Learning

Learning

Learning Definitions

- *Learning* is improvement of performance (time, accuracy).
- *Inductive inference* is generalizing from experience.
- An *example* is an input-output pair, output optional.
- In *unsupervised learning*, examples do not have outputs. The most common task is *clustering*.
- In *supervised learning*, all exs. have outputs.
- In *semi-supervised learning*, some examples have outputs. For example, in *reinforcement learning*, an input is a series of actions, and the output is intermittent feedback.

Supervised Learning

- Given a *training set* of m examples:
  $$(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)$$
  assumed to be generated by an unknown function $y = f(x)$, discover a *hypothesis* h that approximates f.
- A *test set* of examples is used to estimate the accuracy of h.
- Usually, h is restricted to be an element from a *hypothesis space*.
- *Regression* is when the output is continuous.
- *Classification* is when the output is discrete.
- *Ockham's razor* says to prefer simpler hypotheses.

Example of Examples

No.	Input Attributes				Output
	Outlook	Temp	Humidity	Windy	
1	sunny	hot	high	false	neg
2	sunny	hot	high	true	neg
3	overcast	hot	high	false	pos
4	rain	mild	high	false	pos
5	rain	cool	normal	false	pos
6	rain	cool	normal	true	neg
7	overcast	cool	normal	true	pos
8	sunny	mild	high	false	neg
9	sunny	cool	normal	false	pos
10	rain	mild	normal	false	pos
11	sunny	mild	normal	true	pos
12	overcast	mild	high	true	pos
13	overcast	hot	normal	false	pos
14	rain	mild	high	true	neg

Error in Learning

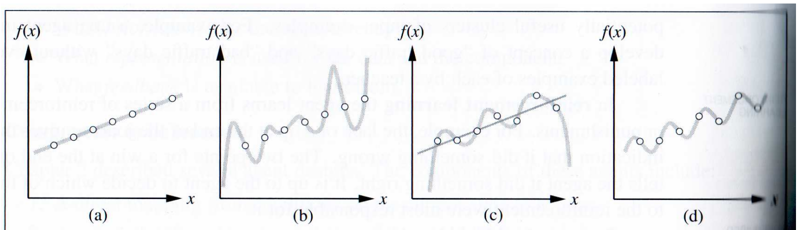


Figure 18.1 (a) Example $(x, f(x))$ pairs and a consistent, linear hypothesis. (b) A consistent, degree-7 polynomial hypothesis for the same data set. (c) A different data set, which admits an exact degree-6 polynomial fit or an approximate linear fit. (d) A simple, exact sinusoidal fit to the same data set.

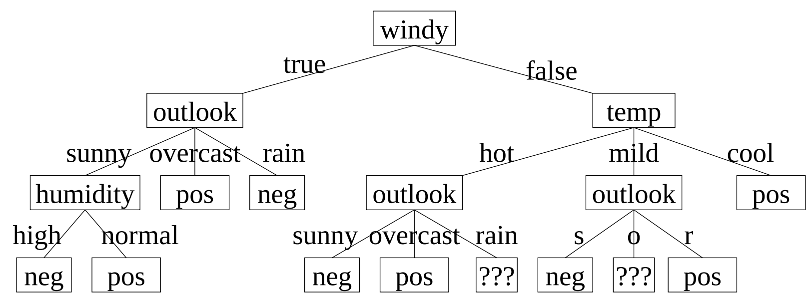
- Perfect learning cannot be guaranteed by any learning algorithm from a finite training set.
- No learning algorithm is best. All learning algorithms make assumptions about the data.

Decision Trees

Definition

- Decision trees are a representation for classification.
  - The root is labeled by an attribute.
  - Edges are labeled by attribute values.
  - Edges go to decision trees or leaves.
  - Each leaf is labeled by a class.
- Growth Phase: Construct the tree top-down.
  - Find the "best" attribute.
  - Split examples based on attribute's values.
- Pruning Phase: Prune the tree bottom-up.
  For each node, keep subtree or change to leaf.

Example of a Decision Tree



Algorithm for Growing Decision Trees

```
Grow_DT(examples)
N ← a new node
N.class ← most common class in examples
if examples have identical class or values
   then return N
N.test ← best attribute (or test)
for each value v_j of N.test
    examples_j ← examples with N.test = v_j
    if examples_j is empty
      then N.branch_j ← N.class
      else N.branch_j ← Grow_DT(examples_j)
return N
```


Comparing Attibutes: Information Gain

- p positive examples and n negative examples
- The information contained is:

  $I(p,n) = -\frac{p}{p+n}\log_2\frac{p}{p+n} - \frac{n}{p+n}\log_2\frac{n}{p+n}$
- Attribute A has v values, p_j positive examples and n_j negative examples when $A = v_j$
- The $Remainder$ of A is:

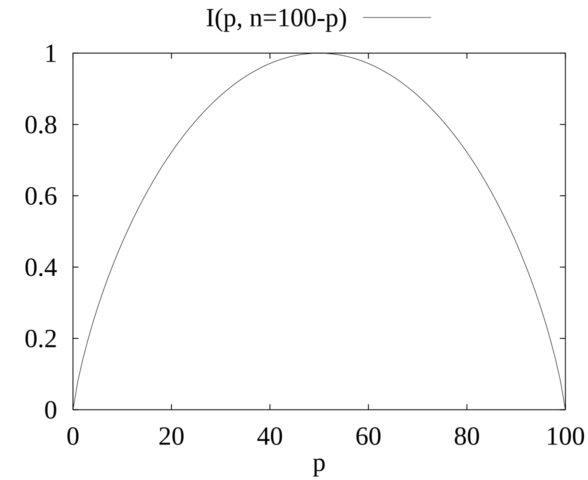
$$Remainder(A) = \sum_{j=1}^{v} \frac{p_j + n_j}{p+n}\, I(p_j, n_j)$$

- The information gain of A is:

$$Gain(A) = I(p,n) - Remainder(A)$$

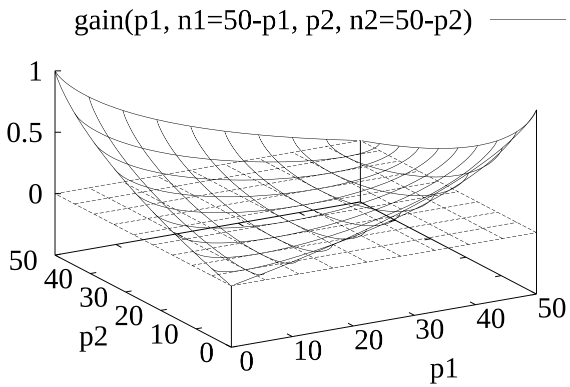
Plot of Information Function

p positive examples and n negative examples



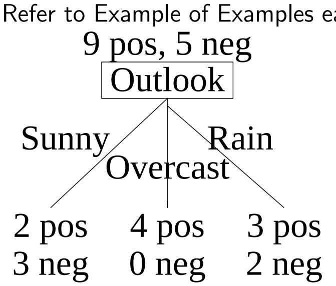
Plot of Information Gain

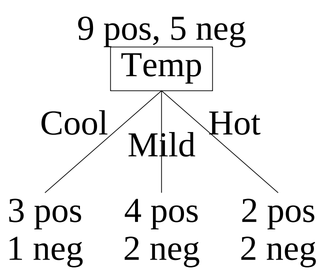
p_1 positive and n_1 negative exs. when attr.$=v_1$
p_2 positive and n_2 negative exs. when attr.$=v_2$



Example of Attribute Selection

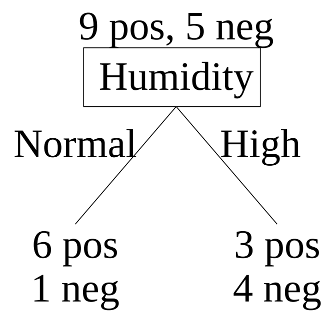
Refer to Example of Examples earlier.

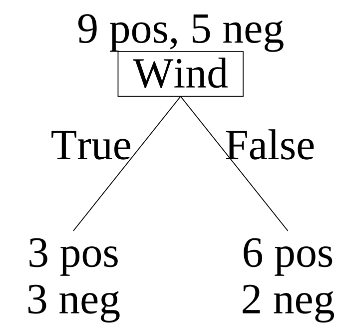




$Gain$(Outlook) ≈ 0.246 $Gain$(Temp) ≈ 0.029

Attribute Selection, Continued





$Gain$(Humidity) ≈ 0.152 $Gain$(Wind) ≈ 0.048

Outlook has the highest gain.
Overcast branch is pure.
Need to construct DTs for two branches.

Alternative Attributes Measures

- Maximize Information Gain Ratio
  $GainRatio(A) = Gain(A)/I(p_1 + n_1, \ldots, p_v + n_v)$
- Minimize Gini Index
  $$Gini(p, n) = 1 - \left(\frac{p}{p+n}\right)^2 - \left(\frac{n}{p+n}\right)^2$$
  $$GiniIndex(A) = \sum_{j=1}^{v} \frac{p_j + n_j}{p+n} Gini(p_j, n_j)$$
- "Maximize" Chi-Squared Statistic
  $$\chi^2 = \sum_{j=1}^{v} \frac{(p_j - p\,s_j)^2}{p\,s_j} + \frac{(n_j - n\,s_j)^2}{n\,s_j}$$
  where $s_j = (p_j + n_j)/(p + n)$

Special Cases in Decision Trees

- Attribute A is numeric.
  - Find best $A \leq v$ test. Requires sorting.
  - Or: Discretization. Partition A into ranges.
- Attribute A has missing values.
  - Pretend missing is just another value.
  - Or: Ignore missing values. Split examples with missing values across branches.
- Attribute A has many discrete values.
  - Find best $A = v$ test. Forms binary tree.
  - Or: Partition values into subsets.

Pruning Decision Trees

- Why are there errors?
  - Statistical fluctuations.
  - Examples might have noise and/or outliers.
  - DT approximates decision boundary.
- Results in overfitting at lower levels of DT
- Pruning
  - Prepruning: Avoid creation of subtrees based on number of examples or attribute relevance.
  - Postpruning: Create overfitting DT and substitute subtrees with leaves if estimated error is reduced.

Estimating Error

- Use a “validation” set of examples.
  (training set, validation set, test set should be disjoint)
- Minimum Description Length principle
  (minimize size of tree and minimize size of errors)
- Add some error to each leaf (C4.5).
  - Suppose a leaf has e errors on n examples.
  - Find 75% confidence interval using binomial dist.
  - Estimate true error as upper limit of interval.

Algorithm for Pruning Decision Trees

Prune_DT(N: node, $examples$)
$leaferr \leftarrow$ number of $examples \neq N.class$
increase $leaferr$ if $examples$ were training set
if N is a leaf **then return** $leaferr$
$treeerr \leftarrow 0$
for each value v_j of $N.test$
    $examples_j \leftarrow examples$ with $N.test = v_j$
    $suberr \leftarrow$ **Prune_DT**($N.branch_j$, $examples_j$)
    $treeerr \leftarrow treeerr + suberr$
if $leaferr < treeerr$
  then make N a leaf; **return** $leaferr$
  else return $treeerr$