

Classification Trees

Lecturer: Rob Nowak

Scribe: Rui Castro

1 Minimum Complexity Penalized Function

Recall the basic results of the last lectures: let \mathcal{X} and \mathcal{Y} denote the input and output spaces respectively. Let $X \in \mathcal{X}$ and $Y \in \mathcal{Y}$ be random variables with unknown joint probability distribution P_{XY} . We would like to use X to “predict” Y . Consider a loss function $0 \leq \ell(y_1, y_2) \leq 1$, $\forall y_1, y_2 \in \mathcal{Y}$. This function is used to measure the accuracy of our prediction. Let \mathcal{F} be a collection of candidate functions (models), $f : \mathcal{X} \rightarrow \mathcal{Y}$. The expected risk we incur is given by $R(f) \equiv E_{XY}[\ell(f(X), Y)]$. We have access only to a number of i.i.d. samples, $\{X_i, Y_i\}_{i=1}^n$. These allow us to compute the empirical risk $\widehat{R}_n(f) \equiv \frac{1}{n} \sum_{i=1}^n \ell(f(X_i), Y_i)$.

Assume in the following that \mathcal{F} is countable. Assign a positive number $c(f)$ to each $f \in \mathcal{F}$ such that $\sum_{f \in \mathcal{F}} 2^{-c(f)} \leq 1$. If we use a prefix code to describe each element of \mathcal{F} and define $c(f)$ to be the codeword length (in bits) for each $f \in \mathcal{F}$, the last inequality is automatically satisfied.

We define the *minimum complexity penalized estimator* as

$$\widehat{f}_n \equiv \arg \min_{f \in \mathcal{F}} \left\{ \widehat{R}_n(f) + \sqrt{\frac{c(f) \log 2 + \frac{1}{2} \log n}{2n}} \right\}.$$

As we showed previously we have the bound

$$E[R(\widehat{f}_n)] \leq \min_{f \in \mathcal{F}} \left\{ R(f) + \sqrt{\frac{c(f) \log 2 + \frac{1}{2} \log n}{2n}} + \frac{1}{\sqrt{n}} \right\}.$$

The performance (risk) of \widehat{f}_n is on average better than

$$R(f_n^*) + \sqrt{\frac{c(f_n^*) \log 2 + \frac{1}{2} \log n}{2n}} + \frac{1}{\sqrt{n}},$$

where

$$f_n^* = \arg \min_{f \in \mathcal{F}} \left\{ R(f) + \sqrt{\frac{c(f) \log 2 + \frac{1}{2} \log n}{2n}} \right\}.$$

If it happens that the optimal function, that is

$$f^* = \arg \min_{f \text{ measurable}} R(f),$$

is close to an $f \in \mathcal{F}$ with a small $c(f)$, then \widehat{f}_n will perform almost as well as the optimal function.

Example 1 Suppose $f^* \in \mathcal{F}$, then

$$E[R(\widehat{f}_n)] \leq R(f^*) + \sqrt{\frac{c(f^*) \log 2 + \frac{1}{2} \log n}{2n}} + \frac{1}{\sqrt{n}}.$$

Furthermore if $c(f^*) = O(\log n)$ then

$$E[R(\hat{f}_n)] \leq R(f^*) + O\left(\sqrt{\frac{\log n}{n}}\right),$$

that is, only within a small offset ($O\left(\sqrt{\frac{\log n}{n}}\right)$) of the optimal risk.

2 Classification

Consider the particularization of the above to a classification scenario. Let $\mathcal{X} = [0, 1]^d$, $\mathcal{Y} = \{0, 1\}$ and $\ell(\hat{y}, y) \equiv \mathbf{1}_{\{\hat{y} \neq y\}}$. Then $R(f) = E_{XY}[\mathbf{1}_{\{f(X) \neq Y\}}] = P(f(X) \neq Y)$. The Bayes risk is given by

$$R^* = \inf_{f \text{ measurable}} R(f).$$

As it was observed before, the Bayes classifier (*i.e.*, a classifier that achieves the Bayes risk) is given by

$$f^*(x) = \begin{cases} 1, & P(Y = 1|X = x) \geq \frac{1}{2} \\ 0, & P(Y = 1|X = x) < \frac{1}{2} \end{cases}.$$

This classifier can be expressed in a different way. Consider the set $G^* = \{P(Y = 1|X = x) \geq 1/2\}$. The Bayes classifier can be written as $f^*(x) = \mathbf{1}_{\{x \in G^*\}}$. Therefore the classifier is characterized entirely by the set G^* , if $X \in G^*$ then the “best” guess is that Y is one, and vice-versa. The boundary of this set corresponds to the points where the decision is harder. The boundary of G^* is called the *Bayes Decision Boundary*. In Figure 1(a) this concept is illustrated. If $\eta(x) = P(Y = 1|X = x)$ is a continuous function then the Bayes decision boundary is simply given by $\{x \in \mathcal{X} : P(Y = 1|X = x) = 1/2\}$. Clearly the structure of the decision boundary provides important information on the difficulty of the problem.

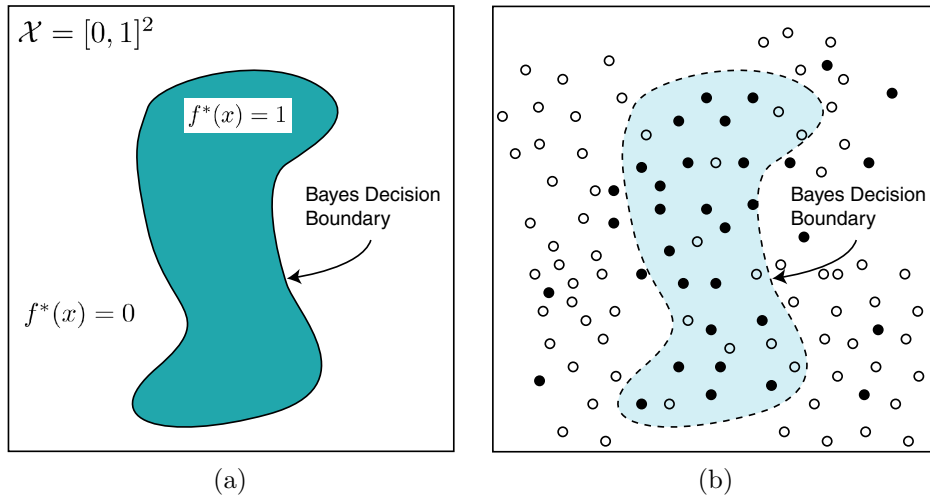


Figure 1: (a) The Bayes classifier and the Bayes decision boundary ; (b) Example of the i.i.d. training pairs.

2.1 Empirical Classifier Design

Given n i.i.d. training pairs, $\{X_i, Y_i\}_{i=1}^n$, we want to construct a classifier \hat{f}_n that performs well on average, *i.e.*, we want $E[R(\hat{f}_n)]$ as close to R^* as possible. In Figure 1(b) an example of the i.i.d. training pairs is depicted.

The construction of a classifier boils down to the estimation of the Bayes decision boundary. The histogram rule, discussed in a previous lecture, approaches the problem by subdividing the feature space into small boxes and taking a majority vote of the training data in each box. A typical result is depicted in Figure 2(a).

The main problem with the histogram rule is that it is solving a more complicated problem than it is actually necessary. We do not need to determine the correct label for each individual box directly (the histogram rule is essentially estimating $\eta(x)$). In principle we only need to locate the decision boundary and assign the correct label on either side (notice that the accuracy of a majority vote over a region increases with the size of the region). The next example illustrates this.

Example 2 (Three Different Classifiers) *The pictures below correspond to the approximation of the Bayes classifier by three different classifiers:*

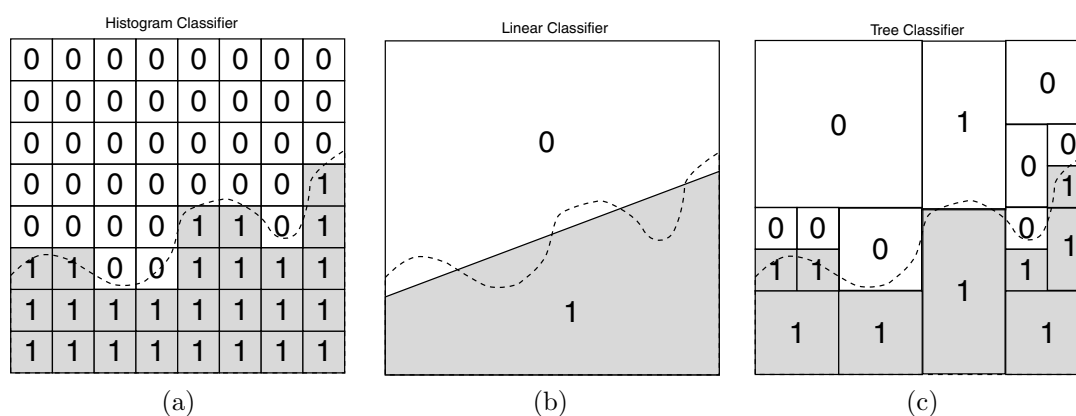


Figure 2: (a) Histogram classifier ; (b) Linear classifier; (c)Tree classifier.

The linear classifier and the tree classifier (to be defined formally later) both attack the problem of finding the boundary more directly than the histogram classifier, and therefore they tend to produce much better results in theory and practice. In the following we will demonstrate this for classification trees.

3 Binary Classification Trees

Binary classification trees are constructed by a two-step process:

1. Tree growing
2. Tree pruning

The basic idea is to first grow a very large, complicated tree classifier, that explains the the training data very accurately, but has poor generalization characteristics, and then prune this tree, to avoid overfitting.

3.1 Growing Trees

The growing process is based on recursively subdividing the feature space. Usually the subdivisions are splits of existing regions into two smaller regions (*i.e.*, binary splits) and usually the splits are perpendicular to one of the feature axes. An example of such construction is depicted in Figure 3.

Often the splitting process is designed to separate the training data with different labels as much as possible, therefore the “splits” are data dependent. Alternatively, the splitting and subdivision could be independent from the training data. The later approach is the one we are going to use in the following.

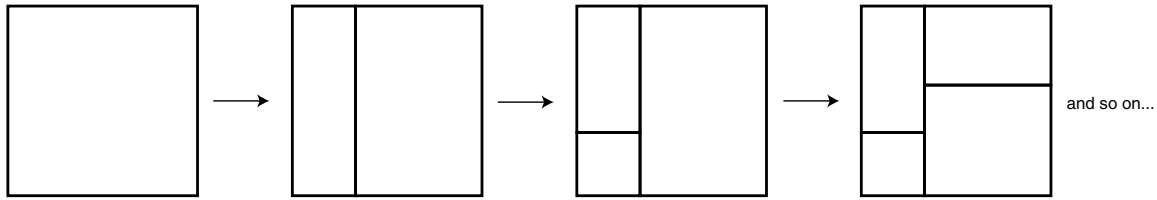


Figure 3: Growing a recursive binary tree ($\mathcal{X} = [0, 1]^2$).

Until now we have been referring to trees, but did not made clear how do trees relate to dyadic partitions. It turns out that any Recursive Dyadic Partition (RDP) can be associated with a (binary) tree. In fact, this is the most efficient way of describing a RDP. In Figure 4 we illustrate the procedure. Each leaf of the tree corresponds to an element of the partition. The nodes in the tree correspond to the various partition elements we go through in the construction of the tree. The orientation of the dyadic split alternates between the levels of the tree (for the example of Figure 4, at the root level the split is done in the horizontal axis, at the level below that (the level of nodes 2 and 3) the split is done in the vertical axis, and so on...).

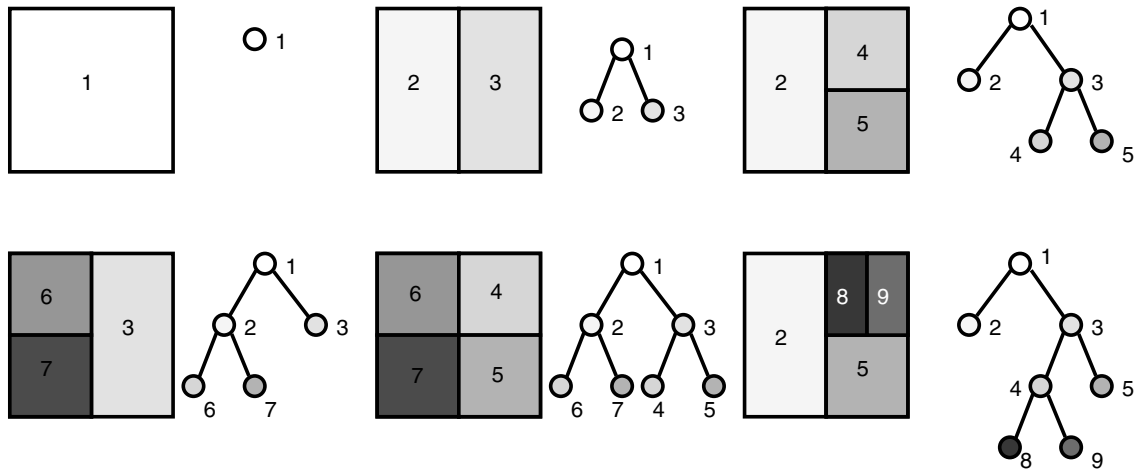


Figure 4: Example of Recursive Dyadic Partition (RDP) growing ($\mathcal{X} = [0, 1]^2$).

In the following we are going to consider the 2-dimensional case, but all the results can be easily generalized for the d -dimensional case ($d \geq 2$), provided the dyadic tree construction is defined properly.

Consider a recursive dyadic partition of the feature space into k boxes of equal size. Associated with this partition there is a tree T . Minimizing the empirical risk with respect to this partition produces the histogram classifier with k equal-sized bins. Consider also all the possible partitions corresponding to pruned versions of the tree T . Minimizing the empirical risk with respect to those other partitions results in other classifiers (classification trees) that are fundamentally different than the histogram rule we analyzed earlier.

3.2 Pruning

Let \mathcal{F} be the collection of all possible classification trees corresponding to recursive dyadic partitions of the feature space. Each such tree can be prefix encoded with a bit-string proportional to the number of leafs in the tree as follows; encode the structure of the tree in a top-down fashion: (i) assign a zero at each branch node and a one at each leaf node (terminal node) (ii) read the code in a breadth-first fashion, top-down, left-right. Figure 5 exemplifies this coding strategy. Notice that, since we are considering binary trees, the

total number of nodes is twice the number of leaves minus one, that is, if the number of leaves in the tree is k then the number of nodes is $2k - 1$. Therefore to encode a tree with k leaves we need $2k - 1$ bits.

Since we want to use the partition associated with this tree for classification we need to assign a decision label (either zero or one) to each leaf. Hence, to encode a decision tree in this fashion we need $3k - 1$ bits, where k is the number of leaves. For a tree with k leaves the first $2k - 1$ bits of the codeword encode the tree structure, and the remaining k bits encode the classification labels. This is easily shown to be a prefix code (why?), therefore we can use this under our classification scenario.

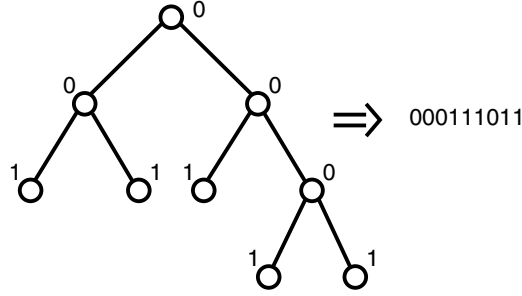


Figure 5: Illustration of the tree coding technique: example of a tree and corresponding prefix code.

Let

$$\hat{f}_n^* = \arg \min_{f \in \mathcal{F}} \left\{ \hat{R}_n(f) + \sqrt{\frac{(3k-1) \log 2 + \frac{1}{2} \log n}{2n}} \right\}.$$

This optimization can be solved through a bottom-up pruning process (starting from a very large initial tree T_0) in $O(|T_0|^2)$ operations, where $|T_0|$ is the number of leaves in the initial tree. The complexity regularization theorem tells us that

$$E[R(\hat{f}_n)] \leq \min_{f \in \mathcal{F}} \left\{ R(f) + \sqrt{\frac{(3k-1) \log 2 + \frac{1}{2} \log n}{2n}} \right\} + \frac{1}{\sqrt{n}}. \quad (1)$$

4 Comparison between Histogram Classifiers and Classification Trees

In the following we will illustrate the idea behind complexity regularization by applying the basic theorem to histogram classifiers and classification trees (using our setup above).

Consider the classification setup described in Section 2, with $\mathcal{X} = [0, 1]^2$.

4.1 Histogram Risk Bound

Recall the setup and results of a previous lecture¹. Let

$$\mathcal{F}_k^H = \{\text{histogram rules with } k^2 \text{ bins}\}.$$

Then $|\mathcal{F}_k^H| = 2^{k^2}$. Let $\mathcal{F}^H = \bigcup_{k \geq 1} \mathcal{F}_k^H$. We can encode each element f of \mathcal{F}^H with $c_H(f) = k + k^2$ bits, where the first k bits indicate the smallest k such that $f \in \mathcal{F}_k^H$ and the following k^2 bits encode the labels of each bin. This is a prefix encoding of all the elements in \mathcal{F}^H .

We define our estimator as

$$\hat{f}_n^H = \hat{f}_n^k,$$

¹The description here is slightly different than the one in the previous lecture.

where

$$\widehat{f}_n^{(k)} = \arg \min_{f \in \mathcal{F}_k^H} \widehat{R}_n(f),$$

and

$$\widehat{k} = \arg \min_{k \geq 1} \left\{ \widehat{R}_n(\widehat{f}_n^{(k)}) + \sqrt{\frac{(k + k^2) \log 2 + \frac{1}{2} \log n}{2n}} \right\}.$$

Therefore \widehat{f}_n^H minimizes

$$\widehat{R}_n(f) + \sqrt{\frac{c_H(f) \log 2 + \frac{1}{2} \log n}{2n}},$$

over all $f \in \mathcal{F}^H$. We showed before that

$$E[R(\widehat{f}_n^H)] \leq \min_{f \in \mathcal{F}^H} \left\{ R(f) + \sqrt{\frac{c_H(f) \log 2 + \frac{1}{2} \log n}{2n}} \right\} + \frac{1}{\sqrt{n}}.$$

To proceed with our analysis we need to make some assumptions on the intrinsic difficulty of the problem. We will assume that the Bayes decision boundary is a “well-behaved” 1-dimensional set, in the sense that it has box-counting dimension one (see Appendix A). This implies that, for an histogram with k^2 bins, the Bayes decision boundary intersects less than Ck bins, where C is a constant that does not depend on k . Furthermore we assume that the marginal distribution of X satisfies $P_X(A) \leq K|A|$, for any measurable subset $A \subseteq [0, 1]^2$. This means that the samples collected do not accumulate anywhere in the unit square.

Under the above assumptions we can conclude that

$$\min_{f \in \mathcal{F}_k^H} R(f) \leq \frac{K}{k^2} Ck = \frac{CK}{k}.$$

Therefore

$$E[R(\widehat{f}_n^H)] \leq CK/k + \sqrt{\frac{(k + k^2) \log 2 + \frac{1}{2} \log n}{2n}} + \frac{1}{\sqrt{n}}.$$

We can balance the terms in the right side of the above expression using $k = n^{1/4}$ (for n large) therefore

$$E[R(\widehat{f}_n^H)] = O(n^{-1/4}), \quad \text{as } n \rightarrow \infty.$$

4.2 Tree Classifiers

Now let's consider the tree classifiers, under the scenario above, and contrast these with the histogram classifier. Let

$$\mathcal{F}_k^T = \{\text{tree classifiers with } k \text{ leaves}\}.$$

Let $\mathcal{F}^T = \bigcup_{k \geq 1} \mathcal{F}_k^T$. We can prefix encode each element f of \mathcal{F}^T with $c_T(f) = 3k - 1$ bits, as described before.

Let

$$\widehat{f}_n^T = \widehat{f}_n^k,$$

where

$$\widehat{f}_n^{(k)} = \arg \min_{f \in \mathcal{F}_k^T} \widehat{R}_n(f),$$

and

$$\widehat{k} = \arg \min_{k \geq 1} \left\{ \widehat{R}_n(\widehat{f}_n^{(k)}) + \sqrt{\frac{(3k - 1) \log 2 + \frac{1}{2} \log n}{2n}} \right\}.$$

Hence \hat{f}_n^T minimizes

$$\hat{R}_n(f) + \sqrt{\frac{c_T(f) \log 2 + \frac{1}{2} \log n}{2n}},$$

over all $f \in \mathcal{F}^T$. Moreover

$$E[R(\hat{f}_n^T)] \leq \min_{f \in \mathcal{F}^T} \left\{ R(f) + \sqrt{\frac{c_T(f) \log 2 + \frac{1}{2} \log n}{2n}} \right\} + \frac{1}{\sqrt{n}}.$$

If the Bayes decision boundary is a 1-dimensional set, as in Section 4.1, there exists a tree with at most $8Ck$ leafs such that the boundary is contained in at most Ck squares, each of volume $1/k^2$. To see this, start with a tree yielding the histogram partition with k^2 boxes (*i.e.*, the tree partitioning the unit square into k^2 equal sized squares). Now prune all the nodes that do not intersect the boundary. In Figure 6 we illustrate the procedure. If you carefully bound the number of leafs you need at each level you can show that you will have in total less than $8Ck$ leafs. We conclude then that there exists a tree with at most $8Ck$ leafs that has the same risk as a histogram with $O(k^2)$ bins. Therefore, using equation (1) we have

$$E[R(\hat{f}_n^T)] \leq CK/k + \sqrt{\frac{(3(8Ck) - 1) \log 2 + \frac{1}{2} \log n}{2n}} + \frac{1}{\sqrt{n}}.$$

We can balance the terms in the right side of the above expression using $k = n^{1/3}$ (for n large) therefore

$$E[R(\hat{f}_n^T)] = O(n^{-1/3}), \text{ as } n \rightarrow \infty.$$

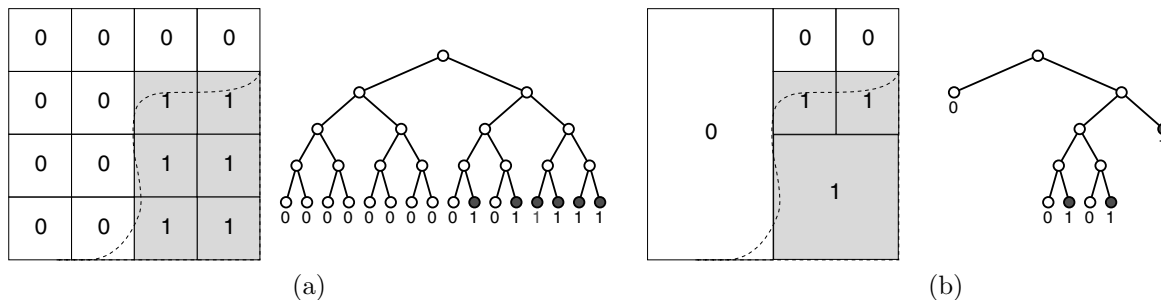


Figure 6: Illustration of the tree pruning procedure: (a) Histogram classification rule, for a partition with 16 bins, and corresponding binary tree representation (with 16 leafs). (b) Pruned version of the histogram tree, yielding exactly the same classification rule, but now requiring only 6 leafs. (*Note:* The trees were constructed using the procedure of Figure 4)

5 Final Comments

Trees generally work much better than histogram classifiers. This is essentially because they provide much more efficient ways of describing a classifier (as we saw in our example, under reasonable assumptions on the Bayes boundary, a tree encoded with $O(k)$ bits can describe the same classifier as an histogram that requires $O(k^2)$ bits).

The tree classifiers studied here are different than classical tree rules, such as CART (Breiman et al., 1984) or C4.5 (Quinlan, 1993). Those techniques select a tree according to

$$\hat{k} = \arg \min_{k \geq 1} \left\{ \hat{R}_n(\hat{f}_n^{(k)}) + \alpha k \right\},$$

for some $\alpha > 0$ whereas ours was roughly

$$\hat{k} = \arg \min_{k \geq 1} \left\{ \widehat{R}_n(\widehat{f}_n^{(k)}) + \alpha \sqrt{k} \right\},$$

for $\alpha \approx \sqrt{\frac{3 \log 2}{2n}}$. The square root penalty is essential for the risk bound. No such bound exists for CART or C4.5. Moreover, recent experimental work has shown that the square root penalty often performs better in practice.

A Box Counting Dimension

The notion of dimension of a sets arises in many aspects of mathematics, and it is particularly relevant to the study of fractals (that besides some important applications make really cool t-shirts). The dimension somehow indicates how we should measure the contents of a set (length, area, volume, etc...). The box-counting dimension is a simple definition of the dimension of a set. The main idea is to cover the set with boxes with sidelength r . Let $N(r)$ denote the smallest number of such boxes, then the box counting dimension is defined as

$$\lim_{r \rightarrow 0} \frac{\log N(r)}{-\log r}.$$

Although the boxes considered above do not need to be aligned on a rectangular grid (and can in fact overlap) we can usually consider them over a grid and obtain an upper bound on the box-counting dimension. To illustrate the main ideas let's consider a simple example, and connect it to the classification scenario considered before.

Let $f : [0, 1] \rightarrow [0, 1]$ be a Lipschitz function, with Lipschitz constant L (i.e., $|f(a) - f(b)| \leq L|a - b|$, $\forall a, b \in [0, 1]$). Define the set

$$A = \{x = (x_1, x_2) : x_2 = f(x_1)\},$$

that is, the set A is the graphic of function f .

Consider a partition with k^2 squared boxes (just like the ones we used in the histograms), the points in set A intersect at most $C'k$ boxes, with $C' = (1 + \lceil L \rceil)$ (and also the number of intersected boxes is greater than k). The sidelength of the boxes is $1/k$ therefore the box-counting dimension of A satisfies

$$\begin{aligned} \dim_B(A) &\leq \lim_{1/k \rightarrow 0} \frac{\log C'k}{-\log(1/k)} \\ &= \lim_{k \rightarrow \infty} \frac{\log C' + \log(k)}{\log(k)} \\ &= 1. \end{aligned}$$

The result above will hold for any “normal” set $A \subseteq [0, 1]^2$ that does not occupy any area. For most sets the box-counting dimension is always going to be an integer, but for some “weird” sets (called fractal sets) it is not an integer. For example, the Koch curve (see for example <http://classes.yale.edu/fractals/IntroToFrac/InitGen/InitGenKoch.html>) has box-counting dimension $\log(4)/\log(3) = 1.26186\dots$. This means that it is not quite as small as a 1-dimensional curve, but not as big as a 2-dimensional set (hence occupies no area).

To connect these concepts to our classification scenario consider a simple example. Let $\eta(x) = P(Y = 1|X = x)$ and assume $\eta(x)$ has the form

$$\eta(x) = \frac{1}{2} + x_2 - f(x_1), \quad \forall x \equiv (x_1, x_2) \in \mathcal{X}, \quad (2)$$

where $f : [0, 1] \rightarrow [0, 1]$ is Lipschitz with Lipschitz constant L . The Bayes classifier is then given by

$$f^*(x) = \mathbf{1}_{\{\eta(x) \geq 1/2\}} \equiv \mathbf{1}_{\{x_2 \geq f(x_1)\}}.$$

This is depicted in Figure 7. Note that this is a special, restricted class of problems. That is, we are considering the subset of all classification problems such that the joint distribution P_{XY} satisfies $P(Y = 1|X = x) = 1/2 + x_2 - f(x_1)$ for some function f that is Lipschitz. The Bayes decision boundary is therefore given by

$$A = \{x = (x_1, x_2) : x_2 = f(x_1)\}.$$

As we observed before this set has box-counting dimension 1.

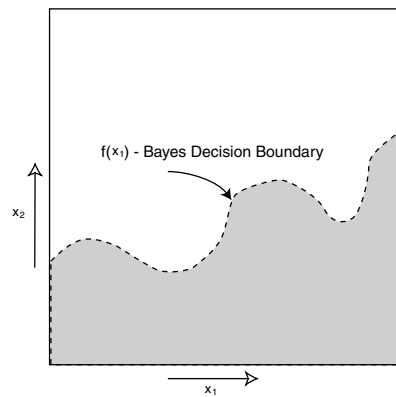


Figure 7: Bayes decision boundary for the setup described in Appendix A.