

Intro. to machine learning (CSI 5325)

Lecture 6: decision trees

Greg Hamerly

Fall 2008

Some content from Tom Mitchell.

1 Other issues in decision tree learning

- Continuous valued attributes
- Attributes with many values
- Attributes with costs
- Unknown attribute values
- Other attribute value functions
- Regression trees
- Testing multiple attributes

Continuous Valued Attributes

Create a discrete attribute to test continuous

- $Temperature = 82.5$
- $(Temperature > 72.3) = true, false$

<i>Temperature:</i>	40	48	60	72	80	90
<i>PlayTennis:</i>	No	No	Yes	Yes	Yes	No

Continuous-value split point

How do we choose a splitting point for a continuous attribute?

- try them all?
- try a grid search?
- sort, then look for changes in the target value?
- other?

Testing multiple times

With a continuous-valued attribute, we might need to test it multiple times in the tree – even on the same branch.

Each test would partition the attribute values differently:

- Root: $Temp > 32$
- Level 1: $Temp < 50$

Of course, there's no benefit from multiple tests (along the same branch) of the an attribute using the same threshold value.

Attributes with many values

Problem: some attributes have lots of possible values, such as *Date*

- Imagine using $Date = Jun_3_1996$ as attribute.

If each example occurs on a different date value, then *Date* perfectly classifies all the examples!

Thus, attributes with lots of values will tend to have higher *Gain*

- but the attribute is not useful in practice

Attributes with many values

Attributes with lots of values will tend to have higher *Gain*

One approach: use *GainRatio* instead

$$\text{GainRatio}(S, A) \equiv \frac{\text{Gain}(S, A)}{\text{SplitInformation}(S, A)}$$

$$\text{SplitInformation}(S, A) \equiv - \sum_{i=1}^c \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}$$

where S_i is subset of S for which A has value v_i

Attributes with Costs

Consider

- medical diagnosis, *BloodTest* has cost \$150
- robotics, *Width_from_1ft* has cost 23 sec.

How to learn a consistent tree with low expected cost?

Attributes with Costs

How to learn a consistent tree with low expected cost?

One approach: replace gain by

- Tan and Schlimmer (1990)

$$\frac{Gain^2(S, A)}{Cost(A)}$$

- Nunez (1988)

$$\frac{2^{Gain(S,A)} - 1}{(Cost(A) + 1)^w}$$

where $w \in [0, 1]$ determines importance of cost

Unknown Attribute Values

What if some examples missing values of A ?

Use training example anyway, sort through tree

- If node n tests A , assign most common value of A among other examples sorted to node n
- assign most common value of A among other examples with same target value
- assign probability p_i to each possible value v_i of A
 - assign fraction p_i of example to each descendant in tree

Classify new examples in same fashion

Other attribute value functions

We have used entropy to measure the value of different attributes.

Any attribute value function $\phi(p, 1 - p)$ could be used that satisfies:

- $\phi(p, 1 - p) \geq 0$
- $\phi(1/2, 1/2) \geq \phi(p, 1 - p) \forall p \in [0, 1]$
- $\phi(0, 1) = \phi(1, 0) = 0$
- $\phi(p, 1 - p)$ increases for $p \in [0, 1/2]$, decreases for $p \in [1/2, 1]$

(Devroye, Györfi, and Lugosi, 1996)

This ϕ is specifically for 2 classes; we can generalize to more.

Other attribute value functions

Examples (for 2-class problems):

- Entropy: $\phi(p, 1 - p) = -p \log_2(p) - (1 - p) \log_2(1 - p)$
- Gini index: $\phi(p, 1 - p) = 2p(1 - p)$
- Misclassification error: $1 - \max(p, 1 - p)$

Real-valued targets

The target values we're training on have been classes – discrete values, without order.

What if we want to learn/predict real values with decision trees?

Such things are called regression trees (as opposed to classification trees).

Regression trees

Attributes could be discrete or continuous (just like in classification trees).

Prediction is a real value, like *Length_of_stay* or *Temp*.

How do we model this?

Leaf nodes

Leaves can represent:

- constant value, e.g. mean or median of training examples sorted to the leaf
- linear (or higher-order polynomial) model based on attributes

For now, we'll assume leaves are constant values.

Choosing attributes for splitting

How do we measure the goodness of an attribute split?

- entropy is unreliable for continuous values, or requires binning/density estimation – a pain
- simpler method: variance (statistic) of a leaf

For a set of target values X ,

$$\text{Var}(X) = \frac{1}{|X|} \sum_{x \in X} (x - \bar{x})^2$$

$$\bar{x} = \frac{1}{|X|} \sum_{x \in X} x$$

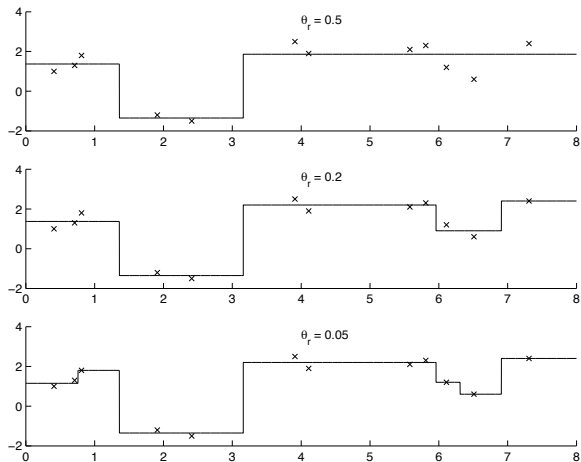
(other measures are possible, such as max error)

When do we stop splitting?

We expect noise in real-valued measurements

- it's likely that two measurements by an instrument will be slightly different, even for the same conditions

We can set a threshold for the 'purity' of a leaf – e.g. $\text{Var}(X) < \theta$.

Effect of θ 

Choosing θ

If θ is too small, we will see overfitting...

If θ is too large, we will see underfitting...

We could use the validation set to help us choose θ (by trying different values).

Testing one attribute per node

If we are using two real-valued attributes, what does the hypothesis look like?

Testing multiple attributes

It might be useful, especially in real-valued attributes, to test multiple attributes simultaneously.

How would this look?

How could we learn these types of decision nodes?

- problem: number of attribute combinations is huge
- need to restrict the search to make it feasible