudo-code for linear discriminant analysis. Here, we assume that there are two classes, and that $\mathbf{S}$ is nonsingular (i.e., $\mathbf{S}^{-1}$ exists). The vector $\mathbf{1}_{n_i}$ is the vector of all ones, with the appropriate dimension for each class, i.e., $\mathbf{1}_{n_i} \in \mathbb{R}^{n_i}$ for class $i = 1,2$. After dividing $\mathbf{D}$ into the two groups $\mathbf{D}_1$ and $\mathbf{D}_2$, LDA proceeds to compute the between-class and within-class scatter matrices, $\mathbf{B}$ and $\mathbf{S}$. The optimal LD vector is obtained as the dominant eigenvector of $\mathbf{S}^{-1}\mathbf{B}$. In terms of computational complexity, computing $\mathbf{S}$ takes $O(nd^2)$ time, and computing the dominant eigenvalue-eigenvector pair takes $O(d^3)$ time in the worst case. Thus, the total time is $O(d^3 + nd^2)$.

ALGORITHM 20.1. Linear Discriminant Analysis

```
LINEARDISCRIMINANT ($\mathbf{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$):
1 $\mathbf{D}_i \leftarrow \{\mathbf{x}_j \mid y_j = c_i, j = 1, \ldots, n\}, i = 1, 2$ // class-specific subsets
2 $\boldsymbol{\mu}_i \leftarrow \text{mean}(\mathbf{D}_i), i = 1, 2$ // class means
3 $\mathbf{B} \leftarrow (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T$ // between-class scatter matrix
4 $\mathbf{Z}_i \leftarrow \mathbf{D}_i - \mathbf{1}_{n_i}\boldsymbol{\mu}_i^T, i = 1, 2$ // center class matrices
5 $\mathbf{S}_i \leftarrow \mathbf{Z}_i^T\mathbf{Z}_i, i = 1, 2$ // class scatter matrices
6 $\mathbf{S} \leftarrow \mathbf{S}_1 + \mathbf{S}_2$ // within-class scatter matrix
7 $\lambda_1, \mathbf{w} \leftarrow \text{eigen}(\mathbf{S}^{-1}\mathbf{B})$ // compute dominant eigenvector
```

Example 20.2 (Linear Discriminant Analysis). Consider the 2-dimensional Iris data (with attributes `sepal length` and `sepal width`) shown in Example 20.1. Class c_1, corresponding to `iris-setosa`, has $n_1 = 50$ points, whereas the other class c_2 has $n_2 = 100$ points. The means for the two classes c_1 and c_2, and their difference is given as

$$\boldsymbol{\mu}_1 = \begin{pmatrix} 5.01 \\ 3.42 \end{pmatrix}^T \qquad \boldsymbol{\mu}_2 = \begin{pmatrix} 6.26 \\ 2.87 \end{pmatrix}^T \qquad \boldsymbol{\mu}_1 - \boldsymbol{\mu}_2 = \begin{pmatrix} -1.256 \\ 0.546 \end{pmatrix}^T$$

The between-class scatter matrix is

$$\mathbf{B} = (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T = \begin{pmatrix} -1.256 \\ 0.546 \end{pmatrix} \begin{pmatrix} -1.256 & 0.546 \end{pmatrix} = \begin{pmatrix} 1.587 & -0.693 \\ -0.693 & 0.303 \end{pmatrix}$$

and the within-class scatter matrix is

$$\mathbf{S}_1 = \begin{pmatrix} 6.09 & 4.91 \\ 4.91 & 7.11 \end{pmatrix} \qquad \mathbf{S}_2 = \begin{pmatrix} 43.5 & 12.09 \\ 12.09 & 10.96 \end{pmatrix} \qquad \mathbf{S} = \mathbf{S}_1 + \mathbf{S}_2 = \begin{pmatrix} 49.58 & 17.01 \\ 17.01 & 18.08 \end{pmatrix}$$

$\mathbf{S}$ is nonsingular, with its inverse given as

$$\mathbf{S}^{-1} = \begin{pmatrix} 0.0298 & -0.028 \\ -0.028 & 0.0817 \end{pmatrix}$$

Therefore, we have

$$\mathbf{S}^{-1}\mathbf{B} = \begin{pmatrix} 0.0298 & -0.028 \\ -0.028 & 0.0817 \end{pmatrix} \begin{pmatrix} 1.587 & -0.693 \\ -0.693 & 0.303 \end{pmatrix} = \begin{pmatrix} 0.066 & -0.029 \\ -0.100 & 0.044 \end{pmatrix}$$

The direction of most separation between c_1 and c_2 is the dominant eigenvector corresponding to the largest eigenvalue of the matrix $\mathbf{S}^{-1}\mathbf{B}$. The solution is

$$J(\mathbf{w}) = \lambda_1 = 0.11$$

$$\mathbf{w} = \begin{pmatrix} 0.551 \\ -0.834 \end{pmatrix}$$

Figure 20.2 plots the optimal linear discriminant direction $\mathbf{w}$, translated to the mean of the data. The projected means for the two classes are shown in black. We can clearly observe that along $\mathbf{w}$ the circles appear together as a group, and are quite well separated from the triangles. Except for one outlying circle corresponding to the point $(4.5, 2.3)^T$, all points in c_1 are perfectly separated from points in c_2.

For the two class scenario, if $\mathbf{S}$ is nonsingular, we can directly solve for $\mathbf{w}$ without computing the eigenvalues and eigenvectors. Note that $\mathbf{B} = (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T$ is a $d \times d$ rank-one matrix, and thus $\mathbf{Bw}$ must point in the same direction as $(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)$ because

$$\begin{aligned}\mathbf{Bw} &= \Big((\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T\Big)\mathbf{w}\\ &= (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)\Big((\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T\mathbf{w}\Big)\\ &= b(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)\end{aligned}$$

where $b = (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T\mathbf{w}$ is just a scalar multiplier.

We can then rewrite Eq. (20.9) as

$$\begin{aligned}\mathbf{Bw} &= \lambda\mathbf{Sw}\\ b(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) &= \lambda\mathbf{Sw}\\ \mathbf{w} &= \frac{b}{\lambda}\mathbf{S}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)\end{aligned}$$

Because $\frac{b}{\lambda}$ is just a scalar, we can solve for the best linear discriminant as

$$\mathbf{w} = \mathbf{S}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \tag{20.10}$$

Once the direction $\mathbf{w}$ has been found we can normalize it to be a unit vector. Thus, instead of solving for the eigenvalue/eigenvector, in the two class case, we immediately obtain the direction $\mathbf{w}$ using Eq. (20.10). Intuitively, the direction that maximizes the separation between the classes can be viewed as a linear transformation (by $\mathbf{S}^{-1}$) of the vector joining the two class means $(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)$.

Example 20.3. Continuing Example 20.2, we can directly compute $\mathbf{w}$ as follows:

$$\begin{aligned}\mathbf{w} &= \mathbf{S}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)\\ &= \begin{pmatrix} 0.066 & -0.029\\ -0.100 & 0.044\end{pmatrix}\begin{pmatrix}-1.246\\ 0.546\end{pmatrix} = \begin{pmatrix}-0.0527\\ 0.0798\end{pmatrix}\end{aligned}$$

After normalizing, we have

$$\mathbf{w} = \frac{\mathbf{w}}{\|\mathbf{w}\|} = \frac{1}{0.0956}\begin{pmatrix}-0.0527\\ 0.0798\end{pmatrix} = \begin{pmatrix}-0.551\\ 0.834\end{pmatrix}$$

Note that even though the sign is reversed for $\mathbf{w}$, compared to that in Example 20.2, they represent the same direction; only the scalar multiplier is different.

20.2 KERNEL DISCRIMINANT ANALYSIS

Kernel discriminant analysis, like linear discriminant analysis, tries to find a direction that maximizes the separation between the classes. However, it does so in *feature space* via the use of kernel functions.

Given a dataset $\mathbf{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, where $\mathbf{x}_i$ is a point in input space and $y_i \in \{c_1, c_2\}$ is the class label, let $\mathbf{D}_i = \{\mathbf{x}_j | y_j = c_i\}$ denote the data subset restricted to class c_i, and let $n_i = |\mathbf{D}_i|$. Further, let $\phi(\mathbf{x}_i)$ denote the corresponding point in feature space, and let K be a kernel function.

The goal of kernel LDA is to find the direction vector $\mathbf{w}$ in feature space that maximizes

$$\max_{\mathbf{w}} \; J(\mathbf{w}) = \frac{(m_1 - m_2)^2}{s_1^2 + s_2^2} \tag{20.11}$$

where m_1 and m_2 are the projected means, and s_1^2 and s_2^2 are projected scatter values in feature space. We first show that $\mathbf{w}$ can be expressed as a linear combination of the points in feature space, and then we transform the LDA objective in terms of the kernel matrix.

Optimal LD: Linear Combination of Feature Points

The mean for class c_i in feature space is given as

$$\boldsymbol{\mu}_i^\phi = \frac{1}{n_i} \sum_{\mathbf{x}_j \in \mathbf{D}_i} \phi(\mathbf{x}_j) \tag{20.12}$$

and the covariance matrix for class c_i in feature space is

$$\boldsymbol{\Sigma}_i^\phi = \frac{1}{n_i} \sum_{\mathbf{x}_j \in \mathbf{D}_i} \left(\phi(\mathbf{x}_j) - \boldsymbol{\mu}_i^\phi\right)\left(\phi(\mathbf{x}_j) - \boldsymbol{\mu}_i^\phi\right)^T$$

Using a derivation similar to Eq. (20.2) we obtain an expression for the between-class scatter matrix in feature space

$$\mathbf{B}_\phi = \left(\boldsymbol{\mu}_1^\phi - \boldsymbol{\mu}_2^\phi\right)\left(\boldsymbol{\mu}_1^\phi - \boldsymbol{\mu}_2^\phi\right)^T = \mathbf{d}_\phi \mathbf{d}_\phi^T \tag{20.13}$$

where $\mathbf{d}_\phi = \boldsymbol{\mu}_1^\phi - \boldsymbol{\mu}_2^\phi$ is the difference between the two class mean vectors. Likewise, using Eqs. (20.5) and (20.6) the within-class scatter matrix in feature space is given as

$$\mathbf{S}_\phi = n_1 \boldsymbol{\Sigma}_1^\phi + n_2 \boldsymbol{\Sigma}_2^\phi$$

$\mathbf{S}_\phi$ is a $d \times d$ symmetric, positive semidefinite matrix, where d is the dimensionality of the feature space. From Eq. (20.9), we conclude that the best linear discriminant vector $\mathbf{w}$ in feature space is the dominant eigenvector, which satisfies the expression

$$\left(\mathbf{S}_\phi^{-1} \mathbf{B}_\phi\right) \mathbf{w} = \lambda \mathbf{w} \tag{20.14}$$

where we assume that $\mathbf{S}_\phi$ is non-singular. Let δ_i denote the ith eigenvalue and $\mathbf{u}_i$ the ith eigenvector of $\mathbf{S}_\phi$, for $i = 1, \ldots, d$. The eigen-decomposition of $\mathbf{S}_\phi$ yields $\mathbf{S}_\phi = \mathbf{U}\boldsymbol{\Delta}\mathbf{U}^T$, with the inverse of $\mathbf{S}_\phi$ given as $\mathbf{S}_\phi^{-1} = \mathbf{U}\boldsymbol{\Delta}^{-1}\mathbf{U}^T$. Here $\mathbf{U}$ is the matrix whose columns are the eigenvectors of $\mathbf{S}_\phi$ and $\boldsymbol{\Delta}$ is the diagonal matrix of eigenvalues of $\mathbf{S}_\phi$. The inverse $\mathbf{S}_\phi^{-1}$ can thus be expressed as the spectral sum

$$\mathbf{S}_\phi^{-1} = \sum_{r=1}^{d} \frac{1}{\delta_r}\mathbf{u}_r\mathbf{u}_r^T \tag{20.15}$$

Plugging Eqs. (20.13) and (20.15) into Eq. (20.14), we obtain

$$\lambda\mathbf{w} = \left(\sum_{r=1}^{d}\frac{1}{\delta_r}\mathbf{u}_r\mathbf{u}_r^T\right)\mathbf{d}_\phi\mathbf{d}_\phi^T\mathbf{w} = \sum_{r=1}^{d}\frac{1}{\delta_r}\left(\mathbf{u}_r(\mathbf{u}_r^T\mathbf{d}_\phi)(\mathbf{d}_\phi^T\mathbf{w})\right) = \sum_{r=1}^{d} b_r\mathbf{u}_r$$

where $b_r = \frac{1}{\delta_r}(\mathbf{u}_r^T\mathbf{d}_\phi)(\mathbf{d}_\phi^T\mathbf{w})$ is a scalar value. Using a derivation similar to that in Eq. (7.32), the rth eigenvector of $\mathbf{S}_\phi$ can be expressed as a linear combination of the feature points, say $\mathbf{u}_r = \sum_{j=1}^n c_{rj}\phi(\mathbf{x}_j)$, where c_{rj} is a scalar coefficient. Thus, we can rewrite $\mathbf{w}$ as

$$\begin{aligned}
\mathbf{w} &= \frac{1}{\lambda}\sum_{r=1}^{d} b_r\left(\sum_{j=1}^{n} c_{rj}\phi(\mathbf{x}_j)\right)\\
&= \sum_{j=1}^{n}\phi(\mathbf{x}_j)\left(\sum_{r=1}^{d}\frac{b_r c_{rj}}{\lambda}\right)\\
&= \sum_{j=1}^{n} a_j\phi(\mathbf{x}_j)
\end{aligned}$$

where $a_j = \sum_{r=1}^d b_r c_{rj}/\lambda$ is a scalar value for the feature point $\phi(\mathbf{x}_j)$. Therefore, the direction vector $\mathbf{w}$ can be expressed as a linear combination of the points in feature space.

LDA Objective via Kernel Matrix

We now rewrite the kernel LDA objective [Eq. (20.11)] in terms of the kernel matrix. Projecting the mean for class c_i given in Eq. (20.12) onto the LD direction $\mathbf{w}$, we have

$$\begin{aligned}
m_i = \mathbf{w}^T\boldsymbol{\mu}_i^\phi &= \left(\sum_{j=1}^{n} a_j\,\phi(\mathbf{x}_j)\right)^T\left(\frac{1}{n_i}\sum_{\mathbf{x}_k\in\mathbf{D}_i}\phi(\mathbf{x}_k)\right)\\
&= \frac{1}{n_i}\sum_{j=1}^{n}\sum_{\mathbf{x}_k\in\mathbf{D}_i} a_j\,\phi(\mathbf{x}_j)^T\phi(\mathbf{x}_k)\\
&= \frac{1}{n_i}\sum_{j=1}^{n}\sum_{\mathbf{x}_k\in\mathbf{D}_i} a_j K(\mathbf{x}_j,\mathbf{x}_k)\\
&= \mathbf{a}^T\mathbf{m}_i
\end{aligned} \tag{20.16}$$

where $\mathbf{a} = (a_1, a_2, \ldots, a_n)^T$ is the weight vector, and

$$\mathbf{m}_i = \frac{1}{n_i} \begin{pmatrix} \sum_{\mathbf{x}_k \in \mathbf{D}_i} K(\mathbf{x}_1, \mathbf{x}_k) \\ \sum_{\mathbf{x}_k \in \mathbf{D}_i} K(\mathbf{x}_2, \mathbf{x}_k) \\ \vdots \\ \sum_{\mathbf{x}_k \in \mathbf{D}_i} K(\mathbf{x}_n, \mathbf{x}_k) \end{pmatrix} = \frac{1}{n_i} \mathbf{K}^{c_i} \mathbf{1}_{n_i} \tag{20.17}$$

where $\mathbf{K}^{c_i}$ is the $n \times n_i$ subset of the kernel matrix, restricted to columns belonging to points only in $\mathbf{D}_i$, and $\mathbf{1}_{n_i}$ is the n_i-dimensional vector all of whose entries are one. The n-length vector $\mathbf{m}_i$ thus stores for each point in $\mathbf{D}$ its average kernel value with respect to the points in $\mathbf{D}_i$.

We can rewrite the separation between the projected means in feature space as follows:

$$\begin{aligned}(m_1 - m_2)^2 &= \left(\mathbf{w}^T \boldsymbol{\mu}_1^\phi - \mathbf{w}^T \boldsymbol{\mu}_2^\phi\right)^2 \\ &= \left(\mathbf{a}^T \mathbf{m}_1 - \mathbf{a}^T \mathbf{m}_2\right)^2 \\ &= \mathbf{a}^T (\mathbf{m}_1 - \mathbf{m}_2)(\mathbf{m}_1 - \mathbf{m}_2)^T \mathbf{a} \\ &= \mathbf{a}^T \mathbf{M} \mathbf{a} \end{aligned} \tag{20.18}$$

where $\mathbf{M} = (\mathbf{m}_1 - \mathbf{m}_2)(\mathbf{m}_1 - \mathbf{m}_2)^T$ is the between-class scatter matrix.

We can also compute the projected scatter for each class, s_1^2 and s_2^2, purely in terms of the kernel function, as

$$\begin{aligned} s_1^2 &= \sum_{\mathbf{x}_i \in \mathbf{D}_1} \left\| \mathbf{w}^T \phi(\mathbf{x}_i) - \mathbf{w}^T \boldsymbol{\mu}_1^\phi \right\|^2 \\ &= \sum_{\mathbf{x}_i \in \mathbf{D}_1} \left\| \mathbf{w}^T \phi(\mathbf{x}_i) \right\|^2 - 2 \sum_{\mathbf{x}_i \in \mathbf{D}_1} \mathbf{w}^T \phi(\mathbf{x}_i) \cdot \mathbf{w}^T \boldsymbol{\mu}_1^\phi + \sum_{\mathbf{x}_i \in \mathbf{D}_1} \left\| \mathbf{w}^T \boldsymbol{\mu}_1^\phi \right\|^2 \\ &= \left(\sum_{\mathbf{x}_i \in \mathbf{D}_1} \left\| \sum_{j=1}^n a_j \phi(\mathbf{x}_j)^T \phi(\mathbf{x}_i) \right\|^2 \right) - 2 \cdot n_1 \cdot \left\| \mathbf{w}^T \boldsymbol{\mu}_1^\phi \right\|^2 + n_1 \cdot \left\| \mathbf{w}^T \boldsymbol{\mu}_1^\phi \right\|^2 \\ &= \left(\sum_{\mathbf{x}_i \in \mathbf{D}_1} \mathbf{a}^T \mathbf{K}_i \mathbf{K}_i^T \mathbf{a} \right) - n_1 \cdot \mathbf{a}^T \mathbf{m}_1 \mathbf{m}_1^T \mathbf{a} \qquad \text{by using Eq. (20.16)} \\ &= \mathbf{a}^T \left(\left(\sum_{\mathbf{x}_i \in \mathbf{D}_1} \mathbf{K}_i \mathbf{K}_i^T \right) - n_1 \mathbf{m}_1 \mathbf{m}_1^T \right) \mathbf{a} \\ &= \mathbf{a}^T \mathbf{N}_1 \mathbf{a} \end{aligned}$$

where $\mathbf{K}_i$ is the ith column of the kernel matrix, and $\mathbf{N}_1$ is the class scatter matrix for c_1. Let $K(\mathbf{x}_i, \mathbf{x}_j) = K_{ij}$. We can express $\mathbf{N}_1$ more compactly in matrix notation as follows:

$$\begin{aligned} \mathbf{N}_1 &= \left(\sum_{\mathbf{x}_i \in \mathbf{D}_1} \mathbf{K}_i \mathbf{K}_i^T \right) - n_1 \mathbf{m}_1 \mathbf{m}_1^T \\ &= (\mathbf{K}^{c_1}) \left(\mathbf{I}_{n_1} - \frac{1}{n_1} \mathbf{1}_{n_1 \times n_1} \right) (\mathbf{K}^{c_1})^T \end{aligned} \tag{20.19}$$

where $\mathbf{I}_{n_1}$ is the $n_1 \times n_1$ identity matrix and $\mathbf{1}_{n_1 \times n_1}$ is the $n_1 \times n_1$ matrix, all of whose entries are 1's.

In a similar manner we get $s_2^2 = \mathbf{a}^T\mathbf{N}_2\mathbf{a}$, where

$$\mathbf{N}_2 = (\mathbf{K}^{c_2})\left(\mathbf{I}_{n_2} - \frac{1}{n_2}\mathbf{1}_{n_2 \times n_2}\right)(\mathbf{K}^{c_2})^T$$

where $\mathbf{I}_{n_2}$ is the $n_2 \times n_2$ identity matrix and $\mathbf{1}_{n_2 \times n_2}$ is the $n_2 \times n_2$ matrix, all of whose entries are 1's.

The sum of projected scatter values is then given as

$$s_1^2 + s_2^2 = \mathbf{a}^T(\mathbf{N}_1 + \mathbf{N}_2)\mathbf{a} = \mathbf{a}^T\mathbf{N}\mathbf{a} \tag{20.20}$$

where $\mathbf{N}$ is the $n \times n$ within-class scatter matrix.

Substituting Eqs. (20.18) and (20.20) in Eq. (20.11), we obtain the kernel LDA maximization condition

$$\max_{\mathbf{w}} J(\mathbf{w}) = \max_{\mathbf{a}} J(\mathbf{a}) = \frac{\mathbf{a}^T\mathbf{M}\mathbf{a}}{\mathbf{a}^T\mathbf{N}\mathbf{a}}$$

Notice how all the terms in the expression above involve only kernel functions. The weight vector $\mathbf{a}$ is the eigenvector corresponding to the largest eigenvalue of the generalized eigenvalue problem:

$$\mathbf{M}\mathbf{a} = \lambda_1\mathbf{N}\mathbf{a} \tag{20.21}$$

If $\mathbf{N}$ is nonsingular, $\mathbf{a}$ is the dominant eigenvector corresponding to the largest eigenvalue for the system

$$(\mathbf{N}^{-1}\mathbf{M})\mathbf{a} = \lambda_1\mathbf{a}$$

As in the case of linear discriminant analysis [Eq. (20.10)], when there are only two classes we do not have to solve for the eigenvector because $\mathbf{a}$ can be obtained directly:

$$\mathbf{a} = \mathbf{N}^{-1}(\mathbf{m}_1 - \mathbf{m}_2)$$

Once $\mathbf{a}$ has been obtained, we can normalize $\mathbf{w}$ to be a unit vector by ensuring that

$$\mathbf{w}^T\mathbf{w} = 1, \text{ which implies that}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j) = 1, \text{ or}$$

$$\mathbf{a}^T\mathbf{K}\mathbf{a} = 1$$

Put differently, we can ensure that $\mathbf{w}$ is a unit vector if we scale $\mathbf{a}$ by $\frac{1}{\sqrt{\mathbf{a}^T\mathbf{K}\mathbf{a}}}$.

Finally, we can project any point $\mathbf{x}$ onto the discriminant direction, as follows:

$$\mathbf{w}^T\phi(\mathbf{x}) = \sum_{j=1}^{n} a_j\phi(\mathbf{x}_j)^T\phi(\mathbf{x}) = \sum_{j=1}^{n} a_j K(\mathbf{x}_j, \mathbf{x}) \tag{20.22}$$

Algorithm 20.2 shows the pseudo-code for kernel discriminant analysis. The method proceeds by computing the $n \times n$ kernel matrix $\mathbf{K}$, and the $n \times n_i$ class

ALGORITHM 20.2. Kernel Discriminant Analysis

```
KernelDiscriminant ($\mathbf{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n, K$):
1 $\mathbf{K} \leftarrow \{K(\mathbf{x}_i, \mathbf{x}_j)\}_{i,j=1,\dots,n}$ // compute $n \times n$ kernel matrix
2 $\mathbf{K}^{c_i} \leftarrow \{\mathbf{K}(j,k) \mid y_k = c_i, 1 \le j,k \le n\}, i = 1,2$ // class kernel matrix
3 $\mathbf{m}_i \leftarrow \frac{1}{n_i}\mathbf{K}^{c_i}\mathbf{1}_{n_i}, i = 1,2$ // class means
4 $\mathbf{M} \leftarrow (\mathbf{m}_1 - \mathbf{m}_2)(\mathbf{m}_1 - \mathbf{m}_2)^T$ // between-class scatter matrix
5 $\mathbf{N}_i \leftarrow \mathbf{K}^{c_i}(\mathbf{I}_{n_i} - \frac{1}{n_i}\mathbf{1}_{n_i \times n_i})(\mathbf{K}^{c_i})^T, i = 1,2$ // class scatter matrices
6 $\mathbf{N} \leftarrow \mathbf{N}_1 + \mathbf{N}_2$ // within-class scatter matrix
7 $\lambda_1, \mathbf{a} \leftarrow \text{eigen}(\mathbf{N}^{-1}\mathbf{M})$ // compute weight vector
8 $\mathbf{a} \leftarrow \frac{\mathbf{a}}{\sqrt{\mathbf{a}^T\mathbf{K}\mathbf{a}}}$ // normalize $\mathbf{w}$ to be unit vector
```

specific kernel matrices $\mathbf{K}^{c_i}$ for each class c_i. After computing the between-class and within-class scatter matrices $\mathbf{M}$ and $\mathbf{N}$, the weight vector $\mathbf{a}$ is obtained as the dominant eigenvector of $\mathbf{N}^{-1}\mathbf{M}$. The last step scales $\mathbf{a}$ so that $\mathbf{w}$ will be normalized to be unit length. The complexity of kernel discriminant analysis is $O(n^3)$, with the dominant steps being the computation of $\mathbf{N}$ and solving for the dominant eigenvector of $\mathbf{N}^{-1}\mathbf{M}$, both of which take $O(n^3)$ time.

Example 20.4 (Kernel Discriminant Analysis). Consider the 2-dimensional Iris dataset comprising the `sepal length` and `sepal width` attributes. Figure 20.3a shows the points projected onto the first two principal components. The points have been divided into two classes: c_1 (circles) corresponds to *iris-virginica* and c_2 (triangles) corresponds to the other two Iris types. Here $n_1 = 50$ and $n_2 = 100$, with a total of $n = 150$ points.

Because c_1 is surrounded by points in c_2 a good linear discriminant will not be found. Instead, we apply kernel discriminant analysis using the homogeneous quadratic kernel

$$K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T\mathbf{x}_j)^2$$

Solving for $\mathbf{a}$ via Eq. (20.21) yields

$$\lambda_1 = 0.0511$$

However, we do not show $\mathbf{a}$ because it lies in $\mathbb{R}^{150}$. Figure 20.3a shows the contours of constant projections onto the best kernel discriminant. The contours are obtained by solving Eq. (20.22), that is, by solving $\mathbf{w}^T\phi(\mathbf{x}) = \sum_{j=1}^n a_j K(\mathbf{x}_j, \mathbf{x}) = c$ for different values of the scalars c. The contours are hyperbolic, and thus form pairs starting from the center. For instance, the first curve on the left and right of the origin $(0,0)^T$ forms the same contour, that is, points along both the curves have the same value when projected onto $\mathbf{w}$. We can see that contours or pairs of curves starting with the fourth curve (on the left and right) from the center all relate to class c_2, whereas the first three contours deal mainly with class c_1, indicating good discrimination with the homogeneous quadratic kernel.

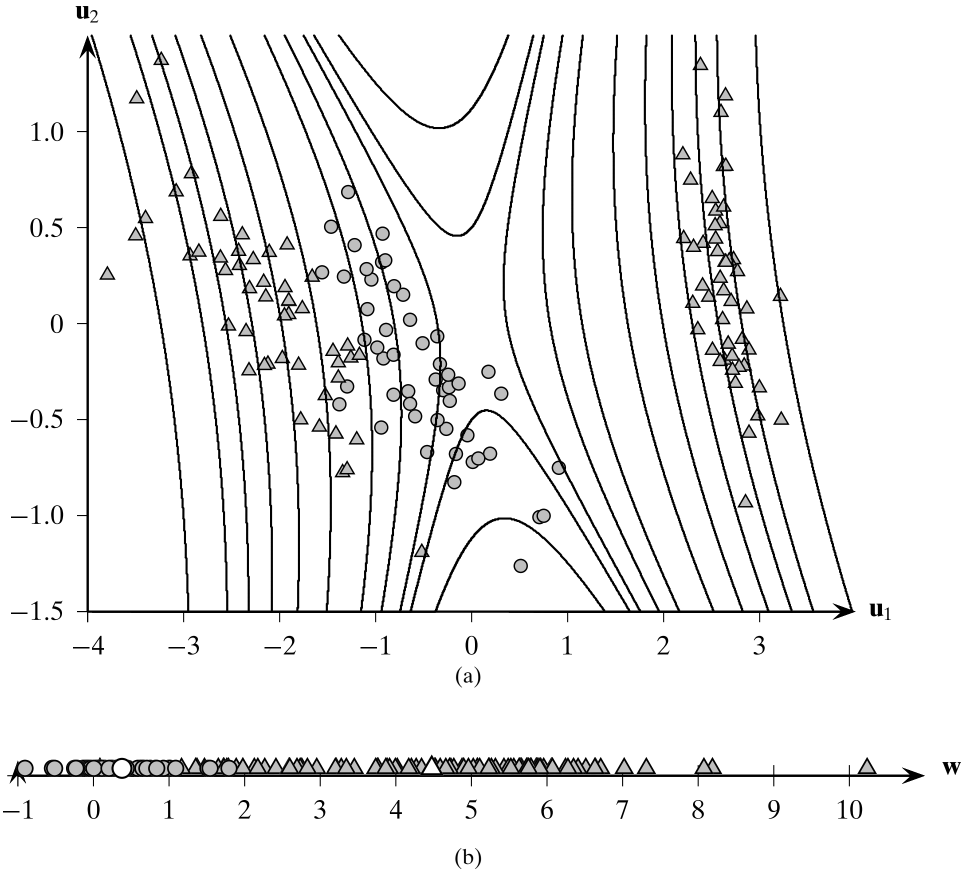


Figure 20.3. Kernel discriminant analysis: quadratic homogeneous kernel.

A better picture emerges when we plot the coordinates of all the points $\mathbf{x}_i \in \mathbf{D}$ when projected onto $\mathbf{w}$, as shown in Figure 20.3b. We can observe that $\mathbf{w}$ is able to separate the two classes reasonably well; all the circles (c_1) are concentrated on the left, whereas the triangles (c_2) are spread out on the right. The projected means are shown in white. The scatters and means for both classes after projection are as follows:

$$m_1 = 0.338 \qquad\qquad m_2 = 4.476$$

$$s_1^2 = 13.862 \qquad\qquad s_2^2 = 320.934$$

The value of $J(\mathbf{w})$ is given as

$$J(\mathbf{w}) = \frac{(m_1 - m_2)^2}{s_1^2 + s_2^2} = \frac{(0.338 - 4.476)^2}{13.862 + 320.934} = \frac{17.123}{334.796} = 0.0511$$

which, as expected, matches $\lambda_1 = 0.0511$ from above.

In general, it is not desirable or possible to obtain an explicit discriminant vector $\mathbf{w}$, since it lies in feature space. However, because each point $\mathbf{x} = (x_1, x_2)^T \in \mathbb{R}^2$ in

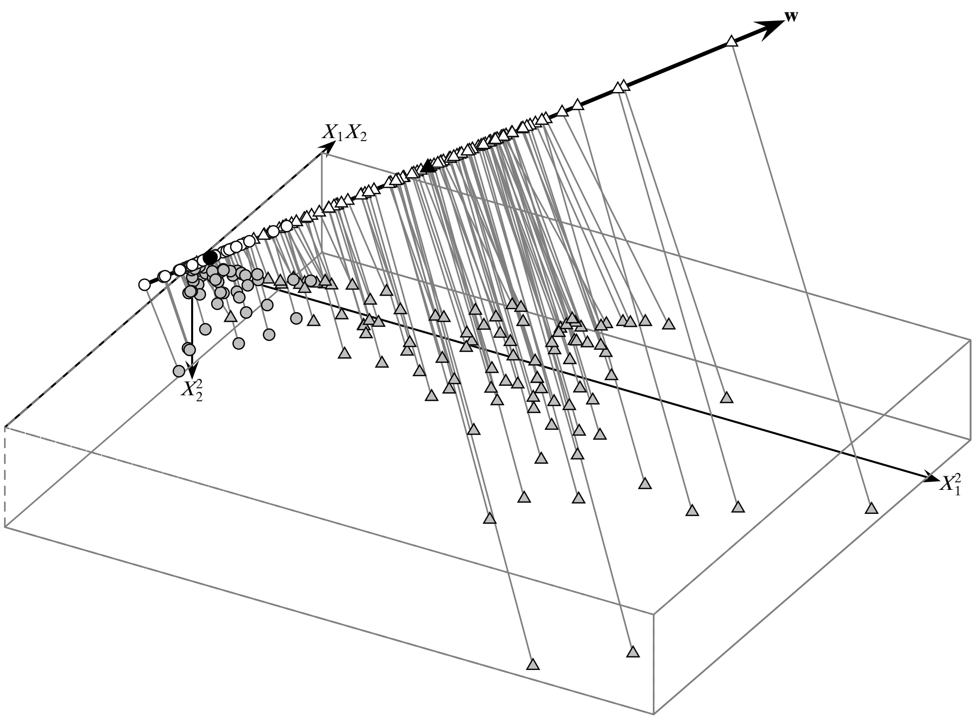


Figure 20.4. Homogeneous quadratic kernel feature space.

input space is mapped to the point $\phi(\mathbf{x}) = (\sqrt{2}x_1x_2, x_1^2, x_2^2)^T \in \mathbb{R}^3$ in feature space via the homogeneous quadratic kernel, for our example it is possible to visualize the feature space, as illustrated in Figure 20.4. The projection of each point $\phi(\mathbf{x}_i)$ onto the discriminant vector $\mathbf{w}$ is also shown, where

$$\mathbf{w} = 0.511x_1x_2 + 0.761x_1^2 - 0.4x_2^2$$

The projections onto $\mathbf{w}$ are identical to those shown in Figure 20.3b.

20.3 FURTHER READING

Linear discriminant analysis was introduced in Fisher (1936). Its extension to kernel discriminant analysis was proposed in Mika et al. (1999). The 2-class LDA approach can be generalized to $k > 2$ classes by finding the optimal $(k-1)$-dimensional subspace projection that best discriminates between the k classes; see Duda, Hart, and Stork (2012) for details.

Duda, R. O., Hart, P. E., and Stork, D. G. (2012). *Pattern Classification*. New York: Wiley-Interscience.

Fisher, R. A. (1936). "The use of multiple measurements in taxonomic problems." *Annals of Eugenics*, 7 (2): 179–188.

Mika, S., Ratsch, G., Weston, J., Scholkopf, B., and Mullers, K. (1999). "Fisher discriminant analysis with kernels." *In Proceedings of the IEEE Neural Networks for Signal Processing Workshop,* IEEE, pp. 41–48.

20.4 EXERCISES

Q1. Consider the data shown in Table 20.1. Answer the following questions:

(a) Compute $\boldsymbol{\mu}_{+1}$ and $\boldsymbol{\mu}_{-1}$, and $\mathbf{S}_B$, the between-class scatter matrix.

(b) Compute $\mathbf{S}_{+1}$ and $\mathbf{S}_{-1}$, and $\mathbf{S}_W$, the within-class scatter matrix.

(c) Find the best direction $\mathbf{w}$ that discriminates between the classes. Use the fact that the inverse of the matrix $\mathbf{A} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ is given as $\mathbf{A}^{-1} = \frac{1}{\det(\mathbf{A})}\begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$.

(d) Having found the direction $\mathbf{w}$, find the point on $\mathbf{w}$ that best separates the two classes.

Table 20.1. Dataset for Q1

i	$\mathbf{x}_i$	y_i
$\mathbf{x}_1$	(4,2.9)	1
$\mathbf{x}_2$	(3.5,4)	1
$\mathbf{x}_3$	(2.5,1)	−1
$\mathbf{x}_4$	(2,2.1)	−1

Q2. Given the labeled points (from two classes) shown in Figure 20.5, and given that the inverse of the within-class scatter matrix is

$$\begin{pmatrix} 0.056 & -0.029 \\ -0.029 & 0.052 \end{pmatrix}$$

Find the best linear discriminant line $\mathbf{w}$, and sketch it.

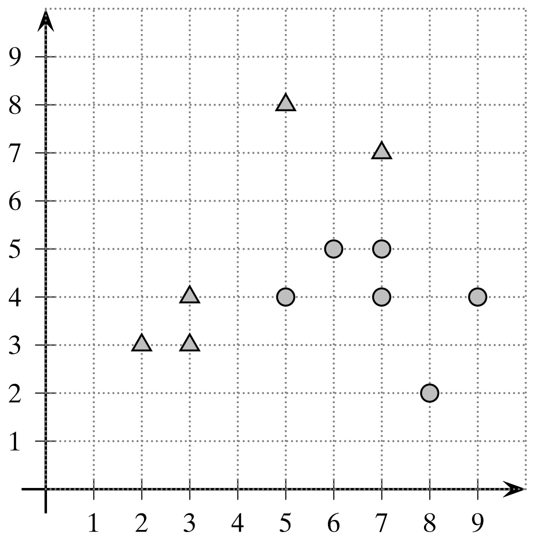


Figure 20.5. Dataset for Q2.

Q3. Maximize the objective in Eq. (20.7) by explicitly considering the constraint $\mathbf{w}^T\mathbf{w} = 1$, that is, by using a Lagrange multiplier for that constraint.

Q4. Prove the equality in Eq. (20.19). That is, show that

$$\mathbf{N}_1 = \left(\sum_{\mathbf{x}_i \in \mathbf{D}_1} \mathbf{K}_i \mathbf{K}_i^T \right) - n_1 \mathbf{m}_1 \mathbf{m}_1^T = (\mathbf{K}^{c_1}) \, (\mathbf{I}_{n_1} - \frac{1}{n_1} \mathbf{1}_{n_1 \times n_1}) \, (\mathbf{K}^{c_1})^T$$