

Decision Trees on Real Problems

Must consider the following issues:

1. How to assess the performance of the learning algorithm (or, more importantly, of the decision tree).
2. Inadequate attributes.
3. Noise in the data.
4. Missing values.
5. Attributes with numeric values.
6. Bias in attribute selection.

Note that we will need to be concerned about these for *all* machine learning problems – not just those to which we are applying a decision tree learning algorithm.

Assessing the Performance of the Decision Tree

Performance task is to predict the class of unseen examples.

Assessing the quality of the decision tree involves checking its classifications of test examples.

This requires that we leave some of our data out of the training phase, so that we can test appropriately. For certain problems where data are hard to obtain, this can be problematic.

Test Methodology

1. Collect a large set of examples.
 2. Divide it into two disjoint sets: the **training set** and the **test set**.
 3. Run the algorithm on the training set to generate a decision tree.
 4. Measure the classification accuracy of the tree on the test set.
 5. Repeat steps 2-4 for different splits of the data. We want to be sure that the performance we measure is not skewed due to a particularly odd distribution of examples in the training and test sets.
-

The Complexities of Assessing the “Goodness” of a Decision Tree

Say we have a 2-class performance task:

- What if the distribution of examples in the training and test sets is 98% class A and 2% class B? Guessing class A on all examples yields 98% accuracy, but this is not a useful way to classify examples.

- What if there is a difference between different kinds of classification errors? For example, what if we don't mind that A examples are classified as B, but we do care when class B examples are classified as A. A single measure of classification accuracy does not capture this information.
- What if there is a difference in the cost of measuring different attribute values?
- What if the decision tree is accurate, but is also very large? What if it is too large for an expert to read and interpret easily?

While we will focus on classification accuracy as our measure of a decision tree's performance, it is important to be aware of these other considerations.

Inadequate Attributes

Cause examples to appear inconsistent.

Result in larger decision trees as more splits are required to compensate for the lack of good attributes.

Noisy Data

Two types:

- incorrect attribute values
- incorrect class labels

Can be caused by many factors, including:

- faulty measurements
- subjective interpretation
- data entry problems

Additional complications:

- might not know whether data are noisy
- might not know how much noise is in the data

Dealing with Noise

Inconsistent examples cause the algorithm to fail to find a tree consistent with all training examples. That is, it will not find a tree with "pure" leaf nodes.

Solutions:

1. have each leaf report the majority class of its examples.

2. have each leaf report the estimated probability of each classification using the relative frequencies.

Dealing with Noise: The Problem of Overfitting

The problem of overfitting = the problem of finding meaningless regularity in the data.

This is a potential problem for all learning algorithms.

Solution for decision trees: decide that testing more attributes along a particular path will not improve the classification accuracy of the decision tree.
called **pruning**.

Reduced Error Pruning

Idea is to trim unnecessary leaves from the decision tree.

Pruning consists of taking a non-leaf node and replacing it with a leaf node, assigning it the most common classification of the training examples affiliated with that node.

We do so only if the pruned tree performs no worse than the original over a validation set – i.e., a **pruning set**. The pruning set is distinct from the training set.

The Reduced Error Pruning Algorithm:

After a full tree has been constructed from the training data,

In a postorder fashion traverse the decision tree, replacing a non-leaf node with a leaf node when the error rate of the leaf node is no worse than the error rate of the subtree.

Unknown Attribute Values

Say that a given training example $\langle x, f(x) \rangle$ has a missing attribute value. How will we handle this training example when trying to compute the information gain of the attribute at a node n in the decision tree?

Assign it a value for purposes of calculation!

Two simple options:

- assign it the value that is most common among training examples at node n .
- assign it the value that is most common among examples at the node that have the same classification.

Attributes with Numeric Values

For continuous, rather than discrete, attributes, we'll split the numeric range into two groups: values \leq threshold and values $>$ threshold. The important issue, of course, is how we select the threshold:

- Sort the examples by the values of the attribute.
- Search the examples, noting adjacent examples that belong to different classes.
In the following table, we would note class mismatches between the values of 15 and 20, and again between 24 and 36.

10	15	15	20	22	23	24	36
Class A	Class A	Class A	Class B	Class B	Class B	Class B	Class A

- Consider the average values at those transition points to be potential splits.
In the example here, those values would be 17.5 and 30.
- Evaluate each split found by applying the information gain formula. Choose the split that is best.
- Compare information gain for the best split against information gain for the remaining attributes.

An Example. Handling humidity in the golf data set

Consider the golf examples. In particular, we'll focus on those examples that have Outlook = Sunny. There are five such examples:

	Outlook	Temperature	Humidity	Wind	Class
Example 1	Sunny	85	85	False	Don't Play
Example 2	Sunny	80	90	True	Don't Play
Example8	Sunny	72	95	False	Don't Play
Example9	Sunny	69	70	False	Play
Example11	Sunny	75	70	True	Play

To consider the Humidity attribute as a possible test, we sort the examples by Humidity:

	Outlook	Temperature	Humidity	Wind	Class
Example9	Sunny	75	70	True	Play
Example11	Sunny	75	70	True	Play
Example 1	Sunny	85	85	False	Don't Play
Example 2	Sunny	80	90	True	Don't Play

Example8	Sunny	72	95	False	Don't Play
----------	-------	----	----	-------	------------

We note a class transition between Example 11 and Example 1. Their humidity values are 70 and 85, respectively. We average those values, to obtain a boolean split of the attribute:

$$\begin{aligned} \text{Humidity} &\leq 77.5 \\ \text{Humidity} &> 77.5 \end{aligned}$$

We then compute information gain for humidity and compare that to information gain for the other attributes (Temperature and Wind).

So why is the value in the decision tree 75???

Because C4.5 (which created the tree), chooses as the threshold the closest value from the entire data set that is less than or equal to the average.

Bias in Attribute Selection

The information gain criterion has a strong bias in favor of tests with many outcomes. Consider the following trivial example: Consider a medical diagnosis data set, where one of the attributes is social security number. This attribute will have many values – in fact, it will have as many values as there are examples. Splitting on this attribute will necessarily partition the examples into 1-element sets, each of which is “pure” in that it contains examples (one!) of only one class. So this attribute appears to be very informative – but in fact, it is not.

Gain Ratio criterion: attempt to normalize the apparent gain of attribute X.

Let

$$\text{split info (X)} = -\sum (|T_i| / |T|) * \log_2 |T_i| / |T|$$

where the sum is taken over the n different possible values of attribute X.

Then

$$\text{gain ratio (X)} = \text{gain(T, X)} / \text{split info (X)}$$

Select a test to maximize gain ratio, as long as the information gain is large – i.e., at least as great as the average gain over all tests examined.