Decision and Regression Tree Learning

CS 586

Prepared by Jugal Kalita

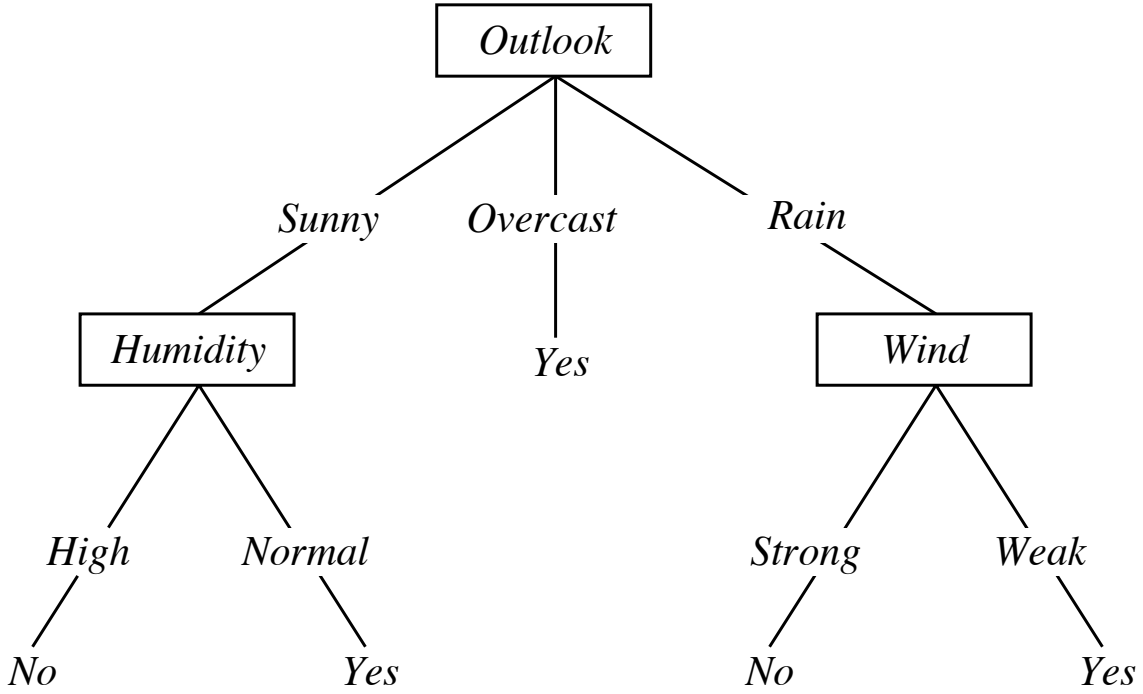
With help from Tom Mitchell's *Machine Learning*, Chapter 3
Alpaydin's *Ethem Introduction to Machine Learning*, Chapter 9
Jang, Sun and Mizutani's *Neuro-Fuzzy and Soft Computing*, Chapter 14
Dan Steinberg, CART: Classification and Regression Trees, 2009

Training Examples

Suppose we are given a table of training examples. Here *PlayTennis* is the concept to be learned.

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

A Possible Decision Tree for $PlayTennis$



From Page 53 of Mitchell

Decision or Regression Trees

- A decision tree solves a classification problem based on labeled examples. It is a supervised learning technique.
- Each internal node tests an attribute.
- Each branch corresponds to attribute value.
- Each leaf node assigns a classification.
- Although in this example, all attribute values are discrete, the values can be continuous as well.
- If the target attribute (i.e., the attribute to be learned, here $PlayTennis$) is continuous, the tree that results is called a regression tree. Otherwise, it is a decision tree.
- In a regression tree, the leaf nodes may have specific values for the concept or a function to compute the value of the target attribute.

What does a Decision or Regression Tree Represent?

- We can traverse the decision or regression tree from root down to classify an unseen example.
- A decision or regression tree represents a disjunct of conjuncts. The decision tree given earlier corresponds to the expression
  $(Outlook = Sunny \wedge Humidity = Normal)$
  $\vee\ (Outlook = Overcast)$
  $\vee\ (Outlook = Rain \wedge Wind = Weak)$
- A decision or regression tree can be expressed in terms of simple rules as well.

Representing a Decision or RegressionTree in Rules

- We can express the example decision tree in terms of rules.
- The last or default rule may or may not be necessary, depending on how rules are processed.
- Representing knowledge in readable rules is convenient.

  If $(Outlook = Sunny \wedge Humidity = Normal)$, then $PlayTennis$
  If $(Outlook = Overcast)$, then $PlayTennis$
  If $(Outlook = Rain \wedge Wind = Weak)$, then $PlayTennis$
  Otherwise, $\neg PlayTennis$

- The conclusion of each rule obtained from a decision tree is a value of the target attribute from a small set of possible values. The conclusion of each rule obtained from a regression tree is a value of the target attribute or a function to compute the value of the target attribute.

Basic Issues in Decision or Regression Tree Learning

- Under what situations can we use a decision tree for classification? Or a regression tree for regression?
- How do we build a decision or regression tree from the data?
- Which one of the many possible decision (or, regression) trees do we build and why?

When do we use Decision Trees

We use Decision Trees for Classification problems when the following conditions arise.

- Instances describable by attribute–value pairs; the number of possible values for an attribute is ideally small.
- Target function is discrete valued. The number of possible values is low.
- Disjunctive hypothesis may be required, i.e., we may have to explain what was learned.
- Possibly noisy training data
- The training data may contain missing attribute values

We will discuss decision trees first and then discuss regression trees.

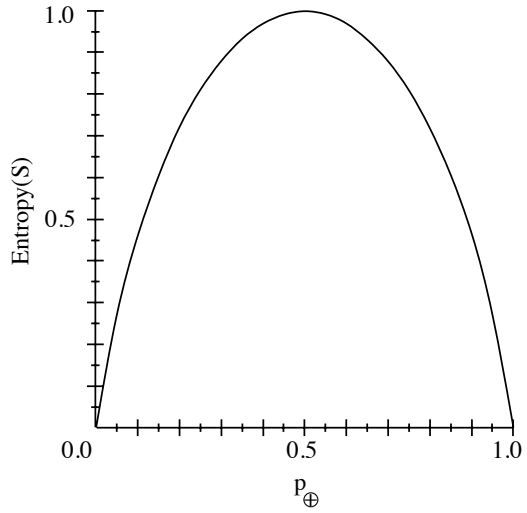
Examples of usage of Decision Trees

- Equipment or medical diagnosis
- Credit risk analysis
- Classifying living species based on characterisitics
- Stock trading
- Intrusion Detection
- Speech Processing
- Classifying Mammography data, etc.

Entropy in Data: A Measure of Impurity of a Set of Data

- Entropy in a collection of data samples represents the amount of impurity or chaos in the data.
- We will measure entropy by considering the classes data samples in a set belong to.
- Entropy in a set of data items should be 0 if there is no chaos or impurity in the data. In other words, if all items in the set belong to the same class, entropy should be 0.
- Entropy in a set of data belonging to several classes should be 1 if the data is equally divided into several classes. For examples, if we have two classes and our samples in a set are equally distributed in the two classes, entropy is 1.

Computing Entropy in Data



- Let T be a set of training examples belonging to two classes, $+$ and $-$.
- $p_{\oplus}$ is the proportion of positive examples in T
- $p_{\ominus}$ is the proportion of negative examples in T
- Entropy measures the impurity of T

$$Entropy(T) = -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus}$$

Entropy in Information Theory

- $Entropy(T)$ = expected number of bits needed to encode class ($\oplus$ or $\ominus$) of randomly drawn member of T (under the optimal, shortest-length code)
- Why?
  In Information theory: optimal length code assigns $-\log_2 p$ bits to message having probability p.
- So, expected number of bits to encode $\oplus$ or $\ominus$ of random member of T:

$$p_{\oplus}(-\log_2 p_{\oplus}) + p_{\ominus}(-\log_2 p_{\ominus})$$

$$Entropy(T) = -p_{\oplus}\log_2 p_{\oplus} - p_{\ominus}\log_2 p_{\ominus}$$

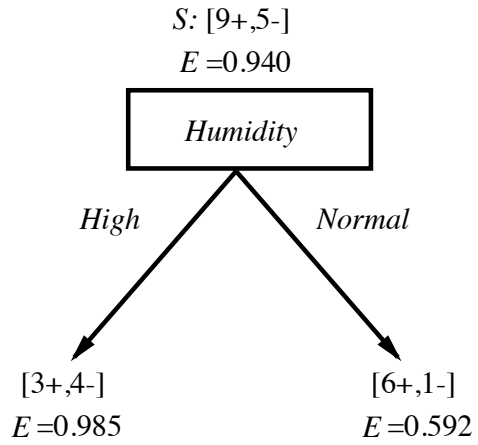
Information Gain

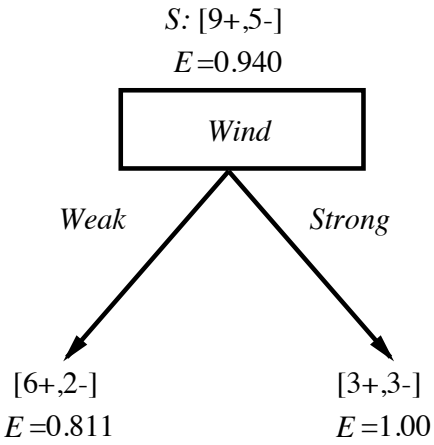
- Suppose we take the training examples in T and split T into subsets based on one of the attributes A.
- Let $Gain(T, A)$ = expected reduction in entropy due to splitting on A. It is the difference between entropies before splitting and after splitting on attribute A.

$$Gain(T, A) \equiv Entropy(T) - \sum_{v \in Values(A)} \frac{|T_v|}{|T|} Entropy(T_v)$$

Selecting the First Attribute

Which attribute is the best classifier?





Gain (S, Humidity)

= .940 - (7/14).985 - (7/14).592
= .151

Gain (S, Wind)

= .940 - (8/14).811 - (6/14)1.0
= .048

Selecting the First Attribute

$$Gain(T, Outlook) = 0.246$$

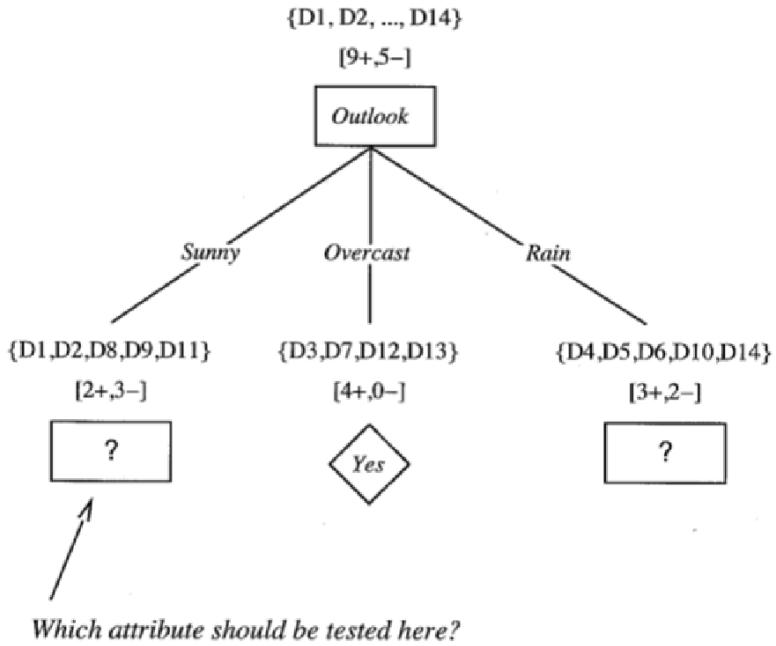
$$Gain(T, Humidity) = 0.151$$

$$Gain(T, Wind) = 0.048$$

$$Gain(T, Temperature) = 0.029$$

Choose $Outlook$ as the root attribute since it gives the most information gain.

Selecting the Next Attribute



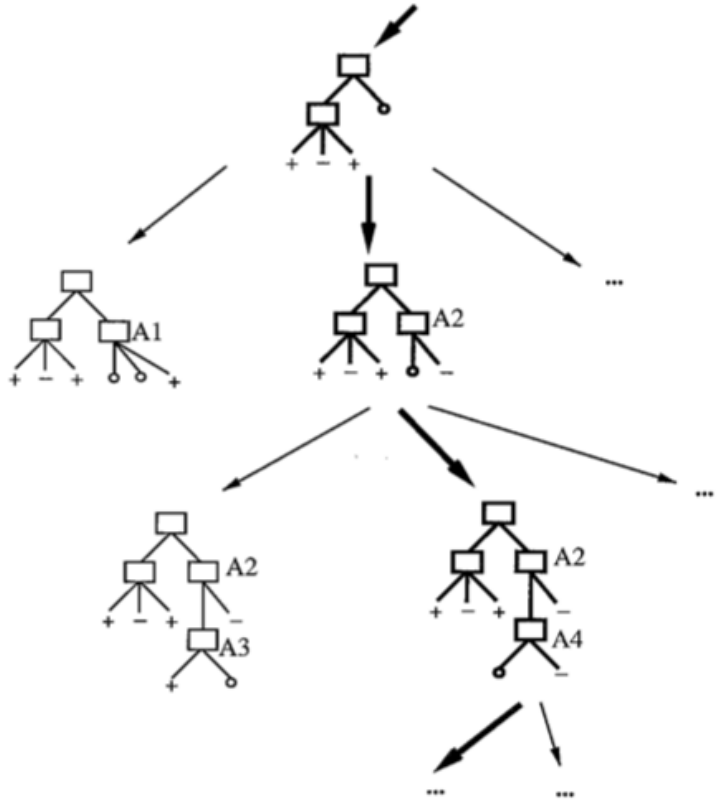
$$T_{Sunny} = \{D1, D2, D8, D9, D11\}$$

$$\begin{aligned} Gain(T_{sunny}, Humidity) &= .970 - (3/5)0.0 - (2/5)0.0 = .970 \\ Gain(T_{sunny}, Temperature) &= .970 - (2/5)0.0 - (1/5)0.0 = .570 \\ Gain(T_{sunny}, Wind) &= .970 - (2/5)1.0 - (3/5)0.918 = .019 \end{aligned}$$

Choose $Humidity$ as next attribute under $Sunny$ because it gives most information gain.

From Page 61 of Mitchell.

Hypothesis Space Search by a Decision Tree Learner



From Page 62 of Mitchell

Hypothesis Space Search by a Decision Tree Learner

- A decision tree learner searches the space of all decision trees that can be built from the data. The target function is in this space. Whether we find it or not is a different question.
- A learner maintains only a single current hypothesis. This is unlike some other learning techniques that maintain several hypotheses that are consistent with the training examples (e.g., Genetic algorithms keep consider several hypotheses at the same time). It is a greedy technique.
- The learner cannot backtrack. Once it makes a choice, it is stuck with it. It may get stuck in local minima.
- The learner searches through choices based on statistical computation using all training examples relevant at that point. Thus, it provides some robustness to noisy data or missing attribute values.

Inductive Bias in Decision Tree Learning

- Inductive Bias: We are searching in the space of trees or equivalently logical formulas that are disjuncts of conjuncts
- Preference for short trees, and for those with high information gain attributes near the root.
- Occam's razor: prefer the shortest hypothesis that fits the data. Why prefer short hypotheses?
  - Fewer short hypotheses than long hypotheses.
  - A short hypothesis that fits data unlikely to be coincidence.
  - A long hypothesis that fits data might be coincidence.

Some Issues in Decision and Regression Tree Learning

- Avoid overfitting the data.
- Incorporate continuous-valued attributes.
- Other measures for selecting attributes.
- Handle training examples with missing attributes.
- Handle attributes with different costs.
- Univariate, multivariate, or omnivariate trees.
- Regression trees. These are like decision trees, but the target attribute is continuous.
- Balancing subtrees.
- Testing the hypothesis obtained.

Overfitting the training examples

- Consider error of hypothesis h over
  - training data: $error_{train}(h)$
  - entire distribution $\mathcal{D}$ of data: $error_{\mathcal{D}}(h)$
- Hypothesis $h \in H$ **overfits** training data if there is an alternative hypothesis $h' \in H$ such that

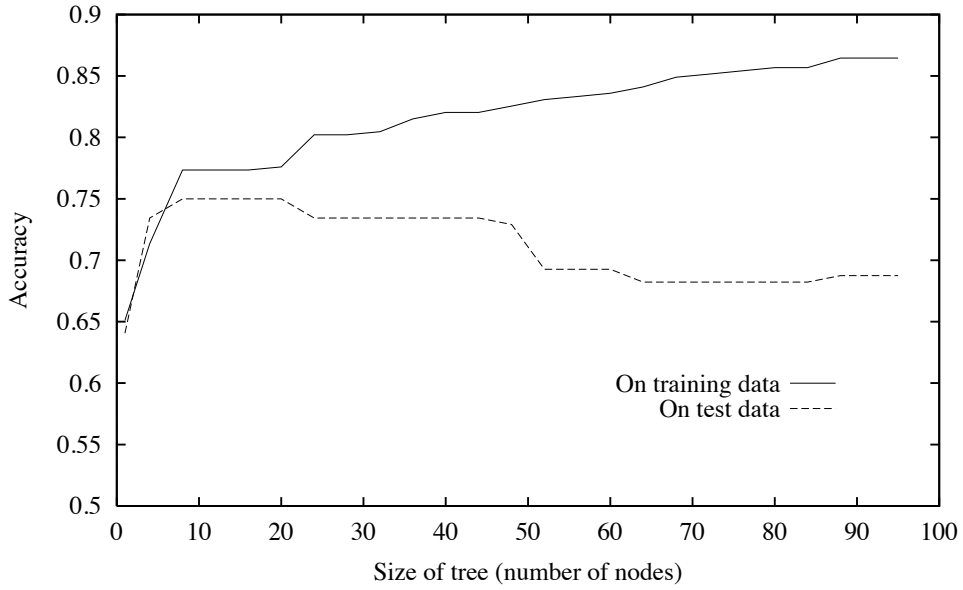
$$error_{train}(h) < error_{train}(h')$$

but

$$error_{\mathcal{D}}(h) > error_{\mathcal{D}}(h')$$

- In other words, h fits the training data better than h', but h does worse on the entire distribution of the data.

Overfitting in Decision Tree Learning



As new nodes are added to the decision tree, the accuracy of the tree over the training examples increased monotonically. However, measured over an independent set of training examples, the accuracy first increases and then decreases. From Page 67 of Mitchell.

How to Avoid Overfitting

Two possible approaches:

- Stop growing the tree before the tree fits the data perfectly. It is difficult to figure out when to stop.
- Grow full tree, then post-prune. This is the method normally used.

How to select "best" tree

- Use a separate set of examples, distinct from the training examples, to evaluate how post-pruning of nodes from the tree is working. This set is called the *validation set*.
- Use all the available data for training, but apply a statistical test to estimate whether expanding or pruning a particular node is likely to produce improvement beyond the training set. Quinlan (1986) uses a chi-square test to estimate whether further expanding a node is likely to improve performance over the entire distribution, not just the training examples.
- Use an explicit measure of complexity for encoding training examples and the decision tree, halting growth when this encoding size is minimized. It uses the so-called *Minimum Description Length principle*.

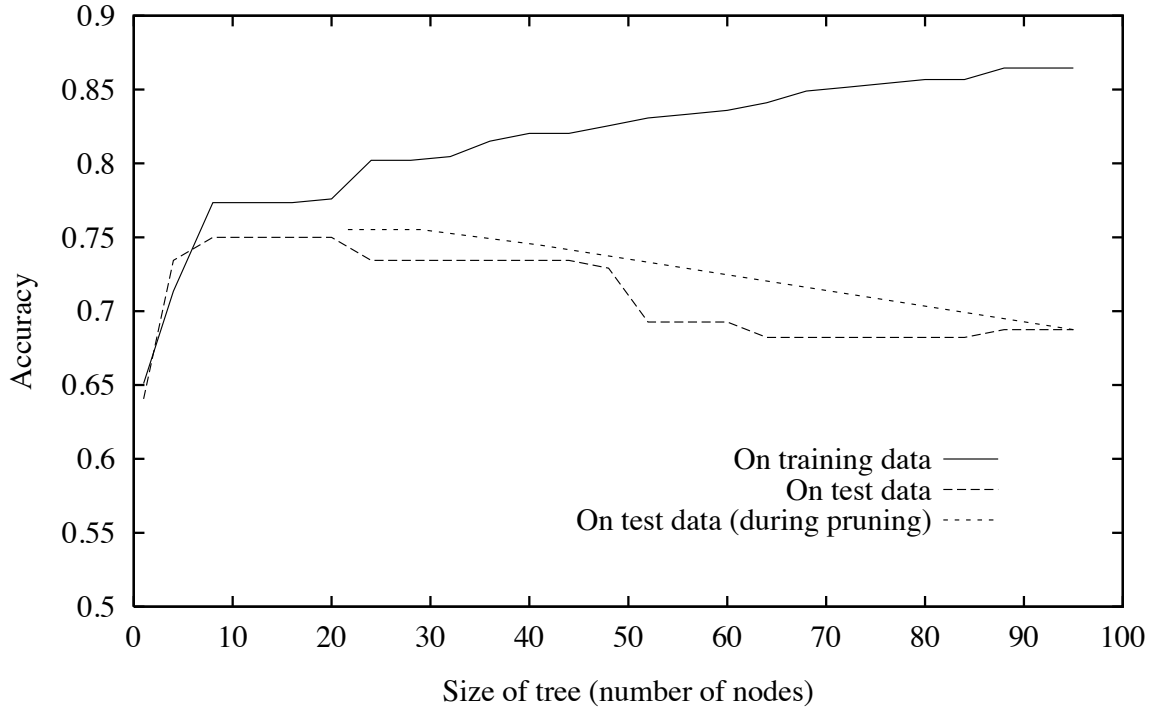
Reduced-Error Pruning

- It is one of the most common ways to prune a fully grown decision tree.
- Split data into $training$ and $validation$ set
- Do until further pruning is harmful:
  - Evaluate impact on $validation$ set of pruning each possible node (plus those below it); i.e., pick a node (in some traversal order, e.g., reverse depth-first order) and remove it with all branches below it, and examine how well the new tree classifies examples in the validation set.
  - Greedily remove the one that most improves $validation$ set accuracy

- produces smallest version of most accurate subtree

Note: Some people use three independent sets of data: training set for training the decision tree, validation set for pruning, and test set for testing on a pruned tree. Assuming the data elements are randomly distributed, the same regularities do not occur in the three sets.

Effect of Reduced-Error Pruning



From Page 70 of Mitchell.

Rule Post-Pruning

- Grow the tree till overfitting occurs.
- Convert tree to equivalent set of rules by creating one rule for each path from root to a leaf.
- Prune i.e., generalize each rule independently of others by removing any precondition that results in improving estimated accuracy over a test/validation set. Accuracy is measured by what proportion of examples in the test/validation set does the rule cover.
- Sort final rules by estimated accuracy; consider the rules in this sequence.

Pruning Rules

Suppose we have the rule:

IF $(Outlook = Sunny)\ and\ (Humidity = High)$
THEN $PlayTennis = No$.

- We can consider dropping one antecedent, and then the second to see if the accuracy doesn't become worse.
- In this simple case, dropping an antecedent may not be helpful, but if we have many antecedents (say 5-10 or more), dropping one or more may not affect performance.
- When we evaluate the performance of a rule, we can be pessimistic; it will perform worse with real data than with the training/test/validation data. We can assume that the data and thus the rule's accuracy will be distributed according to a certain distribution and we assume that the rule's accuracy will be within a certain confidence level (say 95%).

Why Convert Decision Tree to Rules before Pruning?

- Converting to rules allows distinguishing among the different contexts in which a decision node occurs. Pruning decision regarding an attribute can be made differently for each path (or equivalently, rule). In contrast, if the tree itself is pruned, the only two choices are to remove the node or not remove it.
- Converting to rules removes the distinction between attribute tests that occur near the root of the tree and those that occur near the leaves. In contrast, if we prune nodes in a tree, we have to make a decision regarding reorganizing the tree if a non-leaf node is pruned and nodes below are kept.
- Rules are easy for people to understand. When rule preconditions are pruned, people understand what is actually being pruned better.

Other Splitting Rules for Discrete Variables

- The Gini Diversity Index for a set of training examples T is frequently used.

$$Gini(T) = \sum p_i \ (1 - p_i) = 1 - \sum_{j=1}^{J} p_j^2$$

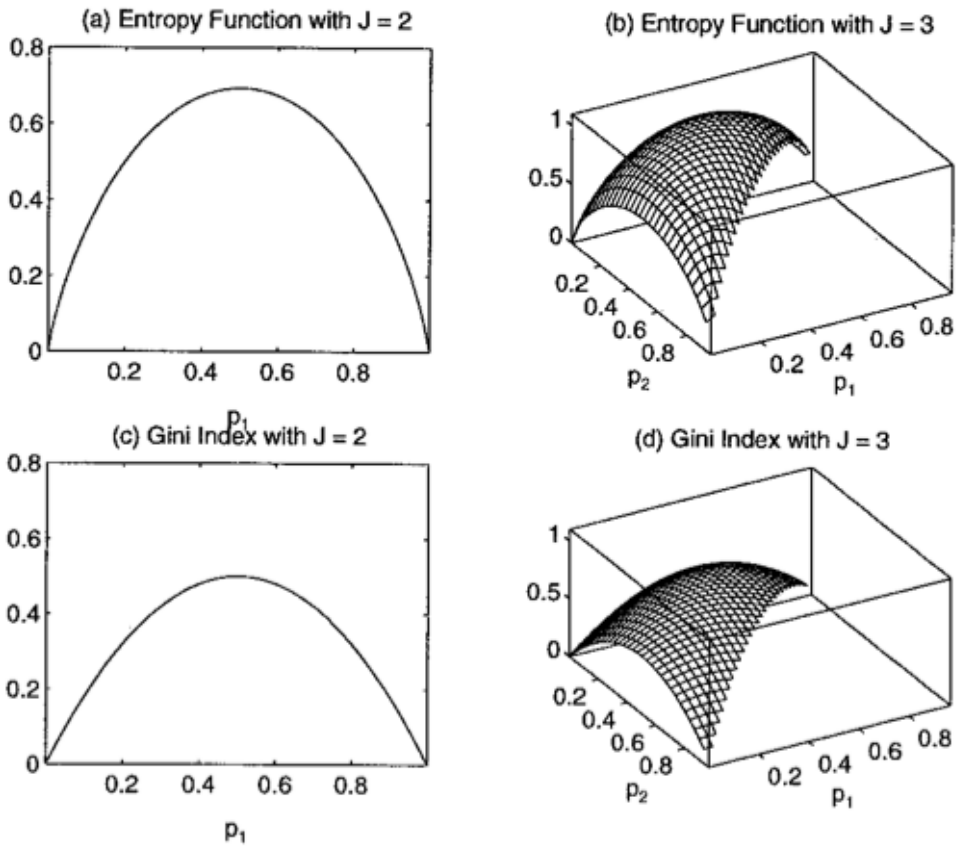
  where p_i is the probability (proportion) of class i occurring at a certain node in the tree, j is the total number of classes into which examples can belong.

- The split selected maximizes the value of the difference between Gini impurity before splitting and Gini impurity after splitting.

$$\begin{aligned} GiniGain(T, A) \quad = \quad & Gini(T) - p(T_l)\ Gini(T_l) \\ & -p(T_r)\ Gini(T_r) \end{aligned}$$

  where $p(T_l)$ and $p(T_r)$ are the proportion of examples in the left and right subtrees after split.

Comparing Entropy and Gini Index of Diversity



As we can see, the two functions are not very different. Here J is the total number of classes. From Page 409 of Jang et al.

Other Splitting Rules for Discrete Variables

- This page contains a bunch of diversity indices: `http://en.wikipedia.org/wiki/Diversity_index`

- True diversity (The effective number of types), Richness, Shannon index, Simpson index, Inverse Simpson index, Gini-Simpson index, Berger-Parker index, Renyi entropy

Continuous Valued Attributes

We need to have a method to convert continuous values to discrete values. Once we have done that, there are at least two choices.

- The same entropy reduction computation can be used for node splitting.
- Use a different splitting rule for continuous attributes. Such a splitting rule can use the idea of variance.
- Both methods have their adherents. There must be some paper comparing performance somewhere.

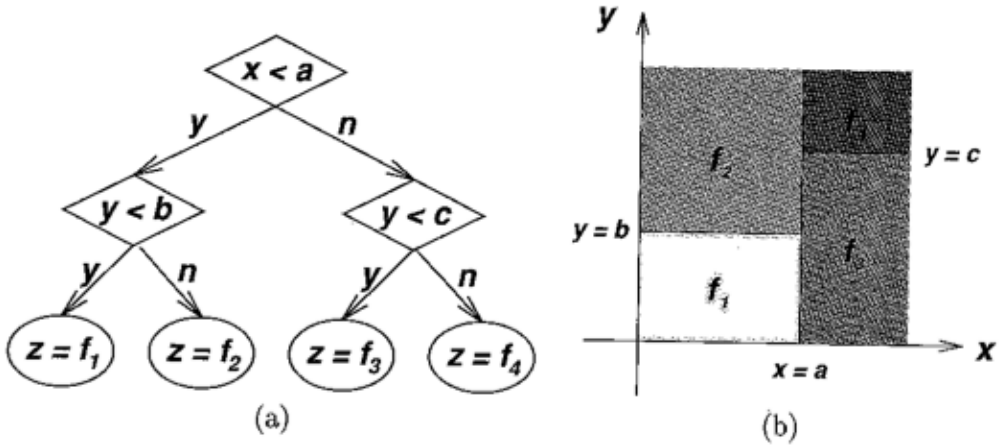
Discretizing a Continuous Valued Attribute

Consider independent attribute temperature and the target attribute PlayTennis in the table below.

Temperature:	40	48	60	72	80	90
PlayTennis:	No	No	Yes	Yes	Yes	No

- We may consider creating discrete variables considering where the target variable changes value: between 48 and 60, and between 80 and 90.
- For example, we can create a discrete variable that correspond to $Temperature > 40$ and $Temperature \leq \frac{48+60}{2}$
- This variable takes true or false values.
- One can use other methods: binary search type, statistical searches.

Continuous Valued Attribute and Space Partitioning



From Page 409 of Jang et al.

- It is easy to see that discretizing a continuous valued item and creating a decision tree using such a discretized attributes in a decision tree leads to partitioning of the search space as we go down the tree.

Alternative Splitting Rule for Continuous Attribute

- Assume we are allowed to create only binary splits for continuous attributes.
- Let the variance of the target variable at the root node be $Var(T)$. Let the variance of the target variable at the proposed left and right subtrees be $Var(T_l)$ and $Var(T_r)$.
- Let the proportion of samples in the left and right subtrees be $p(T_l)$ and $p(T_r)$
- One splitting rule can be: Choose the split that makes the reduction in variance the most, i.e., makes the value of $Var(T) - p(T_l)\ Var(T_l) - p(T_r)\ Var(T_r)$ the highest.

Attributes with Many Values

- If attribute has many values, the approach where we reduce entropy the most in a split, will select such an attribute.
- Imagine using $Date = Jun\_3\_1996$ or $SSN = 111223333$ as attribute.
- Assume each row has a unique attribute value for this attribute.
- Then splitting n samples into n branches below the parent will make each branch pure, but containing only one sample in each split. This is not a good way to branch.

Attributes with Many Values

- Use a measure for splitting different from information gain.
- One approach: use $GainRatio$ instead

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

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

  where T_i is subset of T for which A has value v_i.
- $GainRatio(T, A)$ is the gain in entropy due to splitting using attribute A: We used this before.
- $SplitInformation(T, A)$ is the sum of the entropy of the nodes after splitting on attribute A.

- The value of the denominator is high for attributes such as $Date$ and SSN.
- If an attribute is like $Date$, where each date is unique for each row of data, then $SplitInformation(T, A) = \log_2 n$ if there are n attributes.
- If an attribute is boolean and divides the set exactly in half, the value of $SplitInformation = 1$.
- Use attributes which have high gain ratio. It makes gain ratio low for attributes like $Date$.

Unknown Attribute Values

- What if some examples missing values of A?
- When computing $Gain(T, A)$ at a certain attribute,
  - Assign most common value of A among other examples sorted to node n
  - Or, Assign most common value of A among other examples with same target value
  - There are other more complex methods using probability and statistics.
- Classify new examples in same fashion

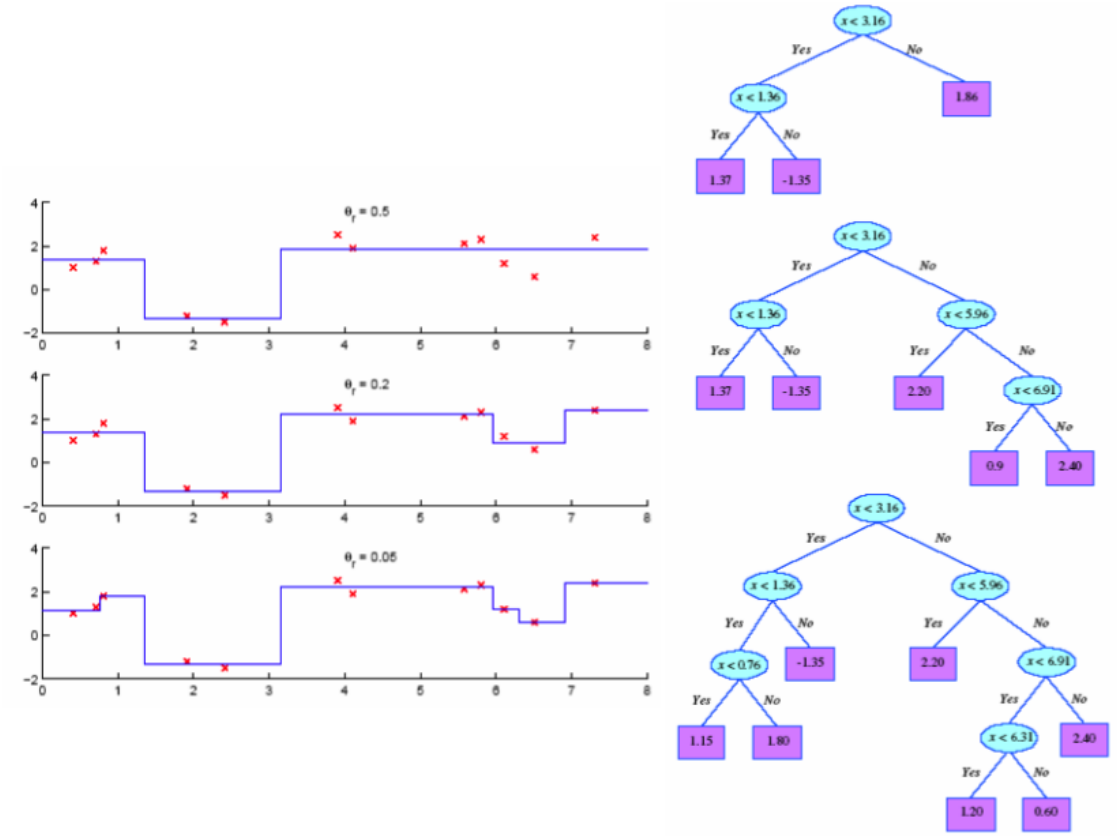
Attributes with Differing Costs

- There may be differing costs associated with attributes. E.g., in medical diagnosis, attributes such as Age, $Race$, $BloodTest$. $XRay$, $Temperature$, $BiopsyResult$, etc., may not have the same costs. The costs may be monetary, or to patient comfort.
- We want to decision trees that use low-cost attributes when possible, using high-cost attributes only when needed.
- A simple approach: Divide the $Gain$ value by the cost of the attributes, i.e., compute $\frac{Gain(T,A)}{Cost(A)}$ when deciding which attribute to choose to make a split in the tree.
- Others have used formulas such as $\frac{Gain^2(T,A)}{Cost(A)}$ or $\frac{2^{Gain(T,A)}-1}{(Cost(A)+1)^w}$ where $w \in \{0, 1\}$ in various situations to account for varying costs of attributes.

Regression Trees

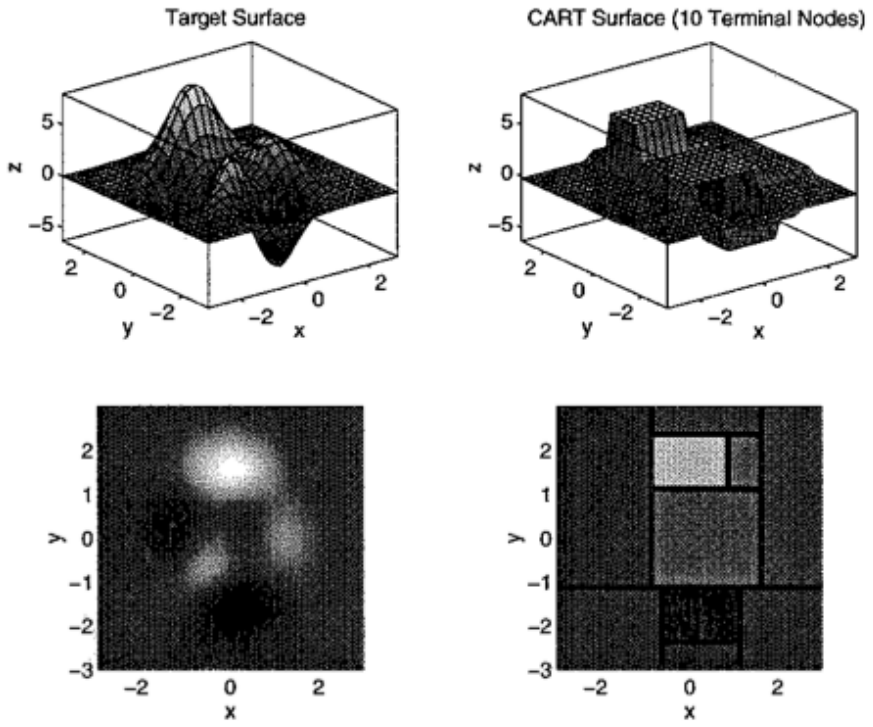
- In a regression tree, the target attribute has a continuous value.
- If all the independent attributes have continuous values, then a regression tree does what least-squares or other types of regression do. Instead of approximating the data points by a single function as in regression, a regression tree does it in steps.
- The leaf nodes of a regression tree can be constants or functions of the independent (non-target) attributes.

Growing Regression Trees



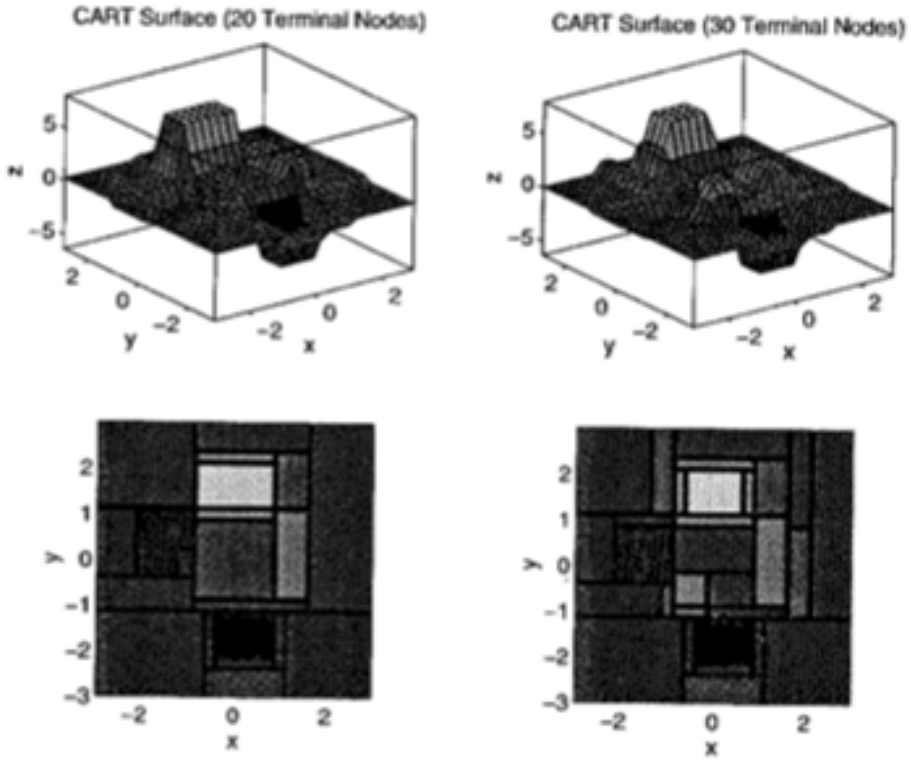
As the regression tree is grown, the learned function which is discontinuous, becomes better defined. From Alpaydin, Chapter 13's instructor slides. The output y is a function of one variable only: x. From Alpaydin, instructor slides.

Growing Regression Trees



As the regression tree is grown, the learned function which is discontinuous, becomes better defined. The output z is a function of two variables x and y. The output has 10 nodes now. From Jang et al. page 413.

Growing Regression Trees

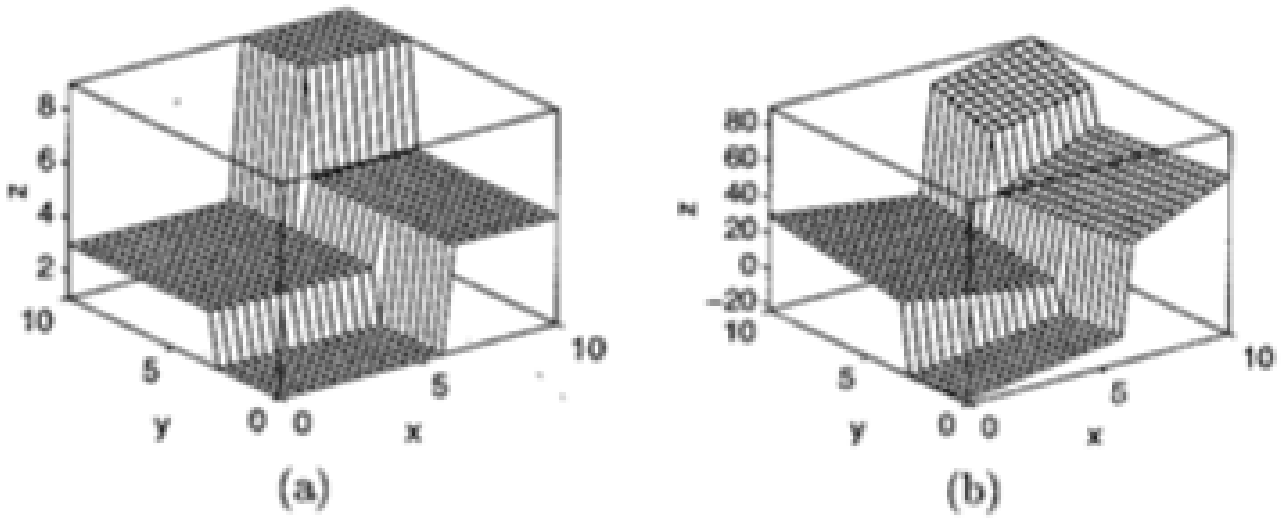


As the regression tree is grown, the learned function which is discontinuous, becomes better defined. The output z is a function of two variables x and y. The tree has 20 nodes now. From Jang et al. page 414.

Pruning Regression Trees

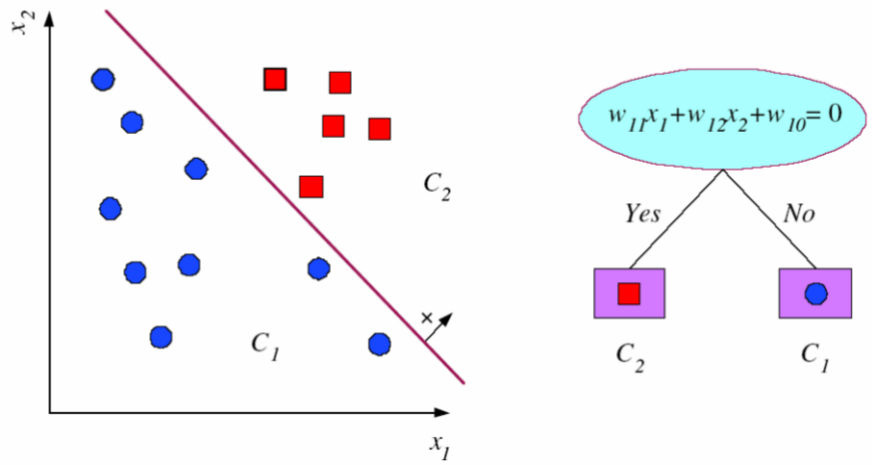
- A regression tree is grown to an overfitted size just like a decision tree.
- The tree is then pruned. First prune any node whose removal doesn't reduce performance of the tree on a test or validation set. Rearrange the tree if necessary. Repeat the pruning process till the tree's performance doesn't deteriorate.
- As a tree is pruned, the output regression function becomes less smooth (opposite of what we see in the regression tree growing examples).

Constant Leaves Vs. Linear Equations in Leaves



The Input-Output surfaces of Regression Trees with terminal nodes that are constant-valued vs. terminal nodes that store functions of the input. From Jang et al., Page 406.

Mutlivariate Decision/Regression Trees



- A node can have a condition composed of more than one feature in linear or non-linear combination. From Alpaydin, instructor slides.
- Having very complex conditions defeats the one purpose of building a decision tree which is creating classification rules that can be explained.

Omnivariate Decision/Regression Trees

- An omnivariate tree allows both univariate or multivariate nodes, both linear and non-linear.
- Different types of nodes are compared with statistical tests and the best one chosen.
- A simpler node is chosen unless a complex one gives much better accuracy.
- Usually more complex nodes occur near root when we have more data and simpler nodes occur near leaves when we have less data to work with.

Balancing subtrees

- Many machine learning algorithms do not perform well if the training data is unbalanced in class sizes.
- However, in real life, unbalanced classes are common. One needs a way not to be biased by the bigger classes.
- The CART program uses some techniques to keep the classes balanced as much as possible.