CS 760: Machine Learning Fall 2020

Topic 8: Decision Trees

Instructor: Daniel L. Pimentel-Alarcón

GO GREEN. AVOID PRINTING, OR PRINT 2-SIDED OR MULTIPAGE.

8.1 Introduction

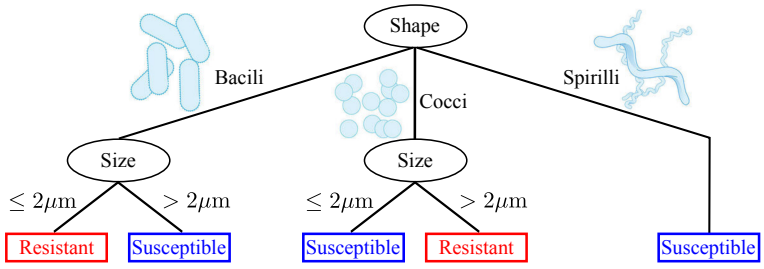
Decision trees are one of the most widely used machine learning algorithms for prediction and classification. They aim to address questions such as:

- Will I get Covid?
- Will I develop diabetes?
- Will this bacteria develop antibiotic resistance?
- What virus is this?

8.2 Setup

Suppose you have a data matrix $\mathbf{X} \in \mathbb{R}^{D\times N}$ containing D *features* about a collection of N bacteria, such as size, shape, and whether specific genes are active or inactive. Also suppose you have a vector $\mathbf{y} \in \mathbb{R}^N$ containing the *response* of the N bacteria in your sample, for example, whether they are resistant to penicillin. Given a new bacteria with feature vector $\mathbf{x} \in \mathbb{R}^D$, our goal is to predict its *response* y indicating whether it is resistant to penicillin.

As the name suggests, *decision trees* use trees to predict y. Each internal node in the tree tests a feature x_i, each branch from an internal node represents an outcome (either a nominal feature or a range), and each leaf predicts y:



Based on Occam's razor (when you have two competing theories that make exactly the same predictions, the simpler one is the better), to minimize overfitting we would like to find the *simplest* tree that correctly

predicts y for the training data. This in turn depends on selecting the correct features for each split. The problem is that finding *the smallest* possible decision tree that fits the training data is an NP-hard problem [1]. Hence, to narrow our search we will use an information-theoretic heuristic to choose the splits in a greedy fashion.

8.3 Information Theory

As the name suggests, information theory deals with quantifying information in data, efficient ways to store it, and reliable ways to communicate it. One of its most important concepts is *entropy*, which measures the amount of information in a random variable. For a discrete random variable x with support on a set $\mathcal{X}$, entropy is defined as follows:

$$H(x) \;:=\; \mathbb{E}\left[\log_2\left(\frac{1}{\mathbb{P}(x)}\right)\right] \;=\; \sum_{\mathrm{x}\in\mathcal{X}} \mathbb{P}(x=\mathrm{x})\log_2\left(\frac{1}{\mathbb{P}(x=\mathrm{x})}\right).$$

Intuitively, $\log_2\left(\frac{1}{\mathbb{P}(x=\mathrm{x})}\right)$ quantifies the number of bits that one should spend encoding outcome x: the higher the probability of x, the more frequent we will see it, and so the fewer bits we should spend on it; the lower the probability of x the less likely it will appear, and so we can spend more bits on it.

To build some intuition, consider the following example from [2], where 8 horses race with the following odds:

Horse	1	2	3	4	5	6	7	8
$\mathbb{P}$(winning)	1/2	1/4	1/8	1/16	1/64	1/64	1/64	1/64

Suppose you want to keep track of which horse wins each race. You can use the following code, which has an average length of 3 bits per race:

Horse	1	2	3	4	5	6	7	8
Code	000	001	010	011	100	101	110	111

However, as entropy suggests, if you assign each horse (outcome x) a code of length $\log_2\left(\frac{1}{\mathbb{P}(x=\mathrm{x})}\right)$, such as the following, then you can achieve an average length of 2 bits per race:

Horse	1	2	3	4	5	6	7	8
Code	0	10	110	1110	111100	111101	111110	111111

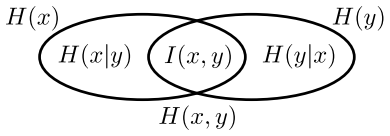
To summarize, entropy quantifies the amount of information in a variable. In decision trees we are interested in the information of each feature x_{i} relative to the response y. To assess this, we can further consider the *conditional entropy*:

$$\begin{aligned} H(x|y) &:= \mathbb{E}\left[\log_2\left(\frac{1}{\mathbb{P}(x|y)}\right)\right] := \sum_{\mathrm{x}\in\mathcal{X}}\sum_{\mathrm{y}\in\mathcal{Y}} \mathbb{P}(x=\mathrm{x}|y=\mathrm{y})\log_2\left(\frac{1}{\mathbb{P}(x=\mathrm{x}|y=\mathrm{y})}\right) \\ &= \sum_{\mathrm{y}\in\mathcal{Y}} \mathbb{P}(y=\mathrm{y}) \sum_{\mathrm{x}\in\mathcal{X}} \mathbb{P}(x=\mathrm{x}|y=\mathrm{y})\log_2\left(\frac{1}{\mathbb{P}(x=\mathrm{x}|y=\mathrm{y})}\right) \\ &=: \sum_{\mathrm{y}\in\mathcal{Y}} \mathbb{P}(y=\mathrm{y}) H(x|y=\mathrm{y}). \end{aligned}$$

Intuitively, $H(x|y)$ quantifies the amount of information contained in x once y is known. Similarly, we can define the *mutual information*, measuring the information that one variable contains about another:

$$I(x,y) := H(x) - H(x|y).$$

The following diagram depicts the relationship between entropy and mutual information:



In a nutshell, decision trees split data iteratively, according to the most informative features with respect to y (highest mutual information), and predict according to this split.

8.4 Decision Tree Recipe

Suppose we have the following data indicating active genes of 16 bacteria, and whether they are antibiotic resistant:

Bacteria	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
x_1 = Gene 1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
x_2 = Gene 2	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
x_3 = Gene 3	0	1	1	0	0	0	1	1	1	1	0	0	0	1	0	1
y = **Resistant**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**

To construct a decision tree that predicts y, we follow the next steps:

Step 1: Compute Mutual Information

First compute the entropy for each feature. For example, for Gene 1:

$$\begin{aligned} H(x_1) &= \sum_{\mathrm{x}\in\mathcal{X}} \mathbb{P}(x_1 = \mathrm{x}) \log_2 \left(\frac{1}{\mathbb{P}(x_1 = \mathrm{x})} \right) \\ &= \mathbb{P}(x_1 = 0) \log_2 \left(\frac{1}{\mathbb{P}(x_1 = 0)} \right) + \mathbb{P}(x_1 = 1) \log_2 \left(\frac{1}{\mathbb{P}(x_1 = 1)} \right) \\ &= \frac{1}{16} \log_2(16) + \frac{15}{16} \log_2(\frac{16}{15}) \\ &= 0.3373. \end{aligned}$$

Next we compute conditional entropy on y:

$$\begin{aligned} H(x_1|y) &= \sum_{\mathrm{x}\in\mathcal{X}} \sum_{\mathrm{y}\in\mathcal{Y}} \mathbb{P}(x_1 = \mathrm{x}, y = \mathrm{y}) \log_2 \left(\frac{1}{\mathbb{P}(x_1 = \mathrm{x}|y = \mathrm{y})} \right) \\ &= \mathbb{P}(x_1 = 0, y = 0) \log_2 \left(\frac{1}{\mathbb{P}(x_1 = 0|y = 0)} \right) + \mathbb{P}(x_1 = 0, y = 1) \log_2 \left(\frac{1}{\mathbb{P}(x_1 = 0|y = 1)} \right) \\ &\quad + \mathbb{P}(x_1 = 1, y = 0) \log_2 \left(\frac{1}{\mathbb{P}(x_1 = 1|y = 0)} \right) + \mathbb{P}(x_1 = 1, y = 1) \log_2 \left(\frac{1}{\mathbb{P}(x_1 = 1|y = 1)} \right) \\ &= 0 + 0.0625(1) + 0.875(0) + 0.0625(1) \\ &= 0.125 \end{aligned}$$

And finally the mutual information:

$$I(x_1, y) = H(x_1) - H(x_1|y) = 0.3373 - 0.125 = 0.2123.$$

Similarly, for the other genes we have:

$$\begin{array}{lll} H(x_2) = 0.3373 & H(x_2|y) = 0.125 & I(x_2, y) = 0.2123 \\ H(x_3) = 1 & H(x_3|y) = 1 & I(x_2, y) = 0. \end{array}$$

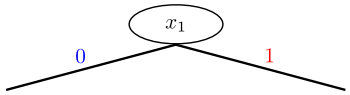
At this point we can identify the most informative feature relative to y (the one with highest mutual information). In this case there is a tie between x_1 and x_2, so we can pick randomly. Say we choose x_1.

Step 2: Split Data

We split our data according to the most informative feature relative to y (the one with highest mutual information), and make such feature a node in our decision tree. In our example we would split data according to x_1:

	Bacteria	14	1	2	3	4	5	6	7	8	9	10	11	12	13	15	16
→	x_1 = Gene 1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	x_2 = Gene 2	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
	x_3 = Gene 3	1	0	1	1	0	0	0	1	1	1	1	0	0	0	0	1
	y = **Resistant**	**1**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

and our tree would start to look as follows:



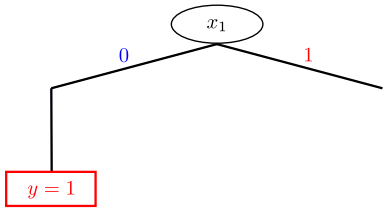
Step 3: Iterate on in Each Subset

A decision tree then recursively iterates Steps 1 (find most informative feature in each subset) and 2 (split data) until a *stopping criteria* is met, for example:

- The entropy of the response y in the current subset of data is zero (or close to zero), indicating y provides little or no information (i.e., it is determined) given the existing features.
- The current subset of data contains too few samples (e.g., less than 5% of the total data).

Whenever the stopping criteria is met for a subset of data, we simply add a leaf to the tree with the corresponding value of y.

For example, in our case, $H(y|x_1 = 0) = 0$. This implies that y provides no new information given $x_1 = 0$, which makes perfect sense, as given $x_1 = 0$, the only option for y is 0. At this point we can add a leaf to this branch of our tree:

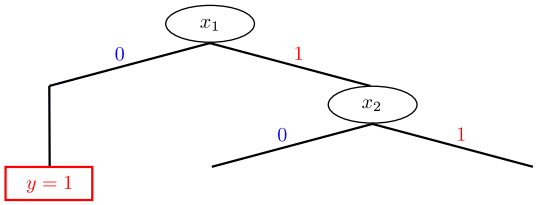


On the other hand, $H(y|x_1 = 1) = 0.3534$, which means y still contains additional information given $x_1 = 1$. Hence we need to add another split to our tree according to the most informative of the remaining features, restricted to this subset of the data. To identify such feature we proceed as in Step 1:

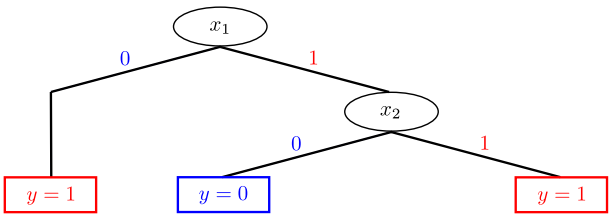
$$H(x_2|x_1 = 1) = 0.3534 \qquad H(x_2|y, x_1 = 1, y) = 0 \qquad I(x_2, y|x_1 = 1) = 0.3534$$
$$H(x_3|x_1 = 1) = 0.9968 \qquad H(x_3|y, x_1 = 1, y) = 0.9333 \qquad I(x_3, y|x_1 = 1) = 0.0635.$$

From this we conclude x_2 is the most informative feature given $x_1 = 1$, and so as in Step 2, we split our data accordingly, and add another node to our tree:

	Bacteria	1	2	3	4	5	7	8	9	10	11	12	13	15	16	6
→	x_2 = Gene 2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
	x_3 = Gene 3	0	1	1	0	0	1	1	1	1	0	0	0	0	1	0
	y = **Resistant**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**



At this point we can again see that $H(y|x_1 = 1, x_2 = 0) = H(y|x_1 = 1, x_2 = 1) = 0$, so we add leaves to the corresponding branches, and we are done:



8.5 Using a Decision Tree

Once we have a decision tree, we can follow it from the root to predict/classify new data. For instance, given the following data:

Bacteria	17	18	19	20
x_1 = Gene 1	0	1	1	0
x_2 = Gene 2	1	0	1	0
x_3 = Gene 3	0	1	0	1

we can use our tree to classify/predict whether these additional bacteria are antibiotic resistant:

y = **Resistant**	**1**	**0**	**1**	**1**

8.6 Caveats with Mutual Information

One limitation of mutual information is that it is biased towards tests with many outcomes. For example, a feature that uniquely identifies each sample instance. Splitting on this feature would result in many branches, each with samples from only one class, resulting in maximal mutual information.

One option to address this issue is to use the so-called *gain ratio* [3] as selection criteria, rather than mutual information (a.k.a. information gain):

$$G(x, y) \;:=\; \frac{I(x, y)}{H(x)}.$$

The main idea is to normalize the mutual information by the entropy of the split being considered.

8.7 Managing Overfitting

To avoid overfitting in decision trees there are two main strategies:

- **Early stopping.** As the name suggests, this involves no longer splitting/growing the tree unless it is justified by a statistical test, e.g., not high enough variance or entropy in y, or too few samples. One prominent example of this approach is the ID3 algorithm [4].

- **Pruning.** Another alternative is to grow a large tree, and then greedily remove leaves according to a cross-validation approach. More precisely, data is initially split into training and testing. Then a large tree is grown using the training data. Finally, each leaf is iteratively removed until the accuracy on the testing data no longer improves. This strategy is known to be more robust to *myopia*, that is, missing a feature that may not appear to be informative until it is used in conjunction with other features.

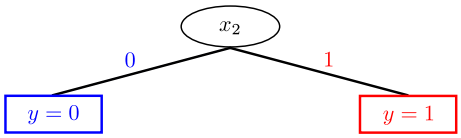
8.8 From Decision Trees to Random Forests

As you can imagine, decision trees can be very sensitive to the specific data we observe. In fact, they can be quite biased, and tend to overfit. One option to overcome these drawbacks is to extend decision trees to random forests, which are essentially a *bootstrap*, or *bagging* version of *decision trees*.

As the name suggests, the main idea of random forests is to generate many decision trees, each time starting with a random subset of the data, either in terms of samples, or features. In our example, we can obtain a random forest by select a random subset $\mathcal{S}_1 \subset [\mathrm{N}] := \{1, \ldots, \mathrm{N}\}$ of the samples in this table, e.g., $\mathcal{S}_1 = [\mathrm{N}] \backslash \{3, 10, 14, 16\}$:

Bacteria	1	2	4	5	6	7	8	9	11	12	13	15
x_1 = Gene 1	1	1	1	1	1	1	1	1	1	1	1	1
x_2 = Gene 2	0	0	0	0	1	0	0	0	0	0	0	0
x_3 = Gene 3	0	1	0	0	0	1	1	1	0	0	0	0
y = **Resistant**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

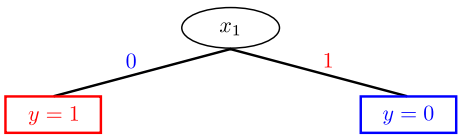
Using this subset of the data we build a decision tree (as described before), which we will denote as $\mathcal{T}_1$. We repeat this process T times, each with a different random subset of data $\mathcal{S}_\mathrm{t}$ to obtain a sequence of decision trees $\mathcal{T}_1, \ldots, \mathcal{T}_\mathrm{T}$, which together form a random forest. Notice that the decision trees are not necessarily going to be the same, because they are run on different subsets of the data. For instance, with $\mathcal{S}_1$ above, the most informative feature will be x_2, so instead of our tree from before, we will obtain a tree $\mathcal{T}_1$ with a single level:



In contrast, if $\mathcal{S}_2 = [\mathrm{N}] \backslash \{1, 3, 6, 15\}$, then our data looks as follows:

Bacteria	2	4	5	7	8	9	10	11	12	13	14	16
x_1 = Gene 1	1	1	1	1	1	1	1	1	1	1	0	1
x_2 = Gene 2	0	0	0	0	0	0	0	0	0	0	0	0
x_3 = Gene 3	1	0	0	1	1	1	1	0	0	0	1	1
y = **Resistant**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**

In this case the most informative variable will be x_1, and $\mathcal{T}_2$ will be:



Consequently, $\mathcal{T}_1$ will not be the same as $\mathcal{T}_2$. In fact, each tree may produce a different prediction/classification. In a random forest, the final prediction/classification is obtained by consensus over all the predictions/classifications given by all its decision trees. For example, if our random forest has 5 decision trees $\mathcal{T}_1, \ldots, \mathcal{T}_5$, classifying drug resistance of bacteria 17 as $\{1, 1, 1, 0, 1\}$, then we would conclude that bacteria 17 is drug resistant.

8.9 Regression Trees

So far our examples have mainly involved discrete variables. One can similarly define entropy and mutual information for continuous variables:

$$\begin{aligned} H(x) &:= -\int_{\mathcal{X}} \mathbb{P}(x) \log \mathbb{P}(x) dx, \\ H(x|y) &:= -\int_{\mathcal{X}} \int_{\mathcal{Y}} \mathbb{P}(x, y) \log \mathbb{P}(x|y) dx dy, \\ I(x, y) &:= H(x) - H(x|y), \end{aligned}$$

and we can use these quantities to build decision trees as before. However, in practice people often use a slightly different approach for continuous variables: minimizing a *loss* or error, in this case the *within-leaf sum of squared errors* of a tree, given by:

$$\ell(\mathcal{T}) := \sum_{\mathcal{L} \in \mathcal{T}} \sum_{\mathrm{i} \in \mathcal{L}} (\mathrm{y_i} - \mu_{\mathcal{L}})^2,$$

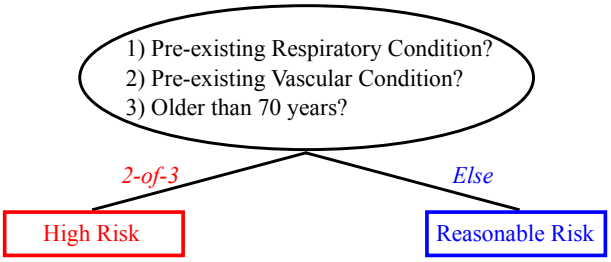
where $\mu_{\mathcal{L}} := \frac{1}{\mathrm{N}_{\mathcal{L}}} \sum_{\mathrm{i} \in \mathcal{L}} \mathrm{y_i}$ is the mean of the responses $\mathrm{y_i}$ in leaf $\mathcal{L}$ from tree $\mathcal{T}$. Notice that $\mathrm{N}_{\mathcal{L}} \sum_{\mathrm{i} \in \mathcal{L}} (\mathrm{y_i} - \mu_{\mathcal{L}})^2$ is the within-leaf variance, so $\ell(\mathcal{T})$ is essentially the weighted variance of the leaves in the tree. To minimize this quantity we use the following basic growing algorithm:

1. Start with a single node containing all data.
2. For each node that does not satisfy a predefined stopping criteria (e.g., if a node has within-node variance lower than some threshold, or has fewer than some fraction of the overall data), find the feature x_{j} whose split minimizes $\ell(\mathcal{T})$ as much as possible, and add two nodes to your tree accordingly.
3. Repeat Step 2 until all leaves of the tree meet the stopping criteria.

8.10 Other Variants

Being one of the most popular machine learning algorithms, decision trees have countless variants. A few common examples include:

- **M-of-N splits.** Here each split represents a test with N conditions. The test is satisfied if at least M out of those N conditions are met. For example, a 2-of-3, single-node tree to assess COVID risk could look like the following:



The main idea to identify the features to test in each split is to follow an inductive method that starts with 1-of-1 (ordinary case), and then grows it to m-of(n+1) and (m+1)-to-n [5].

- **Probability Estimation Trees.** Rather than making a deterministic decision, each leaf produces the probability of each class. For example, if 5/15 samples in a leaf belong to class 1 and 10/15 samples belong to class 2, then a probability tree will determine that a new sample ending in this leaf has probabilities 1/3 and 2/3 to belong to classes 1 and 2, respectively.
- **Lookahead.** This approach aims to reduce myopia, that is, missing a feature that may not appear to be informative at first. To this end, *lookahead* determines the gain of the most informative feature based on the mutual information of the most informative features at the next level of the tree. According to [6, 7], this can potentially alleviate myopia, however, empirically, it does not improve tree size or accuracy.

References

[1] L. Hyafil and R. Rivest, *Constructing optimal binary decision trees is NP-complete*, Information Processing Letters, 1976.

[2] T. Cover and J. Thomas, *Elements of information theory*, Wiley Series in Telecommunications and Signal Processing.

[3] R. Quinlan, *C4.5: Programs for machine learning*, Morgan Kaufmann Publishers.

[4] R. Quinlan, *Induction of decision trees*, Machine Learning, 1986.

[5] P. Murphy and M. Pazzani, *ID2-of-3: Constructive induction of M-of-N concepts for discriminators in decision trees*, Machine Learning Proceedings, 1991.

[6] S. Norton, *Generating better decision trees*, IJCAI, 1989.

[7] S. Murthy, and S. Salzberg, *Lookahead and pathology in decision tree induction*, IJCAI, 1995.