Intro to Data Science:

Decision Tree Classifiers

Ed Podojil Data Engineer/Scientist, Animoto

Agenda

I. Decision Trees

II. Building Decision Trees

III. Optimization Functions

IV. Preventing Overfit

V. Implementing Decision Trees with Scikit-Learn

I. Decision Trees

Goals

- What is a decision tree?
- How is a decision tree represented?
- What are nodes?

What is a decision tree?

A non-parametric hierachical classification technique

non-parametric: no parameters, no distribution assumptions

hierarchical: consists of a sequence of questions which yield a class label when applied to any record

How is a decision tree represented?

Using a configuration of nodes and edges.

More concretely, as a multiway tree, which is a type of (directed acyclic) graph.

In a decision tree, the nodes represent questions (test conditions) and the edges are the answers to these questions.

What are the types of nodes?

The top node of the tree is called the **root node**. This node has 0 incoming edges, and 2+ outgoing edges.

An **internal node** has 1 incoming edge, and 2+ outgoing edges. Internal nodes represent test conditions.

A **leaf node** has 1 incoming edge and, 0 outgoing edges. Leaf nodes correspond to class labels.

Table 4.1. The vertebrate data set.

Name	Body Temperature	Skin Cover	Gives Birth	Aquatic Creature	Aerial Creature	Has Legs	Hiber-nates	Class Label
human	warm-blooded	hair	yes	no	no	yes	no	mammal
python	cold-blooded	scales	no	no	no	no	yes	reptile
salmon	cold-blooded	scales	no	yes	no	no	no	fish
whale	warm-blooded	hair	yes	yes	no	no	no	mammal
frog	cold-blooded	none	no	semi	no	yes	yes	amphibian
komodo dragon	cold-blooded	scales	no	no	no	yes	no	reptile
bat	warm-blooded	hair	yes	no	yes	yes	yes	mammal
pigeon	warm-blooded	feathers	no	no	yes	yes	no	bird
cat	warm-blooded	fur	yes	no	no	yes	no	mammal
leopard shark	cold-blooded	scales	yes	yes	no	no	no	fish
turtle	cold-blooded	scales	no	semi	no	yes	no	reptile
penguin	warm-blooded	feathers	no	semi	no	yes	no	bird
porcupine	warm-blooded	quills	yes	no	no	yes	yes	mammal
eel	cold-blooded	scales	no	yes	no	no	no	fish
salamander	cold-blooded	none	no	semi	no	yes	yes	amphibian

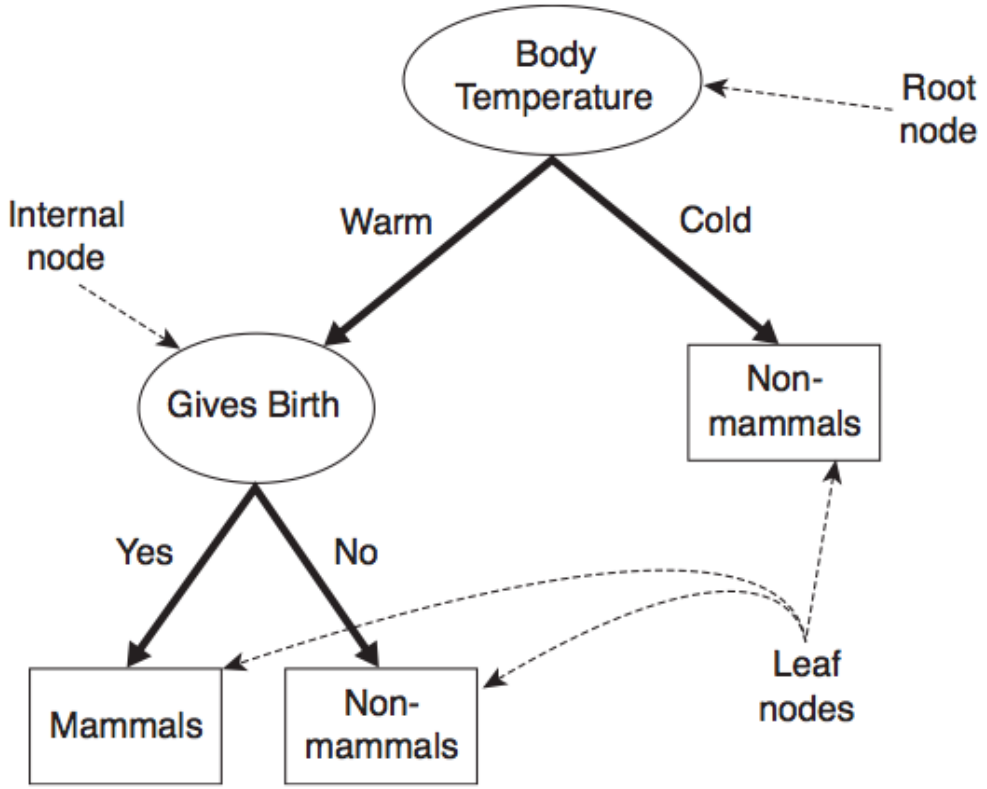

Figure 4.4. A decision tree for the mammal classification problem.

Review: Decision Trees

Classify data using a sequence of questions, assuming nothing about the data

Represented by nodes and splits

Interpreted with a top to bottom approach, using nodes to representing a root, test conditions, and class labels.

II. Building Decision Trees

Goals

- How do we build a decision tree?
- How do we find a practical solution that works?
- How do we interpret various data types in a decision tree?

How do we build a decision tree?

One possibility would be to evaluate all possible decision trees (eg, all permutations of test conditions) for a given dataset.

But this is generally too complex to be practical → O(2n).

How do we find a practical solution that works?

We use a **heuristic** algorithm.

That is, a fast solution good enough to solve the problem

The basic method used to build (or "grow") a decision tree is Hunt's algorithm.

This is a greedy recursive algorithm that leads to a local optimum.

greedy – algorithm makes locally optimal decision at each step

recursive – splits task into subtasks, solves each the same way

local optimum – solution for a given neighborhood of points

Hunt’s algorithm builds a decision tree by recursively partitioning records into smaller & smaller subsets.

The partitioning decision is made at each node according to a metric called purity.

A partition is 100% pure when all of its records belong to a single class.

Consider a binary classification problem with classes X, Y. Given a set of records D_t at node t, Hunt's algorithm proceeds as follows:

1. 1) If all records in d_t belong to class X, then t is a leaf node corresponding to class X.
2. 2) If d_t contains records from both classes, then a test condition is created to partition the records further. In this case, t is an internal node whose outgoing edges correspond to the possible outcomes of this test condition.
   These outgoing edges terminate in child nodes. A record d in d_t is assigned to one of these child nodes based on the outcome of the test condition applied to d.
3. 3) These steps are then recursively applied to each child node.

How do we partition the training records?

Test conditions can create binary splits

How do we partition the training records?

Alternatively, we can create multiway splits

How do we partition the training records?

For continuous features, we can use either method

How do we determine the best split?

Recall that no split is necessary (at a given node) when all records belong to the same class.

Therefore we want each step to create the partition with the highest possible purity.

We need an objective function to optimize!

Review

Decision trees are generated with a greedy algorithm that determines to find the best tree

Splits are generated based on either binary or multi splits of either discrete or continuous data

A decision tree is complete when all data belongs to a single partition

III. Optimization Functions

Goals:

- How do we measure purity of a split?
- How do we measure impurity?

We want our objective function to measure the gain in purity from a particular split.

Therefore we want it to depend on the class distribution over the nodes (before and after the split).

For example, let p(i|t) be the probability of class i at node t (eg, the fraction of records labeled i at node t).

Then for a binary (0/1) classification problem:

The minimum purity partition is given by the distribution:

$$p(0|t) = p(1|t) = 0.5$$

The maximum purity partition is given (eg) by the distribution:

$$p(0|t) = 1 - p(1|t) = 1$$

Some measures of impurity include

$$
\begin{aligned}
\text{Entropy}(t) &= -\sum_{i=0}^{c-1} p(i|t) \log_2 p(i|t), \\
\text{Gini}(t) &= 1 - \sum_{i=0}^{c-1} [p(i|t)]^2, \\
\text{Classification error}(t) &= 1 - \max_i [p(i|t)],
\end{aligned}
$$

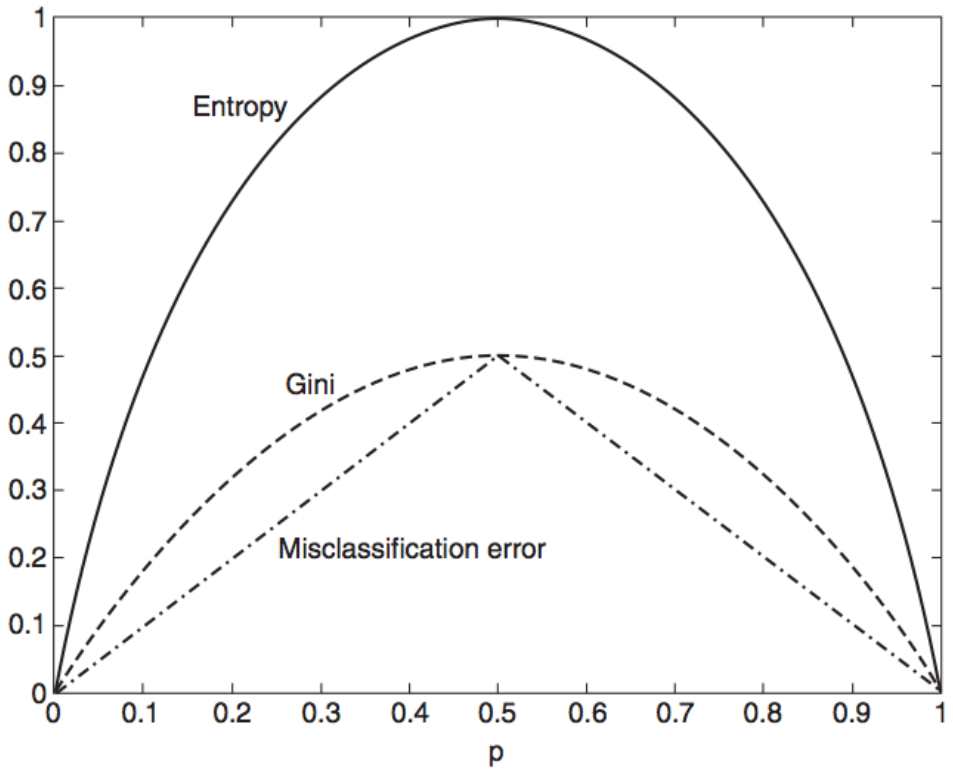

Figure 4.13. Comparison among the impurity measures for binary classification problems.

Generally speaking, a test condition with a high number of outcomes can lead to overfitting (ex: a split with one outcome per record).

One way of dealing with this is to restrict the algorithm to binary splits only (CART).

Another way is to use a splitting criterion which explicitly penalizes the number of outcomes (C4.5)

We can use a function of the information gain called the gain ratio to explicitly penalize high numbers of outcomes:

(Where p(vi) refers to the probability of label i at node v)

Impurity measures put us on the right track, but on their own they are not enough to tell us how our split will do.

Why is this true?

We still need to look at impurity before & after the split.

IV. Prevent Overfitting

Goals

- How do we determine when to stop optimization?
- What is pruning?

In addition to determining splits, we also need a stopping criterion to tell us when we're done.

For example, we can stop when all records belong to the same class, or when all records have the same attributes.

This is correct in principle, but would likely lead to overfitting.

One possibility is pre-pruning, which involves setting a minimum threshold on the gain, and stopping when no split achieves a gain above this threshold.

This prevents overfitting, but is difficult to calibrate in practice (may preserve bias!)

Alternatively we could build the full tree, and then perform pruning as a post-processing step.

To prune a tree, we examine the nodes from the bottom-up and simplify pieces of the tree (according to some criteria).

Complicated subtrees can be replaced either with a single node, or with a simpler (child) subtree.

The first approach is called subtree replacement, and the second is subtree raising.

Review

A decision tree is complete when all results can belong to a single partition

To prevent overfitting, we prune trees after compleition

Pruning is a form of generalization that simplifies a decision tree

Class break

Implementing Decision Trees with Python and SciKit

Group Discussion

Questions to ponder about:

What data seems useful for decision trees? What data doesn't?

How could you use decision trees alongside other classification algorithms?

Are decision trees discriminative or generative? Why do you think so?