PH 240C Supervised Learning (4): Tree

Jingshen Wang

September 29, 2021

The superhero example we worked on is built upon an assumption that similar inputs have similar neighbours. This somewhat implies that data points of various classes are not randomly sprinkled across the space, but instead appear in clusters of more or less homogeneous class assignments. The decision rules we worked on in previous sections all exploit the same heuristic: If a linear combination of (or some metric) attributes exceeds/below certain threshold, then the subject belongs to a certain class. Given the ultimate goal of a classification rule is to given an accurate prediction, we could explore a different type of intuition.

1 Decision tree

Imagine a binary classification problem with positive and negative class labels. If you knew that a test point falls into a cluster of 1 million points with all positive label, you would know that its neighbors will be positive even before you compute the distances to each one of these million distances. It is therefore sufficient to simply know that the test point is an area where all neighbors are positive, its exact identity is irrelevant. Decision trees are exploiting exactly that.

1.1 NP-hard problem to find a good division of attribute space

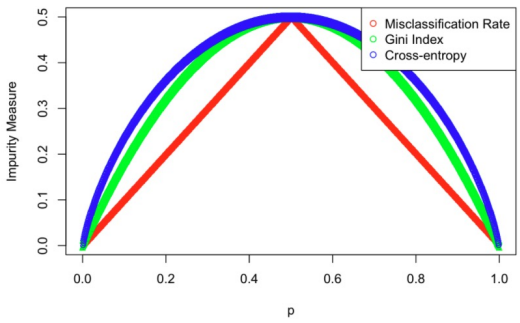

Figure 1: Different impurity measure adopted in tree-based approaches.

Suppose we have an i.i.d. training sample $\{(X_i, Y_i)\}_{i=1}^n$, where Y_i is categorical responses with $Y_i \in \{-1, 1\}$, and $X_i \in \mathbb{R}^d$ contains attributes. Here, we do not store the training data, instead we use the training data to build a structure that divides the space into regions with similar labels. Mathematically, our goal is to segment the attribute space $\mathcal{X}$ into a number of disjoint regions $R_1, \ldots, R_p$. Within each region, the subjects are similar, and between different regions, the units are quite different. Once we have find these disjoint regions, we then predict a new data point with attribute x as

$$\mathrm{y} = \sum_{j=1}^{p} \mathbf{1}(\mathrm{x} \in R_j)\big(2\mathbf{1}_{(\sum_{i\in\mathbb{R}_j} Y_i > 0.5)} - 1\big).$$

Next question we need to resolve is how to find the division R_j? And what kind of criteria they need to follow? Ideally, we want to build a tree that is: (1) maximally compact (why?), and (2) only has pure regions (why?)

Now let's try to work out our objective function to quantify a good division $R = \{R_1, \ldots, R_p\}$ of space in two steps:

1. Quantitative measure for a pure region: For a given region R_j, when the outcomes are binary random variables $Y_i \sim \text{Bernoulli}(p_j)$ with $p_j \in (0, 1)$, we can measure if the region is "pure" with the empirical variance of Y_i's in this region

$$\texttt{Gini}(R_j) = \bar{Y}_j\big(1 - \bar{Y}_j\big), \quad \text{where } \bar{Y}_j = \frac{1}{|R_j|}\sum_{i \in R_j} Y_i.$$

   Alternatively, in a region R_j, we can compare the distribution of $Y_i \sim \text{Bernoulli}(p_j)$ with something that we do not want, i.e., $Y_i \sim \text{Bernoulli}(1/2)$

$$\begin{aligned}\texttt{KL}(R_j) = \texttt{KL}\big(\text{Ber}(\widehat{p}_j)||\text{Ber}(1/2)\big) &= \bar{Y}_j \cdot \log\frac{\bar{Y}_j}{0.5} + \big(1 - \bar{Y}_j\big)\log\frac{1 - \bar{Y}_j}{0.5} \\ &\propto \bar{Y}_j \log \bar{Y}_j + (1 - \bar{Y}_j)\log(1 - \bar{Y}_j).\end{aligned}$$

2. Quantitative measure for a good division:

$$\begin{aligned}&\texttt{Gini}(R) = \sum_{j=1}^{p} s_j \texttt{Gini}(R_j), \\ &\texttt{Entropy}(R) = -\sum_{j=1}^{p} s_j \texttt{KL}(R_j), \quad \text{where } s_j = \text{fraction of subjects in the region } R_j.\end{aligned}$$

   The smaller these quantities are, the better the division is.

Thus, a good division of the attribute space should ideally solve the below optimization problem:

$$\begin{aligned}\min_{R} \quad & \texttt{Gini}(R) \text{ or } \texttt{KL}(R) \\ \text{s.t.} \quad & p \leq \text{Constant}.\end{aligned}$$

Unfortunately, finding a division like this is NP-hard and typically cannot be solved in a polynomial time.

Before ending this section, we can see from Figure 1 that whether we use entropy or Gini impurity does not really matter, because both have the same concave/bell shape which is essential (why?).

1.2 Different types of decision trees

Rather than finding a good division by greedy search, decision trees use the training data to build a tree structure that *recursively* divides the space into regions with similar labels. To see how a decision works in practice, let's first fix some terminologies via introducing several steps (also see Figure 2):

1. The **root node** of the tree represents the entire data set.

2. This entire dataset is then split roughly in half along one dimension by a simple threshold r. All points that have a feature value $\geq t$ fall into the right **child node**, all the others into the left **child node**.

3. The threshold t and the dimension are chosen so that the resulting child nodes are purer than their parent nodes. Ideally all positive points fall into one child node and all negative points in the other:

   - If all child nodes are pure, the tree is done.
   - If not, the child nodes are again split until eventually all nodes are pure (i.e. all its data points contain the same label) or cannot be split any further (in the rare case with two identical points of different labels).

4. Whenever a node is split, we refer to that given node as the **parent node**, and the resulting nodes are called child nodes, respectively.

5. When the tree is done, we all the nodes as **leaf nodes**.

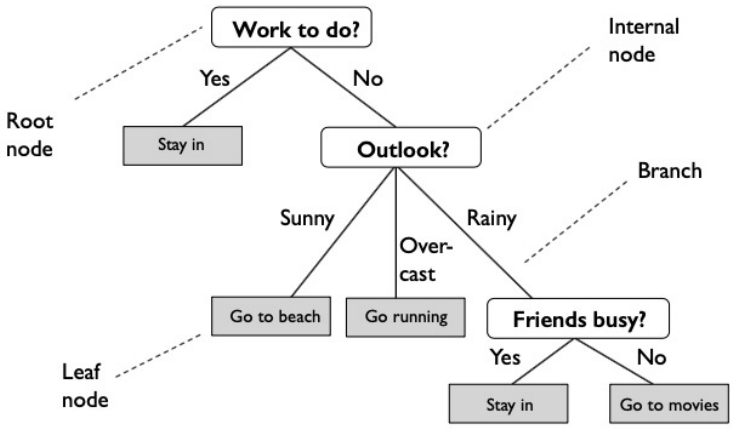

Figure 2: Example of a non-binary decision tree with categorical features.

There exists a relatively large variety of decision tree algorithms. This section lists some of the most influential/popular once. Most decision tree algorithms differ in the following ways: (1) Splitting criterion: information gain (Shannon Entropy, Gini impurity, misclassification error), use of statistical tests, objective function, etc., (2) Binary split vs. multi-way splits, (3) Discrete vs. continuous variables.

ID3 (Iterative Dichotomizer) First described in Quinlan (1986), this is one of the earliest decision tree algorithms. It works with discrete features and cannot handle numeric features. Maximizes information gain/minimizes entropy (calculated by KL-divergence).

C4.5 First described in Quinlan, it handles both continuous and discrete features. Splitting criterion is computed by the gain ratio.

CART First proposed by Breiman (1984), it handles both continuous and discrete features with strict binary splits (resulting trees are taller compared to ID3 and C4.5). Binary splits can generate better trees than C4.5, but tend to be larger and harder to interpret, i.e., for D attributes, we have $2^{d-1} - 1$ways to create a binary partitioning. Uses Gini impurity in classification trees. CART has built-in algorithm to impute missing data with *surrogate splits*.

1.3 Decision trees in action

Take CART as an example, further suppose we have i.i.d. sample with pairs (Y_i, X_i), $i = 1, \ldots, n$, and X_i lives in a discrete sample space $X_i \in \mathcal{X} = \{x_1, \ldots, x_m\}$;

1. For $j = 1 : m$
   (a) Split the entire dataset into two child nodes:
   $$R_{\text{left}}(j) = \{i : X_i \leq x_j\}, \quad R_{\text{right}}(j) = \{i : X_i > x_j\};$$
   (b) Calculate the within node purity with either Gini index or KL-divergence.
   (c) Calculate the *split purity* by appropriately *combining Gini index or Entropy.*
2. Split the node into two child nodes that maximize *information gain.*
3. Keep splitting within each node until some stopping criteria is reached.

To better understand what "information gain" is. We briefly talk about information theory coined by Claude Shannon in Shannon (1948). There, the author defines the entropy of a random variable as the average level of "information", "surprise", or "uncertainty" inherent in the variable's possible outcomes.

Entropy Suppose we have a discrete random variable $Z \sim \text{Bernoulli}(p)$ that characterizes if a student on campus has COVID-19 with some unknown p. Since p is unknown, we are going to take some random samples on campus. Hopefully, this can help us predict if the first student we meet on Oct. 1st has COVID or not. Now how many random samples we have to take in order to achieve an accurate prediction? Naturally, whenever $p = 0$ or $p = 1$ we are 100% certain about an event, less information is required for us to do prediction. Whenever p is a bout one half, we have to collect a lot more information since the samples we collect have random behaviour. In this context, Shannon's entropy uses the following quantity to describe the information:

$$H(p) = p \log p + (1 - p) \log(1 - p).$$

Obviously, $H(p)$ is maximized whenever $p = 1/2$, meaning we need more information to conduct analysis if we want to accurately prediction. In the context of decision trees, as the ultimate goal of the classifier is simply to give an accurate prediction. We do not want to have the a node with $p = 1/2$ because it requires more efforts to classify these points.

Information gain We now consider the more general formula for measuring information gain. Without loss of generality, suppose we are at an internal node with region R, and candidate child nodes are denoted as $R_1, R_2, \ldots, R_p$. The information gain is

$$\texttt{Gain}(R) = \texttt{Information}(R) - \sum_{j=1}^{p} \texttt{Information}(R_j),$$

where the information is measured by either Entropy or Gini index. The larger the information gain is, the better these child nodes can provide decisive prediction results.

Question 1: why don't we stop if no split can improve purity/gain information? See an example in Figure 3, where we hope to classify patients into two groups. Classification tree is likely to stop on the default setting because the first split does not improve impurity. Now the question is, what kind of classifier can we use to avoid this issue? Neither SVM nor GLM is able to do perfect classification in this simple example. Think about if we define $Z_1 = \mathbf{1}(X_1 \geq 5)$ and $Z_2 = \mathbf{1}(X_2 \geq 5)$, we can write down the outcome as

$$Y = Z_1 + Z_2 - 2Z_1Z_2.$$

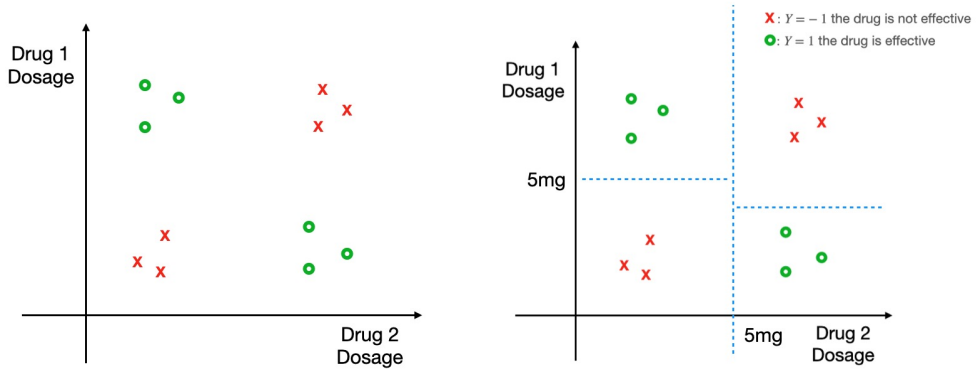

Figure 3: Decision trees may not keep splitting as there is no information gain in the first step.

Question 2. In biological science, very often, we care about more than a single outcome prediction. Many diseases are co-morbid, if our outcome lives in a 2-dimensional tuple, i.e., $\{0,1\}^2$, how can you carry out your prediction?

1.3.1 Regression tree

Regression tree works in a similar fashion as a decision tree. The only difference is that we have a continuous outcome $Y_i \in \mathbb{R}$. In this scenario, we work under similar logic: data points appear in clusters, and subjects fall in the same cluster should have similar outcomes. We can use the standard deviations to measure the outcome heterogeneity in a node. Our goal for a regression tree is to find regions $R_1, \ldots, R_p$ that minimize the residual sum of squares:

$$\sum_{j=1}^{J} \frac{R_j}{n} \sum_{i \in R_j} \left(Y_i - \bar{Y}_{R_j}\right)^2.$$

Implicitly, we assume that the conditional mean of the outcome given predictors is a smooth function. See Figure 4

1.4 Summary of tree-based approach

Tree-based approaches are very light weight classifiers, and can be solved with efficiently. Given it is essentially a non-parametric estimator, it is not competitive in accuracy but can become very strong through bagging (Random Forests) and boosting (Gradient Boosted Trees).

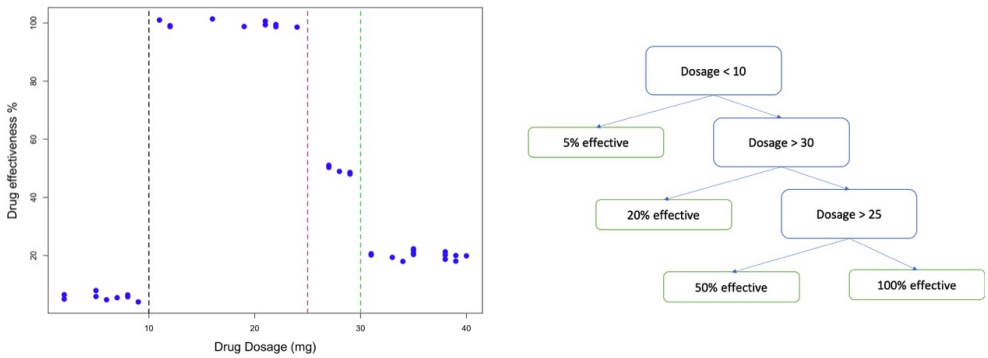

Figure 4: Regression tree example with drug effectiveness.

The tree-based approaches differ from the ones we introduced based on GLM, SVM and metric learning. The later ones are parametric algorithms with a set of parameters which is independent of the number of training samples. One can think about the number of parameters as the amount of space you need to store the trained classifier. In glm and svm, we have β and w–as long as we store these parameters, we can predict the labels for future data.

Decision tree is an interesting case. If they are trained to full depth they are non-parametric, as the depth of a decision tree scales as a function of the training data. In a special case with balanced binary tree, the tree depth is of $\log n$ (why?). If we however limit the tree depth by a maximum value they become parametric (as an upper bound of the model size is now known prior to observing the training data).

References

Leo Breiman. Classification and regression trees. Technical report, 1984.

J Ross Quinlan. C4. 5: Programming for machine learning. morgan kauffmann, 38, 1993.

J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

Claude Elwood Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27 (3):379–423, 1948.