R19-MACHINE LEARNING

Unit-2

Decision Tree Learning: Representing concepts as decision trees, Recursive induction of decision trees, Picking the best splitting attribute: entropy and information gain, Searching for simple trees and computational complexity, Occam's razor, Over fitting, noisy data, and pruning. Experimental Evaluation of Learning Algorithms: Measuring the accuracy of learned hypotheses. Comparing learning algorithms: cross-validation, learning curves, and statistical hypothesis testing.

Decision Tree Learning:

Decision tree learning is a method for approximating discrete-valued target functions, in which the learned function is represented by a decision tree. Learned trees can also be re-represented as sets of if-then rules to improve human readability.

These learning methods are among the most popular of inductive inference algorithms and have been successfully applied to a broad range of tasks from learning to diagnose medical cases, learning to assess credit risk of loan applicants.

Decision Trees are a type of Supervised Machine Learning. The tree can be explained by two entities, namely decision nodes and leaves. The leaves are the decisions or the final outcomes. And the decision nodes are where the data is split.

DECISION TREE REPRESENTATION

Decision trees classify instances by sorting them down the tree from the root to some leaf node, which provides the classification of the instance. Each node in the tree specifies a test of some attribute of the instance, and each branch descending from that node corresponds to one of the possible values for this attribute. An instance is classified by starting at the root node of the tree, testing the attribute specified by this node, then moving down the tree branch corresponding to the value of the attribute in the given example. This process is then repeated for the sub tree rooted at the new node.

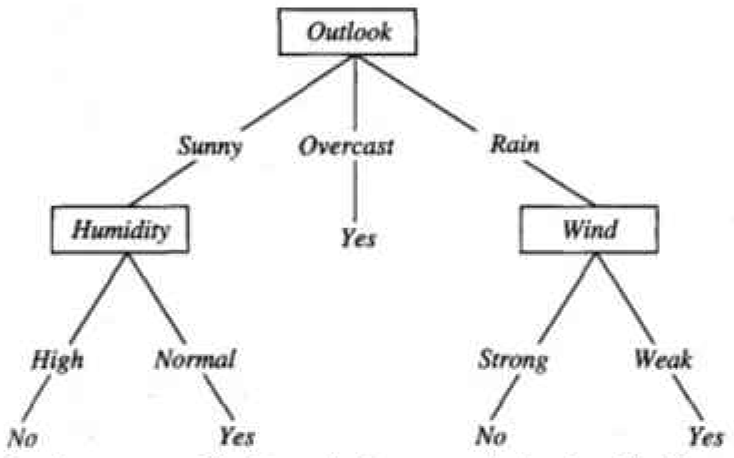


FIGURE : A decision tree for the concept PlayTennis. An example is classified by sorting it through the tree to the appropriate leaf node, then returning the classification associated with this leaf (in this case, Yes or No).

Test Sample: This decision tree classifies Saturday mornings according to whether they are suitable for playing tennis. For example, the instance

<Outlook = Sunny, Temperature = Hot, Humidity = High, Wind = Strong>

would be sorted down the leftmost branch of this decision tree and would therefore be classified as a negative instance (i.e., the tree predicts that PlayTennis = no).

In general, decision trees represent a disjunction of conjunctions of constraints on the attribute values of instances. Each path from the tree root to a leaf corresponds to a conjunction of attribute tests, and the tree itself to a disjunction of these conjunctions.

(Outlook = Sunny Λ Humidity = Normal)
V (Outlook = Overcast)
V (Outlook = Rain Λ Wind = Weak) → PlayTennis=Yes

Appropriate Problems for Decision Tree Learning:

- Instances are represented by attribute-value pair
- The target function has discrete output values
- Disjunctive descriptions may be required
- The training data may contain errors
- The training data may contain missing attribute values.

Recursive induction of decision trees(ID3 algorithm)

ENTROPY MEASURES HOMOGENEITY OF EXAMPLES

Homogeneity means all instances belongs to same class.

Attribute Selection Measures

- Information Gain • Gain ratio • Gini Index•

Let's assume for the moment that we are dealing with Boolean features, so *D* is split into D_1 and D_2. Let's also assume we have two classes, and denote by $D^{\oplus}$ and $D^{\ominus}$ the positives and negatives in *D* .
The question is how to assess the utility of a feature in terms of splitting the examples into positives and negatives.

The best situation is where $D_1^{\oplus} = D^{\oplus}$ and $D_1^{\ominus} = \emptyset$, or where $D_1^{\oplus} = \emptyset$ and $D_1^{\ominus} = D^{\ominus}$.
In that case, the two children of the split are said to be **pure**.

To measure the impurity of a set of $n^{\oplus}$ positives and $n^{\ominus}$ negatives.
impurity can be defined in terms of the proportion

$$\dot{p} = n^{\oplus} / (n^{\oplus} + n^{\ominus}),$$

Entropy:
Entropy refers to a common way to measure impurity. In the decision tree, it measures the randomness or impurity in data sets.
Let us say S is the sample set of training examples. Then, Entropy (S) measuring the impurity of S is defined as

$$\textbf{Entropy}(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

where c is the number of different class labels and p refers to the proportion of values falling into the i-th class label.

To illustrate, suppose S is a collection of 14 examples of some boolean concept, including 9 positive and 5 negatives relative to this boolean classification is negative examples (we adopt the notation [9+, 5-] to summarize such a sample of data). Then the entropy of

Notice that

- the entropy is 0 if all members of S belong to the same class. For example, if all members are positive ($p_{+ve} = I$), then p_{-ve}, is 0, and Entropy(S) = -1 . log2(1) - 0 . log2 0 = -1 . 0 - 0 . log2 0 = 0.
- the entropy is 1 when the collection contains an equal number of positive and negative examples.
- If the collection contains unequal numbers of positive and negative examples, the entropy is between0 and 1

$$\begin{aligned} Entropy([9+, 5-]) &= -(9/14)\log_2(9/14) - (5/14)\log_2(5/14) \\ &= 0.940 \end{aligned}$$

INFORMATION GAIN MEASURES THE EXPECTED REDUCTION IN ENTROPY

Given entropy as a measure of the impurity in a collection of training examples, we can now define a measure of the effectiveness of an attribute in classifying the training data. The measure we will use, called information gain, is simply the expected reduction in entropy caused by partitioning the examples according to this attribute.
The information gain, Gain(S, A) of an attribute A, relative to a collection of examples S, is defined as

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

where Values(A) is the set of all possible values for attribute A, and S_v is the subset of S for which attribute A has value v (i.e., $S_v = \{s \in S | A(s) = v\}$).
Information Gain:
The information gain is created on the basis of the decrease in entropy (S) after a data set is split according to a particular attribute (A).
Information gain for a particular feature A is calculated by the difference in entropy before a split (or S_{bS}) with the entropy after the split (S_{aS}).

Information Gain (S, A) = Entropy (S_{bs}) − Entropy (S_{as})

Information Gain refers to the decline in entropy after the dataset is split. It is also called **Entropy Reduction**. Building a decision tree is all about discovering attributes that return the highest data gain.

For example, suppose S is a collection of training-example days described by attributes including Wind, which can have the values Weak or Strong. As before, assume S is a collection containing 14 examples, [9+, 5-]. Of these 14 examples, suppose 6 of the positive and 2 of the negative examples have Wind = Weak, and the remainder have Wind = Strong. The information gain due to sorting the original 14 examples by the attribute Wind may then be calculated as

$$\begin{aligned}
Values(Wind) &= Weak, Strong \\
S &= [9+, 5-] \\
S_{Weak} &\leftarrow [6+, 2-] \\
S_{Strong} &\leftarrow [3+, 3-] \\
Gain(S, Wind) &= Entropy(S) - \sum_{v \in \{Weak, Strong\}} \frac{|S_v|}{|S|} Entropy(S_v) \\
&= Entropy(S) - (8/14) Entropy(S_{Weak}) \\
&\quad - (6/14) Entropy(S_{Strong}) \\
&= 0.940 - (8/14)0.811 - (6/14)1.00 \\
&= 0.048
\end{aligned}$$

Note: $S_{Weak} \leftarrow [6+, 2-]$ means coverages of weak feature

Information gain is precisely the measure used by ID3 to select the best attribute at each step in growing the tree. The use of information gain to evaluate the relevance of attributes is summarized in below Figure. In this figure the information gain of two different attributes, Humidity and Wind, is computed in order to determine which is the better attribute for classifying the training examples shown in below Table.

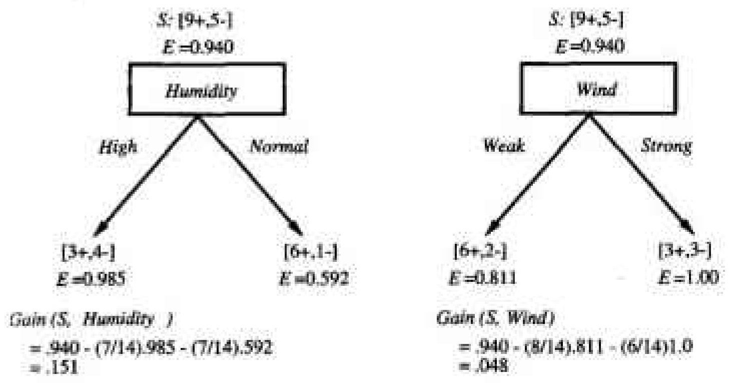


Fig:Humidity provides greater information gain than Wind, relative to the target classification. Here, E stands for entropy and S for the original collection of examples. Given an initial collection S of 9 positive and 5 negative examples, [9+, 5-], sorting these by their Humidity produces collections of [3+, 4-] (Humidity = High) and [6+, 1-] (Humidity = Normal). The information gained by this partitioning is .151, compared to a gain of only .048 for the attribute Wind.

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

TABLE Training examples for the target concept PlayTennis.
The information gain values for all four attributes are

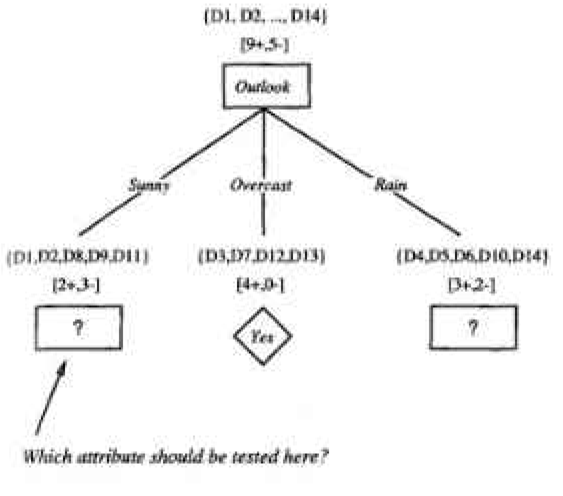
$$Gain(S, Outlook) = 0.246$$

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

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

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

where S denotes the collection of training examples from above Table.
According to the information gain measure, the Outlook attribute provides the best prediction of the target attribute, PlayTennis, over the training examples. Therefore, Outlook is selected as the decision attribute for the root node, and branches are created below the root for each of its possible values (i.e., Sunny, Overcast, and Rain). The resulting partial decision tree is shown in below Figure along with the training examples sorted to each new descendant node.



$S_{sunny} = \{D1,D2,D8,D9,D11\}$

$Gain(S_{sunny}, Humidity) = .970 - (3/5)\,0.0 - (2/5)\,0.0 = .970$

$Gain(S_{sunny}, Temperature) = .970 - (2/5)\,0.0 - (2/5)\,1.0 - (1/5)\,0.0 = .570$

$Gain(S_{sunny}, Wind) = .970 - (2/5)\,1.0 - (3/5)\,.918 = .019$

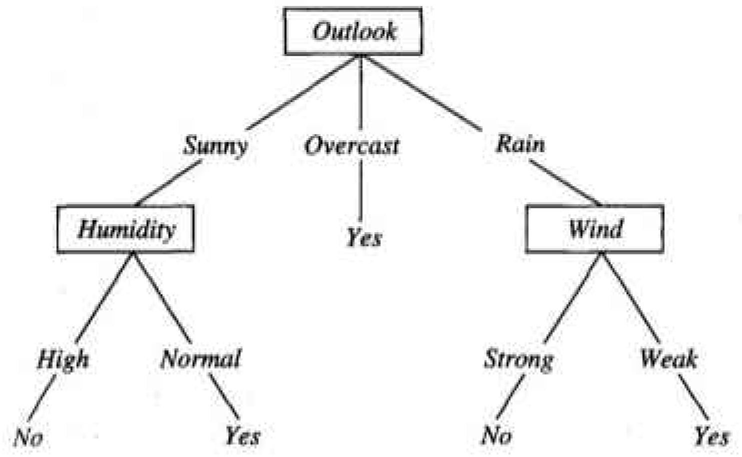
FIGURE:The partially learned decision tree resulting from the first step of ID3. The training examples are sorted to the corresponding descendant nodes. The Overcast descendant has only positive examples and therefore becomes a leaf node with classification Yes. The other two nodes will be further expanded, by selecting the attribute with highest information gain relative to the new subsets of examples.

ALGORITHM:

```
ID3(Examples, Targetattribute, Attributes)
        //Examples are the training examples. Targetattribute is the attribute whose value is to be
        predictedby the tree. Attributes is a list of other attributes that may be tested by the learned decision
        tree. Returns a decision tree that correctly classifies the given Examples.
•  Create a Root node for the tree
•  If all Examples are positive, Return the single-node tree Root, with label = +
•  If all Examples are negative, Return the single-node tree Root, with label = -
•  If Attributes is empty, Return the single-node tree Root, with label = most common value of
   Target_attribute in Examples
•  Otherwise Begin
        o  A ← the attribute from Attributes that best* classifies Examples
        o  The decision attribute for Root ← A
        o  For each possible value, $v_i$, of A,
                ▪  Add a new tree branch below Root, corresponding to the test A = $v_i$
                ▪  Let Examples $v_i$ be the subset of Examples that have value $v_i$ for A
                ▪  If Examples $v_i$ is empty
                        •  Then below this new branch add a leaf node with label = most common value of
                           Target_attribute in Examples
                        •  Else below this new branch add the sub tree
                               ID3(Examples $v_i$, Target_attribute, Attributes – {A}))
•  End
•  Return Root
```

Note: The best attribute is the one with highest information gain

The final decision tree learned by ID3 from the 14 training examples of PlayTennis is shown in Figure:



HYPOTHESIS SPACE SEARCH IN DECISION TREE LEARNING

- ID3 searches the space of possible decision trees: doing hill-climbing on information gain.
- It searches the *complete* space of all finite discrete-valued functions. All functions have at least one tree that represents them.
- It maintains only one hypothesis (unlike Candidate-Elimination). It cannot tell us how many other viable ones there are.
- It does not do back tracking. Can get stuck in local optima.
- Uses all training examples at each step. Results are less sensitive to errors.

INDUCTIVE BIAS IN DECISION TREE LEARNING

- Note H is the power set of instances X

• Inductive Bias in ID3

– Approximate inductive bias of ID3

• Shorter trees are preferred over larger tress

• BFS-ID3

– A closer approximation to the inductive bias of ID3

• Shorter trees are preferred over longer trees. Trees that place high information gain attributes close to the root are preferred over those that do not.

• Difference between ID3 & C-E (Candidate-Elimination)

ID3	**Candidate-Elimination**
Searches a complete hypothesis space incompletely	Searches an incomplete hypothesis space completely
Inductive bias is solely a consequence of the ordering of hypotheses by its search strategy	Inductive bias is solely a consequence of the expressive power of its hypothesis representation

• Restriction bias and Preference bias

Preference bias	**Restriction bias**
ID3	Candidate-Elimination
Preference for certain hypotheses over others	Categorical restriction on the set of hypotheses considered
Work within a complete hypothesis space	Possibility of excluding the unknown target function

Occam's razor

– Prefer the simplest hypothesis that fits the data

– Argument in favor

• Fewer short hypotheses than long hypotheses

– Argument opposed

• There are many ways to define small sets of hypotheses

• What's so special about small sets based on size of hypothesis?

Issues in Decision Tree Learning

• Determine how deeply to grow the decision tree
• Handling continuous attributes
• Choosing an appropriate attribute selection measure
• Handling training data with missing attribute values
• Handling attributes with differing costs
• Improving computational efficiency

Overfitting in decision trees

Definition: Given a hypothesis space H, a hypothesis h ∈ H is said to **overfit** the training data if there exists some alternative hypothesis h' ∈ H, such that h has smaller error than h' over the training examples

i.e, $error_{train}(h) < error_{train}(h')$,

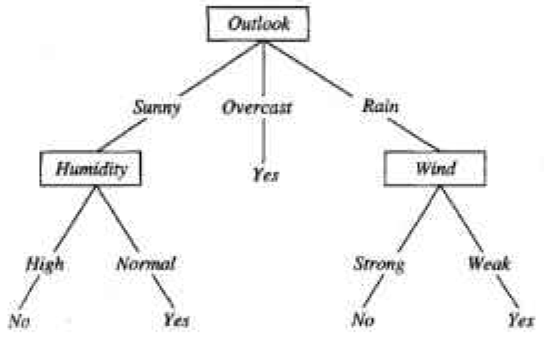
but h' has a smaller error than h over the entire distribution of instances.

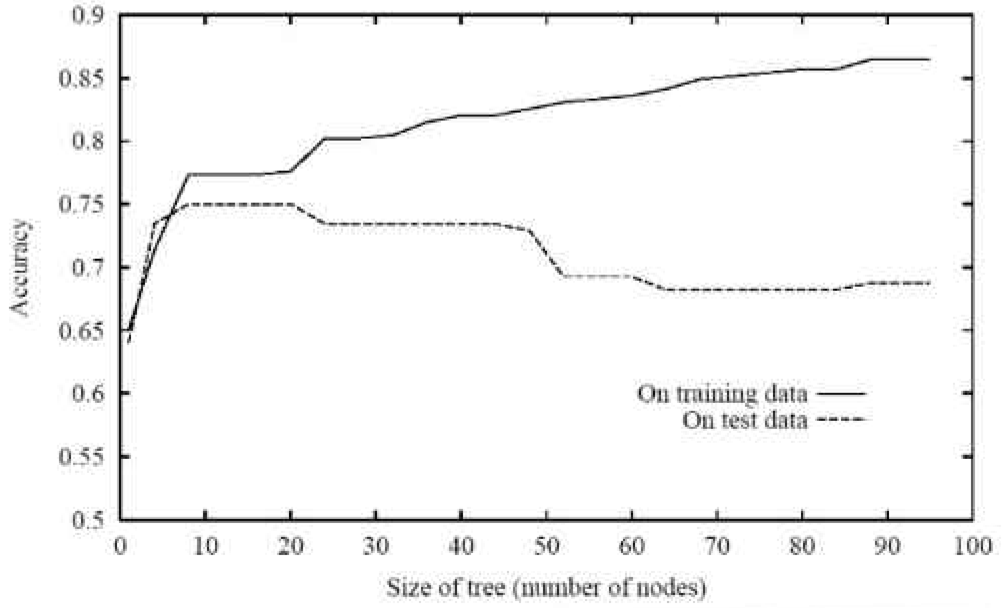
i.e $error_{D}(h) > error_{D}(h')$

Overfitting in decision trees

– Consider adding noisy training example

• <Sunny, Hot, Normal, Strong, PlayTennis = No>
• What effect on earlier tree?

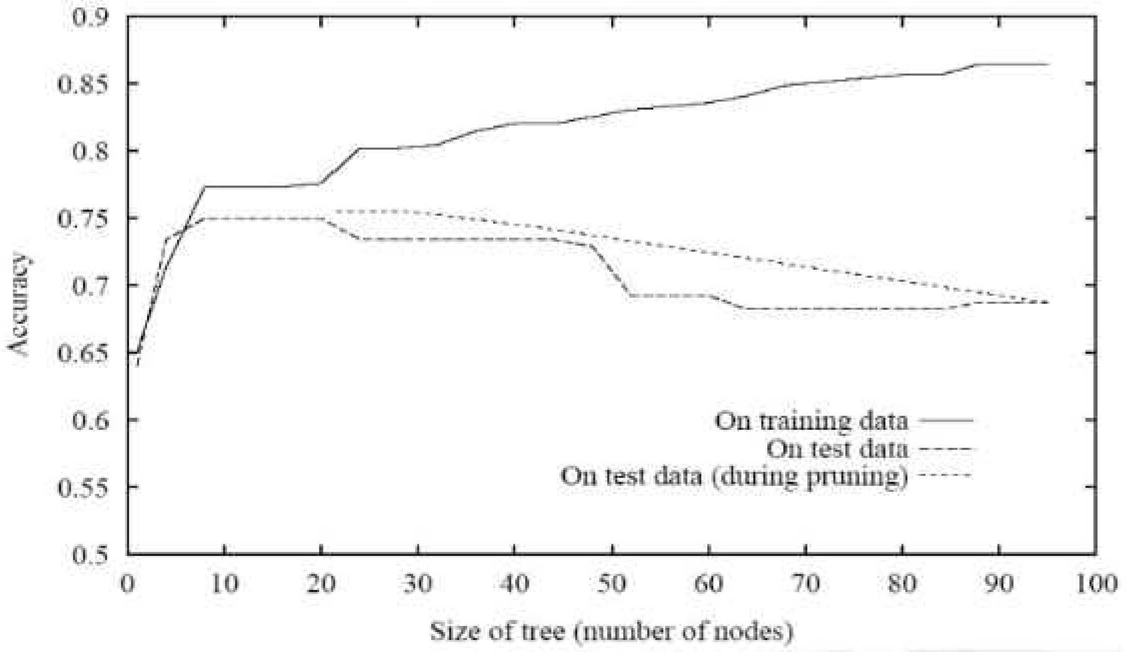




Overfitting in Decision Tree Learning

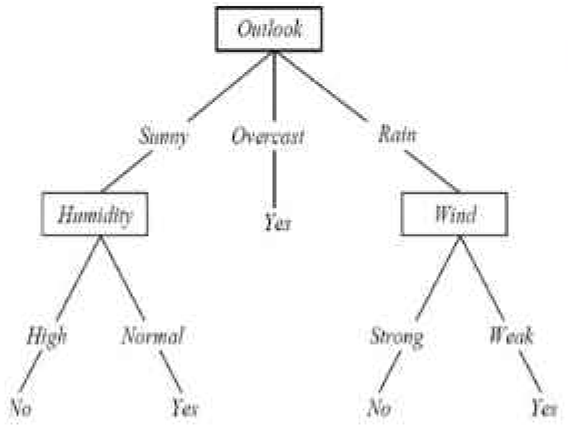
- Avoiding overfitting
  – How can we avoid overfitting?
    • Stop growing before it reaches the point where it perfectly classifies the training data
    • Grow full tree, then post-prune
  – How to select best tree?
    • Measure performance statistically over training data
    • Measure performance over separate validation data set
    • MDL: minimize the complexity for encoding the training examples and the decision trees

- Reduced-error pruning
  – Split data into training set, validation set used for pruning, and test set for measuring accuracy over future unseen examples.
  – Do until further pruning is harmful:
    1. Evaluate impact on validation set of pruning each possible node (plus those below it), starting at its maximum size and lowest accuracy over test set.
    2. Greedily remove the one that most improves validation set accuracy
  – Produces smallest version of most accurate sub tree
  – What if data is limited?



Effect of Reduced-Error Pruning in Decision Tree Learning

- **Rule post-pruning**
  - Most frequently used method (e.g., ch.4.5)
    1. Convert tree to equivalent set of rules
    2. Prune each rule independently of others
    3. Sort final rules into desired sequence for use
  - In C4.5, evaluation of performance is based on the training set itself, using pessimistic estimate :
    - Calculate the rule accuracy over the training set
    - Calculate the standard deviation in this estimated accuracy assuming on a binomial distribution.
    - The lower-bound estimate is taken as the measure of rule performance for a given confidence level.
- **Converting a tree to rules**



IF *(Outlook = Sunny)* $\wedge$ *(Humidity = High)*
THEN *PlayTennis = No*

IF *(Outlook = Sunny)* $\wedge$ *(Humidity = Normal)*
THEN *PlayTennis = Yes*

. . .

Advantages of rule post-pruning over reduced-error Pruning

• Allows distinguishing among the different contexts in which a decision node is used.

• Removes the distinction between attributes near the root and those near the leaves.

• Improves readability

• Continuous valued attributes

– Define new discrete valued attributes that partition the continuous attribute value into a discrete set of intervals

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

– Find a set of thresholds midway Between different target values of the attribute : Temperature>54 and Temperature>85

– Pick a threshold, c, that produces the greatest information gain : temperature>54

• Attributes with many values

– Problem

• If attribute has many values, Gain will select it

• Imagine using Date = Oct_13_2004 as attribute

– One approach: use GainRatio instead

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

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

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

(What if $|S_i|$ is much closer to $|S|$? SplitInformation(S,A) becomes very small so that Attribute A would be selected with large value of GainRatio(S,A) even when Gain(S,A) is small.)

• Unknown attribute values

– What if some examples missing values of A?

Use training examples, 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 and classify new examples in same fashion

• Attributes with costs

– Use low-cost attributes where possible, relying on high-cost attributes only when needed to produce reliable classficiations

– 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 ∈ [0, 1] determines importance of cost

• **Enhancements in C4.5**

– Allows for attributes that have a whole range of discrete or continuous values

– Post-pruning after induction of trees, e.g. based on test sets, in order to increase accuracy

– Uses gain ratio as the information gain measure to replace the old biased method

– Handles training data with missing attribute values by replacing them with the most common or the most probable value

Strengths of decision tree

- It produces very simple understandable rules. For smaller trees, not much mathematical and computational knowledge is required to understand this model.
- Works well for most of the problems.
- It can handle both numerical and categorical variables.
- Can work well both with small and large training data sets.
- Decision trees provide a definite clue of which features are more useful for classification.

Weaknesses of decision tree

- Decision tree models are often biased towards features having more number of possible values, i.e. levels.
- This model gets overfitted or underfitted quite easily.
- Decision trees are prone to errors in classification problems with many classes and relatively small number of training examples.
- A decision tree can be computationally expensive to train.
- Large trees are complex to understand.

Experimental Evaluation of Learning Algorithms:

Evaluating Inductive Hypotheses

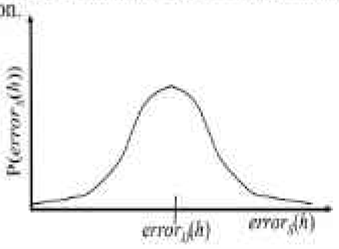
• Accuracy of hypotheses on training data is obviously biased since the hypothesis was constructed to fit this data.

• Accuracy must be evaluated on an independent (usually disjoint) test set.

• The larger the test set is, the more accurate the measured accuracy and the lower the variance observed across different test sets.

Variance in Test Accuracy

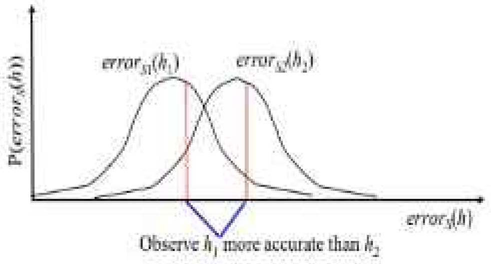
- Let $error_S(h)$ denote the percentage of examples in an independently sampled test set S of size n that are incorrectly classified by hypothesis h.
- Let $error_D(h)$ denote the true error rate for the overall data distribution D.
- When n is at least 30, the *central limit theorem* ensures that the distribution of $error_S(h)$ for different random samples will be closely approximated by a normal (Guassian) distribution.



3

Comparing Two Learned Hypotheses

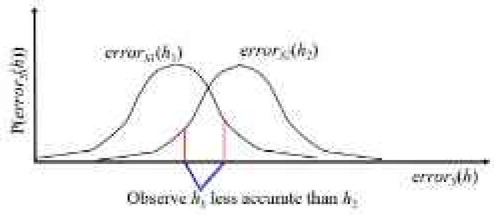
- When evaluating two hypotheses, their observed ordering with respect to accuracy may or may not reflect the ordering of their true accuracies.
  - Assume h_1 is tested on test set S_1 of size n_1
  - Assume h_2 is tested on test set S_2 of size n_2



4

Comparing Two Learned Hypotheses

- When evaluating two hypotheses, their observed ordering with respect to accuracy may or may not reflect the ordering of their true accuracies.
  - Assume h_1 is tested on test set S_1 of size n_1
  - Assume h_2 is tested on test set S_2 of size n_2



5

Statistical Hypothesis Testing

- Determine the probability that an empirically observed difference in a statistic could be due purely to random chance assuming there is no true underlying difference.
- Specific tests for determining the significance of the difference between two means computed from two samples gathered under different conditions.
- Determines the probability of the ***null hypothesis***, that the two samples were actually drawn from the same underlying distribution.
- By scientific convention, we ***reject the null hypothesis*** and say the difference is ***statistically significant*** if the probability of the null hypothesis is less than 5% ($p < 0.05$) or alternatively we accept that the difference is due to an underlying cause with a ***confidence*** of $(1 - p)$.

6

One-sided vs Two-sided Tests

- One-sided test assumes you expected a difference in one direction (A is better than B) and the observed difference is consistent with that assumption.
- Two-sided test does not assume an expected difference in either direction.
- Two-sided test is more conservative, since it requires a larger difference to conclude that the difference is significant.

7

Z-Score Test for Comparing Learned Hypotheses

- Assumes h_1 is tested on test set S_1 of size n_1 and h_2 is tested on test set S_2 of size n_2.
- Compute the difference between the accuracy of h_1 and h_2

$$d = \left| error_{S_1}(h_1) - error_{S_2}(h_2) \right|$$

- Compute the standard deviation of the sample estimate of the difference.

$$\sigma_d = \sqrt{\frac{error_{S_1}(h_1) \cdot (1 - error_{S_1}(h_1))}{n_1} + \frac{error_{S_2}(h_2) \cdot (1 - error_{S_2}(h_2))}{n_2}}$$

- Compute the z-score for the difference

$$z = \frac{d}{\sigma_d}$$

Z-Score Test for Comparing Learned Hypotheses (continued)

- Determine the confidence in the difference by looking up the highest confidence, *C*, for the given z-score in a table.

confidence level	50%	68%	80%	90%	95%	98%	99%
z-score	0.67	1.00	1.28	1.64	1.96	2.33	2.58

- This gives the confidence for a two-tailed test, for a one tailed test, increase the confidence half way towards 100%

$$C' = (100 - \frac{(100 - C)}{2})$$

Sample Z-Score Test 1

Assume we test two hypotheses on different test sets of size 100 and observe:

$$error_{S_1}(h_1) = 0.20 \qquad error_{S_2}(h_2) = 0.30$$

$$d = \left|error_{S_1}(h_1) - error_{S_2}(h_2)\right| = |0.2 - 0.3| = 0.1$$

$$\sigma_d = \sqrt{\frac{error_{S_1}(h_1)\cdot(1 - error_{S_1}(h_1))}{n_1} + \frac{error_{S_2}(h_2)\cdot(1 - error_{S_2}(h_2))}{n_2}}$$

$$= \sqrt{\frac{0.2\cdot(1-0.2)}{100} + \frac{0.3\cdot(1-0.3)}{100}} = 0.0608$$

$$z = \frac{d}{\sigma_d} = \frac{0.1}{0.0608} = 1.644$$

Confidence for two-tailed test: 90%

Confidence for one-tailed test: (100 – (100 – 90)/2) = 95%

Sample Z-Score Test 2

Assume we test two hypotheses on different test sets of size 100 and observe:

$$error_{S_1}(h_1) = 0.20 \qquad error_{S_2}(h_2) = 0.25$$

$$d = \left|error_{S_1}(h_1) - error_{S_2}(h_2)\right| = |0.2 - 0.25| = 0.05$$

$$\sigma_d = \sqrt{\frac{error_{S_1}(h_1)\cdot(1 - error_{S_1}(h_1))}{n_1} + \frac{error_{S_2}(h_2)\cdot(1 - error_{S_2}(h_2))}{n_2}}$$

$$= \sqrt{\frac{0.2\cdot(1-0.2)}{100} + \frac{0.25\cdot(1-0.25)}{100}} = 0.0589$$

$$z = \frac{d}{\sigma_d} = \frac{0.05}{0.0589} = 0.848$$

Confidence for two-tailed test: 50%

Confidence for one-tailed test: (100 – (100 – 50)/2) = 75%

Z-Score Test Assumptions

- Hypotheses can be tested on different test sets; if same test set used, stronger conclusions might be warranted.
- Test sets have at least 30 independently drawn examples.
- Hypotheses were constructed from independent training sets.
- Only compares two specific hypotheses regardless of the methods used to construct them. Does not compare the underlying learning methods in general.

Comparing Learning Algorithms

- Comparing the average accuracy of hypotheses produced by two different learning systems is more difficult since we need to average over multiple training sets. Ideally, we want to measure:

$$E_{S \subset D}(error_D(L_A(S)) - error_D(L_B(S)))$$

where $L_A(S)$ represents the hypothesis learned by method *L* from training data *S*.

- To accurately estimate this, we need to average over multiple, independent training and test sets.
- However, since labeled data is limited, generally must average over multiple splits of the overall data set into training and test sets.

K-Fold Cross Validation

```
Randomly partition data $D$ into $k$ disjoint equal-sized
    subsets $P_1 \ldots P_k$
For $i$ from 1 to $k$ do:
    Use $P_i$ for the test set and remaining data for training
        $S_i = (D - P_i)$
        $h_A = L_A(S_i)$
        $h_B = L_B(S_i)$
        $\delta_i = \text{error}_{P_i}(h_A) - \text{error}_{P_i}(h_B)$
Return the average difference in error:
        $\delta = \frac{1}{k}\sum_{i=1}^{k}\delta_i$
```

K-Fold Cross Validation Comments

- Every example gets used as a test example once and as a training example k–1 times.
- All test sets are independent; however, training sets overlap significantly.
- Measures accuracy of hypothesis generated for $[(k-1)/k]\cdot|D|$ training examples.
- Standard method is 10-fold.
- If k is low, not sufficient number of train/test trials; if k is high, test set is small and test variance is high and run time is increased.
- If $k=|D|$, method is called *leave-one-out* cross validation.

Significance Testing

- Typically $k<30$, so not sufficient trials for a z test.
- Can use (***Student's***) ***t-test***, which is more accurate when number of trials is low.
- Can use a ***paired*** t-test, which can determine smaller differences to be significant when the training/sets sets are the same for both systems.
- However, both z and t test's assume the trials are independent. Not true for k-fold cross validation:
  - Test sets are independent
  - Training sets are **not** independent
- Alternative statistical tests have been proposed, such as McNemar's test.
- Although no test is perfect when data is limited and independent trials are not practical, some statistical test that accounts for variance is desirable.

Sample Experimental Results

Which experiment provides better evidence that SystemA is better than SystemB?

Experiment 1

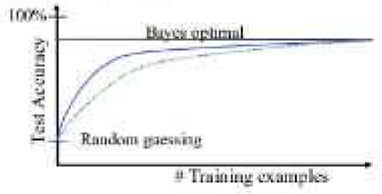
	SystemA	SystemB	Diff
Trial 1	87%	82%	+5%
Trail 2	83%	78%	+5%
Trial 3	88%	83%	+5%
Trial 4	82%	77%	+5%
Trial 5	85%	80%	+5%
Average	85%	80%	+5%

Experiment 2

	SystemA	SystemB	Diff
Trial 1	90%	82%	+8%
Trail 2	93%	76%	+17%
Trial 3	80%	85%	–5%
Trial 4	85%	75%	+10%
Trial 5	77%	82%	– 5%
Average	85%	80%	+5%

Learning Curves

- Plots accuracy vs. size of training set.
- Has maximum accuracy (Bayes optimal) nearly been reached or will more examples help?
- Is one system better when training data is limited?
- Most learners eventually converge to Bayes optimal given sufficient training examples.

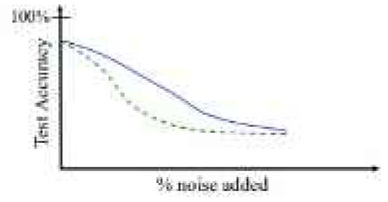


Cross Validation Learning Curves

```
Split data into $k$ equal partitions
For trial $i$ = 1 to $k$ do:
    Use partition $i$ for testing and the union of all other partitions for training.
    For each desired point $p$ on the learning curve do:
        For each learning system $L$
            Train $L$ on the first $p$ examples of the training set and record
                training time, training accuracy, and learned concept complexity.
            Test $L$ on the test set, recording testing time and test accuracy.
Compute average for each performance statistic across $k$ trials.
Plot curves for any desired performance statistic versus training set size.
Use a paired t-test to determine significance of any differences between any
  two systems for a given training set size.
```

Noise Curves

- Plot accuracy versus noise level to determine relative resistance to noisy training data.
- Artificially add category or feature noise by randomly replacing some specified fraction of category or feature values with random values.


Test Accuracy
% noise added