

6. Discretization methods

6.1 The purpose of discretization

Often data are given in the form of continuous values. If their number is huge, model building for such data can be difficult. Moreover, many data mining algorithms operate only in discrete search or variable space. For instance, decision trees typically divide the values of a variable into two parts according to an appropriate threshold value. Many techniques apply computation of various criteria, for example, mutual information, or data mining algorithms that assume discrete values.

The goal of *discretization* is to reduce the number of values a continuous variable assumes by grouping them into a number, b , of intervals or bins.

Two key problems in association with discretization are how to select the number of intervals or bins and how to decide on their width. Discretization can be performed with or without taking class information, if available, into account. These are the *supervised* and *unsupervised* ways. If class labels were known in the training data, the discretization method ought to take advantage of it, especially if the subsequently used learning algorithm for model building is supervised.

In this situation, a discretization method should maximize the interdependence between the variable values and the class labels. In addition, discretization then minimizes information loss while transforming from continuous to discrete values.

Fig. 6.1 illustrates a trivial situation of one variable, employing the same width for all intervals. The top in Fig. 6.1 shows an alternative in which two intervals only were chosen without using class information. The bottom shows the grouping variable values into four intervals while taking into account the class information.

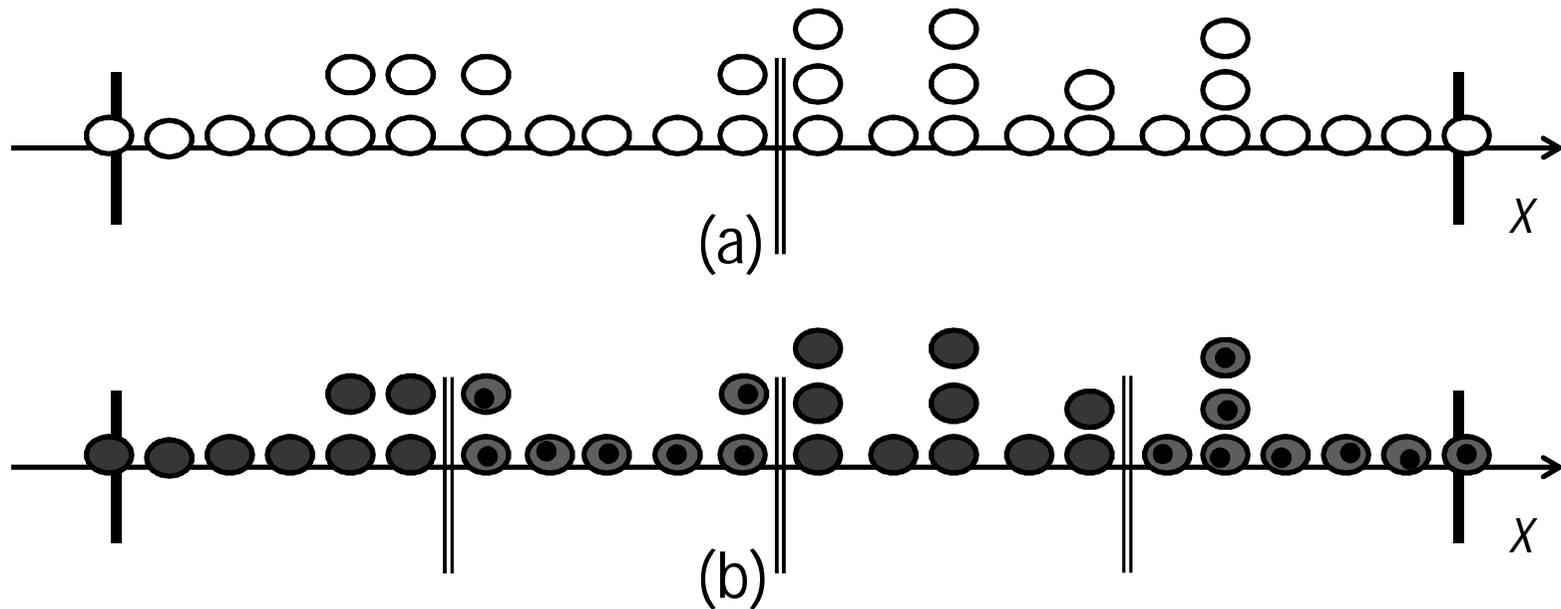


Fig. 6.1 Deciding on the number of intervals or bins without using (a) and after using (b) class information with red and blue ovals (black dot inside the blue).

In Fig. 6.1(a) discretization was unfavorable, since it made more difficult for a classifier to distinguish between the classes when cases of the different classes were grouped into the same intervals. For real data, it would be unusual to encounter such nicely distributed data; usually there would be a mixture of data points from several classes in each interval, as depicted in Fig. 6.2.

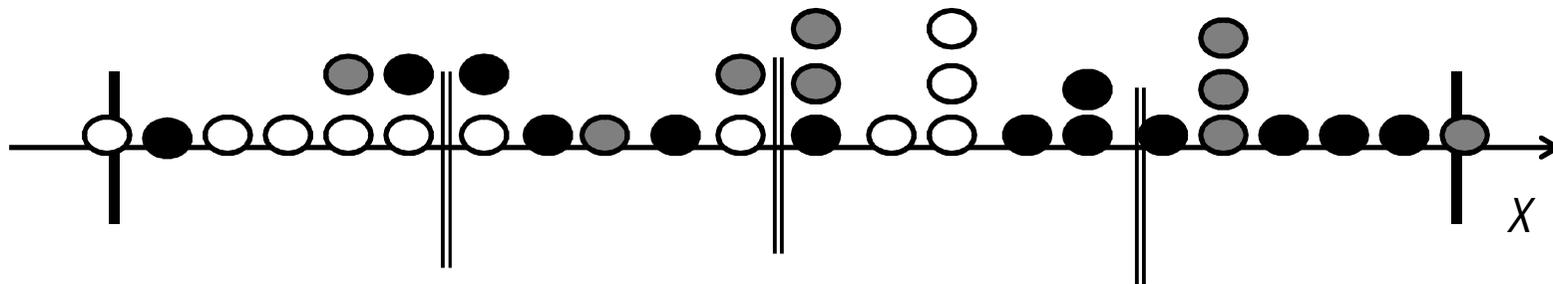


Fig. 6.2 Distribution of values belonging to three classes {white, gray, black} over variable X .

Discretization is mostly performed one variable at a time, known as *static variable discretization*. The opposite approach is known as *dynamic variable discretization*, where all variables are discretized simultaneously while dealing with interdependencies among them. Techniques may also be call *local* or *global*. In the former, not all variables are discretized, and in the latter all are discretized. In the following, terms unsupervised and supervised are used.

6.2 Unsupervised discretization algorithms

These are the simplest to use and implement. The only parameter to specify is the number of intervals or how many values should be included in any given interval.

The following heuristic is often used to choose intervals: *the number of intervals for each variable should not be smaller than the number of classes (if known)*. The other heuristic is to choose the number of intervals, m_{X_i} for each variable, X_i , $i=1, \dots, p$, where p is the number of variables, as follows

$$m_{X_i} = \frac{n}{3C}$$

where n is the number of training cases in data set L and C the number of classes.

We assume that, in general, the user supplies a set of numbers, representing the number of intervals into which each variable is to be discretized:

$$M = \{m_{X_1}, m_{X_2}, \dots, m_{X_p}\}$$

A short description of two unsupervised techniques follows.

Equal-width discretization

The algorithm first finds the minimum and maximum values of every variable, X_i , and then divides this range into a number, m_{X_i} , of user-specified, equal-width intervals.

Equal-frequency discretization

The algorithm determines the minimum and maximum values of the variable, sorts all values in ascending order, and divides the range into a user-defined number of intervals, in such a way that every interval contains the equal number of sorted values.

6.3 Supervised discretization algorithms

A supervised discretization problem can be formalized in view of the class-variable interdependence.

Several supervised discretization algorithms have their origins in *information theory* having a training data set consisting of n samples, examples, objects, items or cases, where each case belongs to only one of C classes. Up to now, such criteria as mutual information and entropy have been presented, but there are also other.

There is a discretization scheme D on variable X that discretizes the continuous variable X into m discrete intervals, bounded by the pairs of numbers

$$D: \{[d_0, d_1], (d_1, d_2], \dots, (d_{m-1}, d_m]\}$$

where d_0 is the minimum and d_m the maximum of variable X , and the values are arranged in ascending order.

Any value of X can be assigned into only one of m intervals defined above.

The membership of each value, within a certain interval, for variable X may change with a change of the discretization D . The class label variable and the discretization variable of variable X are treated as random variables defining a two-dimensional frequency matrix, called the *quanta matrix* (pl. of word quantum) as seen in Table 6.1.

In the table, element q_{ir} is the total number of continuous values belonging to the i^{th} class that are within interval $(d_{r-1}, d_r]$. M_{i+} is the total number of objects or cases belonging to the i^{th} class and M_{+r} is the total number of continuous values of variable X within the interval $(d_{r-1}, d_r]$ for $i=1,2,\dots,C$ and $r=1,2,\dots,p$.

Table 6.1 Two-dimensional quanta matrix for variable X and discretization scheme D .

Class	Interval					Class total
	$[d_0, d_1]$...	$(d_{r-1}, d_r]$...	$(d_{m-1}, d_m]$	
C_1	q_{11}	...	q_{1r}	...	q_{1m}	M_{1+}
.
C_i	q_{i1}	...	q_{ir}	...	q_{im}	M_{i+}
.
C_c	q_{c1}	...	q_{cr}	...	q_{cm}	M_{c+}
Interval total	M_{+1}	...	M_{+r}	...	M_{+m}	M

Example

Let us assume that there are three classes, four intervals and 33 cases distributed as illustrated in Fig. 6.2. The quanta matrix of the data set is shown in Table 6.2.

The values shown in the table have been calculated as follows. Total number of values:

$$M = \sum_{r=1}^m q_{+r} = \sum_{i=1}^c q_{i+}$$

$$M = 8 + 7 + 10 + 8 = 33$$

$$M = 11 + 9 + 13 = 33$$

Table 6.1 Quanta matrix for the data of Fig. 6.2.

Class	Interval				Total
	1 st	2 nd	3 rd	4 th	
White	5	2	4	0	11
Gray	1	2	2	4	9
Black	2	3	4	4	13
Total	8	7	10	8	33

Number of values in the 1st interval:

$$M_{+r} = \sum_{i=1}^C q_{ir}$$

$$M_{+1} = 5 + 1 + 2 = 8$$

Number of values in the White class:

$$M_{i+} = \sum_{r=1}^m q_{ir}$$

$$M_{white+} = 5 + 2 + 4 + 0 = 11$$

We could now calculate various (conditional) probabilities on the basis of the quanta matrix. The estimated joint probability of the occurrence that variable X values are within interval $D_r=(d_{r-1}, d_r]$ and belong to class c_j is given as follows.

$$p_{ir} = p(c_i, D_r | X) = \frac{q_{ir}}{M}$$

Using various probabilities it is possible to apply *Class-variable mutual information* between the class variable and the discretization variable, *Class-variable Information*, *Shannon's entropy*, *Class-variable interdependence redundancy* and *Class-variable interdependence uncertainty*. Nevertheless, we do not deal with those, but view the following.

CVIM: Class-variable interdependency maximization algorithm

This algorithm works in a top-down manner by dividing one of the existing intervals into two new intervals, using a criterion function that results in achieving the optimal class-variable interdependency after the split. It begins with the entire interval $[d_0, d_m]$ and maximizes interdependence between the continuous variable and its class labels to generate a small number of discrete intervals.

The CVIM criterion measures the above dependency between class variable c and discretization variable D for variable X , for a given quanta matrix, as follows

$$CVIM(c, D | X) = \frac{\sum_{r=1}^m \frac{max^2}{M_{+r}}}{m}$$

where m is the number of intervals, r iterates through all intervals, $r=1, 2, \dots, m$, max is the maximum among all q_{ir} values, $i=1, 2, \dots, C$ and M_{+r} is the total number of continuous values of variable X that are from $(d_{r-1}, d_r]$.

The CVIM is a heuristic measure with the following properties:

- The larger the value of CVIM, the higher the interdependence between the classes and intervals. The larger the number of values belonging to class c_j within a particular interval, the higher the interdependence between c_j and the interval. The goal of maximizing this can be translated into the goal of achieving the largest possible number of values that belong to such leading class by using \max_r operation. CVIM achieves the highest value when all values within a particular interval belong to the same class, for all intervals; then $\max_r = M_{rj}$ and $CVIM = M/m$.

- CVIM assumes real values in interval $[0, m]$, where M is the number of values of variable X .
- CVIM generates a discretization scheme, where each interval potentially has the majority of its values grouped within a single class.
- The squared max_r is divided by M_{ri} to account for the (negative) impact that values belonging to classes other than the leading class have on the discretization scheme. The more such values, the bigger the value of M_{ri} , which decreases the CVIM criterion value.

- Since CVIM favors discretization schemes with smaller numbers of intervals, the summed value is divided by the number of intervals m .

The CVIM value is computed with a single pass over the quanta matrix. The optimal discretization scheme can be found by searched over the space of all possible schemes to reach the one with the highest value of CVIM criterion. For the sake of its complexity, the algorithm applies a greedy approach to search for an approximation of the optimal value by finding locally maximal values. Although this does not guarantee the global maximum, it is computationally inexpensive with $O(M \log M)$ time complexity.

Like all discretization algorithms, CVIM consists of two steps: (1) initialization of the candidate interval boundaries and the corresponding initial discretization scheme, and (2) the consecutive additions of a new boundary that results in the locally highest value of CVIM criterion.

CVIM

Data comprising M cases, C classes and continuous variables X_i are given.

For every variable X_i do:

Step 1

(1.1) Find the maximum d_m and minimum d_0 for X_i .

(1.2) Form a set of distinct values of X_i in ascending order, and initialize all possible interval boundaries B with minimum, maximum and all the midpoints of all the adjacent pairs in the set.

(1.3) Set the initial discretization scheme as $D: \{[d_0, d_m]\}$, set GlobalCVIM=0.

Step 2

(2.1) Initialize $k=1$.

(2.2) Tentatively add an inner boundary, which is not already in D , from B , and calculate the corresponding CVIM value.

(2.3) After all the tentative additions have been tried accept the one with the highest value of CVIM.

(2.4) If $(CVIM > GlobalCVIM \text{ or } k < C)$, then update D with the boundary accepted in step (2.3) and set $GlobalCVIM=CVIM$; else terminate.

(2.5) Set $k=k+1$ and go to step (2.2).

Discretization scheme D is the result.

The algorithm starts with a single interval that covers all values of a variable and then divides it iteratively. From all possible division points that are attempted (with replacement) in step (2.2), it selects the division boundary that gives the highest CVIM criterion.

6.4 Other supervised discretization algorithms

Clustering is used for several different purposes. It can also be applied to discretization.

The *K-means algorithm*, the widely used clustering technique, is based on minimization of a performance index defined as the sum of the squared distances of all vectors, in a cluster, to its center (mean). In the following the algorithm is described briefly, for the current context, to find the number and boundaries of intervals.

Note that mostly clustering is applied to unsupervised tasks where no classes are known in advance. Nonetheless, nothing rules out to also use it in the supervised context. Note also that K is its user-defined parameter.

K-means clustering for discretization

A training data set given consists of n cases, C classes and a user-defined number of intervals m_{X_i} for variable X_i .

(1) For $j=1, \dots, C$ do class c_j as follows.

(2) Select $K=m_{X_i}$ as the initial number of cluster centers. At the beginning, the first K values of the variable can be taken as the cluster centers.

(3) Distribute the values of the variable among K cluster centers according to the minimum distance criterion: the cluster is determined for a value by the closest center. As a result, variable values will cluster around the updated K cluster centers.

(4) Compute K new cluster centers such that for each cluster the sum of the squared distances from all points in the same cluster to the new cluster center is minimized, i.e., compute the current means of the clusters.

(5) Check whether the updated K cluster centers are the same as the previous ones. If they are, go to step (1) to treat the next class; otherwise, go to step (3).

As a result, final boundaries are achieved for a single variable that contains of the minimum, midpoints between any two nearby cluster prototypes for all classes and the maximum value of the variable. Of course, this has to be made for all variables.

Note the presented method was a "discretization variation" of K -means clustering. Usually, clustering is performed with all variables together, whereas it was now made for single variables class by class. No classes are usually known, but clusters found correspond to them even if supervised clustering is also possible. There are different manners to

search for the closest clusters in step (3) of the algorithm. For instance, we can also search for the nearest case (for the current one) from each cluster or the farthest case and use this to determine the closest cluster.

The outcome of the algorithm, in an ideal situation, is illustrated in Fig. 6.3. The behavior of the K -means algorithm is influenced by the number of cluster centers K , the choice of initial centers, the order in which cases are considered and the geometric properties of the data.

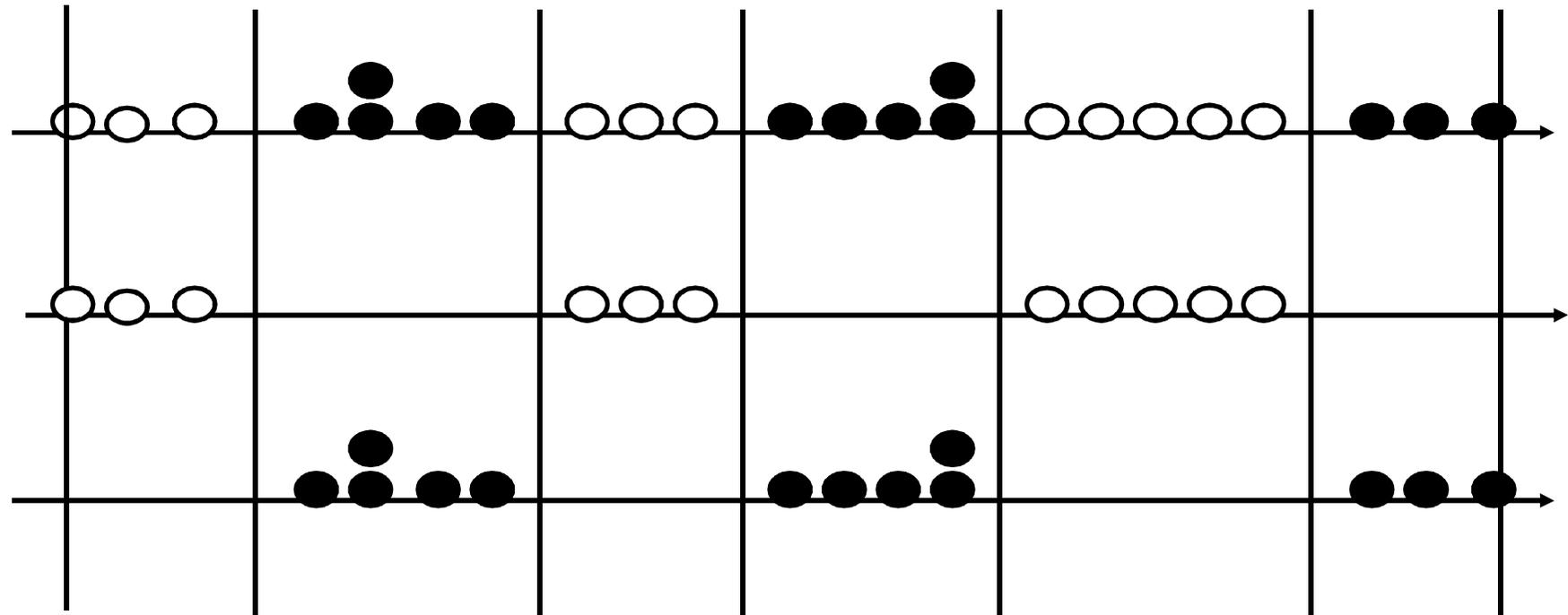


Fig. 6.3 Illustration of an ideal situation of K -means algorithm for discretization. White and black ovals are the two classes. Vertical lines determine the intervals.

In practice, before specifying the number of intervals of each variable, the user could look at the value distribution of each variable, their frequencies in order to determine a good "guess" for the number K of intervals or clusters. In Fig. 6.3 it was correctly guessed to be 6 and, thus the outcome was good. Otherwise, we might have got a number of intervals that did not correspond to the true number of clusters present in a variable given what we are in fact doing in one-dimensional clustering. The problem of selecting the correct number of clusters is inherent generally in clustering. The practical approach is to run several K values and then calculate some measure of the goodness of clustering to find optimal K .

One-level decision tree discretization

The one-level or one-rule (1RD) decision tree algorithm can be used for variable discretization. The method is both general, simple classification algorithm and can be used for discretization. It greedily divides the variable range into intervals, applying the constraint that each interval has to cover at least the user-specified minimum number of continuous values (usually 5). The algorithm begins with initial partition into intervals, each comprising the minimum partition and then moves the initial partition boundaries, by inserting variable values so that each interval contains a strong majority of cases from one class. This is illustrated in Fig. 6.4.

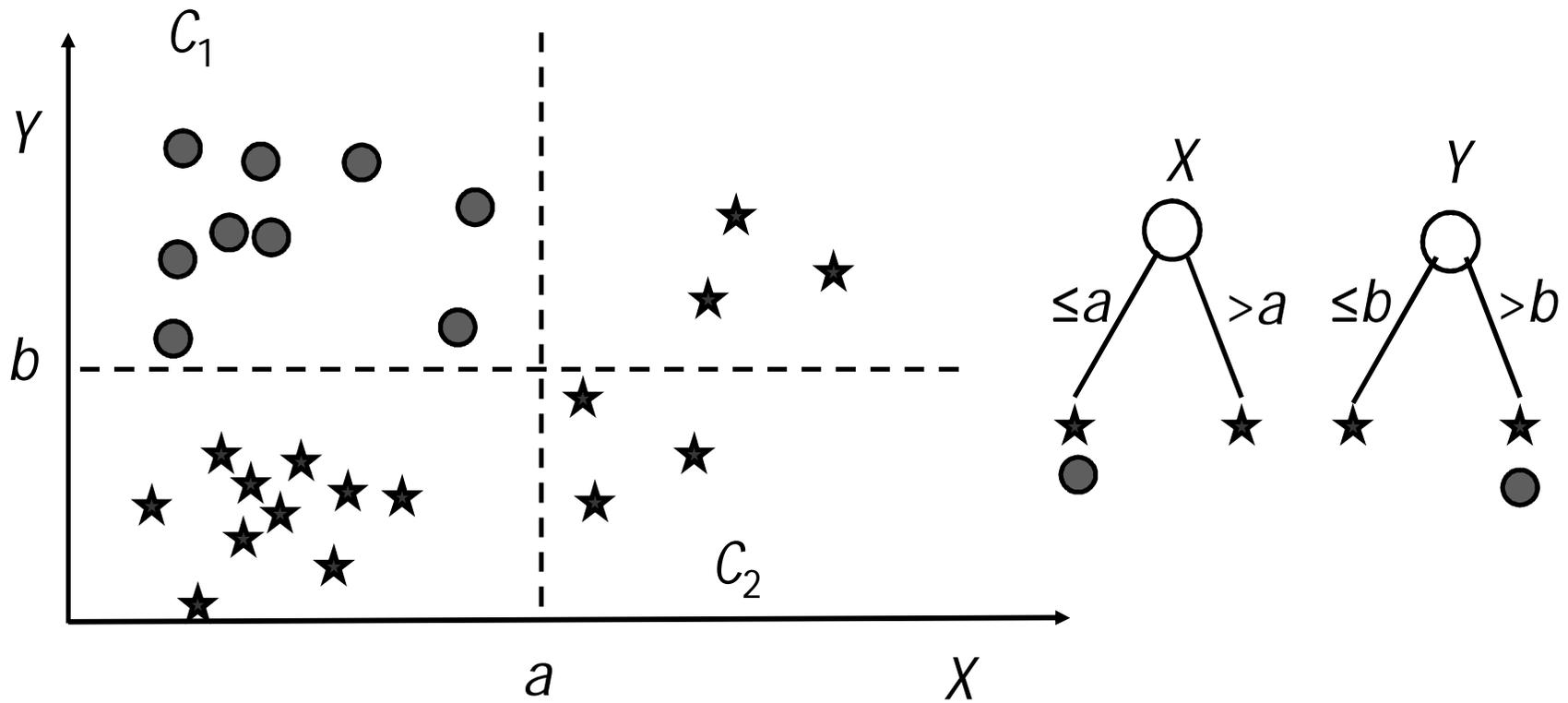


Fig. 6.4 1RD algorithm starts dividing variable X into two intervals with boundary a , when this gives a maximal number of cases of C_2 , related to C_1 , if $X > a$. Then it divides Y with boundary b .