8 CLASSIFICATION AND REGRESSION TREES AND RELATED METHODS

8.1 CLASSIFICATION AND REGRESSION TREES (CART)

ESL, section 9.2.

8.1.1 INTRODUCTION

☛ Methods in last chapter rely on strong **parametric assumptions** (linear model, logistic model or normality).

☛ When these assumptions are far from the truth, the performance can be poor; recall that a linear classifier doesn't work well when the "correct" separation boundary doesn't look linear.

☛ We will introduce **trees**, popularized by Leo Breiman during the 80's; they are less driven by strong parametric assumptions.

Notation:

☛ Recall: we observe i.i.d. samples $(X_1, Y_1), \ldots, (X_n, Y_n)$ coming from the underlying mean model

$$E(Y_i|X_i = x) = m(x),$$

where $Y_i \in \mathbb{R}$ and $X_i = (X_{i1}, \ldots, X_{ip})^T \in \mathbb{R}^p$ are continuous.

☛ In this chapter this notation may become confusing. Therefore, instead, we denote the samples by

$$(\mathcal{X}_1, \mathcal{Y}_1), \ldots, (\mathcal{X}_n, \mathcal{Y}_n),$$

where $\mathcal{Y}_i \in \mathbb{R}$ and $\mathcal{X}_i = (\mathcal{X}_{i1}, \ldots, \mathcal{X}_{ip})^T$.

Notation:

☛ When we don't want to highlight the dependence on the sample, we use the notation (X, Y), where (X, Y) has the same distribution as $(\mathcal{X}_i, \mathcal{Y}_i)$.

☛ In particular, they also come from the model

$$E(Y|X = x) = m(x),$$

where $Y \in \mathbb{R}$ and $X = (X_1, \ldots, X_p)^T \in \mathbb{R}^p$ is continuous.

☛ Hence, $\{X_1, \ldots, X_p\}$ mean the **coordinates** of the generic vector X.

☛ Generally, $m(\cdot)$ may look quite non-smooth over the *feature space*, i.e. the domain of X, or the subset of $\mathbb{R}^p$ of all possible values of X.

☛ However, it is likely

$$m(x_1) \approx m(x_2)$$

if x_1 and x_2 are two close points in the feature space.

☛ Main idea of regression trees:

- **Partition** the feature space into **disjoint** regions $R_1, R_2, \ldots$
- On each region R_i, approximate $m(x)$ by a **constant**.

☛ To fix idea, first consider $p = 2$. ; we are to estimate

$$m(x) = E(Y|X = x)$$

where $Y \in \mathbb{R}$ and $X = (X_1, X_2)^T \in \mathbb{R}^2$ is continuous.

☛ In action, one grow a regression tree via a *sequence* of partitions, roughly as follows:

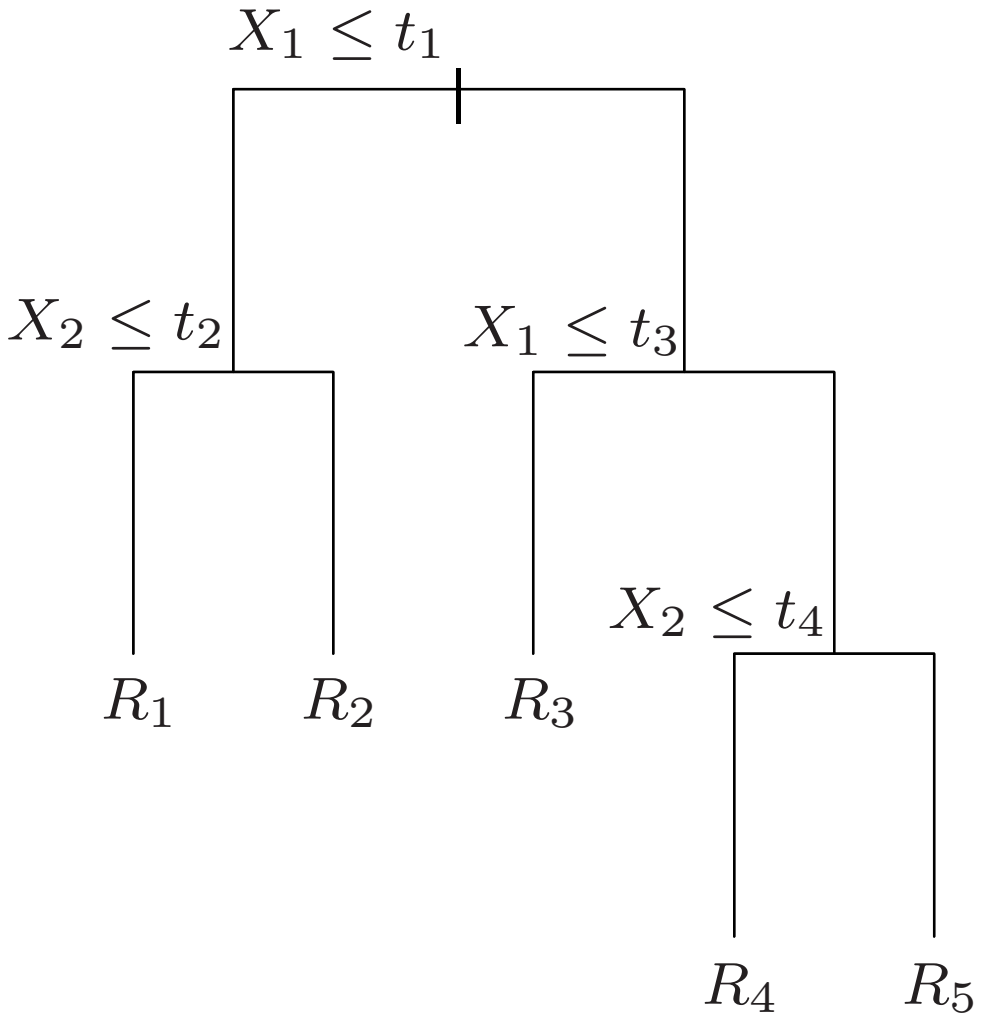
1. First, pick the variable X_1 and a real number t_1. Then consider all the points in the feature space with $X_1 \leq t_1$ as one region, and all points with $X_1 > t_1$ as another. This binary split gives rise to two regions described by
$$\{X_1 \leq t_1\},\ \{X_1 > t_1\}.$$
X_1 and t_1 are respectively called the splitting variable and split point at this step.
2. Next, within the region
$$\{X_1 \leq t_1\}$$
produced from step 1, we may choose the splitting variable X_2 and a split point t_2, and partition $\{X_1 \leq t_1\}$ into two further sub-regions described by $\{X_2 \leq t_2\}$, $\{X_2 > t_2\}$.
3. For the other region
$$\{X_1 > t_1\}$$
from step 1, we may pick X_1 as the splitting variable again and a split point t_3, and split this into $\{X_1 \leq t_3\}$, $\{X_1 > t_3\}$ similarly.

☛ After *recursively* repeating this sequence of binary partitions several times, at the finest level, we will have partitioned the feature space into disjoint rectangular regions, say $R_1, \ldots, R_L$.

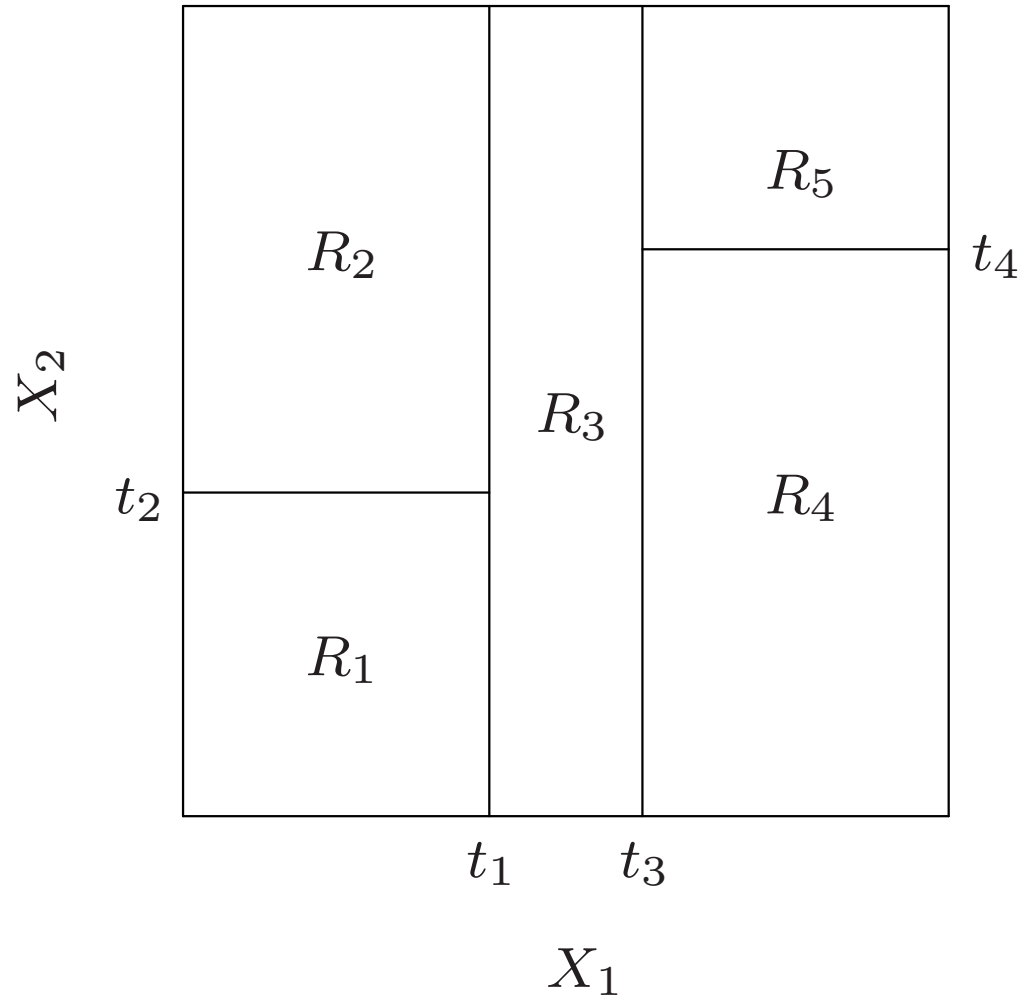
☛ The next two illustrations come from the ESL book.

☛ (I haven't talked about *how to pick* the splitting variable and the split point at each step yet...)

Constructing a tree by a sequence of binary partitions of the type $\{X_j \leq t\}$, $\{X_j > t\}$. After a number of partitions, stop the splitting process and obtain regions $R_1, \ldots, R_L$, which are called *terminal nodes/leaves*.



The feature space is partitioned into a series of rectangles $R_1, R_2, \ldots$:



☛ The regions $R_1, \ldots, R_L$ obtained at the end of the process are called terminal nodes or leaves of the tree.

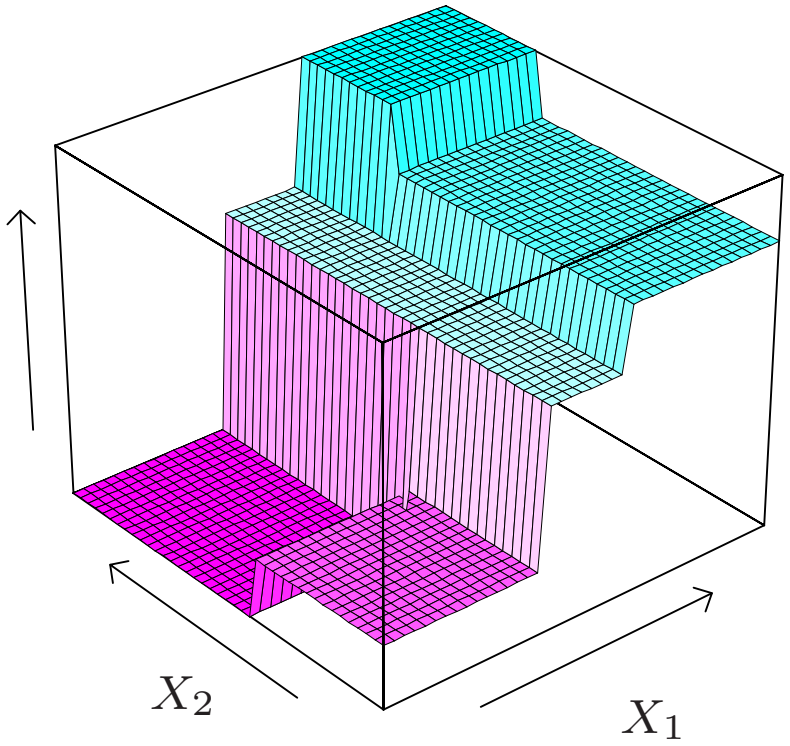
☛ The splits such as $\{X_1 \leq t_1\}$, inside the tree, are called internal nodes.

☛ The segments of the tree connecting the nodes are called the branches.

☛ Once we have partitioned the space into regions $R_1, \ldots, R_L$, on each region we approximate the regression surface m by a constant:

For all x in feature space, $\quad m(x) \approx \sum_{\ell=1}^{L} c_\ell \, I\{x \in R_\ell\} = \begin{cases} c_\ell & \text{if } x \in R_\ell \\ 0 & \text{otherwise.} \end{cases}$

Piecewise constant approximations on regions $R_1, R_2, \ldots$ look like:

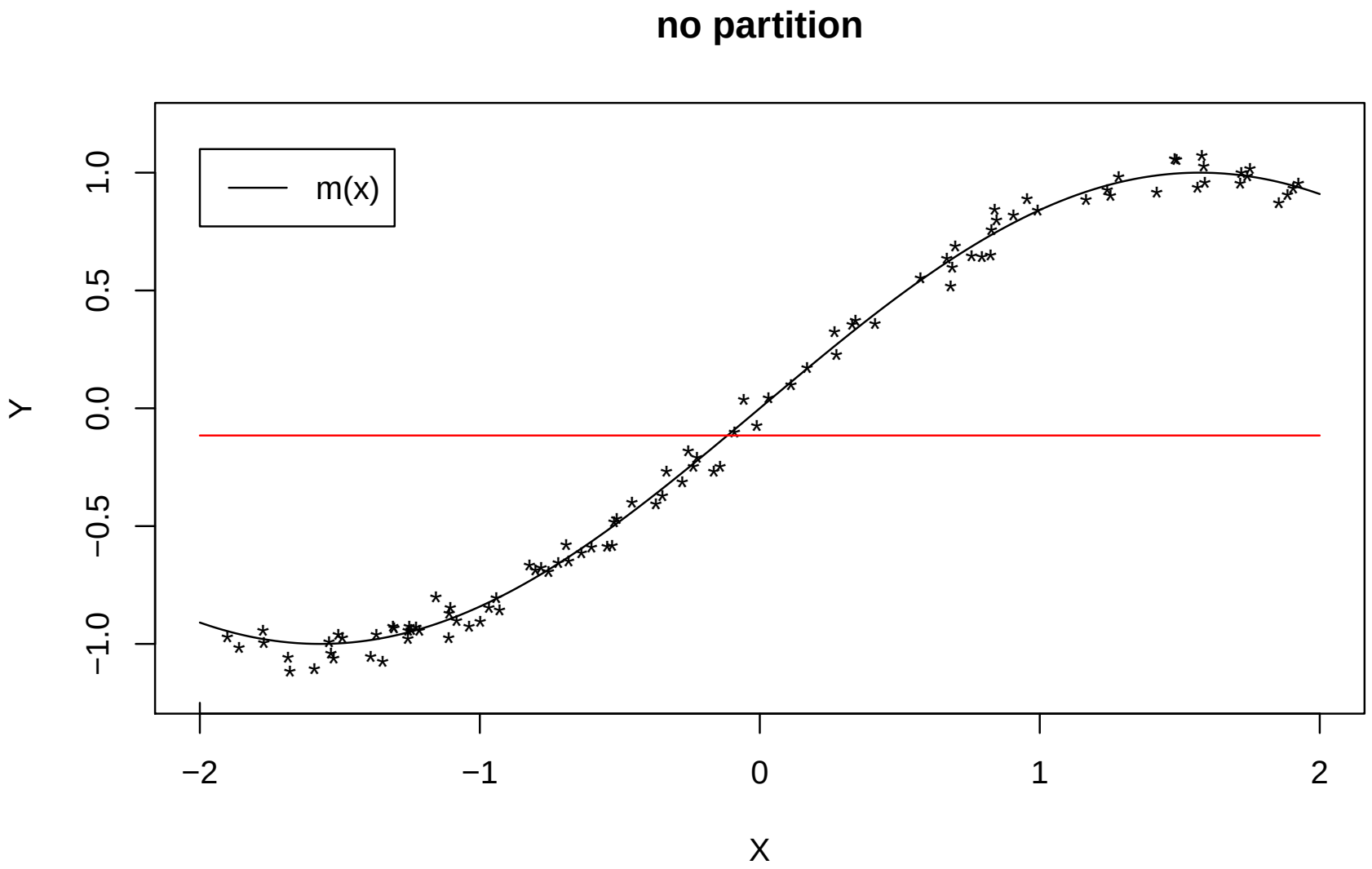


☛ Why is this flexible? As long as we partition a feature space in small enough pieces, we can always approximate a regression surface well enough by constants on each piece.

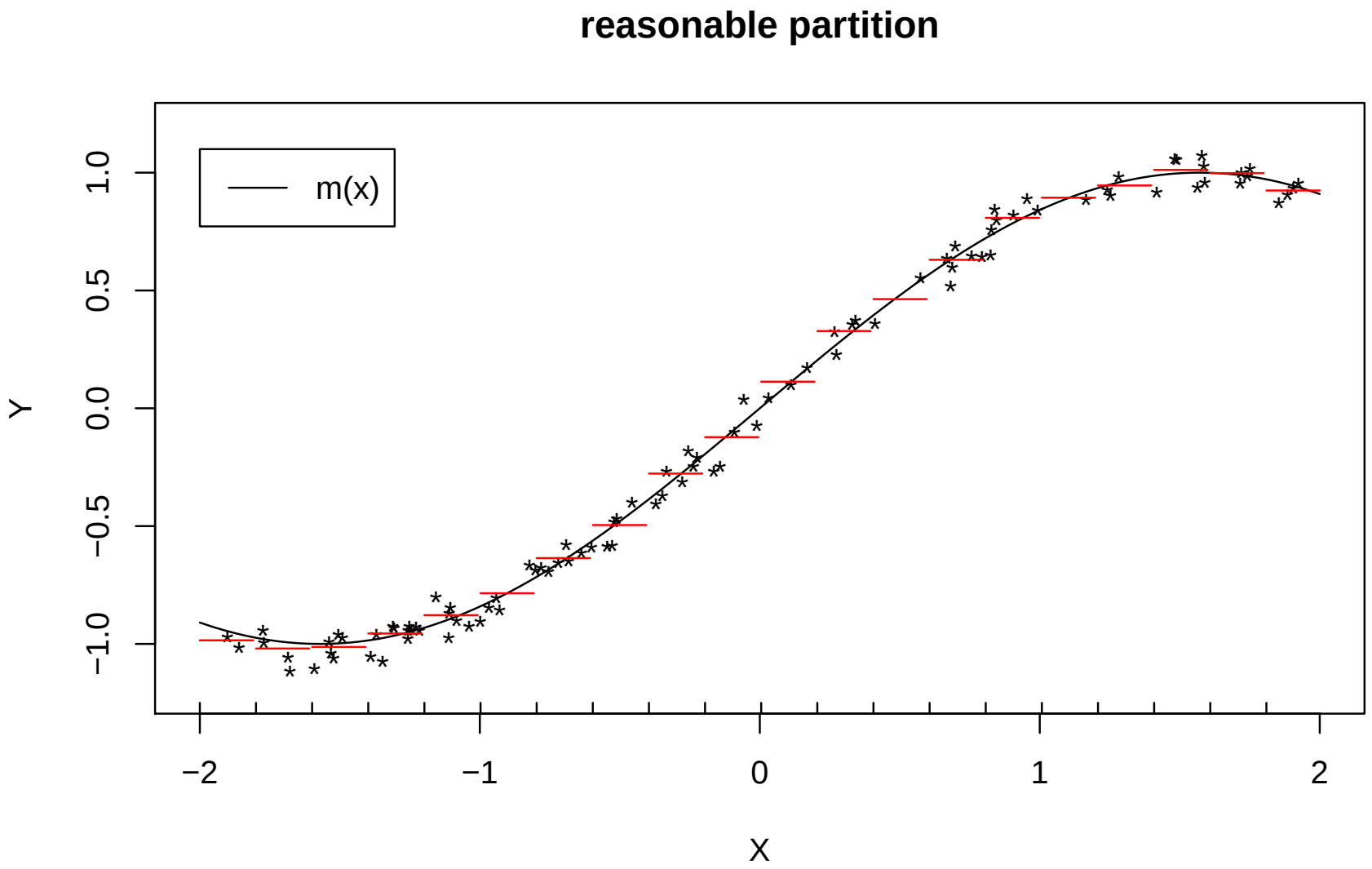
☛ The finer the partition, the better is the approximation of m by constants $c_1, c_2, c_3, \ldots$ on the regions $R_1, R_2, R_3, \ldots$.

☛ Next is an example where $p = 1$. Here, partitioning the feature space means splitting the range of values of X into intervals.

- $*$: data points.
- Red line: the average of the function values $m(x)$ across the whole interval (obtained by integration); it doesn't approximate m well.



- Partition the whole interval into smaller sub-intervals.
- Red lines: the average of the function values $m(x)$ across each smaller sub-interval; $m(\cdot)$ is much better approximated by them.



☛ How to choose the constants $c_1, c_2, \ldots$ in practice, from the actual data?

☛ More generally, when X is a p-vector, how to fit a regression tree?

☛ Say we have already partitioned the feature space into disjoint $R_1, \ldots, R_L$ by using a sequence of binary splits of the type $\{X_j \leq t\}, \{X_j > t\}$, for various $j \in \{1, \ldots, p\}$ and t values.

☛ Then for $\ell \in \{1, \ldots, L\}$ and $x \in R_\ell$, we approximate $m(x)$ by

$$\hat{m}(x) = \hat{c}_\ell = \text{average}(\mathcal{Y}_i) \text{ such that } \mathcal{X}_i \in R_\ell.$$

☛ So on each region R_ℓ, we take the average of the $\mathcal{Y}_i$'s whose $\mathcal{X}_i \in R_\ell$.

☛ This is the same as choosing $c_1, \ldots, c_L$ that minimise the RSS:

$$RSS(c_1, \ldots, c_L) = \sum_{i=1}^{n} \left[\mathcal{Y}_i - \sum_{\ell=1}^{L} c_\ell \, I\{\mathcal{X}_i \in R_\ell\}\right]^2 = \sum_{\ell=1}^{L} \sum_{i:\mathcal{X}_i \in R_\ell} (\mathcal{Y}_i - c_\ell)^2.$$

☛ By differentiating $RSS(c_1, \ldots, c_L)$ with respect to the c_ℓ's, we see that

$$\frac{\partial RSS}{\partial c_\ell} = 2 \left(N_\ell c_\ell - \sum_{i:\mathcal{X}_i \in R_\ell} \mathcal{Y}_i \right),$$

where $N_\ell = |\{i : \mathcal{X}_i \in R_\ell\}|$ is the number of $\mathcal{X}_i$ in R_ℓ.

☛ Setting $\frac{\partial RSS}{\partial c_\ell} = 0$, we get that

$$\hat{c}_\ell = N_\ell^{-1} \sum_{i:\mathcal{X}_i \in R_\ell} \mathcal{Y}_i = \text{average}(\mathcal{Y}_i) \text{ s.t. } \mathcal{X}_i \in R_\ell.$$

☛ Digression:

- Suppose a region R_ℓ is further partitioned into two subregions $R_{\ell 1}$ and $R_{\ell 2}$ in whatever manner, and we let

$$\hat{m}(x) = \hat{c}_{\ell j} = |\{i : \mathcal{X}_i \in R_{\ell j}\}|^{-1} \sum_{i:\mathcal{X}_i \in R_{\ell j}} \mathcal{Y}_i \text{ for each } j \in \{1, 2\} \text{ and } x \in R_{\ell j}.$$

- In this case, the total RSS can only drop.

- Proof:

$$\begin{aligned}
&\sum_{i:\mathcal{X}_i \in R_{\ell 1}} (\mathcal{Y}_i - \hat{c}_{\ell 1})^2 + \sum_{i:\mathcal{X}_i \in R_{\ell 2}} (\mathcal{Y}_i - \hat{c}_{\ell 2})^2 \\
&= \min_{c_{\ell 1} \in \mathbb{R}} \sum_{i:\mathcal{X}_i \in R_{\ell 1}} (\mathcal{Y}_i - c_{\ell 1})^2 + \min_{c_{\ell 2} \in \mathbb{R}} \sum_{i:\mathcal{X}_i \in R_{\ell 2}} (\mathcal{Y}_i - c_{\ell 2})^2 \\
&\leq \min_{c_\ell} \sum_{i:\mathcal{X}_i \in R_\ell} (\mathcal{Y}_i - c_\ell)^2 \\
&= \sum_{i:\mathcal{X}_i \in R_\ell} (\mathcal{Y}_i - \hat{c}_\ell)^2.
\end{aligned}$$

Note that the inequality is true because R_ℓ is precisely a disjoint union of $R_{\ell 1}$ and $R_{\ell 2}$.

☛ But how do we decide the *skeleton* of our tree? In other words, how do we choose the consecutive splits?

☛ Recall: a split is based on the values of *one of* the components of $X = (X_1, \ldots, X_p)^T$, say X_j.

☛ Ideally, we want the entirety of the splits to be RSS mimimizing, but that is not computationally possible.

$\Rightarrow$ Because it would involve comparing *all possible sequences* of splits of the type $\{X_j \leq t\}, \{X_j > t\}$ for all values of $j = 1, \ldots, p$ and *all values* of the split point t for X_j.

$\Rightarrow$ In fact, for a splitting variable X_j, it suffices to consider the values among $\mathcal{X}_{1j}, \ldots, \mathcal{X}_{nj}$ for the split point t; this is b/c a split divides the *observed data* into two parts, and these parts only change when the split point t is one of the observed data.

$\Rightarrow$ However, still too time consuming to consider all possible partitions.

☛ Instead we grow the tree as a greedy algorithm, which only splits optimally *at each step*:

1. Start with all the data. Consider all possible ways of doing the first split, i.e. consider all splits of the data in two regions

$$R_1(j,t) = \{\mathcal{X}_i \text{ s.t. } \mathcal{X}_{ij} \leq t\}, \quad R_2(j,t) = \{\mathcal{X}_i \text{ s.t. } \mathcal{X}_{ij} > t\}$$

   for all values of $j = 1, \ldots, p$ and all values of t that the $\mathcal{X}_{ij}$'s take.

   (As explained above, suffices to consider only $t \in \{\mathcal{X}_{1j}, \ldots, \mathcal{X}_{nj}\}$)

2. Choose the splitting variable j and the split point t that minimise, over j and t, the RSS:

$$\min_{c_1} \sum_{\mathcal{X}_i \in R_1(j,t)} (\mathcal{Y}_i - c_1)^2 + \min_{c_2} \sum_{\mathcal{X}_i \in R_2(j,t)} (\mathcal{Y}_i - c_2)^2.$$

As said earlier, this is the same as minimising over j and t:

$$\sum_{\mathcal{X}_i \in R_1(j,t)} (\mathcal{Y}_i - \hat{c}_1)^2 + \sum_{\mathcal{X}_i \in R_2(j,t)} (\mathcal{Y}_i - \hat{c}_2)^2.$$

where

$$\hat{c}_\ell = \text{average}(\mathcal{Y}_i) \text{ s.t. } \mathcal{X}_i \in R_\ell(j,t).$$

3. Once we have found the best j and t defined above, our first split is determined. It creates two regions $R_1(j,t)$ and $R_2(j,t)$ of data.

4. Then, on each of those two regions we repeat the same procedure:

   ☛ Consider all possible splitting of R_1 in two regions determined by a choice of j and t and find j and t that minimise the RSS.

   ☛ Then consider all possible splitting of R_2 in two regions determined by a choice of j and t and find j and t that minimise the RSS.

   ☛ In total this determines 4 regions at the finest level.

   Here too, when splitting R_ℓ in two parts according to X_j, we only need to consider t among the $\mathcal{X}_{ij}$'s such that $\mathcal{X}_i \in R_\ell$.

5. Note: each R_ℓ has its own splitting variable X_j and splitting point t.

6. We keep repeating the splitting procedure on each new region created in the previous step until we decide to stop.

☛ This is a "greedy" algorithm: you make *locally optimal* decision at each stage (and don't regret).

☛ When should we stop this splitting process? In other words, how large should we grow our tree?

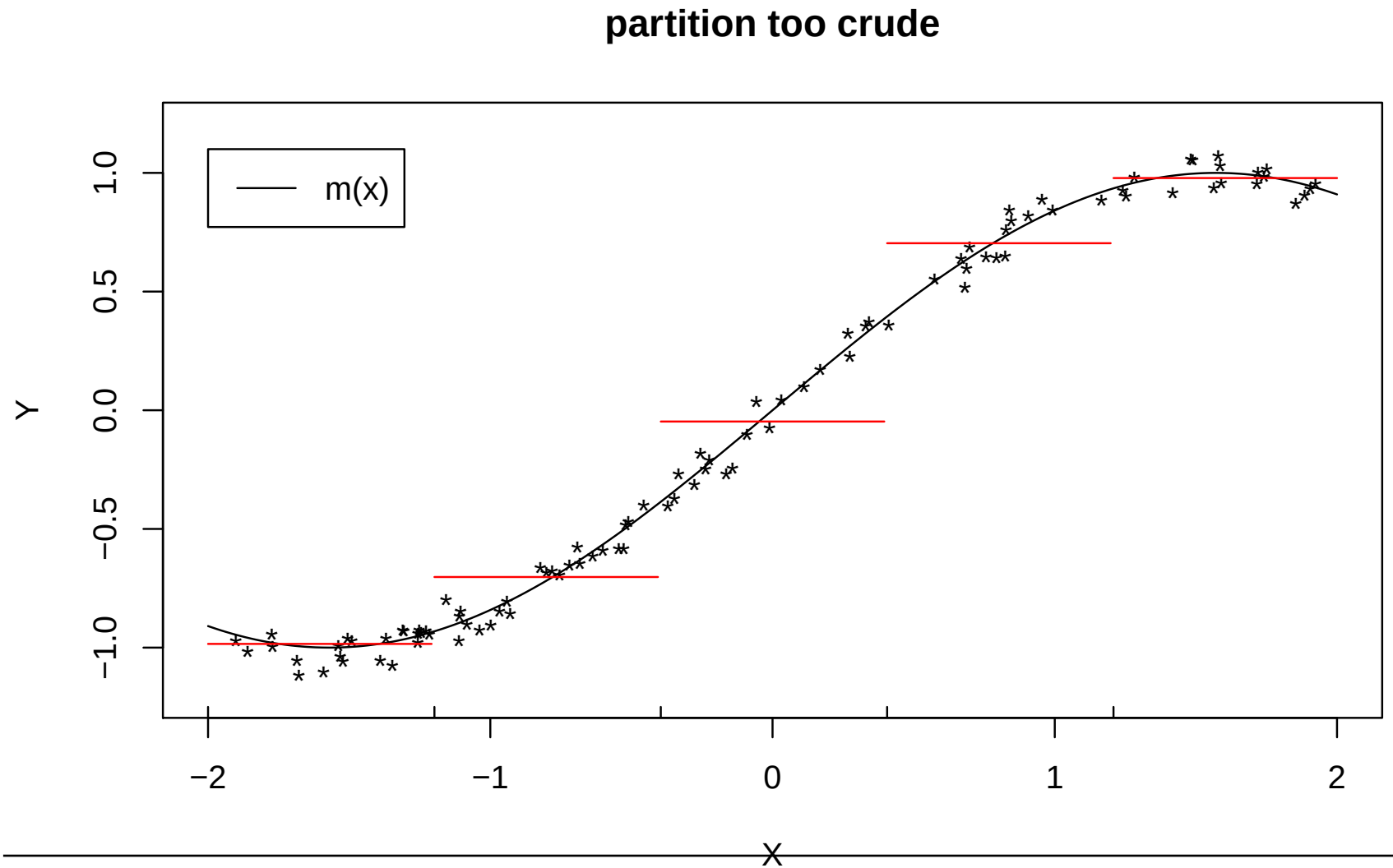
☛ A tree too large will overfit the data and a tree too small won't capture enough structure of $m(\cdot)$.

☛ The size of the tree (L) is a tuning parameter governing the complexities of the model; it should depend on the sample size n.

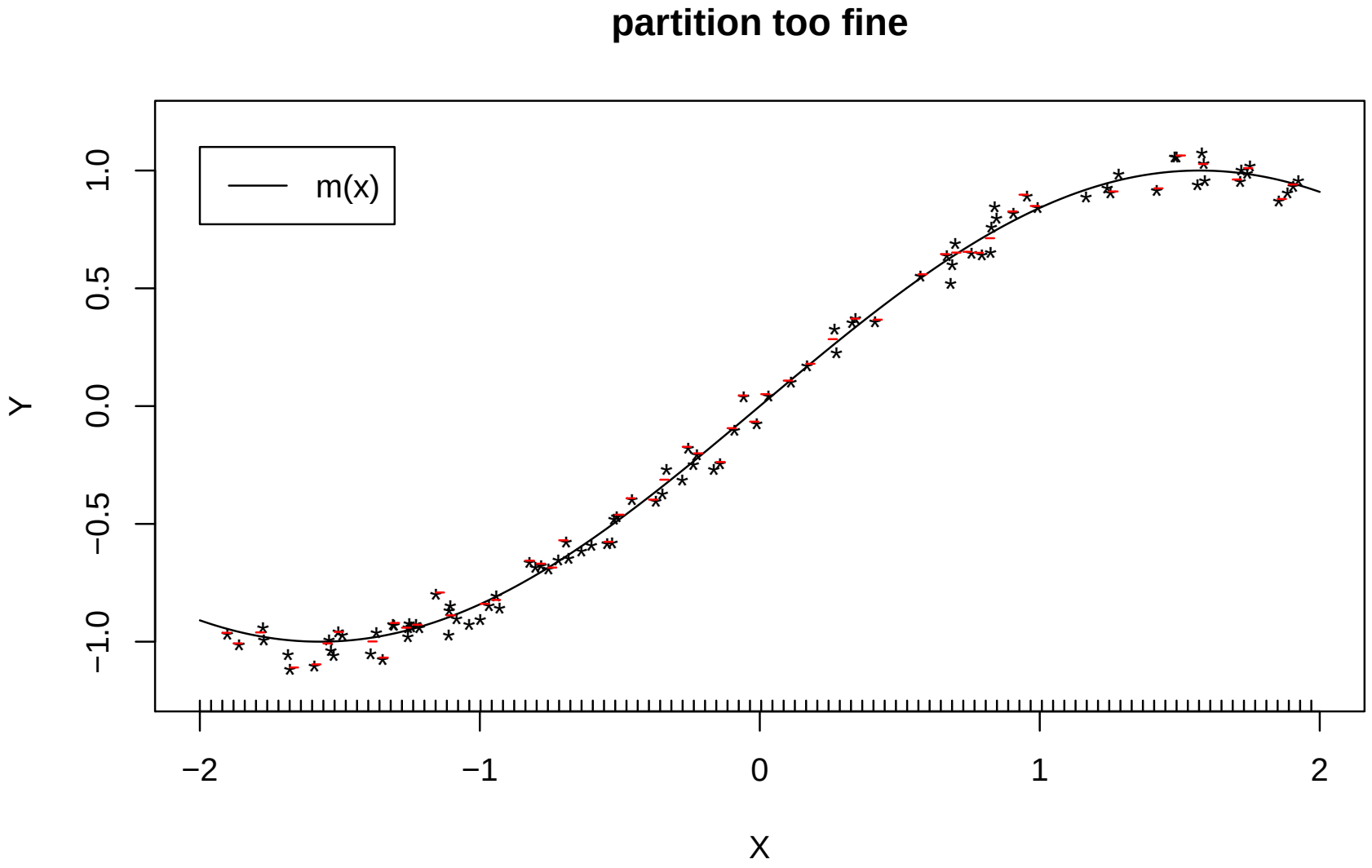
☛ Small n: take crude partitions of the feature space (small L).

☛ As n increases: take finer partitions (larger L).

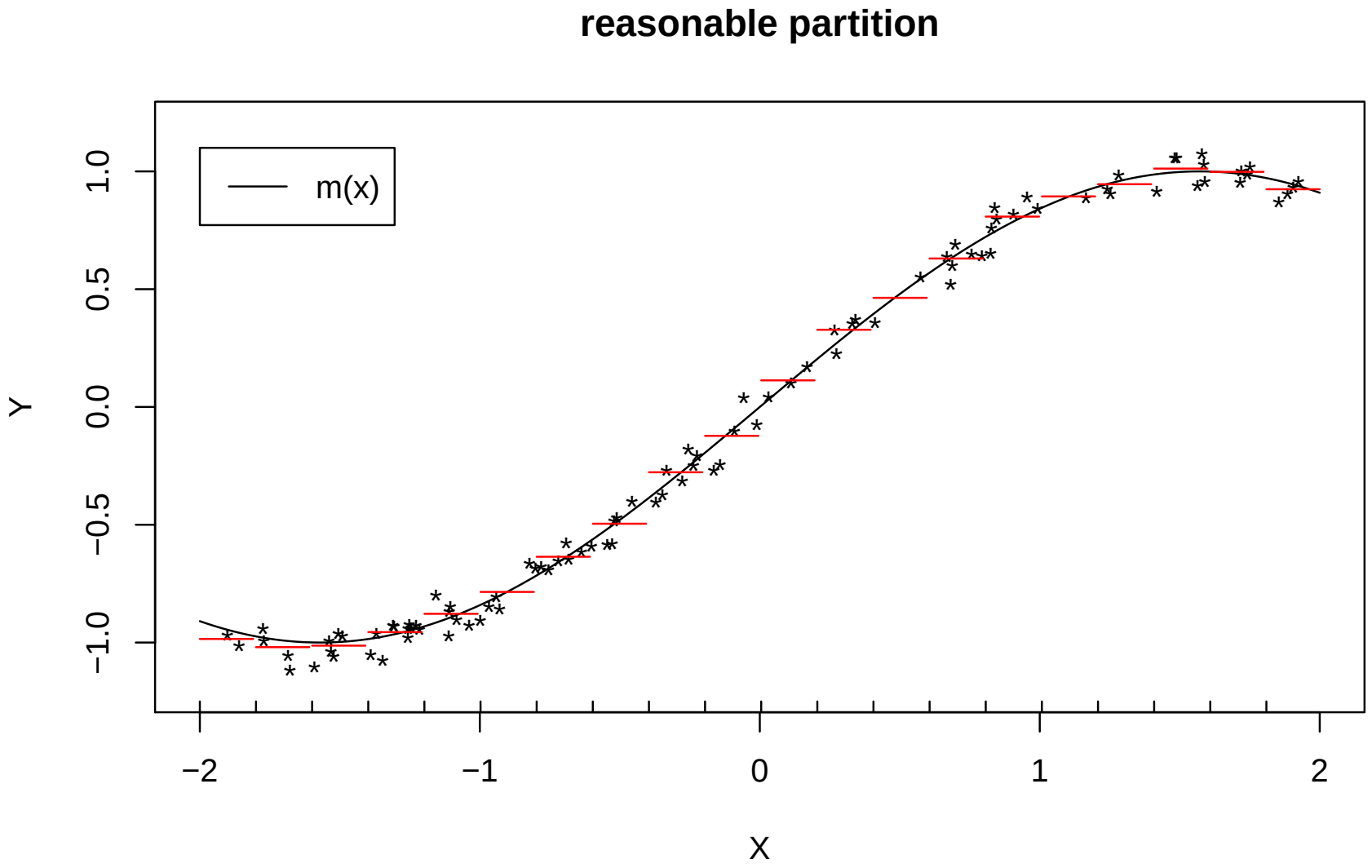
Simple example when $p = 1$: Using too crude partition (L small) does not work well: we don't capture the important structure about the curve m. The level of a red line is computed from the average of the data in corresponding interval.



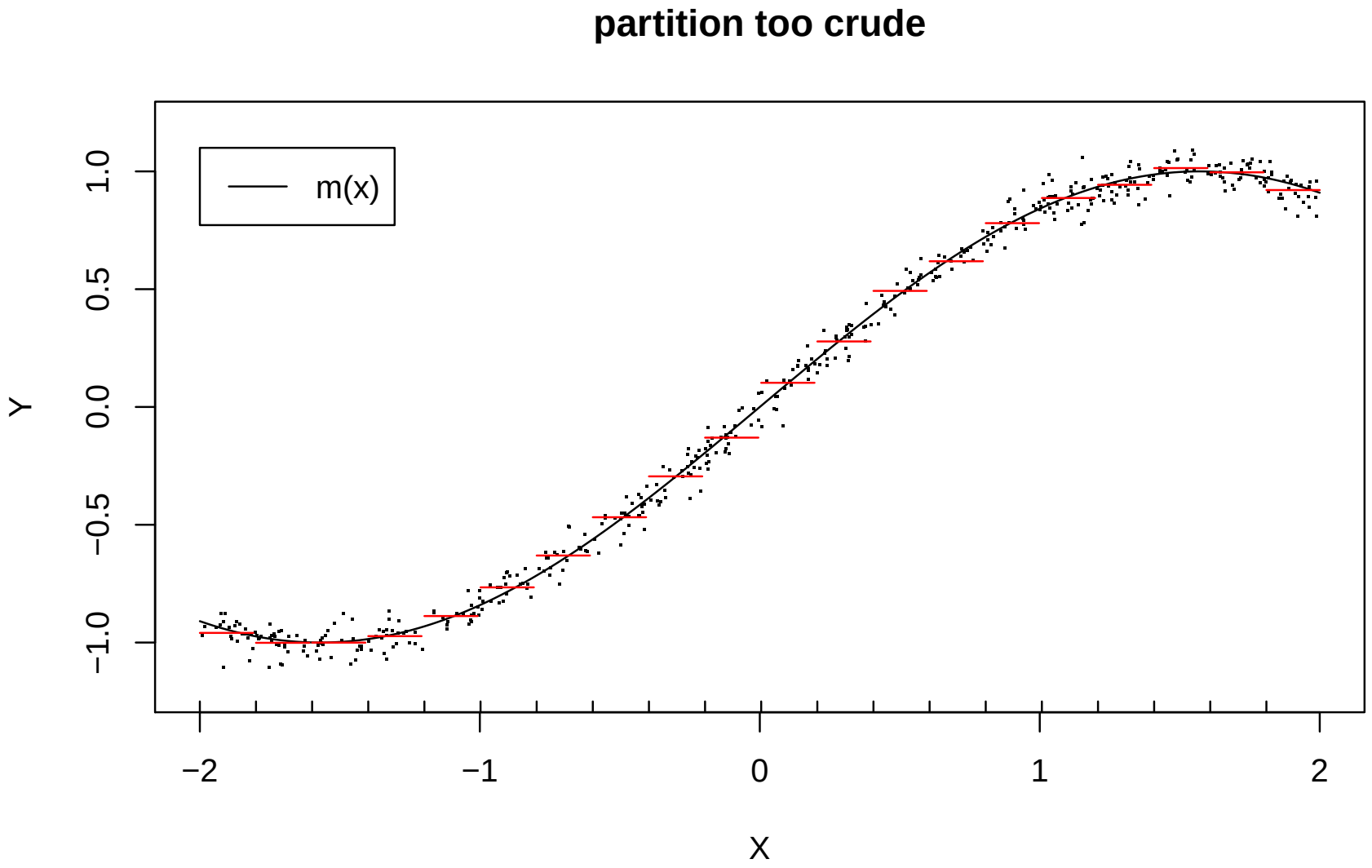
Using too fine partition (L large) does not work well either: we overfit the data and some intervals have no data, thus it is impossible to compute a constant fit on those intervals.



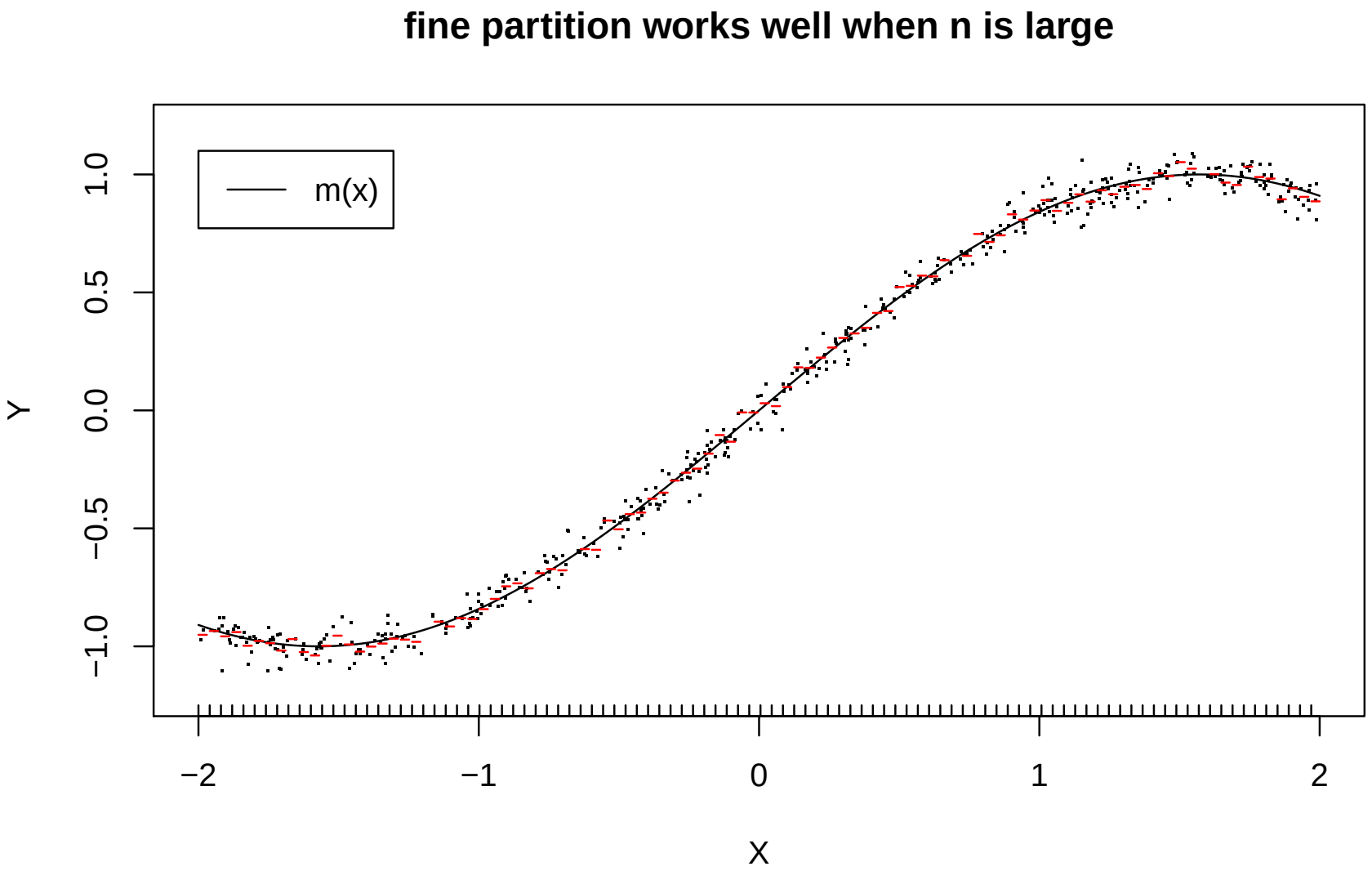
Need to choose the right level of partition, then it works well



As n increases, partition should become finer (L should increase)



With n larger, a fine partition (large L) works well because we have more data to fit the constants on the small regions



☛ One strategy: perform a split into two regions only if that split produces a drop in the RSS *beyond a certain threshold*, where

$$RSS = \sum_{i=1}^{n} \{\mathcal{Y}_i - \hat{m}(\mathcal{X}_i)\}^2$$

and, for $x \in \mathbb{R}^p$,

$$\hat{m}(x) = \sum_{\ell=1}^{L} \hat{c}_\ell \, I\{x \in R_\ell\}.$$

☛ However, it could happen that one split is worthless but the next splits could be very useful, so this strategy is too short-sighted.

☛ The preferred strategy:

- grow a large tree (larger than we need) until each region contains only a small predetermined number of observations (say 5); call this tree T_0.
- T_0 is then "**pruned**" (remove some less useful branches and merge the corresponding regions, i.e. collapse some internal nodes), as follows:

☛ Let $T \subset T_0$ be any tree that can be obtained by pruning T_0 (i.e. T is obtained by collapsing a certain number of internal nodes). We measure the size of T by counting its number of leaves, denoted as $|T|$.

☛ For the ℓ-th leaf, let

$$N_\ell = \text{number of obs } \mathcal{X}_i \in R_\ell \ .$$

☛ Recall for $x \in R_\ell$, we estimate $m(x)$ by

$$\hat{m}(x) = \hat{c}_\ell = \frac{1}{N_\ell} \sum_{\mathcal{X}_i \in R_\ell} \mathcal{Y}_i.$$

☛ For a tree T, denote the average RSS within the region R_ℓ , by

$$Q_\ell(T) = \frac{1}{N_\ell} \sum_{\mathcal{X}_i \in R_\ell} (\mathcal{Y}_i - \hat{c}_\ell)^2.$$

☛ The cost complexity criterion is defined by

$$C_\alpha(T) = \sum_{\ell=1}^{|T|} N_\ell Q_\ell(T) + \alpha|T|.$$

☛ $\alpha \geq 0$: a tuning parameter that governs the trade-off between the tree size $|T|$ and the goodness of fit to the data, $\sum_{\ell=1}^{|T|} N_\ell Q_\ell(T)$.

☛ Large α penalizes the size $|T|$ more $\Rightarrow$ the tree that minimises $C_\alpha(T)$ will be smaller (we'd merge a lot of regions).

☛ small α penalizes the size $|T|$ less $\Rightarrow$ the tree that minimises $C_\alpha(T)$ will be larger (we'd merge fewer regions).

☛ $\alpha = 0$ does't penalise the size of the tree at all $\Rightarrow$ the tree that minimises $C_\alpha(T)$ will be T_0. Indeed, the RSS of a larger tree cannot be larger than that of a subtree.

☛ For each α, one can show there exists a unique smallest subtree $T_\alpha \subset T_0$ that minimises $C_\alpha(T)$.

☛ To find T_α, use the *weakest link pruning*, as follows:

1. Collapse the internal node (merge two leaves/regions produced by a node) which results in the smallest increase in $\sum_{\ell=1}^{|T|} N_\ell Q_\ell(T)$.

2. Repeat this procedure until there is no region to merge (single leaf tree).

3. Then compute $C_\alpha(T)$ for all trees T in this finite sequence of subtrees.

4. One can show that T_α will be the smallest tree, in that sequence, that minimises $C_\alpha(T)$. (ESL refers to Breiman et al. (1984) or Ripley (1996) for a proof)

☛ In practice, α is chosen by cross-validation (leave-one-out, five-fold, or ten-fold, etc.): we choose $\hat{\alpha}$ to minimize the cross-validated sum of squares.

☛ Using $\hat{\alpha}$ and train with all the data, we get the final tree $T_{\hat{\alpha}}$.

☛ Trees are also used for classification: We observe training data

$$(\mathcal{X}_i, G_i) \text{ for } i = 1, \ldots, n,$$

where $G_i \in \{1, \ldots, K\}$ is the class label of the ith individual.

☛ Suppose we have partitioned the feature space in regions $R_1, \ldots, R_L$. We estimate the class label G of a new individual X by

$$\hat{G} = \hat{G}(X) = \sum_{\ell=1}^{L} \hat{c}_\ell \, I\{X \in R_\ell\} = \begin{cases} \hat{c}_1 & \text{if } X \in R_1 \\ \vdots & \\ \hat{c}_L & \text{if } X \in R_L \end{cases}$$

where $\hat{c}_\ell$ is the predicted class of region R_ℓ.

☛ How to compute $\hat{c}_\ell$, the predicted class of region R_ℓ?

☛ In a region R_ℓ, for each $k = 1, \ldots, K$, we count the proportion $\hat{p}_{\ell k}$ of training data whose $G_i = k$:

$$\hat{p}_{\ell k} = \frac{1}{N_\ell} \sum_{i \text{ s.t. } \mathcal{X}_i \in R_\ell} I\{G_i = k\},$$

where N_ℓ is the number of training data in region R_ℓ.

☛ The predicted class $\hat{c}_\ell$ in region R_ℓ is taken to be

$$\hat{c}_\ell = \text{argmax}_{k=1,\ldots,K} \, \hat{p}_{\ell k},$$

i.e. the class occurring most often among the training data in R_ℓ.

☛ Recall that node splitting and tree pruning is based on the cost complexity criterion

$$C_\alpha(T) = \sum_{\ell=1}^{|T|} N_\ell Q_\ell(T) + \alpha |T|,$$

where Q_ℓ is a **node impurity measure**.

☛ For regression, we have picked $Q_\ell = \frac{1}{N_\ell} \sum_{\mathcal{X}_i \in R_\ell} (\mathcal{Y}_i - \hat{c}_\ell)^2$. For classification, we need to use another more suitable impurity measure:

☛ Misclassification error:

$$Q_\ell(T) = \frac{1}{N_\ell} \sum_{i \text{ s.t. } \mathcal{X}_i \in R_\ell} I\{G_i \neq \hat{c}_\ell\} = 1 - \hat{p}_{\ell \hat{c}_\ell}$$

☛ Gini index:

$$Q_\ell(T) = \sum_{k \neq k'} \hat{p}_{\ell k} \, \hat{p}_{\ell k'} = \sum_{k=1}^{K} \hat{p}_{\ell k}(1 - \hat{p}_{\ell k}) = 1 - \sum_{k=1}^{K} \hat{p}_{\ell k}^2.$$

From the last expression, it is obvious why it measures impurity: obviously it is zero (no impurity) if one of the $\hat{p}_{\ell k}$ is 1, and becomes larger than the weights are spread among the $\hat{p}_{\ell k}$'s.

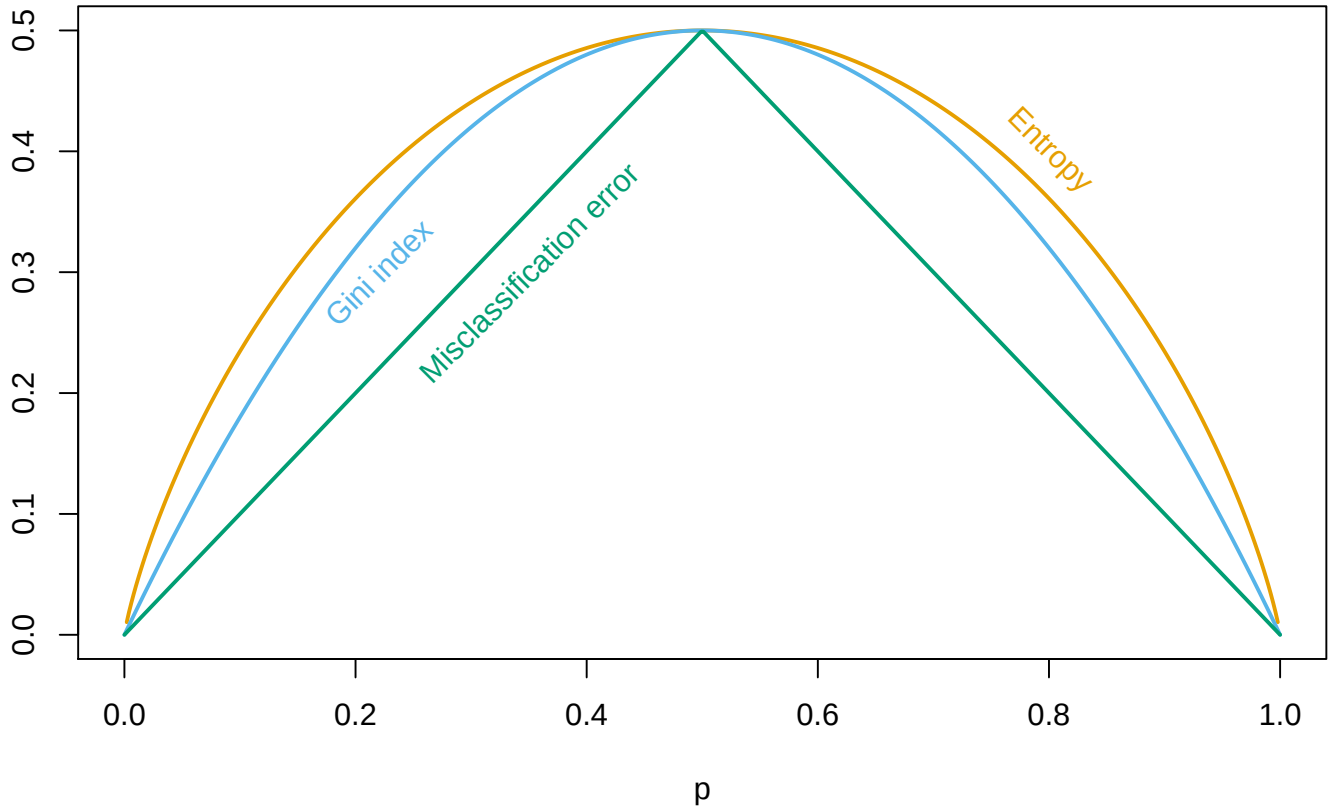
☛ Cross-entropy/deviance:

$$Q_\ell(T) = -\sum_{k=1}^{K} \hat{p}_{\ell k} \log \hat{p}_{\ell k}.$$

Since $0 \leq \hat{p}_{\ell k} \leq 1$, the cross-entropy must be ≥ 0. It is obviously 0 when one of the $\hat{p}_{\ell k}$ is 1.

The quantity can both be interpreted as entropy or cross-entropy; see this discussion in stackexchange.

Comparison of three criteria when $K = 2$. Indices shown as a function of $p = p_{\ell 2}$ (taken from ESL page 309):



As a function in p, Entropy and Gini are differentiable, so they are also used in numerical optimization problems.

☛ For *tree growing*: Entropy and Gini are generally more preferable:

– They are more sensitive to node probabilities and more likely to produce pure nodes containing only one class of data. From the graph, **Entropy** and **Gini** penalize a non-zero probability more than the misclassification error

– Example: $K = 2$ problem with 400 observations in each class; denote a single parent node with all observations as $(400, 400)$. Suppose one split create child nodes

$$(300, 100) \text{ and } (100, 300),$$

and another split create child nodes

$$(200, 400) \text{ and } (200, 0).$$

Both splits give a misclassification rate of 0.25, but Gini and Entropy are lower for the second split, which has one pure child node and hence more preferable.

☛ For *tree pruning*: all three can be used, but typically misclassification rate is used.

☛ Recall for **regression tree**, the overall RSS (i.e. the overall node impurity) will *always* drop by further splitting the tree. (and hence we don't want to keep growing the tree and overfit the data!).

☛ For **classification**, let R_ℓ be a region with N_ℓ points and impurity Q_ℓ.

☛ Suppose we split R_ℓ into two sub-regions $R_{\ell a}$ and $R_{\ell a}$, with $\{N_{\ell a}, Q_{\ell a}\}$ and $\{N_{\ell b}, Q_{\ell b}\}$ being the number of data and impurity of the two sub-regions.

☛ Then it must be that

$$N_\ell Q_\ell \geq N_{\ell a} Q_{\ell a} + N_{\ell b} Q_{\ell b},$$

when Q_ℓ is one of misclassification rate, Gini and Entropy i.e. we always reduce the overall node impurities by splitting, just like for regression trees grown with RSS.

☛ Jensen's inequality: Suppose f is a real-valued *concave* function, then for any x, y in the domain of f and $t \in [0, 1]$, it must be true that

$$f(tx + (1-t)y) \geq tf(x) + (1-t)f(y).$$

☛ "Proof" for $K = 2$: All the three impurity measures are **concave**. By Jensen's inequality, we have

$$Q_\ell(\hat{p}_\ell) \geq \frac{N_{\ell a}}{N_\ell} Q_{\ell a}(\hat{p}_{\ell a}) + \frac{N_{\ell b}}{N_\ell} Q_{\ell b}(\hat{p}_{\ell b}),$$

where $\hat{p}_\ell$, $\hat{p}_{\ell a}$, $\hat{p}_{\ell b}$ are the proportions of class 2 in the three regions. Note that Q_ℓ, $Q_{\ell a}$ and $Q_{\ell b}$ are functions in $\hat{p}_\ell$, $\hat{p}_{\ell a}$ and $\hat{p}_{\ell b}$ respectively. Moreover,

$$\frac{N_{\ell a}}{N_\ell} \hat{p}_{\ell a} + \frac{N_{\ell b}}{N_\ell} \hat{p}_{\ell b} = \hat{p}_\ell$$

and

$$N_{\ell a} + N_{\ell b} = N_\ell.$$

☛ Proof for a general K: Essentially the same, simply use Jensen's inequality for functions on higher dimensional domains.

1. Start with all the data and consider all possible splits of the data in two regions

$$R_1(j,t) = \{\mathcal{X}_i \text{ s.t. } \mathcal{X}_{ij} \leq t\}, \quad R_2(j,t) = \{\mathcal{X}_i \text{ s.t. } \mathcal{X}_{ij} > t\}$$

for all values of $j = 1, \ldots, p$ and all values of t that X_j takes in the training sample, i.e. $t \in \{\mathcal{X}_{1j}, \ldots, \mathcal{X}_{nj}\}$.

2. To create the first tree T with two regions R_1 and R_2, choose the splitting variable X_j and the split point t that minimise, over j and t,

$$\sum_{\ell=1}^{2} N_\ell Q_\ell$$

with Q_ℓ either the entropy or the Gini index.

3. Once we have found the best j and t defined above, our first split is determined. It creates two regions $R_1(j,t)$ and $R_2(j,t)$ of data.

4. Then, on each of those two regions we repeat the same procedure:

   ☛ Consider all possible splitting of R_1 in two regions, say R_1' and R_2' determined by a choice of j and t that minimises

   $$\sum_{\ell=1}^{2} N_\ell Q_\ell$$

   with Q_1 and Q_2 either the entropy or the Gini index evaluated for the regions R_1' and R_2'.

   ☛ Consider all possible splitting of R_2 in two regions, say R_3' and R_4', determined by a choice of j and t that minimises

   $$\sum_{\ell=3}^{4} N_\ell Q_\ell$$

   with Q_3 and Q_4 evaluated for the regions R_3' and R_4'.

5. Keep repeating the splitting procedure on each new region created in the previous step until we stop our tree.

☛ Then how do we prune our tree in classification? As for regression trees, we start by growing a large tree T_0 using the procedure above, which is such that each region contains only a small predetermined number of observations (say 5).

☛ Here, the cost complexity criterion is defined by

$$C_\alpha(T) = \sum_{\ell=1}^{|T|} N_\ell Q_\ell(T) + \alpha|T|$$

and in this case, to prune the tree we take Q_ℓ to be any of the three criteria defined earlier, but the misclassification rate is the one that is the most commonly used.

☛ To find T_α we use the **weakest link pruning**, as for the regression trees.

☛ α is chosen by cross-validation.

8.1.4 EXAMPLE: SPAM DATA (SECTION 9.2.5, ESL)

☛ "Spam" dataset: information from 4602 emails.

☛ Goal: predict whether the email was spam or true email.

☛ The true classes (`email` or `spam`) of all 4602 emails are available.

☛ For each email message, we also have the relative frequencies of 57 of the most commonly occurring words and punctuation marks in the email message. The table below lists the words and characters showing the largest average difference between `spam` and `email`.

TABLE 1.1. *Average percentage of words or characters in an email message equal to the indicated word or character. We have chosen the words and characters showing the largest difference between* `spam` *and* `email`.

	george	you	your	hp	free	hpl	!	our	re	edu	remove
spam	0.00	2.26	1.38	0.02	0.52	0.01	0.51	0.51	0.13	0.01	0.28
email	1.27	1.27	0.44	0.90	0.07	0.43	0.11	0.18	0.42	0.29	0.01

☛ Among the 4602 emails, 1536 of them are singled out as test set which isn't touched at all during the training process.

☛ There are 57 predictors, among which are

- 48 quantitative predictors: the $\%$ of words in the email that match a given word such as `business`, `address`, etc.
- 6 quantitative predictors: the $\%$ of characters in the email that match a given character; the characters are "`;`", "`(`", "`[`", "`!`", "`$`", "`#`".
- Refer to p.301 in ESL for the other predictors.

☛ Deviance is used to grow the tree and misclassification rate to prune the tree.

☛ Compute 10-fold $CV(\alpha)$ for a grid of α values in $[0, 180]$. For each α, $CV(\alpha)$ as a function of the size $|T_\alpha|$ of the pruned *original* tree is shown in blue; recall that as α decreases, tree size increases.

☛ Test error is shown in orange.

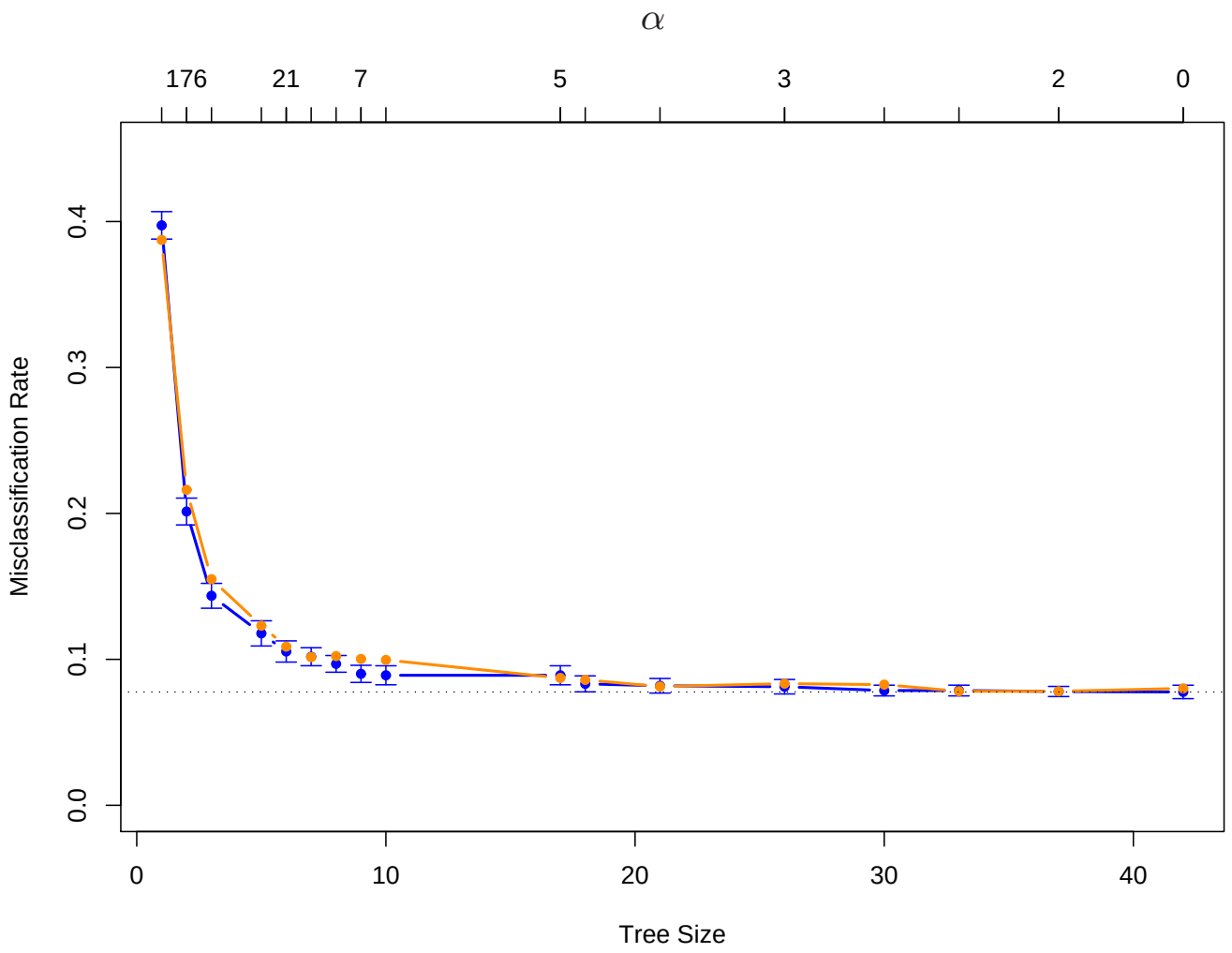


FIGURE 9.4. *Results for* `spam` *example. The blue curve is the* 10*-fold cross-validation estimate of misclassification rate as a function of tree size, with standard error bars. The minimum occurs at a tree size with about* 17 *terminal nodes (using the "one-standard-error" rule). The orange curve is the test error, which tracks the CV error quite closely. The cross-validation is indexed by values of* α*, shown above. The tree sizes shown below refer to* $|T_\alpha|$*, the size of the original tree indexed by* α*.*

☛ We are doing 10-fold cross validation here; so each error bar on the blue curve corresponds to ± 2 standard error from the mean among the 10 replications.

☛ Note that $CV(\alpha)$ is indexed by α but *not* the tree size; for trees grown in different folds, a value of α might imply different sizes.

☛ Alternatively to choosing $|T_\alpha|$ that minimizes $CV(\alpha)$, one can also use the "one-standard-error" rule (p.244 in ESL) to encourage picking a more parsimonious (i.e. smaller-tree-size) model:

This rule says one should choose the most parsimonious model whose $CV(\alpha)$ is no more than 1 standard error above the model that actually minimizes $CV(\alpha)$.

☛ With this rule $|T_\alpha| = 17$ is picked. (Note: This is an errata in the book. Professor Hastie has replied that based on the 1-se-rule, the tree with 21 nodes would have been picked. The 17-node tree is chosen since the curve flattens out around 17 terminal nodes.)

☛ In this example we know the true classes (`email` or `spam`) and so we can compute how many spams and emails have been correctly and wrongly classified as spam or email in the `test data`:

TABLE 9.3. *Spam data: confusion rates for the* 17*-node tree (chosen by cross–validation) on the test data. Overall error rate is* 9.3%*.*

	Predicted	
True	`email`	`spam`
`email`	57.3%	4.0%
`spam`	5.3%	33.4%

☛ 57% emails were classified correctly as emails, and 33.4% spam as spam. The rest was misclassified but the error rate is reasonably small.

☛ It is more annoying to classify email as spam than the other way round. It is possible to use more a modified classifier that puts more weight on misclassifying email than spam. See literature if interested.

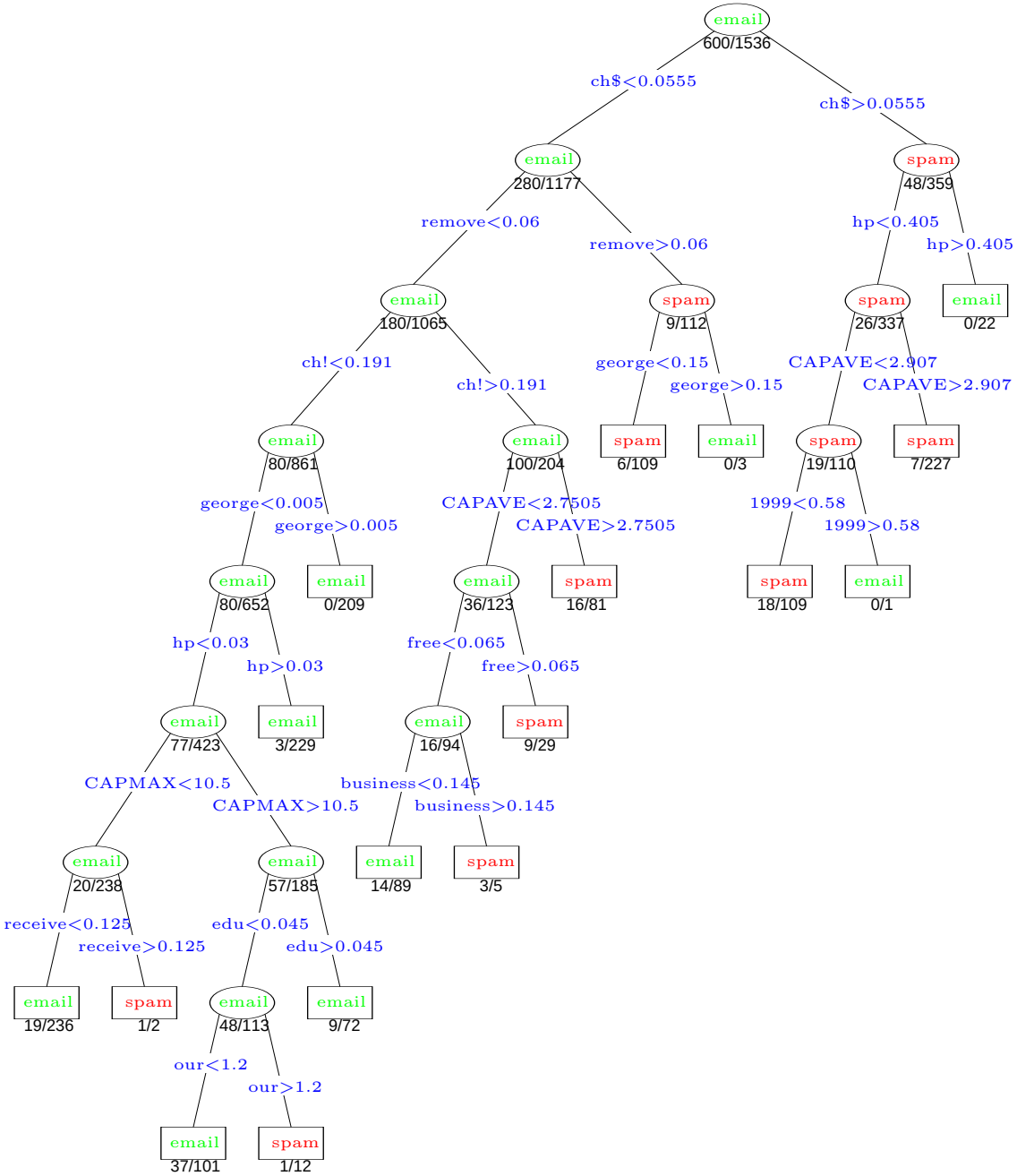


FIGURE 9.5. *The pruned tree for the* `spam` *example. The split variables are shown in blue on the branches, and the classification is shown in every node. The numbers under the terminal nodes indicate misclassification rates on the test data.*