

INTRO TO DATA SCIENCE

LECTURE 9: DECISION TREES

Paul Burkard

11/23/2015

LAST TIME:

- LOGISTIC REGRESSION

QUESTIONS?

I. DECISION TREES

II. BUILDING DECISION TREES

III. OPTIMIZATION FUNCTIONS

IV. PREVENTING OVERFITTING

HANDS-ON: DECISION TREES

I. DECISION TREES

Q: What is a decision tree?

A: A non-parametric hierarchical 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*

Q: How is a decision tree represented?

*A: Using a configuration of **nodes** and **edges**.*

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

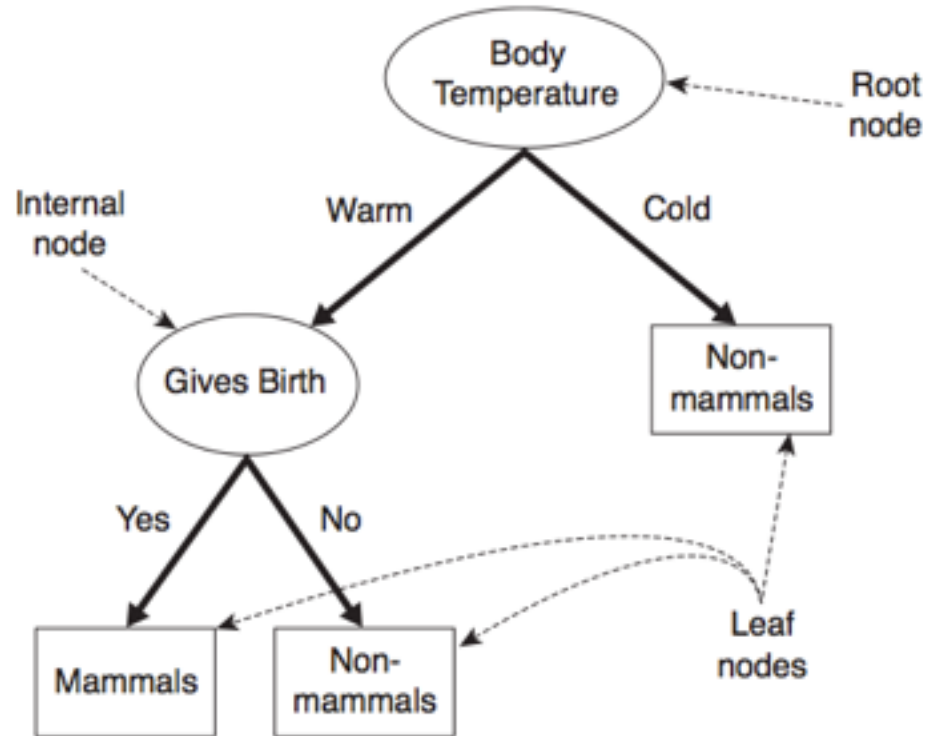


Figure 4.4. A decision tree for the mammal classification problem.

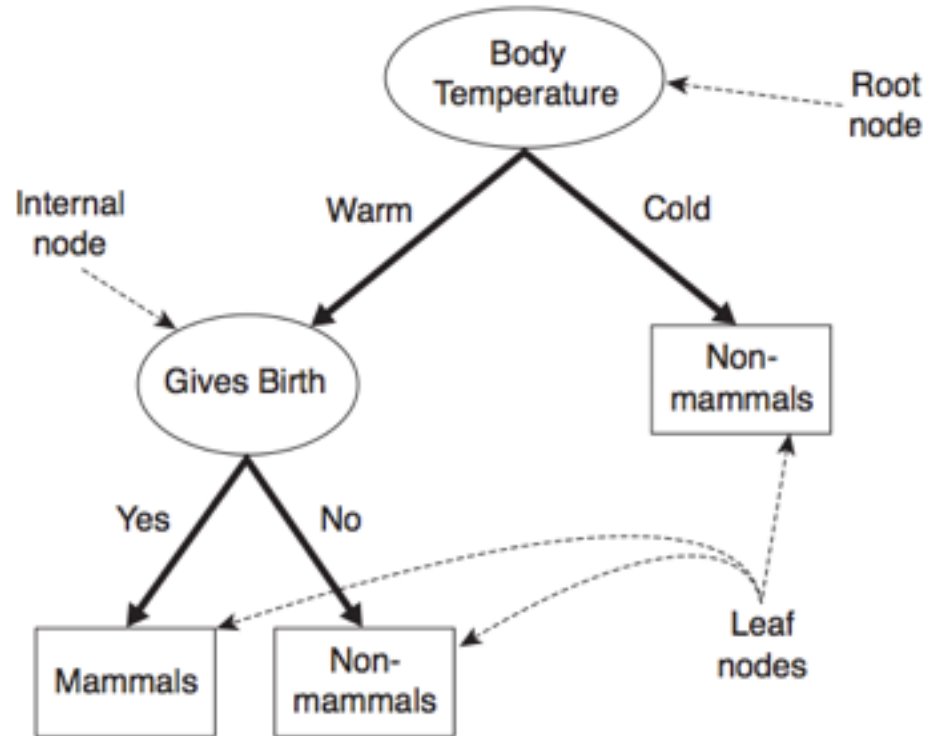
*The top node of the tree is called the **root node**. This node has 0 incoming edges, and 2+ outgoing edges.*

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

*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	Hibernates	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	cold-blooded	scales	yes	yes	no	no	no	fish
shark								
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



NOTE

Internal nodes represent test conditions which partition the records at that node.

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

II. BUILDING DECISION TREES

Q: How do we build a decision tree?

A: 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

Q: How do we find a practical solution that works?

*A: Use a **heuristic** algorithm.*

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) If all records in D_t belong to class X, then t is a leaf node corresponding to class X.*
- 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) Recursively apply 1 & 2 to each child node*

Q: How do we partition the training records?

A: There are a few ways to do this.

*Test conditions can create **binary splits**:*

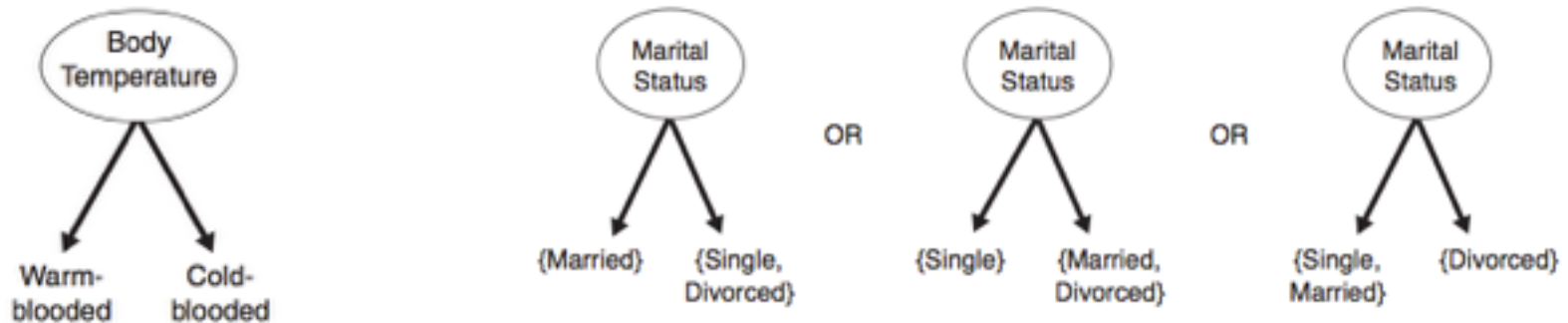


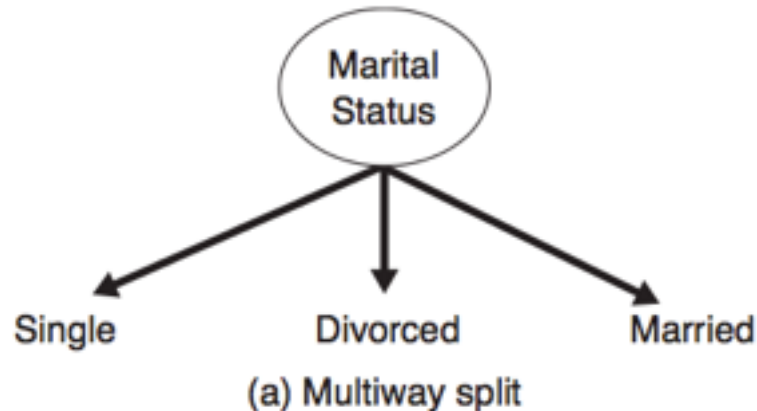
Figure 4.8. Test condition for binary attributes.

(b) Binary split {by grouping attribute values}

Q: How do we partition the training records?

A: There are a few ways to do this.

Alternatively, we can create multiway splits:



NOTE

Multiway splits can produce purer subsets, but may lead to overfitting!

Q: How do we partition the training records?

A: There are a few ways to do this.

For continuous features, we can use either method:

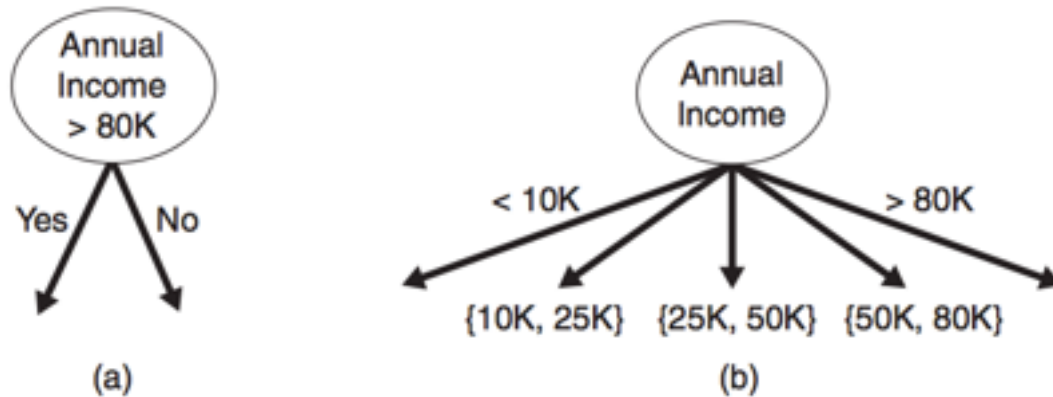


Figure 4.11. Test condition for continuous attributes.

NOTE

There are optimizations that can improve the naïve quadratic complexity of determining the optimum split point for continuous attributes.

Q: How do we determine the best split?

A: 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 (the most class-homogeneous splits).

*We need an objective function for **purity** to optimize!*

III. OBJECTIVE FUNCTIONS

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).

E.G. 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:

$$\text{Entropy}(t) = - \sum_{i=0}^{c-1} p(i|t) \log_2 p(i|t),$$

$$\text{Gini}(t) = 1 - \sum_{i=0}^{c-1} [p(i|t)]^2,$$

$$\text{Classification error}(t) = 1 - \max_i [p(i|t)],$$

1. The data set D has 50% positive examples ($\Pr(\text{positive}) = 0.5$) and 50% negative examples ($\Pr(\text{negative}) = 0.5$).

$$\text{entropy}(D) = -0.5 \times \log_2 0.5 - 0.5 \times \log_2 0.5 = 1$$

2. The data set D has 20% positive examples ($\Pr(\text{positive}) = 0.2$) and 80% negative examples ($\Pr(\text{negative}) = 0.8$).

$$\text{entropy}(D) = -0.2 \times \log_2 0.2 - 0.8 \times \log_2 0.8 = 0.722$$

3. The data set D has 100% positive examples ($\Pr(\text{positive}) = 1$) and no negative examples, ($\Pr(\text{negative}) = 0$).

$$\text{entropy}(D) = -1 \times \log_2 1 - 0 \times \log_2 0 = 0$$

As the data become purer and purer, the entropy value becomes smaller and smaller.

Note that each measure achieves its max at 0.5, min at 0 & 1.

NOTE

Despite consistency, different measures may create different splits.

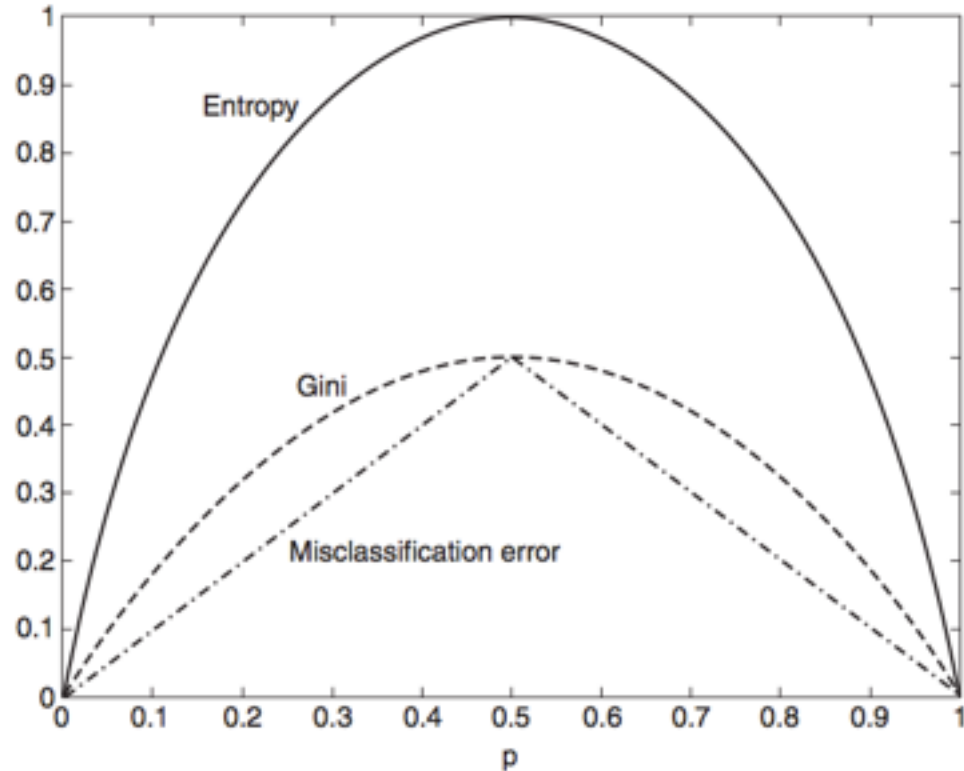


Figure 4.13. Comparison among the impurity measures for binary classification problems.

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

Q: Why is this true?

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

*We can make this comparison using the **gain**:*

$$\Delta = I(\text{parent}) - \sum_{\text{children } j} \frac{N_j}{N} I(\text{child } j)$$

(Here I is the impurity measure, N_j denotes the number of records at child node j , and N denotes the number of records at the parent node.)

*When I is the entropy, this quantity is called the **information gain**.*

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)

IV. PREVENTING OVERFITTING

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

$$\text{gain ratio} = \frac{\Delta_{info}}{-\sum p(v_i) \log_2 p(v_i)}$$

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

NOTE

This is a form of regularization!

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**.*

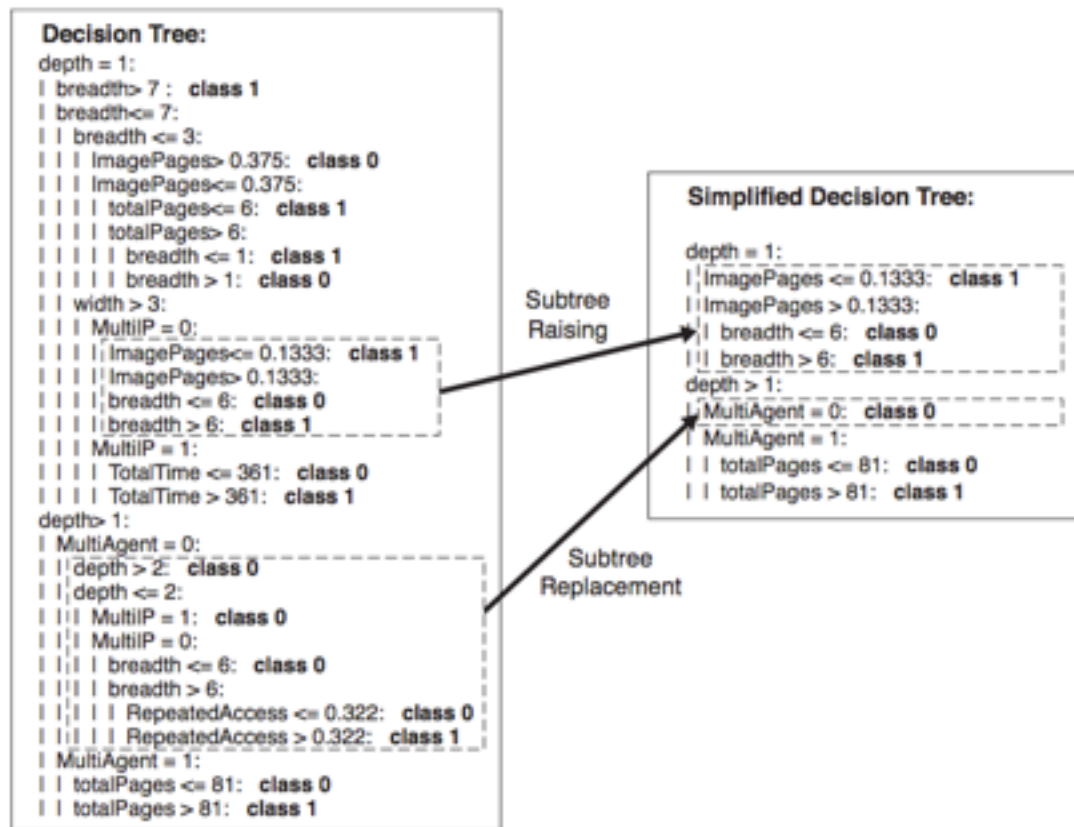


Figure 4.29. Post-pruning of the decision tree for Web robot detection.

INTRO TO DATA SCIENCE

HANDS-ON: DECISION TREES