Examples of Classification Tasks

- An emergency room in a hospital measures 17 variables (e.g. blood pressure, age etc.) of newly admitted patients. A decision has to be taken whether to put the patient in an intensive-care unit. Due to the high cost of ICU, those patients who may survive more a month are given higher priority. The problem is to predict high-risk patients and discriminate them from low-risk patients.
- A credit card company typically receives hundreds of thousands of applications for new cards. The application contains information regarding several different attributes, such as annual salary, any outstanding debts, age etc. The problem is to categorize applications into those who have good credit, bad credit, or fall into a gray area (thus requiring further human analysis).

Examples of Classification Tasks

- A nuclear fuel processing company wishes to improve the yield of its factories. In one such factory, uranium hexafloride gas is converted into uranium-dioxide pellets. Six processing steps are needed to do the conversion. There are 30 controllable variables, such as pressure and flow rates, and temperature. Engineers note that the yield is high on some days and low on others. How can they control the variables to produce high yield on all days?
- Astronomers have been cataloguing distant objects in the sky using long-exposure CCD images. The objects need to be labeled as star, galaxy, nebula etc. The data is highly noisy, and the images are very faint. The cataloguing can take decades to complete. How can physicists automate the cataloguing process, and improve its effectiveness?

The Classification Problem

- **Given** a set of n attributes (ordinal or categorical), a set of k classes, and a set of labeled training instances

$$(i_i, l_i) \ldots (i_j, l_j)$$

$$i = (v_1, \ldots, v_n)$$

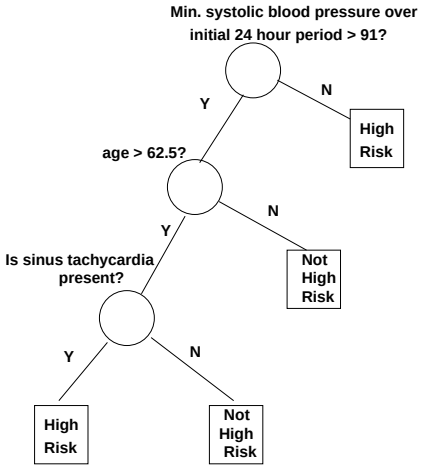
$$l \in \{c_1, \ldots, c_k\}$$

- **Determine** a *classification rule* that predicts the class of any instance from the values of its attributes.
- **Note:** This is a generalization of the concept learning problem since there are typically more than 2 classes.

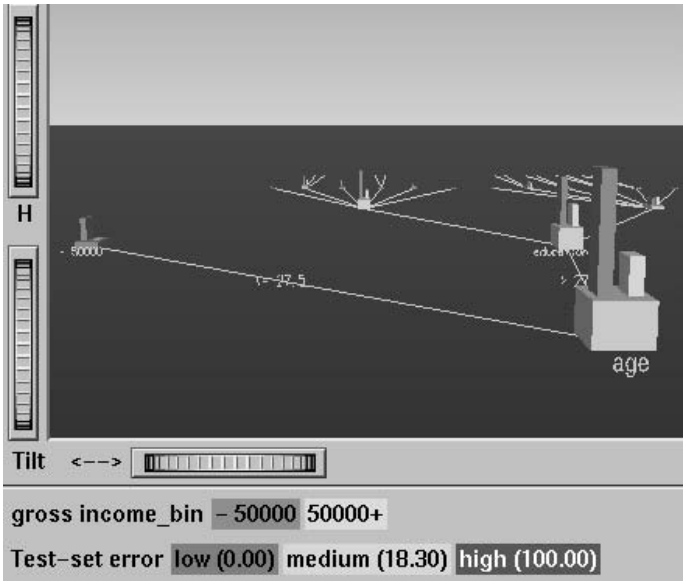
Decision Trees

- Easy-to-understand general representation of a discrete classifier
- Fast learning algorithms (ID3, c4.5, CART)
- Noise immunity (attribute noise, missing values)
- Widely used in solving large realistic classification problems
  - Credit card risk
  - Star classification
  - Medical diagnosis
  - Industrial applications
- Commercial data mining software (SGI Mineset)

Example Decision Tree



SGI Mineset



See course home page for a nice movie!
http://www.sgi.com/Products/software/MineSet/

Decision Tree Representation

- Each node contains a test on an attribute (ordinal or nominal).
- Each node can have many children (one for each test value).
- Leaf nodes are labeled by the class label.
- Decision trees represent disjunctive hypotheses.
- Each path to a leaf node represents a conjunctive term.
- The set of all paths leading to the same class represent alternative (disjunctive) ways in which instances could fall into a class.

Examples

Boolean functions:

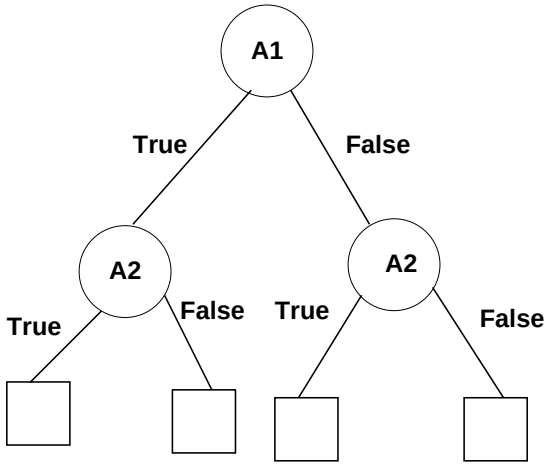
- Each attribute is binary valued (true/false).
- Any arbitrary boolean function can be represented by a decision tree.
- Examples: XOR, parity, AND, OR, etc.

Continuous domains:

- Each attribute is real-valued.
- Each test checks whether $a_i > value$.
- What do the class boundaries look like?

Give an example of a classifier that cannot be represented using a (simple) decision tree.

Boolean Functions using Decision Trees



When are Decision Trees Appropriate?

- Instances are described using a attribute-value representation.
- Target function is discrete-valued.
- The hypothesis needs to include disjunctive descriptions.
- The training instances may be noisy.

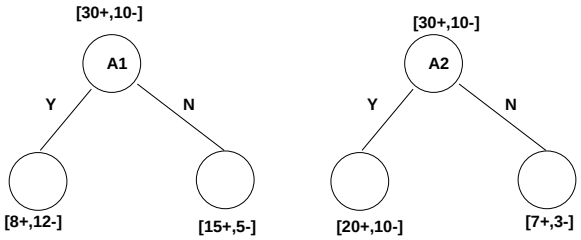
Examples:

- Customer profiling
- Corporate data mining
- Medical diagnosis
- Scientific applications
- World-wide Web

Top-down Induction of Decision Trees

Repeat the following loop until the remaining data is "pure"

- Choose the "best" attribute A for the next decision node.
- For each value of A, create a new descendant node. (for ordinal attributes, choose a value to split on).
- Sort the data into the descendant nodes.



Feature Selection

Define an **impurity function** on datasets that measures the proportion of instances belong to the different classes. Choose the variable (and split value) that maximizes the reduction in impurity from the root node to the children.

The impurity function should be least (zero) for datasets whose elements all fall into one class.

How can we obtain such an impurity function?

A Little Information-Theoretic Analysis

Consider playing a game: guess the right number x from a given set of numbers S. Minimum number of questions $= log_2(|S|)$.

Suppose we split S into P and N, and you know whether x is in P or N. If probability that $x \in P$ is p_p and probability that $x \in N$ is p_n, then (expected) number of questions becomes $p_p log_2(|P|) + p_n log_2(|N|)$

Thus the information *gained* by knowing whether $x \in P$ or $x \in N$ is $I = log_2(|S|) - p_p log_2(|P|) - p_n log_2(|N|)$

Since $p_p = |P|/|S|$ and $p_n = |N|/|S|$ we have

$$\begin{aligned} I &= log_2(|S|) - p_p log_2(p_p|S|) - p_n log_2(p_n|S|) \\ &= log_2(|S|) - p_p log_2(p_p) - p_n log_2(p_n) - (p_p + p_n) log_2(|S|) \\ &= -p_p log_2(p_p) - p_n log_2(p_n) \end{aligned}$$

This is the *entropy* function.

The Entropy Function

Entropy(S) = expected number of bits needed to encode class $(+/-)$ of a randomly chosen element of S.

From information theory, optimal code assigns $-log_2 p$ bits to message having probability p.

So expected number of bits needed to encode class of random element of S is $-p_+ log_2 p_+ - p_- log_2 p_-$.

The Entropy Impurity Function

Let classes be $c_1, \ldots, c_k$. Probability that $x \in c_i$ is denoted by $p(i)$. Then the impurity of the root node corresponding to the entire training data D is

$$I(D) = -\sum_i p^D(i) log_2(p^D(i))$$

Let's say we decide to split data D using some feature f. Then let the left child data be D_l and the right child data be D_r. Then the impurities of the children nodes are

$$I(D_l) = -\sum_i p^{D_l}(i) log_2(p^{D_l}(i))$$

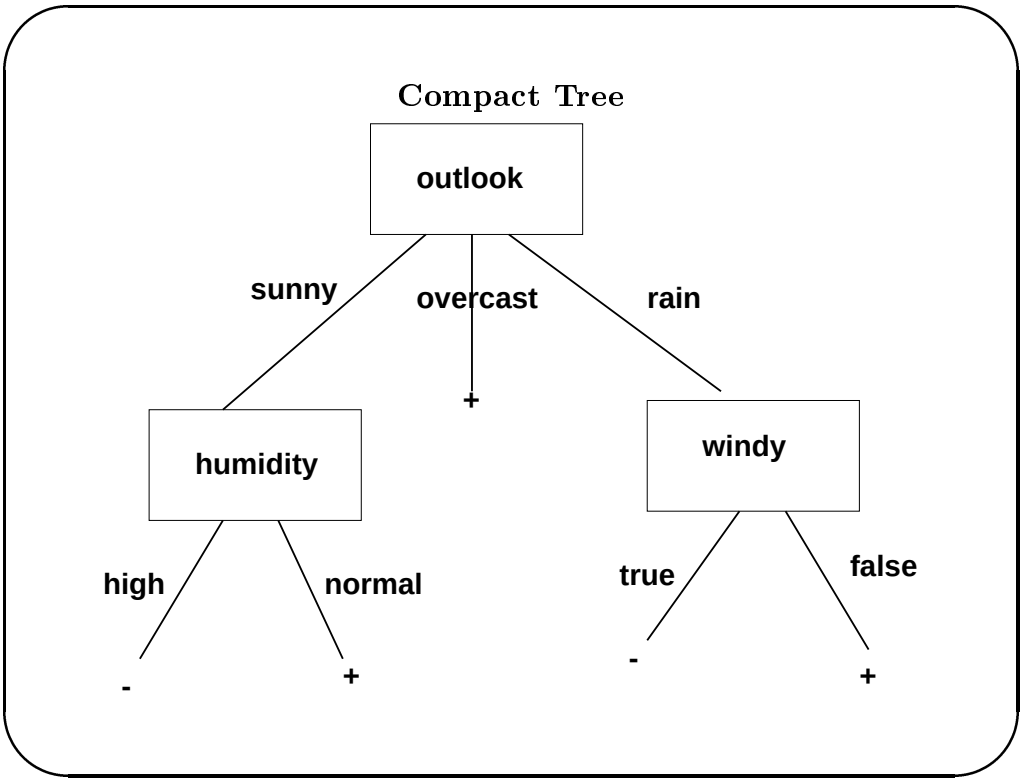
$$I(D_r) = -\sum_i p^{D_r}(i) log_2(p^{D_r}(i))$$

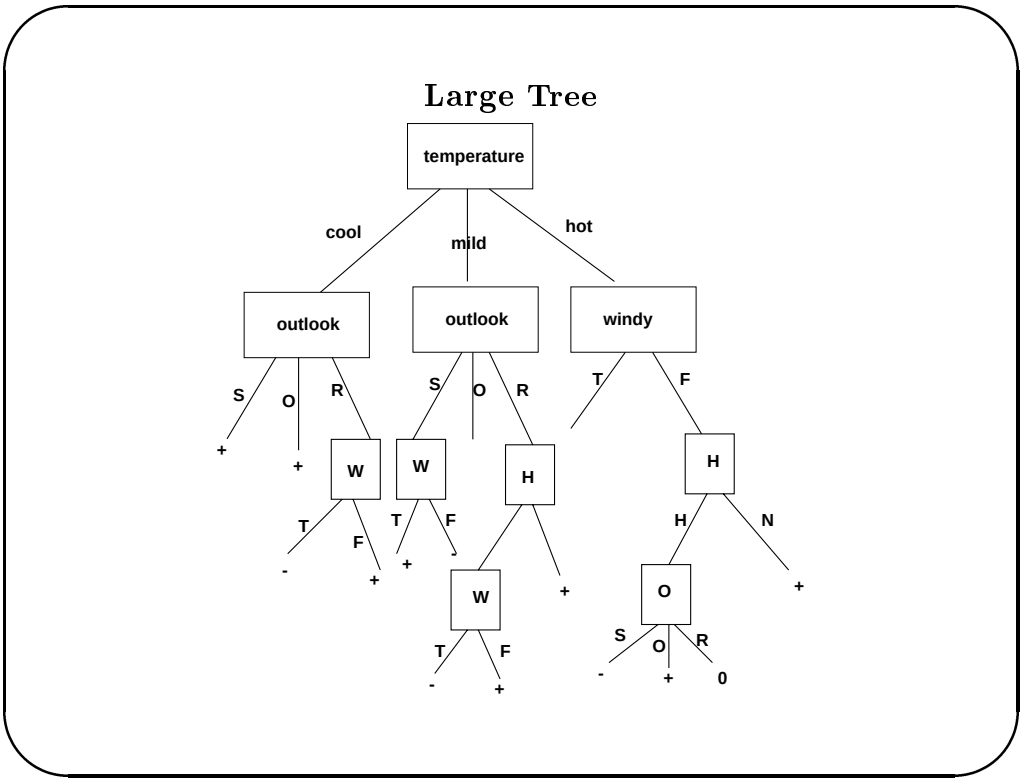
Net decrease in impurity from root to children is

$\Delta(f) = I(D) - p_l I(D_l) - p_r I(D_r)$

PlayTennis?

Class	*Outlook*	*Temp*	*Humidity*	*Windy*
-	sunny	hot	high	false
-	sunny	hot	high	true
+	overcast	hot	high	false
+	rain	mild	high	false
+	rain	cool	normal	false
-	rain	cool	normal	true
+	overcast	cool	normal	true
-	sunny	mild	high	false
+	sunny	cool	normal	false
+	rain	mild	normal	false
+	sunny	mild	normal	true
+	overcast	mild	high	true
+	overcast	hot	normal	false
-	rain	mild	high	true


Compact Tree
outlook
sunny
overcast
rain
+
humidity
windy
high
normal
true
false
-
+
-
+


Large Tree
temperature
cool
mild
hot
outlook
outlook
windy
S
O
R
S
O
R
T
F
+
+
W
W
H
H
T
F
T
F
H
N
-
+
+
W
+
O
T
F
S
O
R
-
+
+
0

Weather Example Continued

Let's assume we select *outlook* first.

$$I(root) = -\frac{9}{14}log_2(\frac{9}{14}) - \frac{5}{14}log_2(\frac{5}{14}) = 0.94\ bits$$

$$I(node_1) = -\frac{2}{5}log_2(\frac{2}{5}) - \frac{3}{5}log_2(\frac{3}{5}) = 0.971\ bits$$

$$I(node_2) = 0, I(node_3) = 0.971\ bits$$

$$\Delta(outlook) = 0.94 - (2)(\frac{5}{14})(0.971) = 0.246\ bits$$

$$\Delta(temperature) = 0.029\ bits$$
$$\Delta(windy) = 0.151\ bits$$
$$\Delta(humidity) = 0.048\ bits$$

So *outlook* is the best attribute to start the spliiting process.

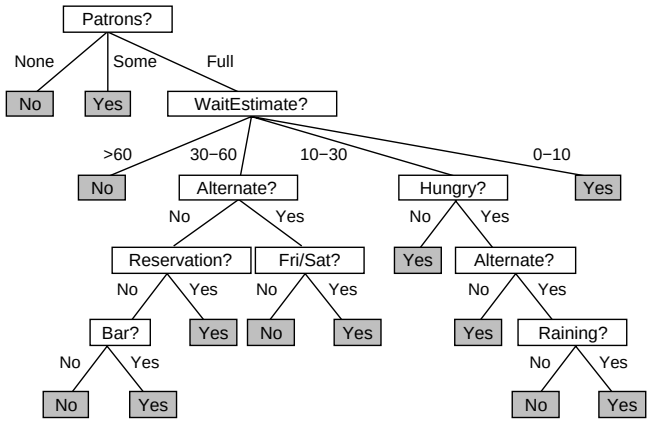
Hypothesis Space Search by ID3

- Complete hypothesis space for nominal attributes (not for ordinal attributes)
- Maintains and outputs a "best" hypothesis
- No back tracking
- Inductive bias: prefer "shortest" tree.

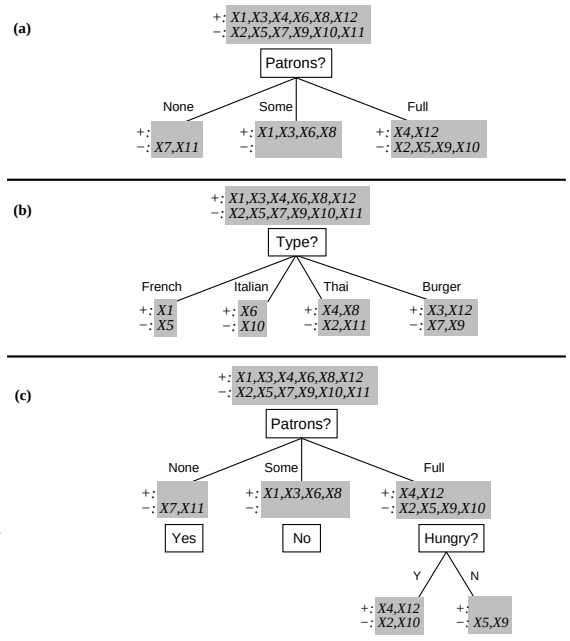
Restaurant Data

Example	Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	*Yes*	*No*	*No*	*Yes*	*Some*	*$$$*	*No*	*Yes*	*French*	*0–10*	*Yes*
X_2	*Yes*	*No*	*No*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*30–60*	*No*
X_3	*No*	*Yes*	*No*	*No*	*Some*	*$*	*No*	*No*	*Burger*	*0–10*	*Yes*
X_4	*Yes*	*No*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*10–30*	*Yes*
X_5	*Yes*	*No*	*Yes*	*No*	*Full*	*$$$*	*No*	*Yes*	*French*	*>60*	*No*
X_6	*No*	*Yes*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Italian*	*0–10*	*Yes*
X_7	*No*	*Yes*	*No*	*No*	*None*	*$*	*Yes*	*No*	*Burger*	*0–10*	*No*
X_8	*No*	*No*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Thai*	*0–10*	*Yes*
X_9	*No*	*Yes*	*Yes*	*No*	*Full*	*$*	*Yes*	*No*	*Burger*	*>60*	*No*
X_{10}	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$$$*	*No*	*Yes*	*Italian*	*10–30*	*No*
X_{11}	*No*	*No*	*No*	*No*	*None*	*$*	*No*	*No*	*Thai*	*0–10*	*No*
X_{12}	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Burger*	*30–60*	*Yes*

Example Decision Tree



Which Attribute to Select?



The Entropy Impurity Function for Restaurant Problem

Entropy(Root) $= -p_{\oplus} log_2(p_{\oplus}) - p_{\ominus} log_2(p_{\ominus})$ bits.

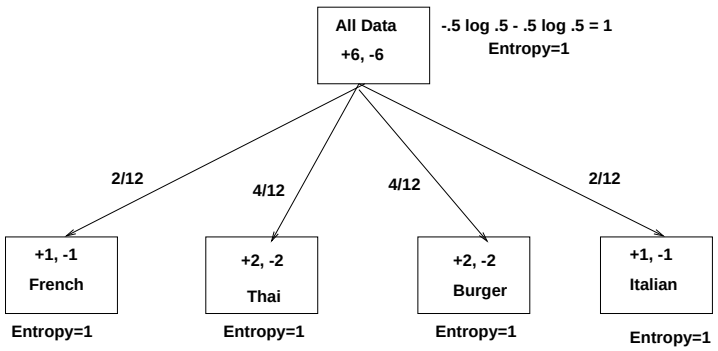
For the restaurant example: Entropy(Root) $=$
$-\frac{6}{12} log_2(\frac{6}{12}) - \frac{6}{12} log_2(\frac{6}{12}) = -2 \times .5 \times -1 = 1.$

If we choose to split on restaurant type, we get four subsets, one for each type of restaurant (Thai, Burger, French, Italian).

We compute the entropy of each subset, and then weight it by the proportion of instances in that subset, and add them all up.

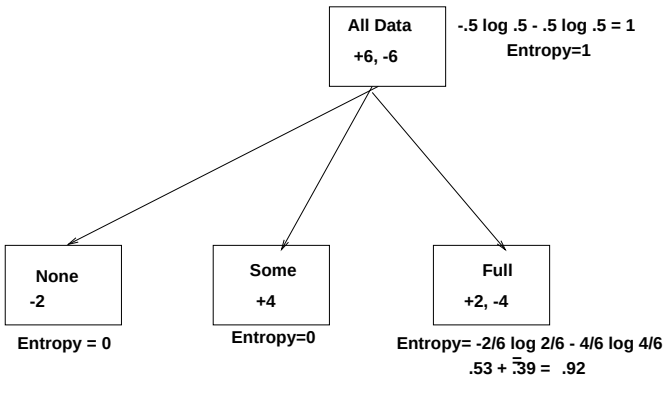
We want to select an attribute that most reduces the entropy.

Splitting on Type of Restaurant



Reduction in Entropy = 1 - 1/6 - 1/3 - 1/3 - 1/6 = 0

Splitting on Number of Patrons



Reduction = 1 - 6/12 * .92 = 0.54

Dealing with continuous attributes

- Continuous attributes (e.g. temperature) can be handled easily by choosing to split at the middle value between two examples which are in different categories

Temperature	PlayTennis?
40	No
48	No
60	Yes
70	Yes
80	Yes
90	No

The right split points are $\frac{48+60}{2}$ or $\frac{80+90}{2}$.

Scaling Decision Trees to Real-world Data

- How to estimate classifier error
- How to avoid overfitting
- How to handle missing/noisy attribute values
- How to incorporate attribute costs

The UCI Irvine Online ML Database

```
==========================================================================
This is the UCI Repository Of Machine Learning Databases and Domain Theories
                          4 October 1992
              ics.uci.edu: pub/machine-learning-databases
      Site Librarian: Patrick M. Murphy (ml-repository@ics.uci.edu)
        Off-Site Assistant: David W. Aha (aha@insight.cs.jhu.edu)
                83 databases and domain theories (24013K)
==========================================================================


1. annealing (David Sterling and Wray Buntine)
4. autos (Jeff Schlimmer)
6. breast-cancer-wisconsin (Wisconsin Breast Cancer D'base, Olvi Mangasarian)
7. bridges (Yoram Reich)
16-17. Credit Screening Database
18. Ein-Dor and Feldmesser's cpu-performance database (David Aha)
21. Evlin Kinney's echocardiogram database (Steven Salzberg)
24. glass (Vina Spiehler)
26-29. heart-disease (Robert Detrano)
30. hepatitis (G. Gong)
33. ionosphere information (Vince Sigillito)
36. labor-negotiations (Stan Matwin)
39. lenses (Cendrowska's database donated by Benoit Julien)
41. liver-disorders (BUPA Medical's database donated by Richard Forsyth)
43. lung cancer (Stefan Aeberhard)
44. lymphography (Ljubjana Institute of Oncology, restricted access)
53. Pima Indians diabetes diagnoses (Vince Sigillito)
56. shuttle-landing-control (Bojan Cestnik)
57. solar flare (Gary Bradshaw)
60. spectrometer (Infra-Red Astronomy Satellite Project Database, John Stutz)
```


Some datasets in more detail

```
16-17. Credit Screening Database
    1. Japanese Credit Screening Database and Domain Theory
       --  Positive instances are people who were granted credit.
       --  The theory was generated by talking to Japanese domain experts
    2. Credit Card Application Approval Database
       -- a good mix of attributes -- continuous, nominal with small numbers
          of values, and nominal with larger numbers of values.
       -- 690 instances, 15 attributes some with missing values.

31. Horse Colic database (Mary McLeish & Matt Cecile)
    -- Well documented attributes
    -- 368 instances with 28 attributes (continuous, discrete, and nominal)
    -- 30% missing values

60. Low resolution spectrometer data (IRAS data -- NASA Ames Research Center)
    -- Documentation: no statistics nor class distribution given
    -- LARGE database...and this is only 531 of the instances
    -- 98 attributes per instance (all numeric)
    -- Contact NASA-Ames Research Center for more information


79. Congressional voting records classified into Republican or Democrat (1984
    United Stated Congressional Voting Records)
    -- Documentation: completed
    -- All attributes are Boolean valued; plenty of missing values; 2 classes
    -- Also, their is a 2nd, undocumented database containing 1986 voting
       records here. (will be)
```

Estimating the Accuracy of a Classifier

Given a classifier, we want to estimate its **misclassification rate**.

Standard approach:

- Build the classifier using all the training data (e.g. all the medical data from 1975-1977).
- Measure the proportion of misclassified instances using a separate test dataset (e.g. all the medical data from some later time period 1978-81).
- **Drawback:** In practice, we may have only one (not very large) dataset.

Three ways of estimating true error

1. **Resubstitution**: Build the classifier using all the training data, and test the classifier using the same data. (*optimistic*)
2. **Test Sample Estimation:** Divide data into training and test sets (say 2/3 and 1/3). (*wastes data*)
3. **Cross-validation:** Divide data into N equal-sized subsets, and train on all but one subset. Average error over all trials. (*pessimistic, but good*) (special case: Leave-one-out)

Overfitting

Error of a hypothesis h over

- Training data: $error_{train}(h)$
- Entire distribution: $error_D(h)$

Hypothesis $h \in H$ overfits the training data if $\exists h' \in H$ such that

- $error_{train}(h) < error_{train}(h')$
- $error_D(h) > error_D(h')$

Example of Overfitting

Imperfect Leaves

- Stop splitting when impurity is low enough
- Assign leaf label to the most numerous class
- Alternatively, use probability estimates of class membership

Tree-Pruning Methods

Reduced-error pruning (c4.5)

- Grow a very large tree (e.g. till each terminal node is pure).
- Prune the tree by deleting subtrees starting at the bottom and working upwards.
- Repeat pruning until the "best" tree is obtained.

Cost-complexity pruning (CART)

- Minimize cost function that is based on tree size and error rate
- Measure error by resubstitution.

Voting Domain

```
democrat, republican |  classes

handicapped infants: n, y, u
water project cost sharing: n, y, u
adoption of the budget resolution: n, y, u
physician fee freeze: n, y, u
el salvador aid: n, y, u
religious groups in schools: n, y, u
anti satellite test ban: n, y, u
aid to nicaraguan contras: n, y, u
mx missile: n, y, u
immigration: n, y, u
synfuels corporation cutback: n, y, u
education spending: n, y, u
superfund right to sue: n, y, u
crime: n, y, u
duty free exports: n, y, u
export administration act south africa: n, y, u
```


Unpruned Tree for Voting Domain

```
physician fee freeze = n:
|   adoption of the budget resolution = y: democrat (151.0)
|   adoption of the budget resolution = u: democrat (1.0)
|   adoption of the budget resolution = n:
|   |   education spending = n: democrat (6.0)
|   |   education spending = y: democrat (9.0)
|   |   education spending = u: republican (1.0)
physician fee freeze = y:
|   synfuels corporation cutback = n: republican (97.0/3.0)
|   synfuels corporation cutback = u: republican (4.0)
|   synfuels corporation cutback = y:
|   |   duty free exports = y: democrat (2.0)
|   |   duty free exports = u: republican (1.0)
|   |   duty free exports = n:
|   |   |   education spending = n: democrat (5.0/2.0)
|   |   |   education spending = y: republican (13.0/2.0)
|   |   |   education spending = u: democrat (1.0)
physician fee freeze = u:
|   water project cost sharing = n: democrat (0.0)
|   water project cost sharing = y: democrat (4.0)
|   water project cost sharing = u:
|   |   mx missile = n: republican (0.0)
|   |   mx missile = y: democrat (3.0/1.0)
|   |   mx missile = u: republican (2.0)
```

Pruned Tree for Voting Domain

```
physician fee freeze = n: democrat (168.0/2.6)
physician fee freeze = y: republican (123.0/13.9)
physician fee freeze = u:
|   mx missile = n: democrat (3.0/1.1)
|   mx missile = y: democrat (4.0/2.2)
|   mx missile = u: republican (2.0/1.0)


Tree saved


Evaluation on training data (300 items):

         Before Pruning           After Pruning
        ----------------   ---------------------------
        Size      Errors   Size      Errors   Estimate

          25    8( 2.7%)      7   13( 4.3%)    ( 6.9%)   <<
```


Resubstitution Estimate for Tree Misclassification Rate

Let T be a decision tree, which has terminal nodes T_n.

Given a terminal node t, and an instance i that falls into t. Then let the class of i be the one for which $p(class|t)$ is highest.

The **resubstitution estimate** of probability of misclassification at node t is

$$r(t) = 1 - p(c|t)$$

where c is the class that maximizes this probability.

Let $p(t)$ be the probability of an instance falling into node t. Then the resubstitution estimate of the overall misclassification rate of tree T is

$$R(T) = \sum_{t \in T_n} r(t)p(t)$$

Let the complexity of a tree T be defined as the number of terminal nodes $|T_n|$.

Then a simple tree-pruning algorithm is to pick the best tree from a sequence of trees $T_1, \ldots, \{t\}$ where "best" is defined as the tree that minimizes the quantity

$$R_\alpha(T) = R(T) + \alpha|T_n|$$

where α is some real number ≥ 0. This strategy is called *minimal cost-complexity pruning.*

Dealing with Errors in the Data

Types of noise:

- Incorrect (or missing) attribute values
- Incorrect classification

Ways of Dealing with Noise

- Use probabilistic criteria for assigning leaf labels
- Test attributes for relevance (chi-square test)
- Use tree-pruning methods
- Fill in missing values

Interesting result regarding the different types of noise:

- Increasing attribute noise only causes bounded test error.
- Increasing classification noise causes linear test error.

Dealing with Unknown Attribute Values

- Use Bayesian approach to fill in missing values

$$p(A = A_i | class = c) = \frac{p(A = A_i \wedge class = c)}{p(class = c)}$$

- Treat values of attribute A as classes, and build a tree for predicting the values. Use this tree to fill in missing values.
- Estimating tree branching probabilities for unknown attributes.
  - The probability that an instance z will take a path from the root to a leaf L passing through branches $B_1, \ldots$ can be written as

$$P_Z(L) = P_Z(B_1) P_Z(B_2|B_1) P_Z(B_3|B_1 \wedge B_2)$$

  - If outcome of all these tests are known, then the probabilities are either 0 or 1.
  - In cases when a decision cannot be made because of an unknown attribute value, the branching probability can be estimated from other instances that take that branch. But note that the

probability that case z will reach a leaf L may be non-zero for more than 1 leaf.

Other Impurity Metrics

Entropy metric is biased towards attributes with many values

- E.g. consider day-of-week in tennis example.
- Solution: use *GainRatio* instead

$$GainRation(S, A) = \frac{Gain(S, A)}{SplitInformation(S, A)}$$

where

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

where S_i is the subset of S for which A has value i.

Incorporating Attribute Costs

Entropy metric ignores attribute costs.

- Some medical tests may be very expensive.
- Some robot actions may be very slow.
- Incorporate cost (Tan & Schlimmer, '90)

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

A Regression Problem

Consider a set of training instances $D = (\vec{x}, y)$, where $\vec{x}$ is a vector of attribute values, and y is a real-valued number.

We define the variables in $\vec{x}$ as *independent* or *predictor* variables, and the variable y as the *dependent* or *response* variable.

A *prediction rule* is a real-valued function $d(\vec{x})$.

Regression analysis revolves around constructing a predictor starting from a learning sample.

In machine learning, we term this the *function approximation* problem.

Error estimates of a prediction rule

Suppose we had a training sample of N cases $(\vec{x_1}, y_1), \ldots, (\vec{x_N}, y_N)$, which was used to construct a predictor d. We can measure the accuracy of the predictor in a number of ways.

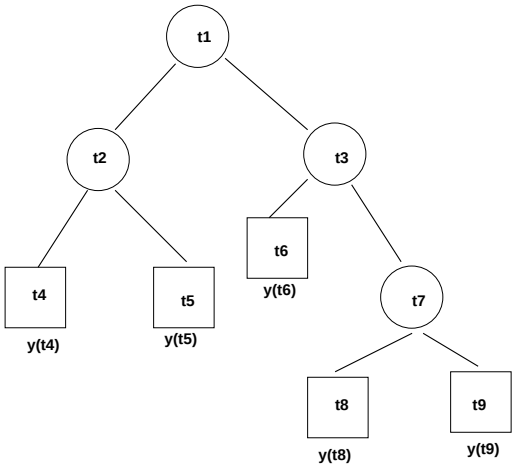
- Resubstitution estimate

$$R(d) = \frac{1}{N} \sum_{i=1}^{N} (y_i - d(\vec{x_i}))^2$$

- Test sample estimate: repeat the above only on a separate test sample.
- Cross-validation: divide the training sample into subsets, and build the predictor on all but one subset, and test on the hold-out subset.

Regression Trees

A decision-tree can be used to build a predictor function. Such trees are called *regression* trees.



Assigning Leaf Values

Assign a leaf t the value that minimizes the resubstitution estimate value

$$R(t) = \frac{1}{N} \sum_{i=1}^{N} (y_i - d(\vec{x_i}))^2$$

Theorem: The leaf value of $y(t)$ that minimizes the resubstitution estimate is the average of y_i for all the cases that fall into the leaf

$$\overline{y}(t) = \frac{1}{N(t)} \sum_{\vec{x_i} \in t} y_i$$

Choosing the Best Split

Given a node t, the best split S of t into t_L and t_R is the one that minimizes

$$\Delta R(S,t) = R(t) - R(t_L) - R(t_R)$$

where

$$R(t) = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - \overline{y}(t)\right)^2$$