



MACHINE LEARNING

Vasant Honavar
Artificial Intelligence Research Laboratory
Department of Computer Science
Bioinformatics and Computational Biology Program
Center for Computational Intelligence, Learning, & Discovery
Iowa State University
honavar@cs.iastate.edu
www.cs.iastate.edu/~honavar/
www.cild.iastate.edu/

Decision Tree Classifiers

- Decision tree Representation for modeling dependencies among input variables using
- Elements of information theory
- How to learn decision trees from data
- Over-fitting and how to minimize it
- How to deal with missing values in the data
- Learning decision trees from distributed data
- Learning decision trees at multiple levels of abstraction

Decision tree representation

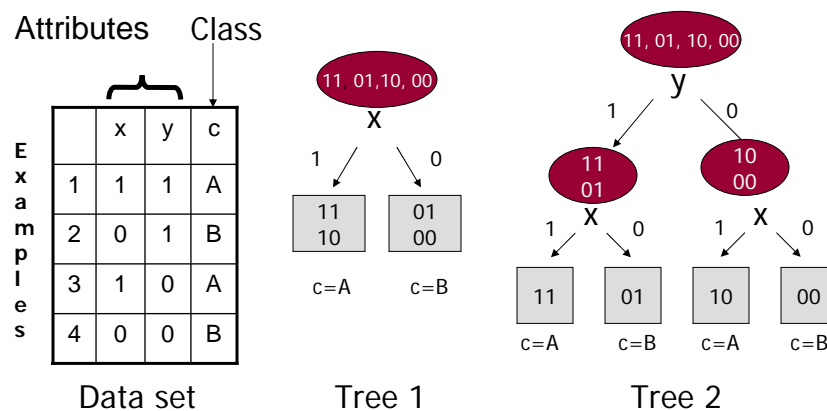
In the simplest case,

- each internal node tests on an attribute
- each branch corresponds to an attribute value
- each leaf node corresponds to a class label

In general,

- each internal node corresponds to a test (on input instances) with mutually exclusive and exhaustive outcomes – tests may be univariate or multivariate
- each branch corresponds to an outcome of a test
- each leaf node corresponds to a class label

Decision Tree Representation



Should we choose Tree 1 or Tree 2? Why?

Decision tree representation

- Any Boolean function can be represented by a decision tree

- Any function $f : A_1 \times A_2 \times \dots \times A_n \rightarrow C$

where each A_i is the domain of the i th attribute and C is a discrete set of values (class labels) can be represented by a decision tree

- In general, the inputs need not be discrete valued

Learning Decision Tree Classifiers

- Decision trees are especially well suited for representing simple rules for classifying instances that are described by discrete attribute values
- Decision tree learning algorithms
 - Implement Ockham's razor as a preference bias (simpler decision trees are preferred over more complex trees)
 - Are relatively efficient – linear in the size of the decision tree and the size of the data set
 - Produce comprehensible results
 - Are often among the first to be tried on a new data set

Learning Decision Tree Classifiers

- Ockham's razor recommends that we pick the simplest decision tree that is consistent with the training set
- Simplest tree is one that takes the fewest bits to encode (why? – information theory)
- There are far too many trees that are consistent with a training set
- Searching for the simplest tree that is consistent with the training set is not typically computationally feasible

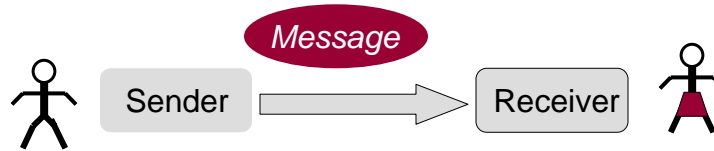
Solution

- Use a greedy algorithm – not guaranteed to find the simplest tree – but works well in practice
- Or restrict the space of hypothesis to a subset of simple

Information – Some intuitions

- Information **reduces uncertainty**
- Information is relative – to what you already know
- Information content of a message is related to how surprising the message is
- Information is related Information **depends on context**

Digression: Information and Uncertainty

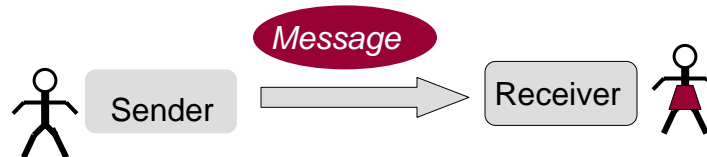


You are stuck inside. You send me out to report back to you on what the weather is like. I do not lie, so you trust me. You and I are both generally familiar with the weather in Iowa

On a **July** afternoon in **Iowa**, I walk into the room and tell you it is **hot** outside

On a **January** afternoon in **Iowa**, I walk into the room and tell you it is **hot** outside

Digression: Information and Uncertainty



How much information does a message contain?

If my message to you describes a scenario that you expect with certainty, the information content of the message for you is zero

The more surprising the message to the receiver, the greater the amount of information conveyed by the message

What does it mean for a message to be surprising?

Digression: Information and Uncertainty

Suppose I have a coin with heads on both sides and you know that I have a coin with heads on both sides.

I toss the coin, and without showing you the outcome, tell you that it came up heads. How much information did I give you?

Suppose I have a fair coin and you know that I have a fair coin.

I toss the coin, and without showing you the outcome, tell you that it came up heads. How much information did I give you?

Information

- Without loss of generality, assume that messages are binary – made of 0s and 1s.
- Conveying the outcome of a fair coin toss requires 1 bit of information – need to identify one out of two equally likely outcomes
- Conveying the outcome one of an experiment with 8 equally likely outcomes requires 3 bits ..
- Conveying an outcome of that is certain takes 0 bits
- In general, if an outcome has a probability p , the information content of the corresponding message is

$$I(p) = -\log_2 p$$

$$I(0) = 0$$

Information is Subjective

- Suppose there are 3 agents – Adrian, Oksana, Jun, in a world where a dice has been tossed. Adrian observes the outcome is a “6” and whispers to Oksana that the outcome is “even” but Jun knows nothing about the outcome.
- Probability assigned by Oksana to the event “6” is a subjective measure of Oksana’s belief about the state of the world.
- Information gained by Adrian by looking at the outcome of the dice = $\log_2 6$ bits.
- Information conveyed by Adrian to Oksana = $\log_2 6 - \log_2 3$ bits
- Information conveyed by Adrian to Jun = 0 bits

Information and Shannon Entropy

- Suppose we have a message that conveys the result of a random experiment with m possible discrete outcomes, with probabilities

$$p_1, p_2, \dots, p_m$$

The **expected information content** of such a message is called the **entropy** of the probability distribution

$$H(p_1, p_2, \dots, p_m) = \sum_{i=1}^m p_i I(p_i)$$

$$I(p_i) = -\log_2 p_i \text{ provided } p_i \neq 0$$

$$I(p_i) = 0 \text{ otherwise}$$

Shannon's entropy as a measure of information

Let $\vec{P} = (p_1 \dots p_n)$ be a discrete probability distribution

The entropy of the distribution P is given by

$$H(\vec{P}) = \sum_{i=1}^n p_i \log_2 \left(\frac{1}{p_i} \right) = - \sum_{i=1}^n p_i \log_2(p_i)$$

$$H\left(\frac{1}{2}, \frac{1}{2}\right) = - \sum_{i=1}^2 p_i \log_2(p_i) = -\left(\frac{1}{2}\right) \log_2\left(\frac{1}{2}\right) - \left(\frac{1}{2}\right) \log_2\left(\frac{1}{2}\right) = 1 \text{ bit}$$

$$H(0,1) = - \sum_{i=1}^2 p_i \log_2(p_i) = -1I(1) - 0I(0) = 0 \text{ bit}$$

Properties of Shannon's entropy

$$\forall \vec{P} \quad H(\vec{P}) \geq 0$$

If there are N possible outcomes, $H(\vec{P}) \leq \log_2 N$

If $\forall i \ p_i = \frac{1}{N}$, $H(\vec{P}) = \log_2 N$

If $\exists i$ such that $p_i = 1$, $H(\vec{P}) = 0$

$H(\vec{P})$ is a continuous function of \vec{P}

Shannon's entropy as a measure of information

- For any distribution \vec{P} , $H(\vec{P})$ is the optimal number of binary questions required on average to determine an outcome drawn from P .
- We can extend these ideas to talk about how much information is conveyed by the observation of the outcome of one experiment about the possible outcomes of another (mutual information)
- We can also quantify the difference between two probability distributions (Kullback-Liebler divergence or relative entropy)

Coding Theory Perspective

- Suppose you and I both know the distribution \vec{P}
- I choose an outcome according to \vec{P}
- Suppose I want to send you a message about the outcome
- You and I could agree in advance on the questions
- I can simply send you the answers
- Optimal message length on average is $H(\vec{P})$
- This generalizes to noisy communication

Entropy of random variables and sets of random variables

For a random variable X taking values $a_1 \dots a_n$,

$$\begin{aligned} H(X) &= -\sum_X P(X) \log_2 P(X) \\ &= -\sum_{i=1}^n P(X = a_i) \log_2 P(X = a_i) \end{aligned}$$

If \mathbf{X} is a set of random variables,

$$H(\mathbf{X}) = -\sum_{\mathbf{x}} P(\mathbf{X}) \log_2 P(\mathbf{X})$$

Joint Entropy and Conditional Entropy

For random variables X and Y , the joint entropy

$$H(X, Y) = -\sum_{X, Y} P(X, Y) \log_2 P(X, Y)$$

Conditional entropy of X given Y

$$\begin{aligned} H(X | Y) &= -\sum_{X, Y} P(X, Y) \log_2 P(X | Y) \\ &= \sum_Y P(Y) H(X | Y = a) \end{aligned}$$

$$H(X | Y = a) = -\sum_X P(X, Y = a) \log_2 P(X | Y = a)$$

Joint Entropy and Conditional Entropy

Some Useful results :

$$\left. \begin{aligned} H(X, Y) &\leq H(X) + H(Y) \\ H(Y | X) &\leq H(Y) \end{aligned} \right\}$$

(When do we have equality?)

Chain rule for Entropy

$$\begin{aligned} H(X, Y) &= H(X) + H(Y|X) \\ &= H(Y) + H(X|Y) \end{aligned}$$

Example of entropy calculations

$$P(X = H; Y = H) = 0.2. \quad P(X = H; Y = T) = 0.4$$

$$P(X = T; Y = H) = 0.3. \quad P(X = T; Y = T) = 0.1$$

$$H(X, Y) = -0.2 \log_2 0.2 + \dots \approx 1.85$$

$$P(X = H) = 0.6. \quad H(X) = 0.97$$

$$P(Y = H) = 0.5. \quad H(Y) = 1.0$$

$$P(Y = H | X = H) = 0.2/0.6 = 0.333$$

$$P(Y = T | X = H) = 1 - 0.333 = 0.667$$

$$P(Y = H | X = T) = 0.3/0.4 = 0.75$$

$$P(Y = T | X = T) = 0.1/0.4 = 0.25$$

$$H(Y|X) \approx 0.88$$

Mutual Information

For a random variable X and Y , the average mutual information between X and Y

$$I(X, Y) = H(X) + H(Y) - H(X, Y)$$

Or by using chain rule

$$H(X, Y) = H(X) + H(Y | X) = H(Y) + H(X | Y)$$

$$I(X, Y) = H(X) - H(X | Y)$$

$$I(X, Y) = H(Y) - H(Y | X)$$

In terms of probability distributions,

$$I(X, Y) = \sum_{X, Y} P(X = a, Y = b) \log_2 \frac{P(X = a, Y = b)}{P(X = a)P(Y = b)}$$

Question: When is $I(X, Y) = 0$?

Relative Entropy

Let P and Q be two distributions over random variable X .
The relative entropy (Kullback - Liebler distance)
is a measure of "distance" from P to Q .

$$D(P \parallel Q) = \sum_x P(X) \log_2 \left(\frac{P(X)}{Q(X)} \right)$$

Note $D(P \parallel Q) \neq D(Q \parallel P)$

$$D(P \parallel Q) \geq 0$$

$$D(P \parallel P) = 0$$

A Recipe for Learning

$$P(h | D) = \frac{P(D | h)P(h)}{P(D)}$$

$P(h)$ = prior probability of hypothesis h

$P(D)$ = prior probability of training data D

$P(h | D)$ = probability of h given D

$P(D | h)$ = probability of D given h

A Recipe for learning

Maximum a posteriori hypothesis

$$\begin{aligned}
 h_{MAP} &= \arg \max_{h \in H} P(h | D) \\
 &= \arg \max_{h \in H} \frac{P(D | h)P(h)}{P(D)} \\
 &= \arg \max_{h \in H} P(D | h)P(h)
 \end{aligned}$$

Maximum likelihood hypothesis

If $\forall h_i, h_j \in H \ P(h_i) = P(h_j)$,

$$h_{ML} = \arg \max_{h \in H} P(D | h)$$

Brute Force MAP Hypothesis Learner

- For each hypothesis h in H , calculate the posterior probability

$$P(h | D) = \frac{P(D | h)P(h)}{P(D)}$$

- Output the hypothesis with the highest posterior probability

$$h_{MAP} = \arg \max_{h_i \in H} P(D | h_i)P(h_i)$$

$$h_{ML} = \arg \max_{h_i \in H} P(D | h_i)$$

Brute Force MAP Hypothesis Learner

$$\begin{aligned}
 h_{MAP} &= \arg \max_{h_i \in H} P(D | h_i) P(h_i) \\
 &= \arg \max_{h_i \in H} (\log_2 P(D|h_i) + \log_2 P(h_i)) \\
 &= \arg \min_{h_i \in H} (-\log_2 P(D|h_i) - \log_2 P(h_i))
 \end{aligned}$$

$-\log_2 P(h_i)$ = description length of h_i under optimal encoding of hypotheses
 $-\log_2 P(D | h_i)$ = description length of data D given h_i under optimal encoding based on $P(D | h_i)$
 \therefore Bayesian learning implies Occam's razor!

MAP Learning of 2-class classifiers

Consider a 2-class learning problem defined over an Instance space X , hypothesis space H , training examples

$$D = \{e_i = (X_i, d_i) | X_i \in X; d_i = c(X_i)\}$$

Consider a learning algorithm which outputs a hypothesis that is *consistent* with the examples in D

- Let $V_{H,D}$ be the subset of hypotheses in H that are consistent with D (the version space)
- What would Bayes rule produce as the MAP hypothesis?

MAP Learning of Binary Concepts (2-class classifiers)

Assumptions

1. The hypothesis space is finite.
2. The training examples are i.i.d.
3. The training data is noise free.

$$\forall d_i, d_i = c(X_i)$$

4. The target concept $c \in H$
5. All hypotheses are a priori equally likely :

$$\forall h \in H, P(h) = \frac{1}{|H|}$$

MAP Learning of 2-class classifiers

i.i.d. samples imply

$$\begin{aligned} P(D | h) &= \prod_i P(e_i | h) = \prod_i P(X_i, d_i | h) \\ &= \prod_i P(d_i | h, X_i) P(X_i | h) \end{aligned}$$

Assuming X_i are independent of h ,

$$P(D | h) = \prod_i P(d_i | h, X_i) P(X_i)$$

$$P(d_i | h, X_i) = 1 \text{ if } h(X_i) = d_i;$$

$$P(d_i | h, X_i) = 0 \text{ otherwise}$$

MAP Learning of 2-class classifiers

If h is consistent with D , that is,

$\forall i \in \{1 \dots m\}, d_i = h(X_i)$ then

$$P(D | h) = \prod_{i=1}^m P(X_i)$$

If h is inconsistent with D , that is,

$\exists i \in \{1 \dots m\}$ such that $d_i \neq h(X_i)$ then

$$P(D | h) = 0$$

MAP Learning of 2-class classifiers

$$\begin{aligned} P(D) &= \sum_{h \in H} P(D|h)P(h) = \sum_{h \in V_{H,D}} \left(\frac{1}{|H|} \right) \left(\prod_i P(X_i) \right) + \sum_{h \notin V_{H,D}} 0 \\ &= \frac{|V_{H,D}|}{|H|} \left(\prod_i P(X_i) \right) \end{aligned}$$

$\therefore \forall h \in V_{H,D}$, we have

$$P(h | D) = \frac{P(D|h)P(h)}{P(D)} = \left(\frac{\left(\prod_i P(X_i) \right)}{|H|} \right) \left(\frac{|H|}{|V_{H,D}| \left(\prod_i P(X_i) \right)} \right) = \left(\frac{1}{|V_{H,D}|} \right)$$

and $\forall h \notin V_{H,D}$, we have $P(h | D) = 0$

Every hypothesis in H that is consistent with D is a MAP hypothesis

Bayesian Recipe for Learning

If the training examples are

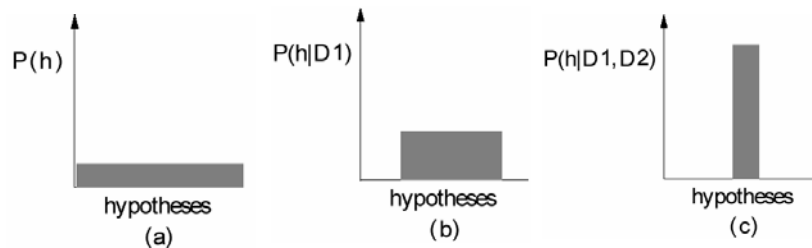
- independently identically distributed
- noise-free

And if each candidate hypothesis in H is equally a priori equally likely

Then every hypothesis that is consistent with the training data (that is, correctly classifies each training example) maximizes $P(h|D)$

In this setting, all that the learner has to do is to produce a hypothesis from H that is consistent with the training data

Effect of Data (Evidence) on the Version Space



Bayesian Recipe for Learning

$$h_{MAP} = \arg \min_{h_i \in H} \left(\underbrace{-\log_2 P(D|h_i)}_{\text{Data encoding}} - \underbrace{\log_2 P(h_i)}_{\text{Hypothesis encoding}} \right)$$

The number of bits needed to encode the Data D given the hypothesis h_i (under optimal encoding) – in other words, error of h or exceptions to h

The number of bits needed to encode the hypothesis h_i (under optimal encoding)

Bayesian Recipe for Learning

Suppose the training data are iid and noise-free but each hypothesis in H is not equally a priori equally likely

$$\begin{aligned} h_{MAP} &= \arg \max_{h_i \in H} P(D | h_i) P(h_i) \\ &= \arg \max_{h_i \in H} (\log_2 P(D|h_i) + \log_2 P(h_i)) \\ &= \arg \min_{h_i \in H} (-\log_2 P(D|h_i) - \log_2 P(h_i)) \end{aligned}$$

Bayesian Recipe for Learning

If

- the training data are iid and noise-free
- but each hypothesis in H is not equally a priori equally likely

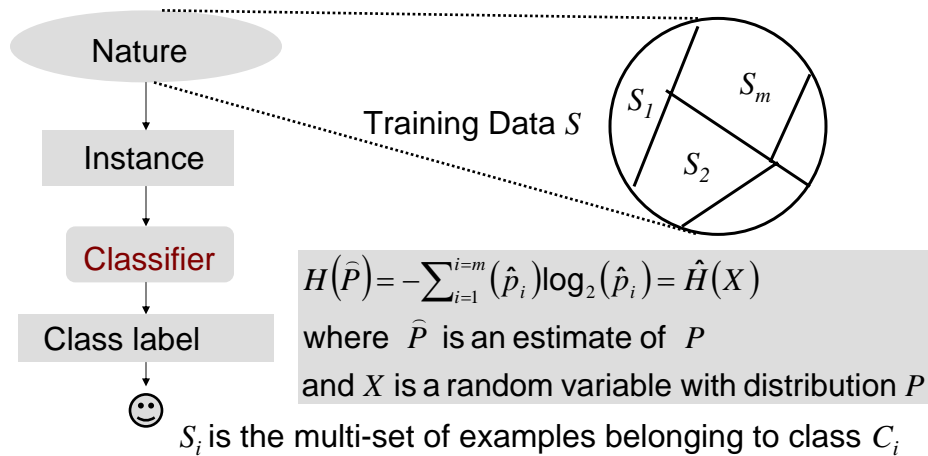
then

- the learner has to produce a hypothesis from H that trades off the error on the training data against complexity of the hypothesis

Decision tree classifiers, which we will look at next, provide an example of this approach

Learning Decision Tree Classifiers

On average, the information needed to convey the class membership of a random instance drawn from nature is $H(\bar{P})$



Learning Decision Tree Classifiers

The task of the learner then is to extract the needed information from the training set and store it in the form of a decision tree for classification

Information gain based decision tree learner

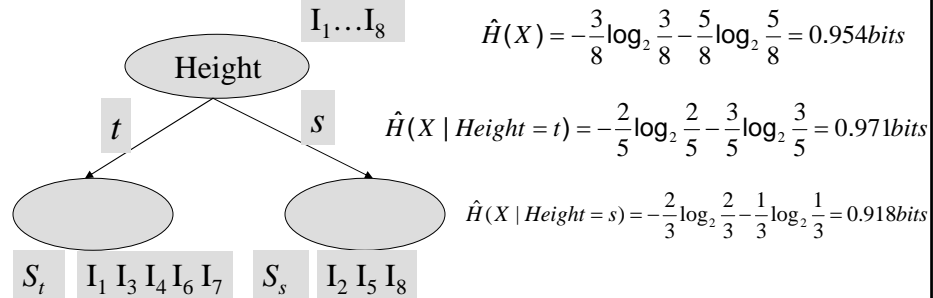
Start with the entire training set at the root

Recursively add nodes to the tree
 corresponding to tests that yield the
 greatest expected reduction in entropy
 (or the largest expected information gain)
 until some termination criterion is met
 (e.g., the training data at every leaf node
 has zero entropy)

Learning Decision Tree Classifiers - Example

	Training Data	
	Instance	Class label
Instances – ordered 3-tuples of attribute values corresponding to <i>Height</i> (<u>t</u> all, <u>s</u> hort) <i>Hair</i> (<u>d</u> ark, <u>b</u> londe, <u>r</u> ed) <i>Eye</i> (<u>b</u> lue, <u>b</u> rown)	I_1 (t, d, l)	+
	I_2 (s, d, l)	+
	I_3 (t, b, l)	–
	I_4 (t, r, l)	–
	I_5 (s, b, l)	–
	I_6 (t, b, w)	+
	I_7 (t, d, w)	+
	I_8 (s, b, w)	+

Learning Decision Tree Classifiers - Example

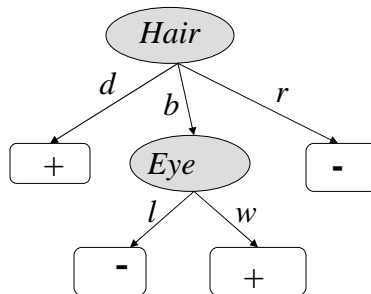


$$\hat{H}(X | \text{Height}) = \frac{5}{8} \hat{H}(X | \text{Height} = t) + \frac{3}{8} \hat{H}(X | \text{Height} = s) = \frac{5}{8}(0.971) + \frac{3}{8}(0.918) = 0.95 \text{bits}$$

Similarly, $\hat{H}(X | \text{Eye}) = 0.607 \text{bits}$ and $\hat{H}(X | \text{Hair}) = 0.5 \text{bits}$

Hair is the most informative because it yields the largest reduction in entropy. Test on the value of *Hair* is chosen to correspond to the root of the decision tree

Learning Decision Tree Classifiers - Example



Compare the result with Naïve Bayes

In practice, we need some way to prune the tree to avoid overfitting the training data – more on this later.

Learning, generalization, overfitting

Consider the error of a hypothesis h over

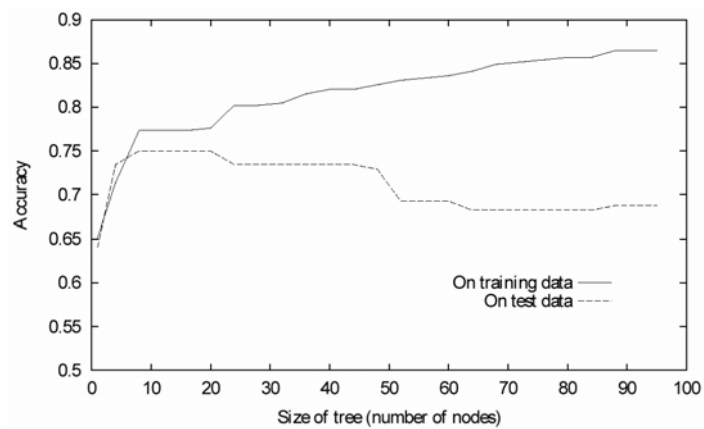
- training data: $Error_{Train}(h)$
- entire distribution D of data: $Error_D(h)$

Hypothesis $h \in H$ **over fits** training data if there is an alternative hypothesis $h' \in H$ such that

and $Error_{Train}(h) < Error_{Train}(h')$

$$Error_D(h) > Error_D(h')$$

Over fitting in decision tree learning



Causes of over fitting

As we move further away from the root, the data set used to choose the best test becomes smaller → poor estimates of entropy

Noisy examples can further exacerbate over fitting

Minimizing over fitting

- Use roughly the same size sample at every node to estimate entropy – when there is a large data set from which we can sample
- Stop when further split fails to yield statistically significant information gain (estimated from validation set)
- Grow full tree, then prune
- minimize $size(tree) + size(exception\ (tree))$

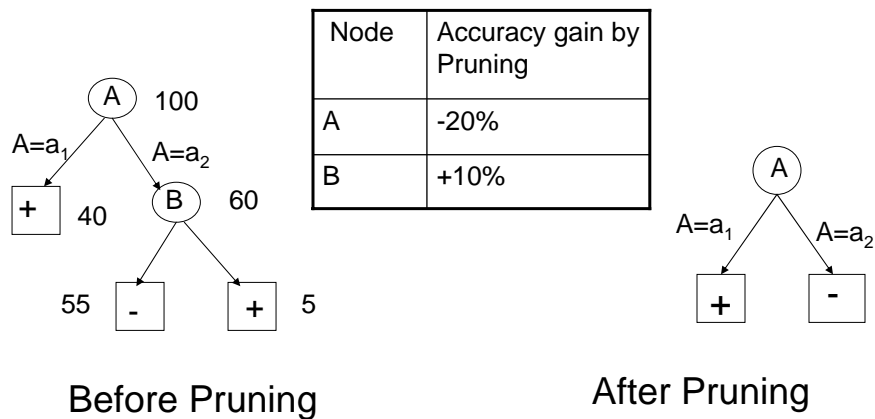
Reduced error pruning

Each decision node in the tree is considered as a candidate for pruning

Pruning a decision node consists of

- removing the sub tree rooted at that node,
- making it a leaf node
- Updating the distribution of training instances at that node

Reduced error pruning – Example



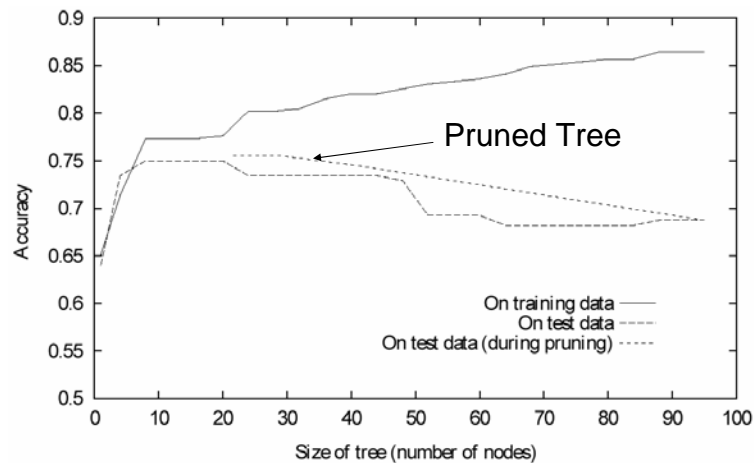
Reduced error pruning

Do until further pruning is harmful:

- Evaluate impact on **validation** set of pruning each candidate node
- Greedily select a node which most improves the performance on the **validation** set when the sub tree rooted at that node is pruned

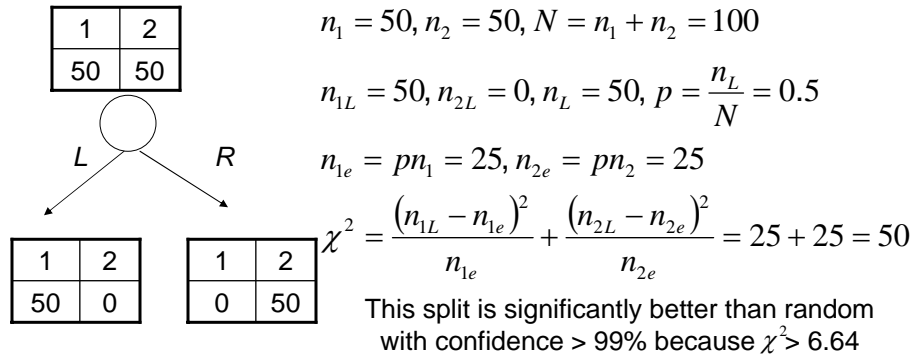
Drawback – holding back the validation set limits the amount of training data available – not desirable when data set is small

Reduced error pruning



Pruning based on whether information gain is **significantly** greater than zero

Evaluate Candidate split to decide if the resulting information gain is significantly greater than zero as determined using a suitable hypothesis testing method at a desired significance level



Copyright Vasant Honavar, 2006.

Pruning based on whether information gain is **significantly** greater than zero

Evaluate Candidate split to decide if the resulting information gain is significantly greater than zero as determined using a suitable hypothesis testing method at a desired significance level

Example: χ^2 statistic

In the 2-class, binary (L,R) split case,

n_1 of class 1, n_2 of class 2; $N = n_1 + n_2$

Split sends pN to L and $(1-p)N$ to R

Random split would send pn_1 of class 1 to L and pn_2 of class 2 to L

The critical value of χ^2 depends on the degrees of freedom which is 1 in this case (for a given p , n_{1L} fully specifies n_{2L} , n_{1R} and n_{2R})

In general, the number of degrees of freedom can be > 1

$$\chi^2 = \sum_{i=1}^2 \frac{(n_{iL} - n_{ie})^2}{n_{ie}}$$

Copyright Vasant Honavar, 2006.

Pruning based on whether information gain is **significantly** greater than zero

$$\chi^2 = \sum_{j=1}^{Branches-1} \sum_{i=1}^{Classes} \frac{(n_{ij} - n_{ie_j})^2}{n_{ie_j}}$$

$$N = n_1 + n_2 + \dots + n_{Classes}$$

$$p = [p_1 p_2 \dots p_{Branches}]; \quad \sum_{j=1}^{Branches} p_j = 1$$

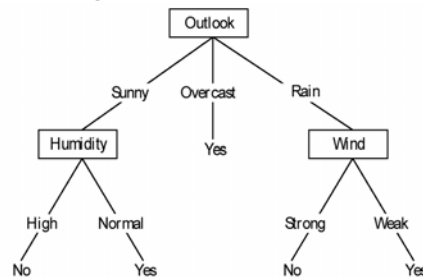
$$n_{ie_j} = p_j n_i$$

$$\text{Degrees of freedom} = (Classes - 1)(Branches - 1)$$

The greater the value of χ^2 the less likely it is that the split is random. For a sufficiently high value of χ^2 , the difference between the expected (random) split is statistically significant and we reject the null hypothesis that the split is random.

Rule post-pruning

Convert tree to equivalent set of rules



IF (Outlook = Sunny) \wedge (Humidity = High)

THEN PlayTennis = No

IF (Outlook = Sunny) \wedge (Humidity = Normal)

THEN PlayTennis = Yes

...

Rule post-pruning

1. Convert tree to equivalent set of rules
2. Prune each rule independently of others by dropping a condition at a time if doing so does not reduce estimated accuracy (at the desired confidence level)
3. Sort final rules in order of lowest to highest error

Advantage – can potentially correct bad choices made close to the root

Post pruning based on validation set is the most commonly used method in practice

Development of pre pruning methods with comparable performance that do not require a validation set is an open problem

Classification of instances

- Unique classification – possible when each leaf has zero entropy and there are no missing attribute values
- Most likely classification – based on distribution of classes at a node when there are no missing attribute values
- Probabilistic classification – based on distribution of classes at a node when there are no missing attribute values

Handling different types of attribute values

Types of attributes

- Nominal – values are names
- Ordinal – values are ordered
- Cardinal (Numeric) – values are numbers
(hence ordered)

....

Handling numeric attributes

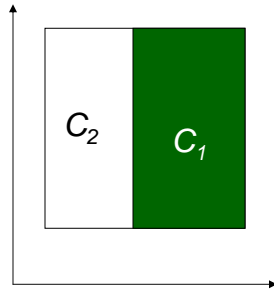
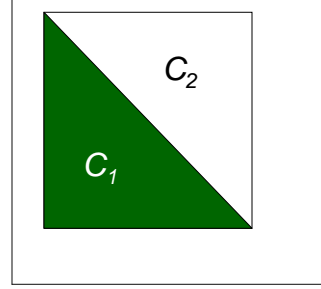
Attribute T	40	48	50	54	60	70
Class	N	N	Y	Y	Y	N

Candidate splits $T > \frac{(48+50)}{2}?$ $T > \frac{(60+70)}{2}?$

$$E(S | T > 49?) = \frac{2}{6}(0) + \frac{4}{6} \left(-\left(\frac{3}{4}\right) \log_2 \left(\frac{3}{4}\right) - \left(\frac{1}{4}\right) \log_2 \left(\frac{1}{4}\right) \right)$$

- Sort instances by value of numeric attribute under consideration
- For each attribute, find the test which yields the lowest entropy
- Greedily choose the best test across all attributes

Handling numeric attributes

Axis-parallel splitOblique split

Oblique splits cannot be realized by univariate tests

Two-way versus multi-way splits

Entropy criterion favors many-valued attributes

Pathological behavior – what if in a medical diagnosis data set, social security number is one of the candidate attributes? $A = value$ versus $A = \neg value$

Solutions

Only two-way splits (CART)

Entropy ratio (C4.5)

$$EntropyRatio(S | A) \equiv \frac{Entropy(S | A)}{SplitEntropy(S | A)}$$

$$SplitEntropy(S | A) \equiv - \sum_{i=1}^{|values(A)|} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}$$

Alternative split criteria

Consider split of set S based on attribute A

$$\text{Impurity}(S | A) = \sum_{j=1}^{|\text{Values}(A)|} \text{Impurity}(S_j)$$

$$\text{Entropy Impurity}(Z) = \sum_{i=1}^{\text{Classes}} -\frac{|Z_i|}{|Z|} \log_2 \frac{|Z_i|}{|Z|}$$

$$\text{Gini Impurity}(Z) = \sum_{i \neq j} \left(\frac{|Z_i|}{|Z|} \right) \left(\frac{|Z_j|}{|Z|} \right) = 1 - \sum_{i=1}^{\text{Classes}} \left(\frac{|Z_i|}{|Z|} \right)^2$$

(Expected rate of error if class label is picked randomly according to distribution of instances in a set)

Alternative split criteria

One-sided split criteria – often useful for exploratory analysis of data

$$\text{Impurity}(S | A) = \text{Min}_{i \in \text{Values}(A)} \{ \text{Impurity}(S_i) \}$$

Incorporating Attribute costs

Not all attribute measurements are equally costly or risky

In Medical diagnosis

Blood-Test has cost \$150

Exploratory-Surgery may have a cost of \$3000

Goal: Learn a Decision Tree Classifier which minimizes cost of classification

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

Incorporating Different Misclassification Costs for different classes

Not all misclassifications are equally costly

An occasional false alarm about a nuclear power plant meltdown is less costly than the failure to alert when there is a chance of a meltdown

Weighted Gini Impurity

$$Impurity(S) = \sum_{ij} \lambda_{ij} \left(\frac{|S_i|}{|S|} \right) \left(\frac{|S_j|}{|S|} \right)$$

λ_{ij} is the cost of wrongly assigning an instance belonging to class i to class j

Dealing with Missing Attribute Values (Solution 1)

Sometimes, the fact that an attribute value is missing might itself be informative –

Missing blood sugar level might imply that the physician had reason not to measure it

Introduce a new value (one per attribute) to denote a missing value

Decision tree construction and use of tree for classification proceed as before

Dealing with Missing Attribute Values Solution 2

During decision tree construction

Replace a missing attribute value in a training example with the most frequent value found among the instances at the node

During use of tree for classification

Replace a missing attribute value in an instance to be classified with the most frequent value found among the training instances at the node

Dealing with Missing Attribute Values (Solution 3)

During decision tree construction

Replace a missing attribute value in a training example with the most frequent value found among the instances at the node that have the same class label as the training example

During use of tree for classification

Assign to a missing attribute the most frequent value found at the node (based on the training sample)

Sort the instance through the tree to generate the class label

Dealing with Missing Attribute Values (Solution 4)

During decision tree construction

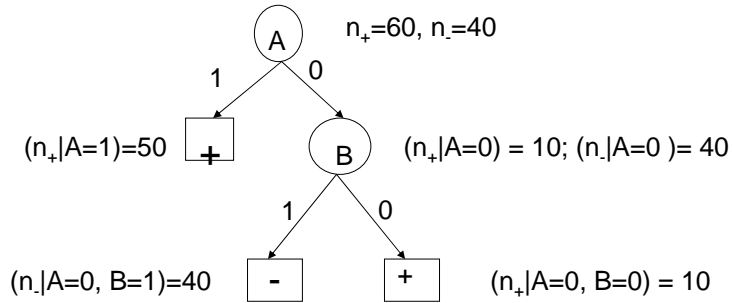
Generate several fractionally weighted training examples based on the distribution of values for the corresponding attribute at the node

During use of tree for classification

Generate multiple *instances* by assigning candidate values for the missing attribute based on the distribution of instances at the node

Sort each such instance through the tree to generate candidate labels and assign the most probable class label or probabilistically assign class label

Dealing with Missing Attribute Values

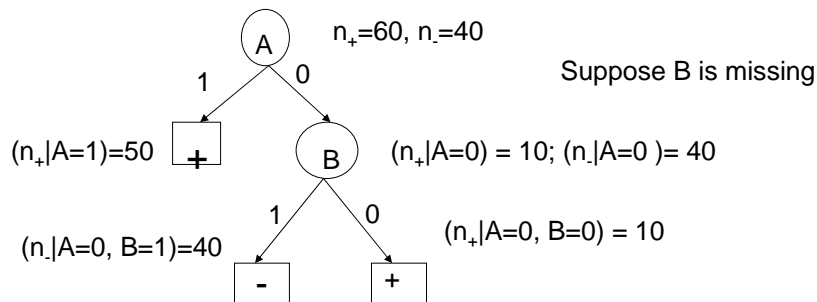


Suppose B is missing

Replacement with most frequent value at the node $\rightarrow B=1$

Replacement with most frequent value if class is + $\rightarrow B=0$

Dealing with Missing Attribute Values



Fractional instance based on distribution at the node .. $4/5$ for $B=1$, $1/5$ for $B=0$

Fractional instance based on distribution at the node for class + ..

$1/5$ for $B=0$, 0 for $B=1$

Recent Developments in Decision Tree Classifiers

- Learning Decision Trees from Distributed Data (Caragea, Silvescu and Honavar, 2004)
- Learning Decision Trees from Attribute Value Taxonomies and partially specified data (Zhang and Honavar, 2003; 2004; Zhang et al., 2005)
- Learning Decision Trees from Relational Data (Atramentov, Leiva, and Honavar, 2003)
- Decision Trees for Multi-label Classification Tasks
- Learning Decision Trees from Very Large Data Sets
- Learning Decision Trees from Data Streams

Summary of Decision Trees

Simple

Fast (Linear in size of the tree, linear in the size of the training set, linear in the number of attributes)

Produce easy to interpret rules

Good for generating simple predictive rules from data with lots of attributes