Pomona College

Senior Thesis in Mathematics

Learning from Loss Functions: A Modified Algorithm for Classification Trees

Author:
Kashvi Tibrewal

Advisor:
Dr. Jo Hardin

Submitted to Pomona College in Partial Fulfillment
of the Degree of Bachelor of Arts

April 12, 2018

Contents

Chapter 1

Introduction

Predictive models like linear or polynomial regression are global models, where a single predictive formula is applied over the entire data space. However, a single global model can perform poorly on many predictor variables that may interact with each other in complex ways. Non-parametric smoothers can be used to fit the model locally over different regions of the predictor space, but such smoothers can be hard to interpret. An alternative approach is to use additive models that recursively (or repeatedly) partition the predictor space into smaller regions using some splitting criteria, until the resulting regions can fit simple models. Prediction trees are an example of such additive models that use recursive splitting of the predictor space into a set of rectangles and then fit simple models to each of the subspaces.

Such predictive models are built using training data, and are used to predict the response of test observations. Ideally, the models should simultaneously achieve *low variance* and *low bias* so as to increase accuracy of prediction on test data. Variance refers to the amount by which the results would change if the model was built using a different training set of data. If a model has high variance, small changes in the training data could lead to large changes in the result. Ideally a model should not change much for different training sets, and therefore have low variance. Bias, on the other hand, refers to the error that arises from trying to estimate a complex real-life problem with a much simpler model. Thus, the simpler the model, the higher the bias, and therefore, higher the error in the model prediction. As a result, as the complexity of a model increases, its variance increases and bias decreases. Therefore, there exists a variance-bias tradeoff, and it is important to think about this concept when looking into predictive models.

Predictive models can be built for classification or regression problems. Most approaches to classification are designed to maximize the homogeneity of the resulting subspaces with respect to the response variable. In other words, common classification algorithms that aim to predict the response variable do so by repeated partitioning of the data until the final subspaces are as homogeneous as possible with respect to the response. Then, all the observations that fall into a particular region are predicted to belong to the most commonly occurring class (or the majority class) in that region. However, there is no well-defined measure for homogeneity, and most measures that are commonly used do not deal with imbalanced data or different loss weights associated with misclassifying different classes of the response.

Many practical applications of classification models deal with *imbalanced data*, where at least one of the classes of the response variable constitutes only a small minority of the data available to train the model. In such cases, the interest is usually focused towards the correct classification of the minority class (or classes). In other words, the loss associated with misclassification of a certain class may be higher than for others. For example, the loss associated with predicting that a patient does not have cancer when they actually do is much higher than the loss associated with its vice versa. However, common classification algorithms perform poorly when working with such imbalanced data or skewed loss functions since they aim to minimize the overall error rate by maximizing the homogeneity of the response. Since the data in such cases is mostly comprised of observations from the dominating classes, the algorithm seeks to correctly classify them rather than paying special attention to the minority class. This leads to over-prediction of the majority class. Thus, in situations such as disease diagnosis, fraud detection, etc. it is important to assign different weights that arise from false positives and false negatives. This leads to the issue of picking suitable weights for different error types, which is often very hard to do. In my thesis, I will focus on classification trees and investigate alternative approaches that evaluate the quality of splits and increase accuracy of prediction when dealing with imbalanced data or data with highly skewed loss functions.

Chapter 2

Background

Tree-based methods are models that are trained on known data, or *training data*, and aim to predict the categorical or numerical response of new test observations. Such models can be used for classification or regression problems, and they recursively partition the predictor space into a finite number of smaller regions so as to maximize the homogeneity of each of the resulting regions with respect to the response variable in the data. In this chapter, we will focus on *classification trees*, which take in numerical or categorical predictor variables and predict a qualitative response. These are binary trees, since the feature space is repeatedly split into two resulting subsets.

Given training data, let the different values of the categorical response variable be referred to as *classes*. Thus each observation in our training data has a response that falls into the set of classes $C = \{1, 2, ..., K\}$. Let the feature space be denoted by $\mathcal{X}$. Then $X \in \mathcal{X}$ is an input vector that contains p features $X_1, X_2, ..., X_p$ that may be numerical or categorical. An observation i in our training data can be represented as (y_i, x_i), where y_i is the categorical response, and x_i is the input vector $(x_{i1}, x_{i2}, ...x_{ip})$.

A classification tree is constructed by splitting the feature space $\mathcal{X}$ into two descendant subsets. These subspaces of $\mathcal{X}$ are known as *nodes*, and each of the resulting nodes is split in the same way as $\mathcal{X}$. This process continues until some stopping criteria is reached (for example, when a minimum node size is reached, or when the resulting nodes are completely homogeneous with respect to the response variable). At each split, the two descendant nodes are referred to as *daughter nodes*, and the node that was split to produce them is called the *parent node*. Each node in a tree is either an *internal node* or a *terminal node*. Terminal nodes represent the final nodes of the tree, implying that further splitting of the feature space does not explain enough of the variance to be relevant in describing the response variable. Once the tree is constructed, it gives a set of rules or conditions at each level that are easy to interpret. Observations fit within a particular node only if they satisfy the condition at that node. Each of the terminal nodes are assigned with a class, and each observation is associated with the class of terminal node that it lands in.

Based on the discussion above, the construction of a tree consists of three important elements:

- The determination of split at each node.
- The decision of when to declare a node terminal or to continue splitting it.
- The prediction or assignment of each terminal node to a particular class of the response variable.

Determination of Splits

In order to grow a classification tree, binary partition of the feature space is carried out at each level of the tree. Each split depends on a particular value of one of the p features or variables. The point at which the partition occurs is called the *split point*, and a goodness of split criterion is evaluated for any split s at any node t. At each node, the goodness of split criterion measures the extent of homogeneity of the resulting nodes with respect to the response variable. These measures of homogeneity are called $impurity functions$, and are discussed in detail in the next chapter. Thus, the algorithm determines the best split for each parent node such that the class distribution of the response variable in each of its daughter nodes is "purer" than that of the parent node. In other words, the algorithm seeks to minimize a measure of impurity with each iterative split.

Each possible split is defined by an explanatory variable X_j and a split point s such that the explanatory variable space can be partitioned into two regions, R_1 and R_2. At node t, consider a continuous predictor variable X_j and a split point s that results in the following pair of half planes:

$$R_1(j,s) = \{X|X_j \le s\}$$
$$R_2(j,s) = \{X|X_j > s\}$$

For example, let X_j be a predictor variable that represents the price of a car, and ranges over \$20,000 to \$100,000. Then the split point $s = 40,000$ divides the feature space into the regions $R_1(j,s) = \{X|X_j \le \$40,000\}$ and $R_2(j,s) = \{X|X_j > \$40,000\}$.

If, on the other hand, X_j is a categorical variable that is defined over a set of classes $1, 2, ...K$,then a split point s results in the following pair of half planes:

$$R_1(j,s) = \{X|X_j \in A\}$$
$$R_2(j,s) = \{X|X_j \notin A\}$$

where A ranges over all subsets of $1, 2, ...K$.

For example, let X_j be a predictor variable that represents the color of a car and ranges over $\{blue, black, red\}$. Then a split point $s = \{blue, red\}$ divides the feature space into the regions $R_1(j,s) = \{X|X_j \in \{blue, red\}\}$ and $R_2(j,s) = \{X|X_j \notin \{blue, red\}\}$.

In a node t representing a region R_t with a total of N_t observations, let $\hat{p}_{tk}$ represent the proportion of observations of class k in node m. Thus,

$$\hat{p}_{tk} = \frac{1}{N_t} \sum_{x_i \in R_t} I(y_i = k)$$

The node is then split using predictor variable X_j and split point s such that the following equation is minimized over all (j,s):

$$\sum_{x_i \in R_1(j,s)} \sum_{k=1}^{K} \hat{p}_{R_1 k}(1 - \hat{p}_{R_1 k}) + \sum_{x_i \in R_2(j,s)} \sum_{k=1}^{K} \hat{p}_{R_2 k}(1 - \hat{p}_{R_2 k})$$

Thus at each node, every possible pair of (j,s) is used to evaluate the node impurity measures, and the best split is determined to be the pair that minimizes the node impurity measure. The data is then partitioned into the two resulting nodes using the best split criteria, and the splitting process described above is repeated with each of the two resulting nodes. This process continues until the stopping criterion is reached.

The algorithm for growing a classification tree is "greedy" - it makes a series of locally optimal decisions about which attribute to use for partitioning the data by minimizing the impurity measures at each level.

To Split or not to Split

Classification trees need some criterion that evaluates whether a particular node should be further split into its daughter nodes or declared a terminal node. In the absence of such a criterion, the algorithm would continue splitting in order to maximize node homogeneity, and would grow a tree in which each case or observation occupied its own terminal node. Such a tree would not only be computationally intensive but would also overfit the training data, and would have low accuracy in predicting classes on new test observations.

Some of the commonly used stopping criteria for classification trees are:

- If the number of observations in a particular node is less than a pre-specified limit, then stop splitting, that is, declare the node to be a terminal node.
- If the node is purely homogeneous, that is, all observations falling into the node have the same response variables, then declare it to be a terminal node.
- If the depth of the node is more than some pre-specified limit, then declare it to be a terminal node.
- If all the observations in the node have identical values for explanatory variables, no rule could be generated to split them, so the node should be declared to be a terminal node.

Prediction

In the process of building the classification tree, the feature space, or the set of all possible values of $X_1, X_2, ...X_j$ is partitioned into M distinct, mutually exclusive regions $R_1, R_2, ..., R_M$. Each terminal node corresponds to one of these regions. All observations in a particular region are classified to belong to the *most commonly occurring* class of training observations in that region. In other words, every observation that belongs to region R_j is classified to belong to the majority class of the response variable for the training observations in R_j.

Let $\hat{p}_{mk}$ represent the proportion of training observations in node m (or region R_m that belong to class k.

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{x_i \in R_m} I(y_i = k)$$

Node m is assigned to class $k(m) = \arg\max_k \hat{p_{mk}}$. Thus, all observations in node m are classified to belong to the majority class in node m.

Given a test observation, we can predict its response value by using its explanatory variables to go down the tree structure and determine which terminal node it falls into. Then, its predicted response value is the class assigned to its terminal node in the process described above.

[Add figure/example]

There are a number of advantages to using classification trees. Their output is a set of rules that is easy to interpret and mirrors the typical human decision-making process. They can handle categorical or numerical predictors (or both) and since they are non-linear models, they make use of interactions between the different variables. They can also handle classification of data points over more than two groups or classes.

At the same time, even though they are incredibly useful, classification trees have their limitations. They suffer from high variance. As discussed in Chapter 1, this means that small changes in the training data can result in large changes in the result of the tree. Thus, if the training data were spit into two parts at random and a tree was fit to both halves, the results could be very different. Without proper pruning of the tree, classification trees can easily overfit the training data. Current methods of binary classification that are used to split nodes at every step do not penalize for misclassifications in specific directions.

Chapter 3

Homogeneity

As discussed in Chapter 2, a goodness of split criterion is evaluated at each node in the tree building process. The goodness of split criterion measures the extent of homogeneity of the resulting nodes with respect to the response variable. The algorithm determines the best split for each parent node such that the class distribution of the response variable in each of its daughter nodes is more homogeneous than that of the parent node. These measures of homogeneity are called $impurity functions$, and while there are several ways of measuring the homogeneity of a node, this chapter discusses the most commonly used measures. Each of them seeks to maximize node homogeneity by minimizing "impurity" withe each iterative split.

Measures of Node Impurity

An impurity function measures the extent of homogeneity of a region containing observations that may belong to a number of different classes. Although there are many ways of measuring impurity, they share some common properties.

Definition. Let the different values of the categorical response variable be referred to as classes. Then each observation in our training data has a response that falls into the set of classes $C = 1, 2, ..., K$. For a node m, an impurity measure is a function f defined on the set of all K-tuples of $(p_{m1}, p_{m2}, ...p_{mk})$ such that $0 \leq \hat{p_{mk}} \leq 1 \ \forall k$ and $\sum_{k=1}^{K} \hat{p_{mk}} = 1$. f has the following properties:

- f is maximized when observations are uniformly distributed across all

K classes, i.e., when $\hat{p}_{mk} = 1/K \ \forall k$.

- $f = 0$ when $\hat{p}_{mk} = 1$ for one class and $\hat{p}_{mk} = 0$ for all other classes.
- f is invariant to which is the majority class (f does not penalize for specific misclassifications).

In classification trees, the measures developed for selecting the best split are often based on the degree of impurity of the child nodes. The impurity function takes a value of 0 for perfectly homogeneous nodes and increases as homogeneity decreases. This means that minimizing measures of impurity implies maximizing homogeneity in nodes.

Some common measures of impurity include the misclassification error, Gini index and Cross-entropy.

1. Misclassification error

The misclassification error is the proportion of observations in a node m that do not belong to the most commonly occurring class in node m. The misclassification error is measured as:

$$E_m = 1 - \max_k \hat{p_{mk}}$$

2. Gini index

The Gini index is a measure of variance across all the different classes. If the response of each observation is recorded to be 1 if it belongs to a particular class k, and 0 if it does not, then the variance over the node of this 0-1 response variable is $\hat{p_{mk}}(1-\hat{p_{mk}})$. Summing over all K classes yields the Gini index, which is, therefore, defined as:

$$G_m = \sum_{k=1}^{K} \hat{p_{mk}}(1 - \hat{p_{mk}})$$

If all the values of $\hat{p_{mk}}$ are close to 0 or 1, it implies that a majority of observations in a particular node belong to a particular class. As a result, the value of the Gini index will be small, implying low variance and higher homogeneity. In contrast, the Gini is maximized when the observations in a node m are more equally distributed among the K classes, thus implying high variance and lesser homogeneity.

3. Cross-entropy

The measure for cross-entropy is defined as:

$$D_m = -\sum_{k=1}^{K} \hat{p_{mk}} \log \hat{p_{mk}}$$

This measure is close to 0 if all values of $\hat{p_{mk}}$ are close to 0 or 1. Thus, the measure is lower when node m is more homogeneous.

The Gini index and cross-entropy are differentiable, and are therefore, more useful in numerical optimization. They are also more sensitive to changes in $\hat{p_{mk}}$ when compared to the misclassification error rate. For example, consider a sample with 800 observations, such that half (=400) belong to class 1 and the other half (=400) belonged to class 2. Split A results in daughter nodes (300,100) and (100,300) each, producing a misclassification error of 0.25. Split B results in daughter nodes (200,0) and (200,400), and also produces a misclassifcation error of 0.25. Thus, even though split B produces a pure node, according to the misclassification error measure, both splits (A and B) have equal measures of impurity and are equally preferred.

The Gini index and cross-entropy measures are lower for split B than split A, implying that split B is preferable over split A. The Gini index and cross entropy are more sensitive to node purity than the misclassification error, and are therefore, more typically used in partitioning trees.

[Add figure on different impurity measures]

Chapter 4

Incorporation of Loss

Introduction

An important aspect of minimizing the risk associated with misclassification that is not used in the previously defined impurity measures is the loss function. In certain situations, the loss associated with misclassifying observations belonging to some classes is higher than the others. For example, when diagnosing cancer in a patient, the loss associated with predicting that the patient will not have cancer when they actually will, is far higher than the vice versa. Thus, sometimes, misclassification of observations need to be weighted differently depending on what class they actually belong to. For this purpose, a K x K loss matrix **L** is used, with rows corresponding to the correct classes and columns corresponding to classes predicted by the tree. The elements of **L** are represented by $L_{kk'}$, indicating the loss incurred for misclassifying a class k observation as a class $k^{'}$ observation. Since no loss is incurred for correct classification, $L_{kk} = 0$ for all values of k. The classification tree can take this into consideration by weighting how much to penalise each incorrect classification in a given choice of split.

Generalized Gini Index

A commonly used approach to incorporating losses in the tree building process is to modify the Gini index to be defined as:

$$G_m = \sum_{k \neq K} L_{kk'\hat{p}_{mk}(1-\hat{p}_{mk})}$$

This is the expected loss incurred by the randomized rule. This approach does not use the majority class in the node to evaluate the splits. At each split, the tree tries to maximize homogeneity irrespective to what is the majority class at each node.

Consider the two class scenario, i.e. when $k = 2$.

$$\begin{aligned} G_m =& \sum_{k \neq k'} L_{kk'_{\hat{p}_{mk}(1-\hat{p}_{mk})}} \\ =& L_{12}\hat{p}_{m1}\hat{p}_{m2} + L_{21}\hat{p}_{m2}\hat{p}_{m1} \\ =& (L_{12} + L_{21})\hat{p}_{m1}\hat{p}_{m2} \\ =& (L_{12} + L_{21}) \sum_{k \neq k'} \hat{p}_{mk}\hat{p}_{mk'} \end{aligned}$$

Therefore, in this case, the generalized Gini index scales the regular Gini index ($G_m = \sum_{k \neq k'} \hat{p_{mk}}\hat{p_{mk'}}$) by a magnitude of $(L_{12} + L_{21})$. Thus, the generalized Gini index is proportional to the regular Gini index, implying that the inclusion of losses has no effect in the split decision in the two case scenario. It only changes the magnitude of the Gini when the split is evaluated, but does not change the split decision.

When $k > 2$, the generalized Gini index can be used to evaluate the splits if as a function of k, L_{kk} does not depend on $k^{'}$. In each terminal node, the observations are then classified to class $k(m) = \arg\min \sum_l L_{lk}\hat{p_{ml}}$. The ordinary Gini index defined in the previous chapter is a special case of the generalized Gini Index where $L_{kk'} = 1 \; \forall k \neq k'$.

Altered Priors

In Bayesian statistics, a prior probability distribution of an uncertain quantity is the probability distribution that would express one's beliefs about this quantity before some evidence is taken into account. In this section, we will investigate the splitting criteria and class assignment rule by using prior probabilities. Given experimental data, the default probability of a class k is the proportion of observations in the data that belong to class k. When working with classification methods, the prior probability of each class k $(= \pi_k)$ can be estimated from the data by using empirical probability of the class k , before the classification technique is applied to the data.

Let the total number of observations in the data be N. Then N_j denotes the total number of observations in the data that belong to class j. Then, the observations across all classes should sum up to the total number of observations in the data.

$$\sum_{j=1}^{K} N_j = N$$

If the prior probability of class j is estimated from the data, then

$$\hat{\pi}_j = \frac{N_j}{N}$$

Let $N(t)$ denote the number of observations in node t, and $N_j(t)$ denote the number of observations belonging to class j in node t. Then for any node t, observations across all classes should sum up to the total number of observations in that node.

$$\sum_{j=1}^{K} N_j(t) = N(t)$$

Let $P(t|j)$ denote the estimated probability of an observation going to node t given that it belongs to class j. $P(t|j)$ is the proportion of observations belonging to class j that go to node t.

$$P(t|j) = \frac{N_j(t)}{N_j}$$

Then, the joint probability of an observation belonging to class j and going to node t is denoted by $P(j,t)$.

$$\begin{aligned} P(j,t) =& P(t|j)P(j) \\ =& P(t|j)\hat{\pi}_j \\ =& \frac{N_j(t)}{N_j}\frac{N_j}{N} \\ =& \frac{N_j(t)}{N} \end{aligned}$$

Here, $P(j)$ is the probability that an observation belongs to class j. Thus, it can be estimated by using our definition of $\hat{\pi}_j$.

Let $P(t)$ be the probability that an observation is in node t. Given that $N(t)$ is the total number of observations in node t,

$$P(t) = \frac{N(t)}{N}$$

Let $P(j|t)$ denote the probability that an observation belongs to class j given that it is in node t.

$$\begin{aligned} P(j|t) =& \frac{P(j,t)}{P(t)} \\ =& \frac{N_j(t)}{N} \frac{N}{N(t)} \\ =& \frac{N_j(t)}{N(t)} \end{aligned}$$

Observations in node t are then classified to belong to the class to which the largest proportion of cases belong in node t. Therefore, the class assignment rule is

$$\begin{aligned} K(t) =& \arg\max_j \frac{N_j(t)}{N(t)} \\ =& \arg\max_j P(j|t) \end{aligned}$$

Altered prior probabilities can be used in order to incorporate loss functions into the splitting criteria. Altered priors can be defined as:

$$\tilde{\pi}_k = \frac{\hat{\pi}_k \sum_{k'} L(k,k')}{\sum_j \hat{\pi}_j \sum_{j'} L(j,j')}$$

Substituting π_j with $\tilde{\pi}_j$ in the equation for $P(j,t)$:

$$\begin{aligned} P(j,t) =& P(t|j)\tilde{\pi}_j \\ =& \frac{N_j(t)}{N_j}\tilde{\pi}_j \end{aligned}$$

Now the probability of an observation belonging to class j given that it is in node t $(= P(j|t))$ is

$$\begin{aligned} P(j|t) =& \frac{P(j,t)}{P(t)} \\ =& \frac{\frac{N_j(t)}{N_j}\tilde{\pi}_j}{\frac{N(t)}{N}} \\ =& \frac{N_j(t)}{N_j}\frac{N}{N(t)}\tilde{\pi}_j \end{aligned}$$

The class assignment rule for node t remains the same, i.e., observations in node t are classified to belong to $K(t) = \arg\max_j P(j|t)$.

Thus, in this method, the prior probabilities of classes are altered by weighting observations with the loss of misclassifying them. When altered priors are used, they affect only the choice of split. The ordinary losses and priors are used to compute the risk of the node. The altered priors simply help the impurity rule choose splits that are likely to be good in terms of the risk.

Oversampling

As explained in chapter 2, decision trees make class assignments based on majority representation - observations in node m are classified to belong to the majority class in node m. Another approach to ensuring that classes that have high misclassification losses associated with them get classified correctly is to increase their representation in the data. This can be achieved by using oversampling. Oversampling observations from class k means that observations belonging to class k in the data are randomly replicated and included in the data. The idea behind this approach is to adjust the sample data, such that the representation of class k in the sample is proportional to its misclassification loss. Then, the regular measures of impurity are used to make the splits, and the observations in each of the terminal nodes are classified to belong to the majority class in that node.

Chapter 5

Performance Metrics

Accuracy

The most commonly used metrics for evaluating the performance of classification models are *accuracy* and *error rate*. A classification model maps each instance to a predicted class. Consider the two-class classification problem. By convention, the class label of the minority class (in the case of imbalanced data) or the class label of the class that has a high cost of misclassification associated with it, is considered *positive*, and the class label of the majority class (or the class with low cost of misclassification) is considered *negative*. Let p, n represent the set of true class labels - positive and negative. Let Y, N denote the set of predicted classes. Then the classification model maps each instance to either a Y or an N. Given a classifier and an instance, there are four possible outcomes.

- If the instance is positive, and is classified as positive, it is counted as a *true positive*
- If the instance is positive, and is classified as negative, it is counted as a *false negative*
- If the instance is negative, and is classified as negative, it is counted as a *true negative*
- If the instance is negative, and is classified as positive, it is counted as a *false positive*

	p	n
p’	TP	FP
n’	FN	TN

Confusion Matrix

Given a classifier, and a set of instances, a 2 x 2 *confusion matrix* can be constructed to represent the performance of the classifier on the set of instances. The table above shows a confusion matrix. The diagonal of the matrix represents the correct classifications made by the model - the count of instances that belong to the positive class and were predicted to belong to the positive class, and those that belong to the negative class and were predicted to belong to the negative class. The accuracy and error rate of a model are defined as

$$Accuracy = \frac{TP + TN}{P + N}$$

$$ErrorRate = 1 - Accuracy$$

where TP is the number of true positives, TN is the number of true negatives, P is the total number instances that belong to the true positive class and N is the total number of instances that belong to the true negative class.

These metrics provide a simple way of assessing the performance of a classification model, but are highly sensitive to changes in the data. A model is considered ”good” if it demonstrates a high rate of accuracy on previously unseen data. However, when dealing with highly imbalanced data, or data that has highly skewed misclassification losses associated with the classes, a high rate of accuracy may not give the most ”accurate” representation of the model’s performance. A major issue with accuracy is that it assumes equal costs of misclassification across all classes. Consider, for example, a dataset that includes 95 percent of instances from the majority class and 5 percent of instances from the minority class. A classification model could classify every instance in the set to belong to the majority class and achieve an accuracy rate of 95%. Such a high accuracy rate across the dataset would imply that the model performs well. However, it fails to reflect the fact that 0% of the instances that actually belonged to the minority class were correctly classified. In classification problems where there are high costs associated with the misclassification of the minority class (like the diagnosis of cancer cells), the accuracy metric does not provide adequate information on a classifiers functionality with respect to the type of classification required.

ROC Analysis

Receiver Operating Characteristics (ROC) graphs are two dimensional graphs that are used to represent the relative tradeoffs between hit rates (or true positives) and false alarm rates of classification models. In such a graph, the *true positive rate* is plotted on the Y axis, and the *false positive rate* is plotted on the X axis. As described by Fawcett, the lower left point (0,0) depicts a strategy where every observation is classified to belong to the negative class - such a classifier commits no false positive errors but also gains no true positives. The upper right point (1,1), on the other hand, represents a strategy where all observations are classified to belong to the positive class - it correctly classifies all cases belonging to the true positive class, but also wrongly classifies all cases belonging to the true negative class. The point (0,1) represents the perfect classification - the model correctly classifies all cases. In general, since we are more interested in correctly classifying observations belonging to the positive class, a point in the ROC space is considered more desirable than another if it lies to the northwest (the TP rate is higher, FP rate is lower or both) of the other.

Classification models such as trees are designed to output the predicted class label for each observation. When applied to a test set, they yield a single confusion matrix that corresponds to a single point in the ROC space. However, it is also possible to obtain *fitted values* for each observation, which are the numerical scores used by the model to predict the class labels. Such numerical scores represent the degree to which an observation belongs to a class. For decision trees, A decision threshold is applied to these scores - if the score is above the threshold value, the observation is predicted to belong to the positive class, and if the score is below the threshold value, it is predicted to belong to the negative class. Each threshold value yields a confusion matrix, and thus, corresponds to a point in the ROC space. A series of threshold values can be used to generate an ROC curve for a particular classification tree.

Chapter 6

Conclusion

The issues that arise from dealing with imbalanced data or data that has unequal costs of misclassification across different classes, can be tackled by using loss functions. From the simulations, we see that the class of interest is more accurately classified when such loss functions are incorporated in the classification tree algorithm. We also explored the accuracy paradox in such situations - a high rate of accuracy may not be indicative of the class of interest actually being classified correctly, and other measures of performance, such as ROC curves can be used to better evaluate the performance of the model. In the future, I would like to explore different approaches to modeling loss, such as if the loss was determined stochastically. I would like to explore whether the different approaches to incorporating loss in the model, through sampling techniques or by modifying the learning algorithm, yield different results.

Bibliography

[1] Friedman, Jerome, Trevor Hastie, and Robert Tibshirani. The elements of statistical learning. Vol. 1. New York: Springer series in statistics, 2001.

[2] Scott, Clayton, and Robert Nowak. "A Neyman-Pearson approach to statistical learning." IEEE Transactions on Information Theory 51.11 (2005): 3806-3819.

[3] Scott, C. "Comparison and design of neyman-pearson classifiers." (2005).

[4] Fawcett, Tom. "An introduction to ROC analysis." Pattern recognition letters 27.8 (2006): 861-874.

[5] Terry Therneau, Beth Atkinson and Brian Ripley (2017). rpart: Recursive Partitioning and Regression Trees. R package version 4.1-11. https://CRAN.R-project.org/package=rpart