Metis

DECISION TREES

I. DECISION TREES
II. BUILDING DECISION TREES
III. SPLITTING METRICS
IV. REGRESSION TREES
V. PREVENTING OVERFITTING
VI. STRENGTHS AND LIMITATIONS

I. DECISION TREES

Q: What is a decision tree classifier?

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**.

*Nodes represent questions (***test conditions)**

Edges are the answers to these questions.

EXAMPLE – DATA

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

source: http://www-users.cs.umn.edu/~kumar/dmbook/ch4.pdf

EXAMPLE – DECISION TREE

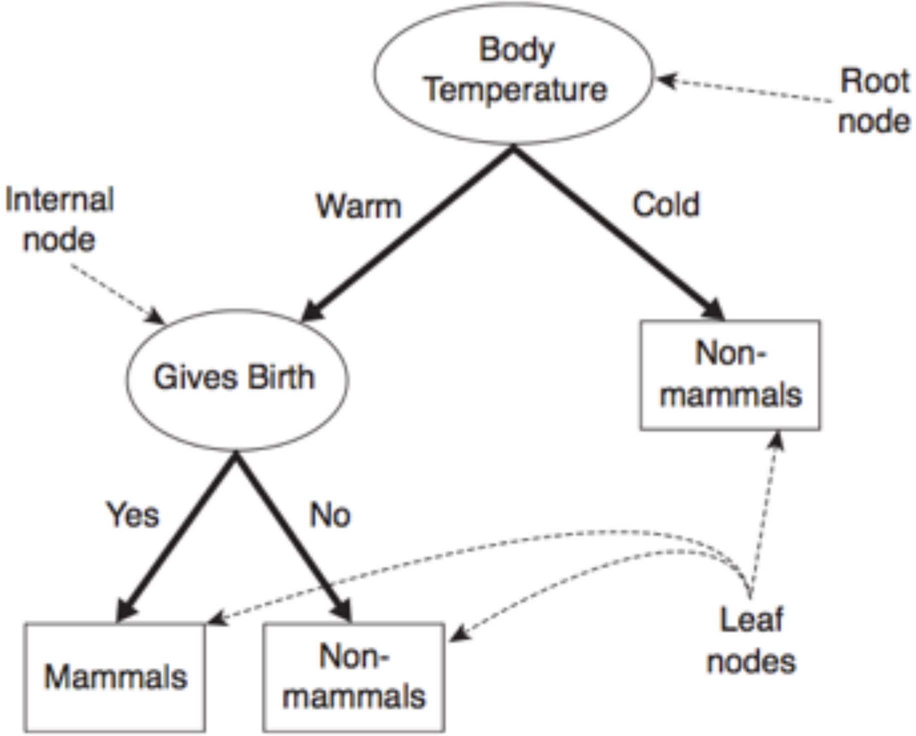


Figure 4.4. A decision tree for the mammal classification problem.

source: http://www-users.cs.umn.edu/~kumar/dmbook/ch4.pdf

Top node of the tree: **root node***.*

- *0 incoming edges, 2+ outgoing edges.*

An **internal node:**

- *1 incoming edge, and 2+ outgoing edges.*
- *Represent test conditions on the features. (if-statements)*

A **leaf node:**

- *1 incoming edge, 0 outgoing edges.*
- *Correspond to decisions on class labels.*

EXAMPLE – DECISION TREE

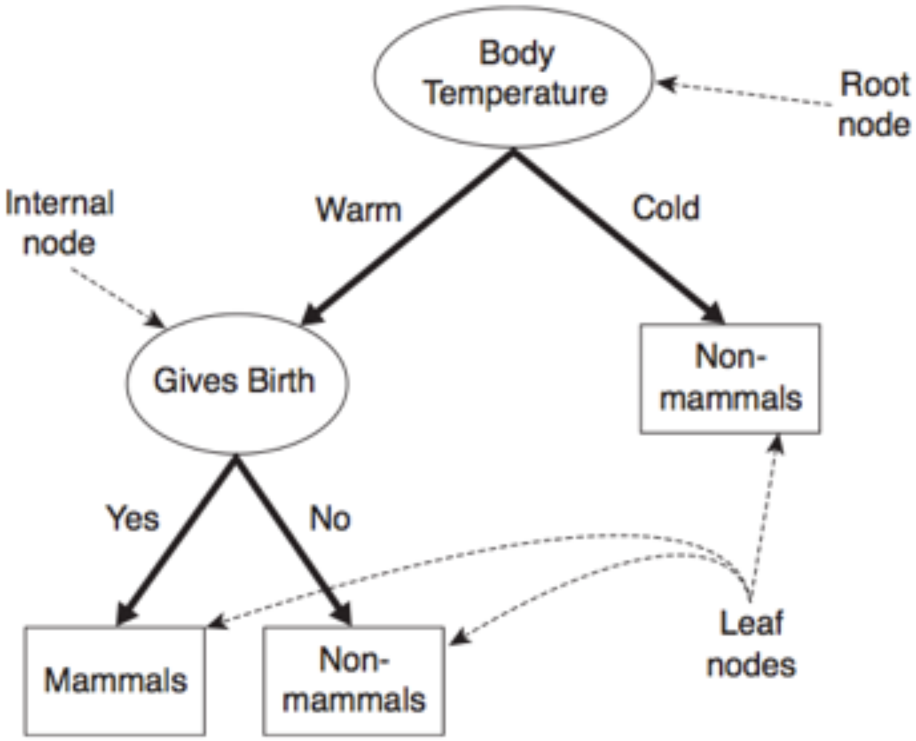


Figure 4.4. A decision tree for the mammal classification problem.

source: http://www-users.cs.umn.edu/~kumar/dmbook/ch4.pdf

NOTE

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

II. BUILDING DECISION TREES

Q: How do we build a decision tree?

A: Evaluate all possible decision trees (eg, all permutations of features) for a given dataset?

No! Too complex! Impractical!

Q: So...a practical solution that works?

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

B*asic method used to build 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*

Build a decision tree by recursively splitting records into smaller & smaller subsets, or **splits***.*

The splitting decision is made at each node according to some metric representing **purity***.*

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

Binary classification problem with **classes X, Y**. *Given* **set of records D_t** *at* **node t**:

1) *If All records in D_t are class X/Y: t is a leaf node with class X/Y.*

2) *If D_t has mixed classes: split into child nodes based on value of some feature(s).*

   t is an internal node whose outgoing edges correspond to the possible values of the chosen splitting feature(s).

   Outgoing edges terminate in **child nodes**.

   Record d is assigned to a child node based on the value of the splitting feature(s) for d.

3) *Recursively apply 1 & 2 to each child node*

Q: How do we know when to stop aka what's a leaf node?

A: Naively, when all nodes are 100% pure or have identical records

- *Not practical in reality*
- *Other Options:*
    - *Specify maximum tree depth*
    - *Specify Node impurity threshold*

Q: How do we split the training records?

A: A few options…

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

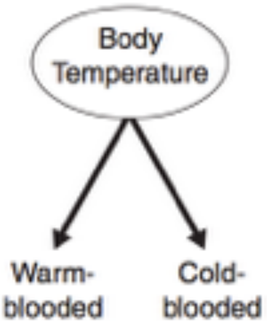
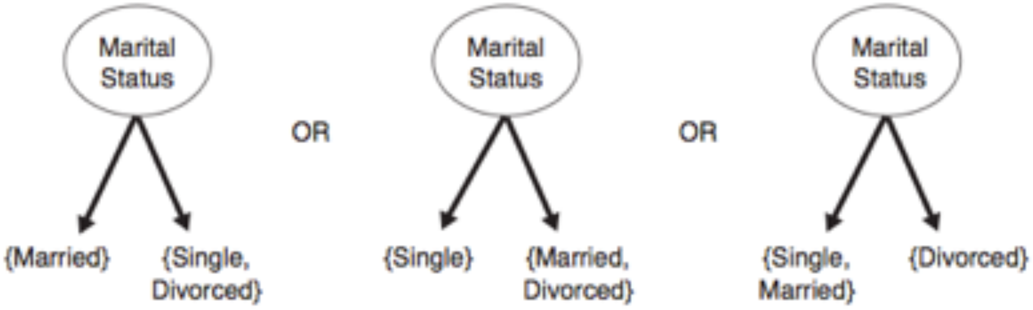


Figure 4.8. Test condition for binary attributes.

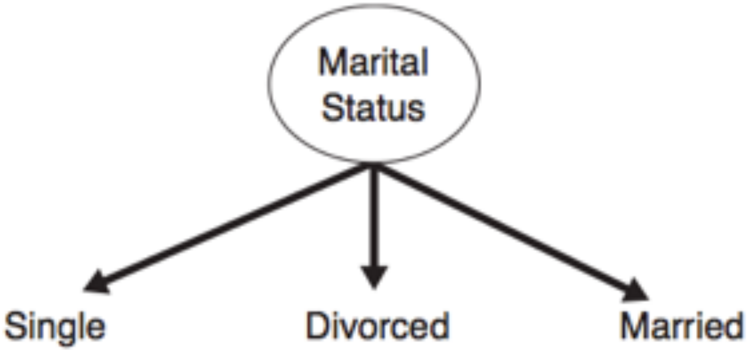


(b) Binary split {by grouping attribute values}

Q: How do we split the training records?

A: A few options...

Alternatively, we can create **multiway splits***:*



(a) Multiway split

NOTE

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

Q: How do we split the training records?

A: A few options…

For continuous features, we can use either method:

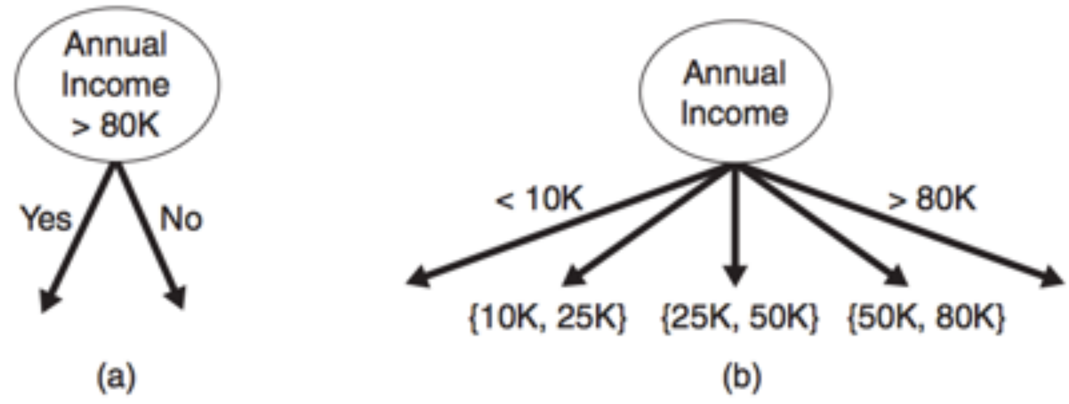


Figure 4.11. Test condition for continuous attributes.

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: no split necessary if records belong to same class.

Thus we want each step to create the split with the highest possible purity (the **most class-homogeneous splits***).*

We need a metric for **purity** *to optimize!*

III. SPLITTING METRICS

We want our metric 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 by the distribution:

$$p(0|t) = 1 - p(1|t) = 1,$$
$$OR$$
$$p(1|t) = 1 - P(0|t) = 1$$

Some measures of **impurity** *at node t over classes i include:*

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

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

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

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

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

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

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

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.

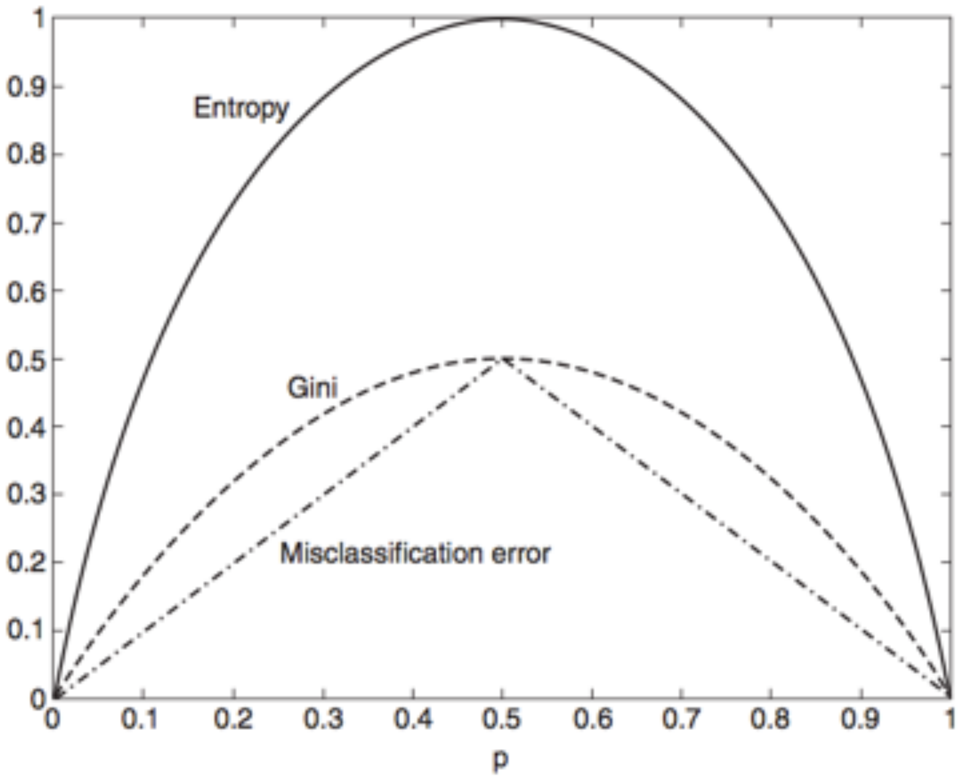


Figure 4.13. Comparison among the impurity measures for binary classification problems.

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***:*

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

I*: Impurity* N_j*: # of records at child node* j N*: # parent records*

Compare parent Impurity to a weighted sum of the impurity of the child nodes.

If I *is entropy, this quantity is called the* **information gain** *for a split.*

Generally, split with high # of outcomes causes overfitting (eg: a split with one outcome per record).

What can we do?

- *Restrict to binary splits only*
- *Explicitly Penalize # of outcomes in Splitting Metric*

IV. REGRESSION TREES

Q: How to extend to **regression** *problems?*

A: We need 3 things:

- *Different splitting metric for continuous targets*
  - *Entropy etc makes no sense for continuous targets!*
- *A stopping criterion*
  - *How do we know when we reach a leaf node?*
- *Decision rule for leaf nodes*
  - *What continuous output do we predict?*

Q: What is our splitting criterion?

A: Idea: **Variance Reduction**

- *Split on feature that yields* **least within-node variance**
- *What quantity can capture this? Variance Reduction!*
- *For Parent node S, child nodes S_t and S_f (binary split):*

$$I_V(N) = \frac{1}{|S|^2}\sum_{i\in S}\sum_{j\in S}\frac{1}{2}(x_i - x_j)^2 - \left(\frac{1}{|S_t|^2}\sum_{i\in S_t}\sum_{j\in S_t}\frac{1}{2}(x_i - x_j)^2 + \frac{1}{|S_f|^2}\sum_{i\in S_f}\sum_{j\in S_f}\frac{1}{2}(x_i - x_j)^2\right)$$

Q: What is our stopping criterion?

A:

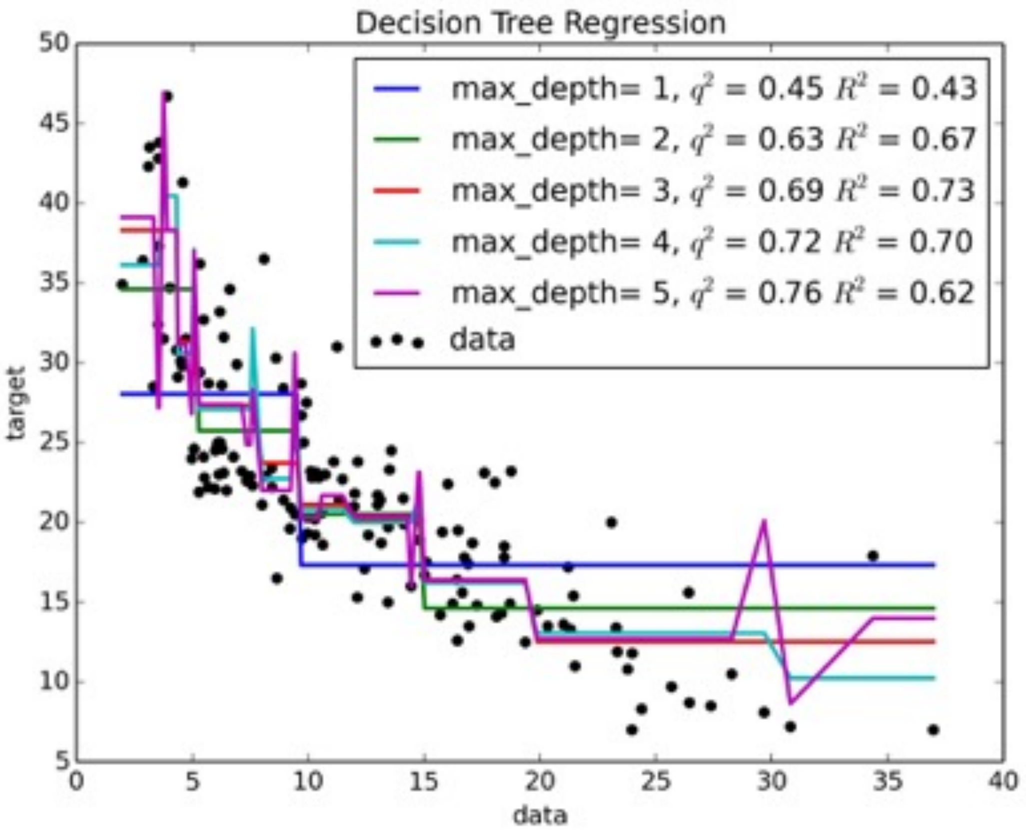
- *Option: Maximum Variance Threshold*
  - *Stop splitting if variance at all nodes below this threshold*
  - *This is a model hyper parameter (tune with CV!)*
- *Option: Maximum Tree Depth*
  - *Stop splitting when tree reaches certain size*
  - *This is a model hyper parameter (tune with CV!)*

Q: What is our evaluation criterion aka what end prediction do we report at a leaf node?

A: Could depend on context!

- *At a leaf node we could:*
  - *Use the mean of the records*
  - *Use the median*
  - *Something else?*
- *Let's assume the mean for now*

DECISION TREE REGRESSORS



V. 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: stop when all records belong to the same class, or when all records have identical features.

This is correct in principle, but will likely overfit.

One possibility: **pre-pruning**

- *Set a minimum threshold on the gain.*
    - *Stop when no split breaks this threshold.*
- *Set a max tree depth*

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

Alternatively: **post-pruning**

- *Build the full tree and perform* **pruning** *as a post-processing step.*

To prune a tree:

- *Examine the nodes from the bottom-up*
- *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**.

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**.

- *ID3 (precursor to C4.5)*
- *C4.5*
- *CART (Classification and Regression Trees)*
- *Others…*

Differ in splitting metric, stopping criterion, pruning strategy, etc

VI. STRENGTHS AND LIMITATIONS

Strengths:

- *Simple interpretation*
- *Little feature preprocessing, dummy variables, scaling, etc*
- *(Mostly) agnostic to feature data type*
- *Mostly robust/scalable*

Drawbacks:

- *Local optimum solution*
- *Unstable (aka high variance)*
  - **Overfitting!!!!**