27 Classification Trees

Lab Objective: *Understand how to build a classification tree and use it to predict survival of Titanic passengers.*

Classification trees are a class of decision trees, and are used in a wide variety of settings where labeled training data is available, and where the desired outcome is a model which is able to accurately assign labels to unlabeled data. We assume that each sample d has P attributes, which can be real-valued or categorical, and that each sample belongs to some class k, where there are K classes. The tree is composed of many *nodes*, which represent a decision point (i.e. a question is asked about the sample which has a boolean response). If the response is `True`, then the sample is "pushed" down the tree to the left child node. If the response is `False`, then the sample is "pushed" down the tree to the right child node. A *leaf* node is a node that has no child node, i.e. it is the end of the line and there is not a question asked. Each leaf has a classification assigned to it, and an unlabeled sample is labeled with that classification upon arrival at the leaf node.

How do we train a classification tree? We start with a labeled data set D and choose the best attribute p and value x by which to *split* the data. We have now partitioned D into two sets, which we may then split as well. We continue in this manner until some stopping criterion is met (often a maximum depth of the tree). To formalize this, we need several definitions.

Definition 27.1. *Let D be a data set with K different classes. Let N_k be the number of samples labeled class k for each $1 \leq k \leq K$, and let $f_k = \frac{N_k}{N}$ where N is the total number of samples in D. We define the* Gini impurity *to be*

$$G(D) = 1 - \sum_{k=1}^{K} f_k^2.$$

Problem 1. Write a function that accepts a list of class assignments and a list of all the K possible classes, and computes the Gini impurity.

Definition 27.2. *We define the* split $s_D(p, x)$ *of the data set D on attribute p using value x, to be a partition D_1, D_2 such that*

1. *$d_p \leq x$ for all $d \in D_1$ and $d_p > x$ for all $d \in D_2$, where d_p is the value of attribute p in d, assuming real values; or*

2. *$d_p = x$ for all $d \in D_1$ and $d_p \neq x$ for all $d \in D_2$, where d_p is the value of attribute p in d, assuming categorical values.*

Problem 2. Write a function that computes the split of a data set for a given variable p and given value x. It should return the partitioned data set, as well as the partitioned class labels.

Definition 27.3. *Let $s_D(p, x) = D_1, D_2$ be a split. We define the* information gain *of this split to be*

$$I(s_D(p, x)) = G(D) - \sum_{i=1}^{2} \frac{|D_i|}{|D|} \cdot G(D_i)$$

Problem 3. Write a function that computes the information gain for the split of a data set for a given variable p, value x.

We define the optimal split of a data set to be

$$s_D^* = s_D(p^*, x^*),$$

where

$$p^*, x^* = \text{argmax}_{p,x} I(s_D(p, x)).$$

From this partition, we create two child nodes, assigning the left child node data set D_1, and right child node data set D_2.

Problem 4. Write a function that computes the optimal split of a data set. You may need to separate this into two tasks: finding the optimal split for each attribute p, and then choosing the optimal split over all the attributes.

Let's put all of this together to create the full classification tree.

Problem 5. Write a class called `Node` that creates and trains a classification tree. It should accept a training data set D, class labels y, current depth (which when initialized should be 1), some maximum depth which is greater than 1, and some tolerance for the Gini impurity (say 0.2). Use recursion to build the tree, i.e. after determining the optimal split, create two new nodes (`leftchild` and `rightchild`), with incremented depth. If the depth equals the maximum depth or the Gini impurity for a node is less than the tolerance, assign the majority label to the node and do not split further.

Problem 6. Write a method for the class `Node` that prints out the tree structure. For each node it should show which attribute p and value x provide the optimal split, and for the leaf nodes, it should show the assigned label. You may use your own creativity for how to display this.

Problem 7. Write a method for the class *Node* that assigns the class label for a new sample. You will probably have to make this method recursive also.

We would like to test our classifier on a real data set. Provided for you is a data set on about 1000 passengers aboard the Titanic. We would like to predict their survival or death depending on several attributes: class ($1^{st}, 2^{nd}$, or 3^{rd}), gender (male or female), and age.

Problem 8. Using the Titanic data set, train a classification tree with a maximum depth of 10 nodes and Gini impurity tolerance .1, and predict labels for the test set. What is your misclassification rate? Print out the tree structure. Is it what you expected? Was there any optimal split which surprised you?

The free parameters which we can vary are the maximum depth and the Gini impurity tolerance. Higher values for the maximum depth creates more refined, specific trees, as does a smaller Gini impurity tolerance. In this case, we are making the classifier more *complex*. As such, it will perform better on the data on which it is trained (it has learned the training data well), but perform worse on new, test data (it is not very generalizable). Keeping the Gini impurity tolerance at 0.1 and increasing the maximum depth yields the following interesting misclassification curves. What is the take away message?

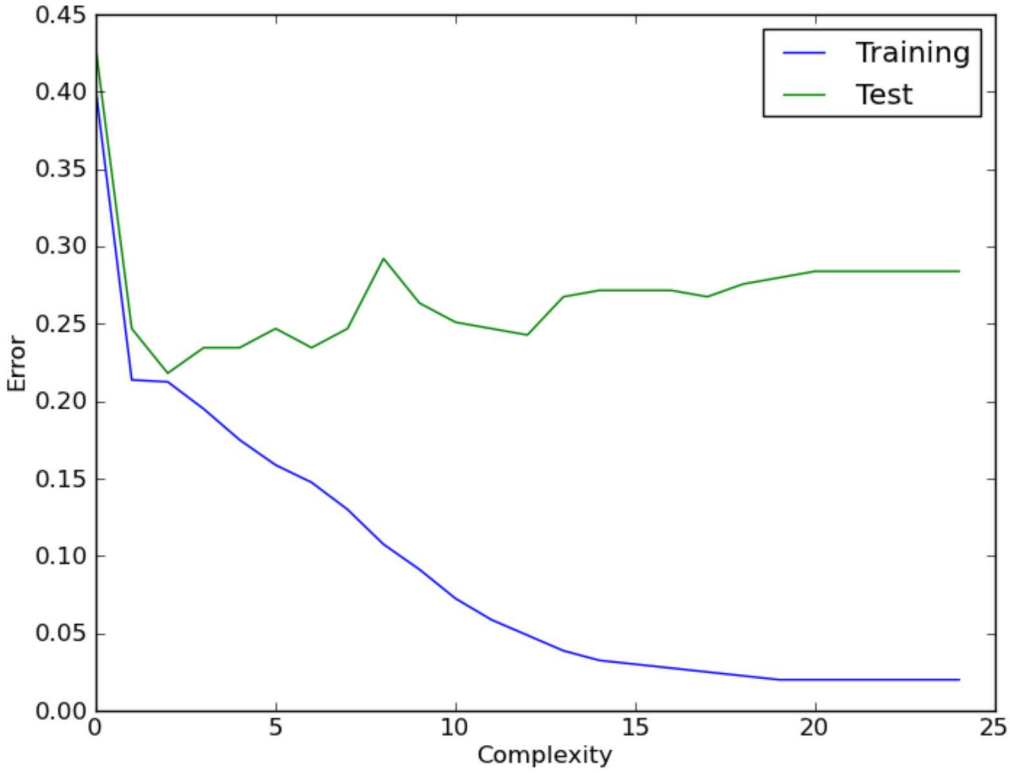

Figure 27.1: Misclassification rate on training data and test data with increasing complexity.

28 Random Forests

Lab Objective: *Understand how to build a random forest and use it to predict survival of Titanic passengers.*

A *random forest* is just what it sounds like–a collection of trees. Each tree is trained randomly, meaning that at each node, only a small, random subset of the attributes is available by which to determine the next split. Each trained tree in the forest casts a vote for the labeling of a new sample, and the sample is labeled according to the majority vote of the trees.

Your approach to the classification tree may have been sloppy, depending on how careful you were about odd cases (say trying to split a data set on gender when each sample is male). It doesn't affect us much when all the attributes are available on which to split, unless we grow the tree too deeply. However, with the random forests these odd cases crop up more frequently, as we only have a small subset of attributes to choose from. We need to be more careful then, and keep track of which attributes are still available to split on and only consider these.

Problem 1. Modify your code for classification trees so that we keep track of which attributes are available to split on at each node. We can only split on an attribute if it assumes two or more distinct values present in the data set. For example, in the Titanic data set, once we have split on gender, we can never split on it again in any descendant node, since one child data set will only have males and the other will only have females.

We must next add the randomness to our trees.

Problem 2. Modify your code for classification trees so that each tree is trained *randomly*, i.e. when determining the optimal split, randomly select a small subset of the available variables, and use them to split on. You should be able to specify the size of the subset. If the number of available variables is smaller than the size of the random subset, then terminate the node (make it a leaf node).

We can now train the whole forest.

Problem 3. Make a class *Forest* which trains a collection of random trees. Use the following implementation:

```
class Forest(data, targets, Gini, max_depth, num_trees, num_vars):
    """
    Train a collection of random trees.

    Parameters
    ----------
    data : ndarray of shape (n,k)
        Each row is an observation.
    targets : ndarray of shape (K,)
        The possible labels or classes.
    Gini : float
        The Gini impurity tolerance
    max_depth : int
        The maximum depth for the the trees
    num_trees : int
        The number of trees in the forest.
    num_vars : int
        The number of variables randomly selected at each node.
    """
    pass
```

Note that `num_vars` should be small, i.e. `num_vars` $\approx \sqrt{P}$ where P is the total number of attributes in the data set. The number of trees in the forest should be somewhat large, say greater than 100.

Problem 4. Write a method that assigns a label to a new sample, by considering the majority vote of the trees.

Let us reexamine the Titanic data set and see if we get any significant improvement.

Problem 5. Train a random forest on the Titanic data set, and for your inputs use `num_vars` = 2 and 100 trees. Let the Gini impurity tolerance be 0.1 and the maximum depth be 10. What is your misclassification rate? Was there any significant improvement?