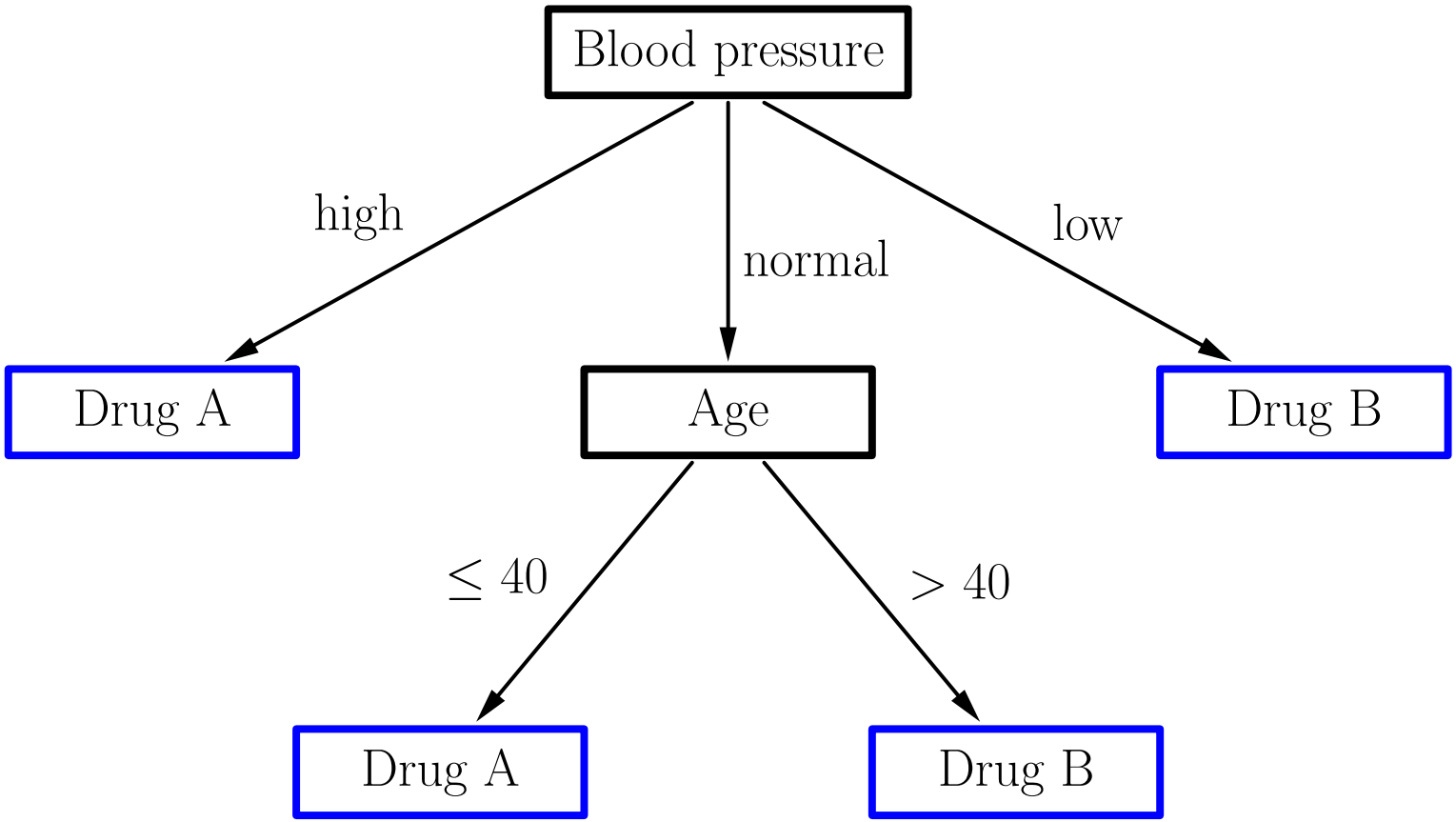
Decision and Regression Trees

Decision and Regression Trees

- **Classification with a Decision Tree**
- **Top-down Induction of Decision Trees**
  - A simple example
  - The general algorithm
  - Attribute selection measures
  - Treatment of numeric attributes and missing values
- **Pruning Decision Trees**
  - General approaches
  - A simple example
- **Regression Trees**
- **Summary**

A Very Simple Decision Tree

Assignment of a drug to a patient:



Classification with a Decision Tree

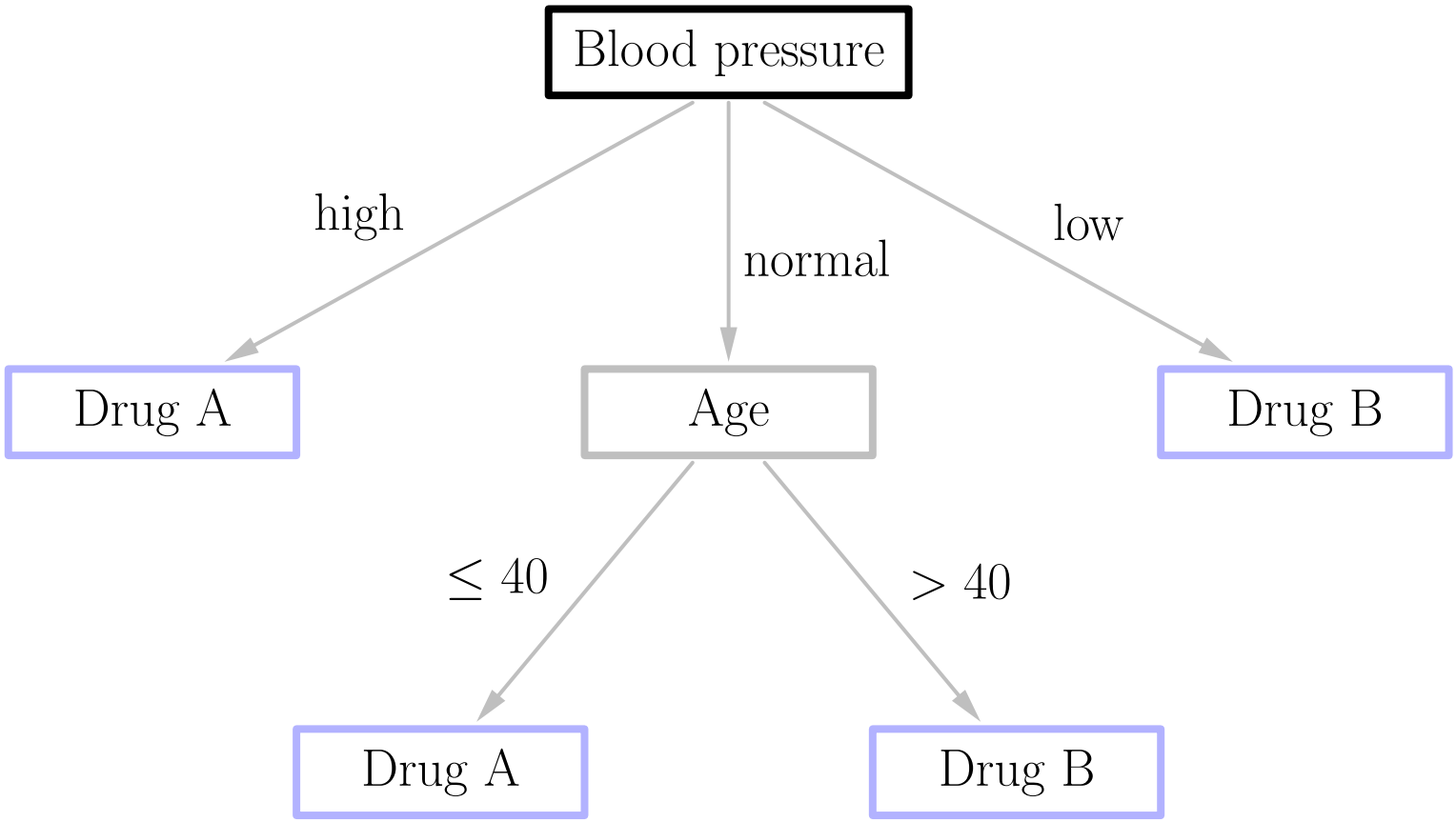
Recursive Descent:

- Start at the root node.
- If the current node is an **leaf node**:
  - Return the class assigned to the node.
- If the current node is an **inner node**:
  - Test the attribute associated with the node.
  - Follow the branch labeled with the outcome of the test.
  - Apply the algorithm recursively.

Intuitively: Follow the path corresponding to the case to be classified.

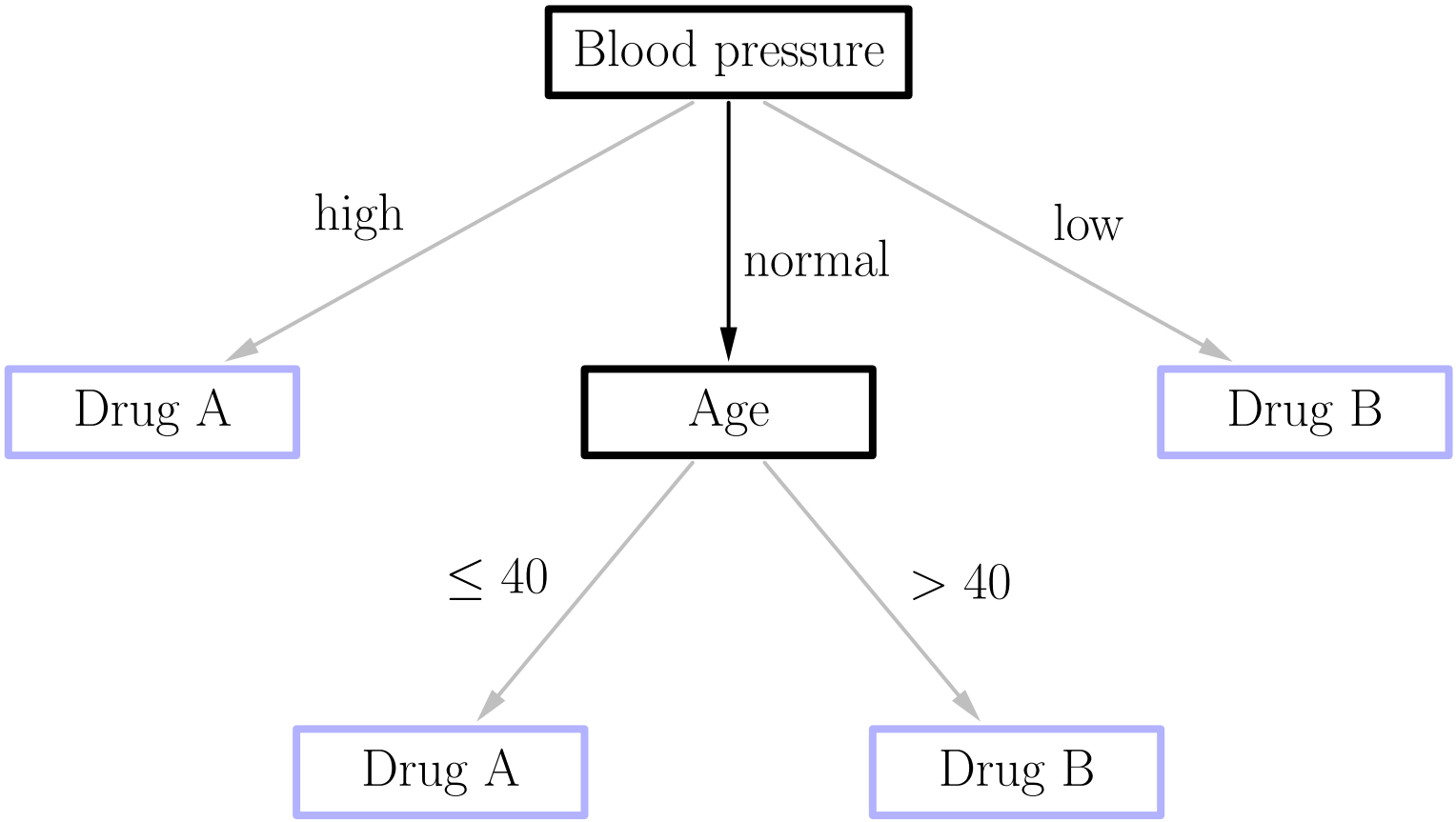
Classification in the Example

Assignment of a drug to a patient:



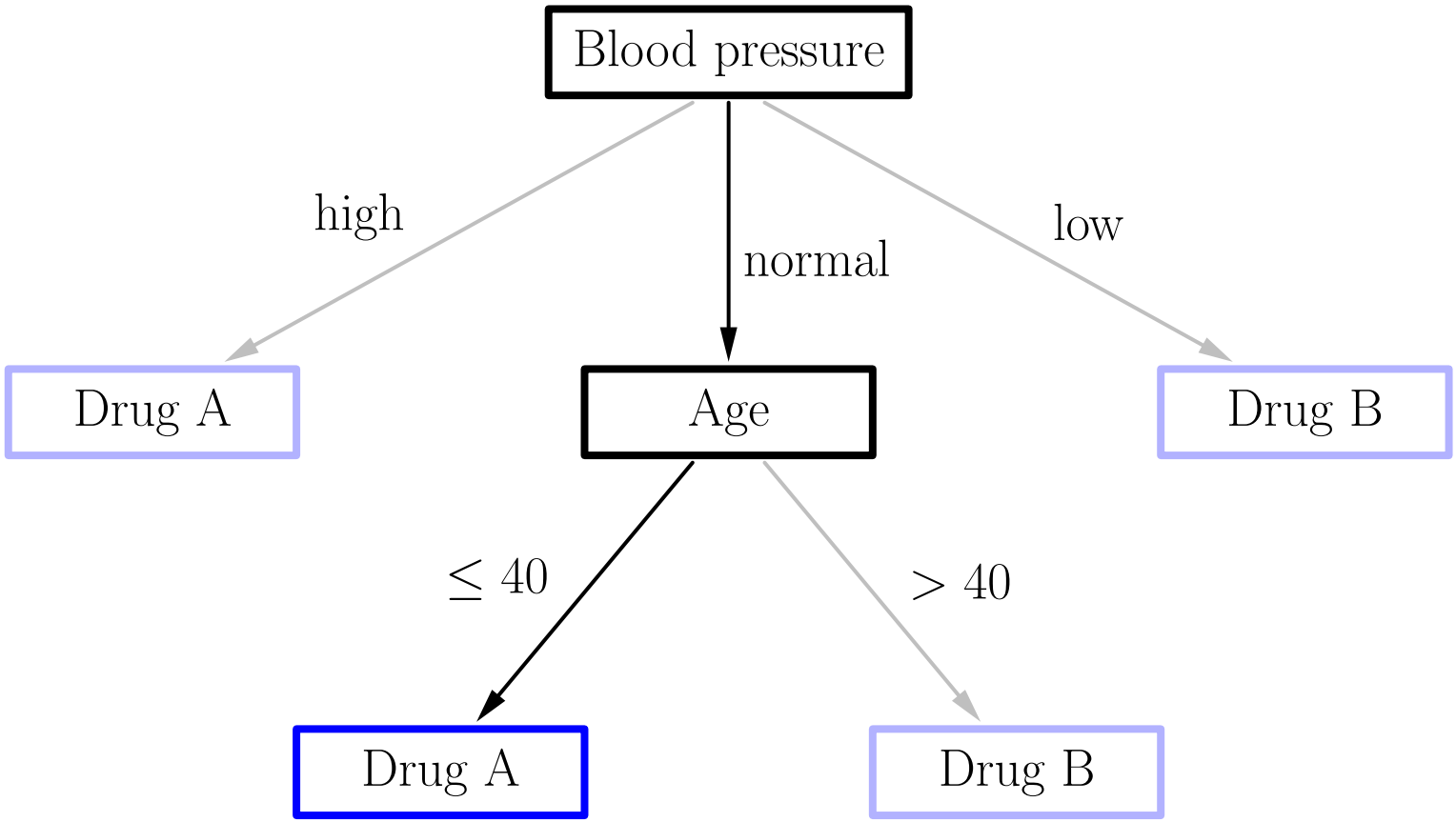
Classification in the Example

Assignment of a drug to a patient:



Classification in the Example

Assignment of a drug to a patient:



Induction of Decision Trees

- **Top-down approach**
  - Build the decision tree from top to bottom (from the root to the leaves).
- **Greedy Selection of a Test Attribute**
  - Compute an evaluation measure for all attributes.
  - Select the attribute with the best evaluation.
- **Divide and Conquer / Recursive Descent**
  - Divide the example cases according to the values of the test attribute.
  - Apply the procedure recursively to the subsets.
  - Terminate the recursion if
    - all cases belong to the same class
    - no more test attributes are available

Induction of a Decision Tree: Example

Patient database

- 12 example cases
- 3 descriptive attributes
- 1 class attribute

Assignment of drug

(without patient attributes)

always drug A or always drug B:

50% correct (in 6 of 12 cases)

No	Sex	Age	Blood pr.	Drug
1	male	20	normal	A
2	female	73	normal	B
3	female	37	high	A
4	male	33	low	B
5	female	48	high	A
6	male	29	normal	A
7	female	52	normal	B
8	male	42	low	B
9	male	61	normal	B
10	female	30	normal	A
11	female	26	low	B
12	male	54	high	A

Induction of a Decision Tree: Example

Sex of the patient

- Division w.r.t. male/female.

Assignment of drug

male:	50% correct	(in 3 of 6 cases)
female:	50% correct	(in 3 of 6 cases)
total:	**50% correct**	(in 6 of 12 cases)

No	Sex	Drug
1	male	A
6	male	A
12	male	A
4	male	B
8	male	B
9	male	B
3	female	A
5	female	A
10	female	A
2	female	B
7	female	B
11	female	B

Induction of a Decision Tree: Example

Age of the patient

- Sort according to age.
- Find best age split.
  here: ca. 40 years

Assignment of drug

≤ 40:	A	67% correct	(in 4 of 6 cases)
> 40:	B	67% correct	(in 4 of 6 cases)
total:		**67% correct**	(in 8 of 12 cases)

No	Age	Drug
1	20	A
11	26	B
6	29	A
10	30	A
4	33	B
3	37	A
8	42	B
5	48	A
7	52	B
12	54	A
9	61	B
2	73	B

Induction of a Decision Tree: Example

Blood pressure of the patient

- Division w.r.t. high/normal/low.

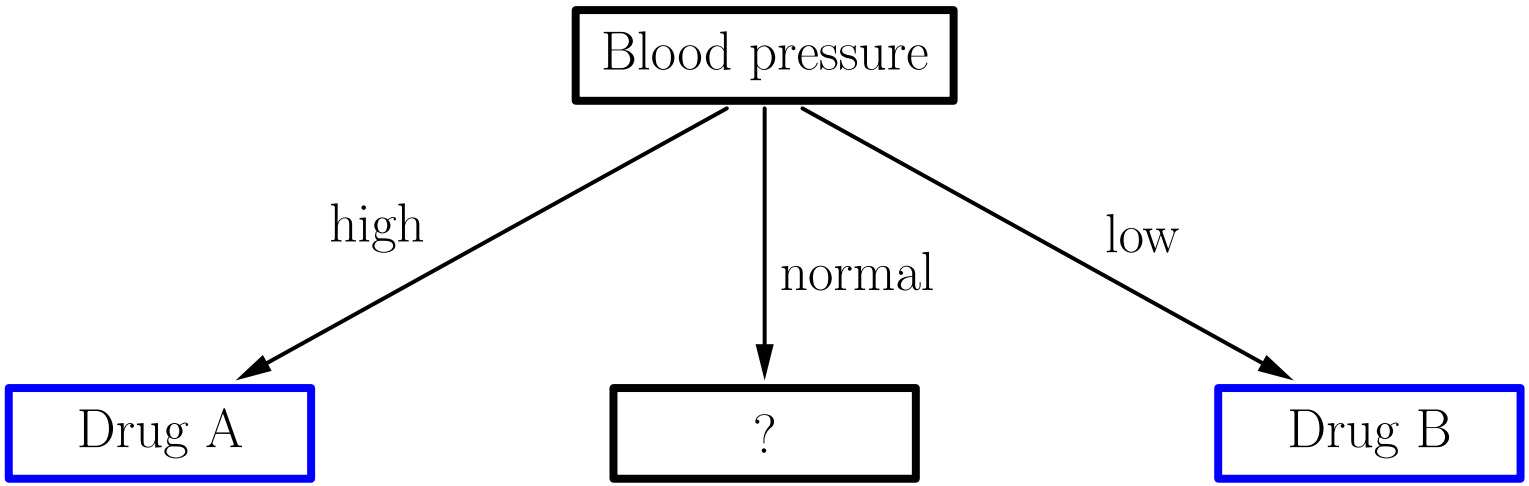
No	Blood pr.	Drug
3	high	A
5	high	A
12	high	A
1	normal	A
6	normal	A
10	normal	A
2	normal	B
7	normal	B
9	normal	B
4	low	B
8	low	B
11	low	B

Assignment of drug

high:	A	100% correct	(in 3 of 3 cases)
normal:		50% correct	(in 3 of 6 cases)
low:	B	100% correct	(in 3 of 3 cases)
total:		**75% correct**	(in 9 of 12 cases)

Induction of a Decision Tree: Example

Current Decision Tree:



Induction of a Decision Tree: Example

Blood pressure and sex

- Only patients
  with normal blood pressure.
- Division w.r.t. male/female.

Assignment of drug

male:	A	67% correct	(2 of 3)
female:	B	67% correct	(2 of 3)
total:		**67% correct**	(4 of 6)

No	Blood pr.	Sex	Drug
3	high		A
5	high		A
12	high		A
1	normal	male	A
6	normal	male	A
9	normal	male	B
2	normal	female	B
7	normal	female	B
10	normal	female	A
4	low		B
8	low		B
11	low		B

Induction of a Decision Tree: Example

Blood pressure and age

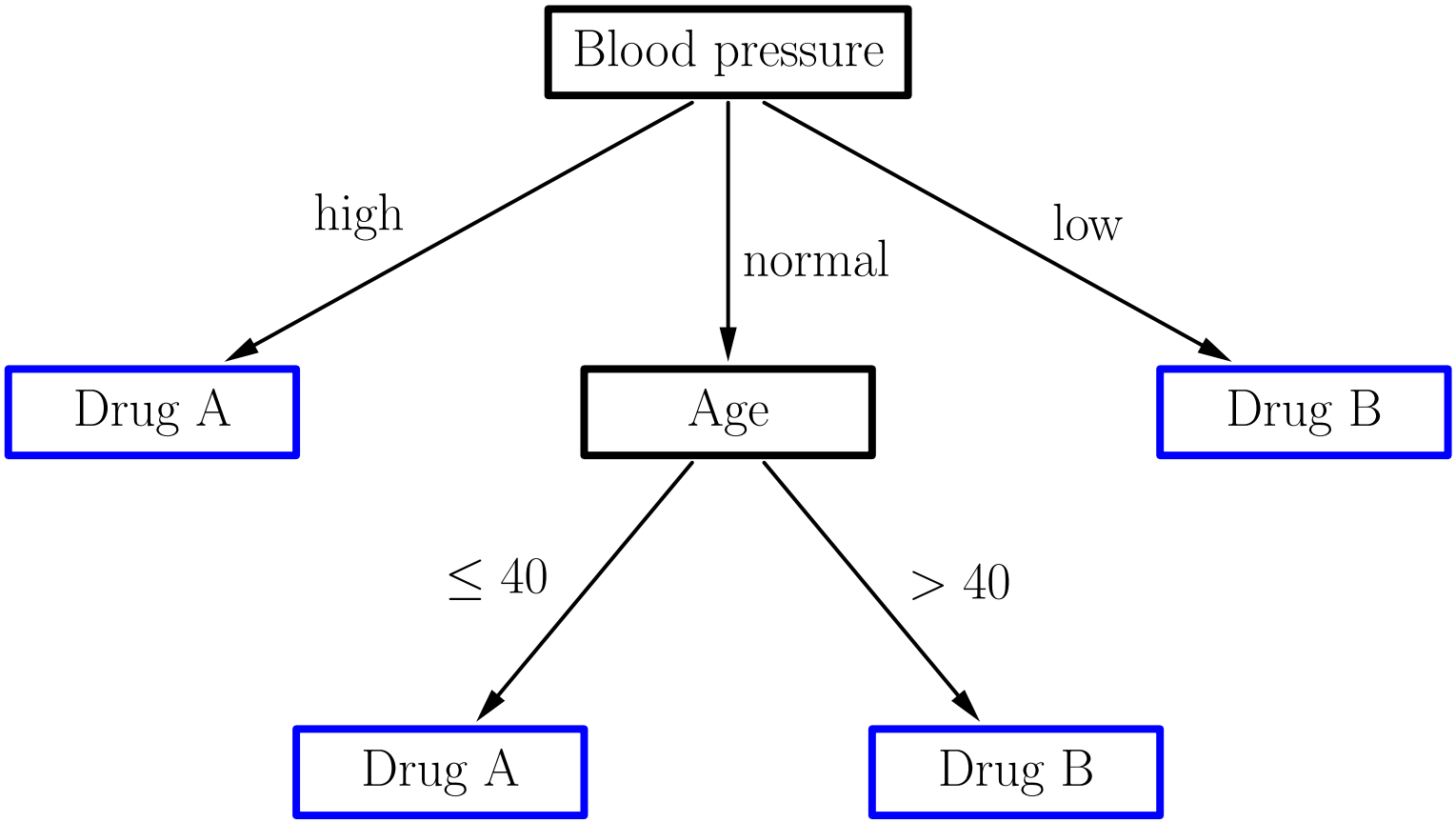
- Only patients
  with normal blood pressure.
- Sort according to age.
- Find best age split.
  here: ca. 40 years

Assignment of drug

≤ 40:	A	100% correct	(3 of 3)
> 40:	B	100% correct	(3 of 3)
total:		**100% correct**	(6 of 6)

No	Blood pr.	Age	Drug
3	high		A
5	high		A
12	high		A
1	normal	20	A
6	normal	29	A
10	normal	30	A
7	normal	52	B
9	normal	61	B
2	normal	73	B
11	low		B
4	low		B
8	low		B

Assignment of a drug to a patient:



Decision Tree Induction: Notation

S	a set of case or object descriptions
C	the class attribute
$A^{(1)}, \ldots, A^{(m)}$	other attributes (index dropped in the following)
$\mathrm{dom}(C)$	$= \{c_1, \ldots, c_{n_C}\}$, n_C: number of classes
$\mathrm{dom}(A)$	$= \{a_1, \ldots, a_{n_A}\}$, n_A: number of attribute values
$N_{\cdot\cdot}$	total number of case or object descriptions i.e. $N_{\cdot\cdot} = \lvert S \rvert$
$N_{i\cdot}$	absolute frequency of the class c_i
$N_{\cdot j}$	absolute frequency of the attribute value a_j
N_{ij}	absolute frequency of the combination of the class c_i and the attribute value a_j. It is $N_{i\cdot} = \sum_{j=1}^{n_A} N_{ij}$ and $N_{\cdot j} = \sum_{i=1}^{n_C} N_{ij}$.
$p_{i\cdot}$	relative frequency of the class c_i, $p_{i\cdot} = \frac{N_{i\cdot}}{N_{\cdot\cdot}}$
$p_{\cdot j}$	relative frequency of the attribute value a_j, $p_{\cdot j} = \frac{N_{\cdot j}}{N_{\cdot\cdot}}$
p_{ij}	relative frequency of the combination of class c_i and attribute value a_j, $p_{ij} = \frac{N_{ij}}{N_{\cdot\cdot}}$
$p_{i\lvert j}$	relative frequency of the class c_i in cases having attribute value a_j, $p_{i\lvert j} = \frac{N_{ij}}{N_{\cdot j}} = \frac{p_{ij}}{p_{\cdot j}}$

Decision Tree Induction: General Algorithm

function grow_tree (S : set of cases) : node;
begin
    $best\_v$:= WORTHLESS;
    for all untested attributes A **do**
        compute frequencies N_{ij}, $N_{i.}$, $N_{.j}$ for $1 \leq i \leq n_C$ and $1 \leq j \leq n_A$;
        compute value v of an evaluation measure using N_{ij}, $N_{i.}$, $N_{.j}$;
        if $v > best\_v$ **then** $best\_v := v$; $best\_A := A$; **end**;
    end
    if $best\_v$ = WORTHLESS
    then create leaf node x;
        assign majority class of S to x;
    else create test node x;
        assign test on attribute $best\_A$ to x;
        for all $a \in \mathrm{dom}(best\_A)$ **do** x.child[a] := grow_tree($S|_{best\_A=a}$); **end**;
    end;
    return x;
end;

Evaluation Measures

- Evaluation measure used in the above example:
  rate of correctly classified example cases.
  - Advantage: simple to compute, easy to understand.
  - Disadvantage: works well only for two classes.
- If there are more than two classes, the rate of misclassified example cases **neglects a lot of the available information**.
  - Only the majority class—that is, the class occurring most often in (a subset of) the example cases—is really considered.
  - The distribution of the other classes has no influence. However, a good choice here can be important for deeper levels of the decision tree.
- **Therefore:** Study also other evaluation measures. Here:
  - **Information gain** and its various normalizations.
  - χ^2 **measure** (well-known in statistics).

An Information-theoretic Evaluation Measure

Information Gain (Kullback and Leibler 1951, Quinlan 1986)

Based on Shannon Entropy $H = -\sum_{i=1}^{n} p_i \log_2 p_i$ (Shannon 1948)

$$I_{\text{gain}}(C, A) = \overbrace{H(C)}^{} - \overbrace{H(C|A)}^{}$$

$$= \overbrace{-\sum_{i=1}^{n_C} p_{i.} \log_2 p_{i.}}^{} - \overbrace{\sum_{j=1}^{n_A} p_{.j} \left(-\sum_{i=1}^{n_C} p_{i|j} \log_2 p_{i|j} \right)}^{}$$

$H(C)$	Entropy of the class distribution (C: class attribute)
$H(C\|A)$	*Expected entropy* of the class distribution if the value of the attribute A becomes known
$H(C) - H(C\|A)$	Expected entropy reduction or *information gain*

Inducing the Decision Tree with Information Gain

- Information gain for drug and sex:

$$H(\text{Drug}) = -\left(\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{2}\log_2\frac{1}{2}\right) = 1$$

$$H(\text{Drug} \mid \text{Sex}) = \frac{1}{2}\underbrace{\left(-\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2}\right)}_{H(\text{Drug}|\text{Sex=male})} + \frac{1}{2}\underbrace{\left(-\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2}\right)}_{H(\text{Drug}|\text{Sex=female})} = 1$$

$$I_{\text{gain}}(\text{Drug}, \text{Sex}) = 1 - 1 = 0$$

- No gain at all since the initial the uniform distribution of drug is splitted into two (still) uniform distributions.

Inducing the Decision Tree with Information Gain

- Information gain for drug and age:

$$H(\text{Drug}) = -\left(\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{2}\log_2\frac{1}{2}\right) = 1$$

$$H(\text{Drug} \mid \text{Age}) = \frac{1}{2}\underbrace{\left(-\frac{2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3}\right)}_{H(\text{Drug}|\text{Age}\leq 40)} + \frac{1}{2}\underbrace{\left(-\frac{1}{3}\log_2\frac{1}{3} - \frac{2}{3}\log_2\frac{2}{3}\right)}_{H(\text{Drug}|\text{Age}>40)} \approx 0.9183$$

$$I_{\text{gain}}(\text{Drug}, \text{Age}) = 1 - 0.9183 = 0.0817$$

- Splitting w. r. t. age can reduce the overall entropy.

Inducing the Decision Tree with Information Gain

- Information gain for drug and blood pressure:

$$H(\text{Drug}) = -\left(\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{2}\log_2\frac{1}{2}\right) = 1$$

$$H(\text{Drug} \mid \text{Blood\_pr}) = \frac{1}{4}\cdot 0 + \frac{1}{2}\Bigg(\underbrace{-\frac{2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3}}_{H(\text{Drug}\mid\text{Blood\_pr=normal})} \Bigg) + \frac{1}{4}\cdot 0 = 0.5$$

$$I_{\text{gain}}(\text{Drug}, \text{Blood\_pr}) = 1 - 0.5 = 0.5$$

- Largest information gain, so we first split w. r. t. blood pressure (as in the example with misclassification rate).

Inducing the Decision Tree with Information Gain

- Next level: Subtree blood pressure is normal.
- Information gain for drug and sex:

$$H(\text{Drug}) = -\left(\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{2}\log_2\frac{1}{2}\right) = 1$$

$$H(\text{Drug} \mid \text{Sex}) = \frac{1}{2}\underbrace{\left(-\frac{2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3}\right)}_{H(\text{Drug}|\text{Sex=male})} + \frac{1}{2}\underbrace{\left(-\frac{1}{3}\log_2\frac{1}{3} - \frac{2}{3}\log_2\frac{2}{3}\right)}_{H(\text{Drug}|\text{Sex=female})} = 0.9183$$

$$I_{\text{gain}}(\text{Drug}, \text{Sex}) = 0.0817$$

- Entropy can be decreased.

Inducing the Decision Tree with Information Gain

- Next level: Subtree blood pressure is normal.
- Information gain for drug and age:

$$H(\text{Drug}) = -\left(\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{2}\log_2\frac{1}{2}\right) = 1$$

$$H(\text{Drug} \mid \text{Age}) = \frac{1}{2}\cdot 0 + \frac{1}{2}\cdot 0 = 0$$

$$I_{\text{gain}}(\text{Drug}, \text{Age}) = 1$$

- Maximal information gain, that is we result in a perfect classification (again, as in the case of using misclassification rate).

Interpretation of Shannon Entropy

- Let $S = \{s_1, \ldots, s_n\}$ be a finite set of alternatives having positive probabilities $P(s_i)$, $i = 1, \ldots, n$, satisfying $\sum_{i=1}^{n} P(s_i) = 1$.
- **Shannon Entropy:**

$$H(S) = -\sum_{i=1}^{n} P(s_i) \log_2 P(s_i)$$

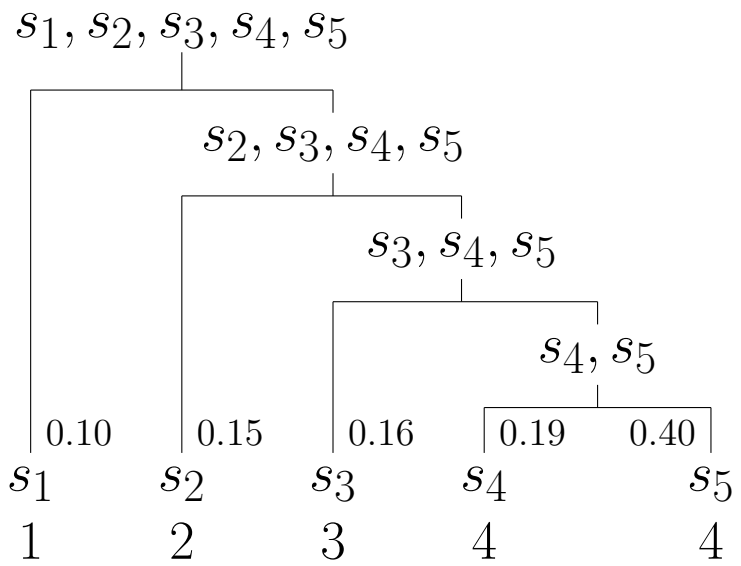
- Intuitively: **Expected number of yes/no questions that have to be asked in order to determine the obtaining alternative.**
  - Suppose there is an oracle, which knows the obtaining alternative, but responds only if the question can be answered with "yes" or "no".
  - A better question scheme than asking for one alternative after the other can easily be found: Divide the set into two subsets of about equal size.
  - Ask for containment in an arbitrarily chosen subset.
  - Apply this scheme recursively $\rightarrow$ number of questions bounded by $\lceil \log_2 n \rceil$.

Question/Coding Schemes

$$P(s_1) = 0.10, \quad P(s_2) = 0.15, \quad P(s_3) = 0.16, \quad P(s_4) = 0.19, \quad P(s_5) = 0.40$$

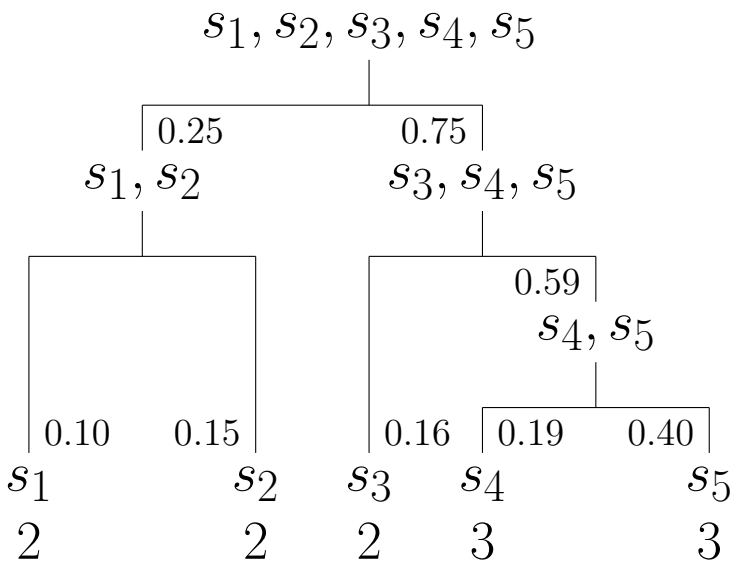
Shannon entropy: $-\sum_i P(s_i) \log_2 P(s_i) = 2.15$ bit/symbol

Linear Traversal



Code length: 3.24 bit/symbol
Code efficiency: 0.664

Equal Size Subsets



Code length: 2.59 bit/symbol
Code efficiency: 0.830

Question/Coding Schemes

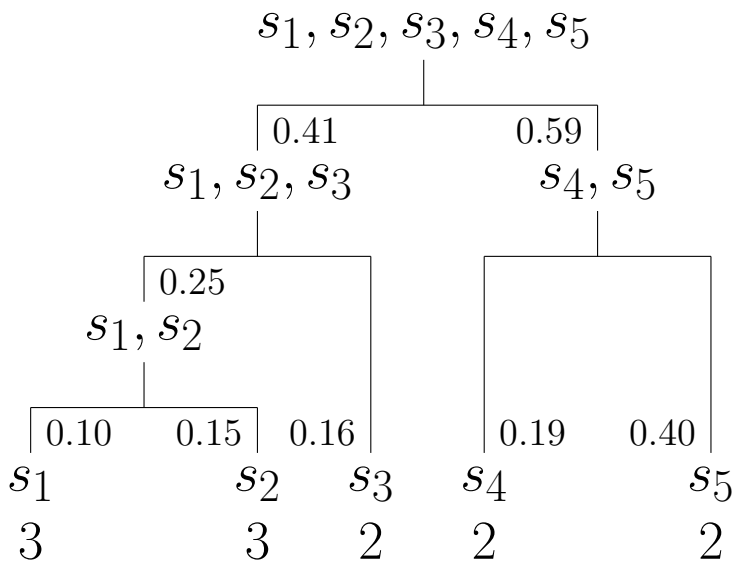
- Splitting into subsets of about equal size can lead to a bad arrangement of the alternatives into subsets → high expected number of questions.
- Good question schemes take the probability of the alternatives into account.
- **Shannon-Fano Coding** (1948)
  - Build the question/coding scheme top-down.
  - Sort the alternatives w.r.t. their probabilities.
  - Split the set so that the subsets have about equal *probability* (splits must respect the probability order of the alternatives).
- **Huffman Coding** (1952)
  - Build the question/coding scheme bottom-up.
  - Start with one element sets.
  - Always combine those two sets that have the smallest probabilities.

Question/Coding Schemes

$$P(s_1) = 0.10, \quad P(s_2) = 0.15, \quad P(s_3) = 0.16, \quad P(s_4) = 0.19, \quad P(s_5) = 0.40$$

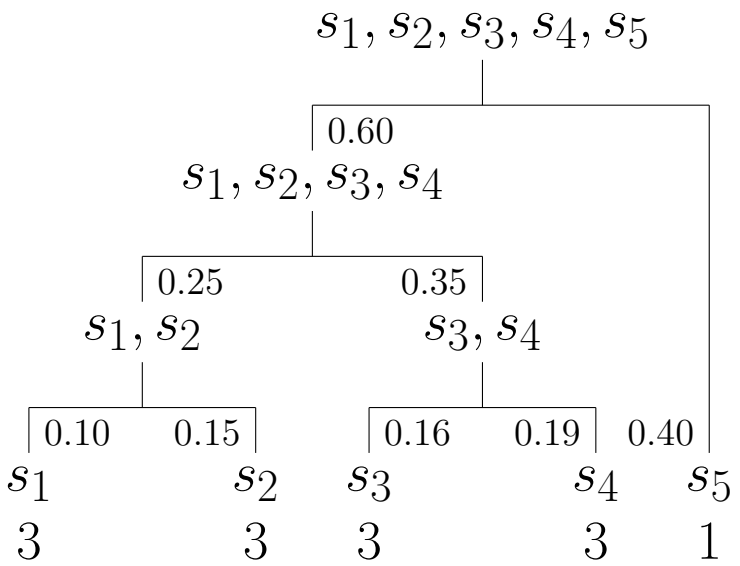
Shannon entropy: $-\sum_i P(s_i) \log_2 P(s_i) = 2.15$ bit/symbol

Shannon–Fano Coding (1948)



Code length: 2.25 bit/symbol
Code efficiency: 0.955

Huffman Coding (1952)



Code length: 2.20 bit/symbol
Code efficiency: 0.977

Question/Coding Schemes

- It can be shown that Huffman coding is optimal if we have to determine the obtaining alternative in a single instance.
  (No question/coding scheme has a smaller expected number of questions.)
- Only if the obtaining alternative has to be determined in a sequence of (independent) situations, this scheme can be improved upon.
- Idea: Process the sequence not instance by instance, but combine two, three or more consecutive instances and ask directly for the obtaining combination of alternatives.
- Although this enlarges the question/coding scheme, the expected number of questions per identification is reduced (because each interrogation identifies the obtaining alternative for several situations).
- However, the expected number of questions per identification cannot be made arbitrarily small. Shannon showed that there is a lower bound, namely the Shannon entropy.

Interpretation of Shannon Entropy

$$P(s_1) = \tfrac{1}{2}, \quad P(s_2) = \tfrac{1}{4}, \quad P(s_3) = \tfrac{1}{8}, \quad P(s_4) = \tfrac{1}{16}, \quad P(s_5) = \tfrac{1}{16}$$

Shannon entropy: $-\sum_i P(s_i) \log_2 P(s_i) = 1.875$ bit/symbol

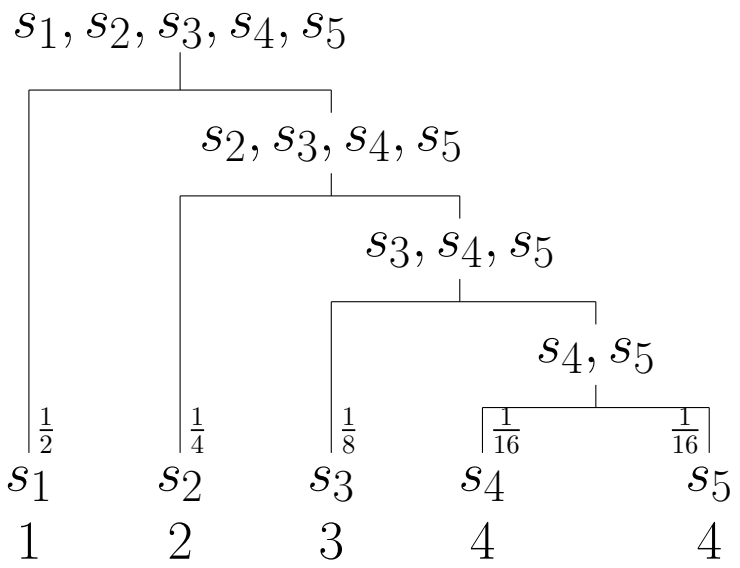
If the probability distribution allows for a perfect Huffman code (code efficiency 1), the Shannon entropy can easily be interpreted as follows:

$$-\sum_i P(s_i) \log_2 P(s_i)$$

$$= \sum_i \underbrace{P(s_i)}_{\substack{\text{occurrence} \\ \text{probability}}} \cdot \underbrace{\log_2 \frac{1}{P(s_i)}}_{\substack{\text{path length} \\ \text{in tree}}} .$$

In other words, it is the expected number of needed yes/no questions.

Perfect Question Scheme



Code length: 1.875 bit/symbol
Code efficiency: 1

Normalized Information Gain

- Information gain is biased towards many-valued attributes.
- Normalization removes / reduces this bias.

Information Gain Ratio (Quinlan 1986 / 1993)

$$I_{\text{gr}}(C, A) = \frac{I_{\text{gain}}(C, A)}{H_A} = \frac{I_{\text{gain}}(C, A)}{-\sum_{j=1}^{n_A} p_{\cdot j} \log_2 p_{\cdot j}}$$

Symmetric Information Gain Ratio (López de Mántaras 1991)

$$I^{(1)}_{\text{sgr}}(C, A) = \frac{I_{\text{gain}}(C, A)}{H_{AC}} \qquad \text{or} \qquad I^{(2)}_{\text{sgr}}(C, A) = \frac{I_{\text{gain}}(C, A)}{H_A + H_C}$$

Bias of Information Gain

- **Information gain is biased towards many-valued attributes**, i.e., of two attributes having about the same information content it tends to select the one having more values.
- The reasons are quantization effects caused by the finite number of example cases (due to which only a finite number of different probabilities can result in estimations) in connection with the following theorem:
- **Theorem:** Let A, B, and C be three attributes with finite domains and let their joint probability distribution be strictly positive, i.e., $\forall a \in \mathrm{dom}(A) : \forall b \in \mathrm{dom}(B) : \forall c \in \mathrm{dom}(C) : P(A = a, B = b, C = c) > 0$. Then

$$I_{\mathrm{gain}}(C, AB) \;\geq\; I_{\mathrm{gain}}(C, B),$$

with equality obtaining only if the attributes C and A are conditionally independent given B, i.e., if $P(C = c \mid A = a, B = b) = P(C = c \mid B = b)$.

(A detailed proof of this theorem can be found, for example, in [Borgelt and Kruse 2002], p. 311ff.)

χ^2 Measure

- Compares the actual joint distribution with a **hypothetical independent distribution**.
- Uses absolute comparison.
- Can be interpreted as a difference measure.

$$\chi^2(C, A) = \sum_{i=1}^{n_C} \sum_{j=1}^{n_A} N_{\cdot\cdot} \frac{(p_{i\cdot} p_{\cdot j} - p_{ij})^2}{p_{i\cdot} p_{\cdot j}}$$

- Side remark: Information gain can also be interpreted as a difference measure.

$$I_{\text{gain}}(C, A) = \sum_{i=1}^{n_C} \sum_{j=1}^{n_A} p_{ij} \log_2 \frac{p_{ij}}{p_{i\cdot} p_{\cdot j}}$$

Treatment of Numeric Attributes

General Approach: Discretization

- **Preprocessing I**
  - Form equally sized or equally populated intervals.
- **During the tree construction**
  - Sort the example cases according to the attribute's values.
  - Construct a binary symbolic attribute for every possible split (values: "$\leq$ threshold" and "$>$ threshold").
  - Compute the evaluation measure for these binary attributes.
  - Possible improvements: Add a penalty depending on the number of splits.
- **Preprocessing II / Multisplits during tree construction**
  - Build a decision tree using only the numeric attribute.
  - Flatten the tree to obtain a multi-interval discretization.

Treatment of Missing Values

Induction

- Weight the evaluation measure with the fraction of cases with known values.
  - Idea: The attribute provides information only if it is known.
- Try to find a surrogate test attribute with similar properties
  (CART, Breiman *et al.* 1984)
- Assign the case to all branches, weighted in each branch with the relative frequency of the corresponding attribute value (C4.5, Quinlan 1993).

Classification

- Use the surrogate test attribute found during induction.
- Follow all branches of the test attribute, weighted with their relative number of cases, aggregate the class distributions of all leaves reached, and assign the majority class of the aggregated class distribution.

Pruning Decision Trees

Pruning serves the purpose

- to simplify the tree (improve interpretability),
- to avoid overfitting (improve generalization).

Basic ideas:

- Replace "bad" branches (subtrees) by leaves.
- Replace a subtree by its largest branch if it is better.

Common approaches:

- Reduced error pruning
- Pessimistic pruning
- Confidence level pruning
- Minimum description length pruning

Reduced Error Pruning

- Classify a set of new example cases with the decision tree.
  (These cases must not have been used for the induction!)
- Determine the number of errors for all leaves.
- The number of errors of a subtree is the sum of the errors of all of its leaves.
- Determine the number of errors for leaves that replace subtrees.
- If such a leaf leads to the same or fewer errors than the subtree,
  replace the subtree by the leaf.
- If a subtree has been replaced,
  recompute the number of errors of the subtrees it is part of.

Advantage: Very good pruning, effective avoidance of overfitting.

Disadvantage: Additional example cases needed.

Pessimistic Pruning

- Classify a set of example cases with the decision tree.
  (These cases may or may not have been used for the induction.)
- Determine the number of errors for all leaves and
  increase this number by a fixed, user-specified amount r.
- The number of errors of a subtree is the sum of the errors of all of its leaves.
- Determine the number of errors for leaves that replace subtrees
  (also increased by r).
- If such a leaf leads to the same or fewer errors than the subtree,
  replace the subtree by the leaf and recompute subtree errors.

Advantage: No additional example cases needed.

Disadvantage: Number of cases in a leaf has no influence.

Confidence Level Pruning

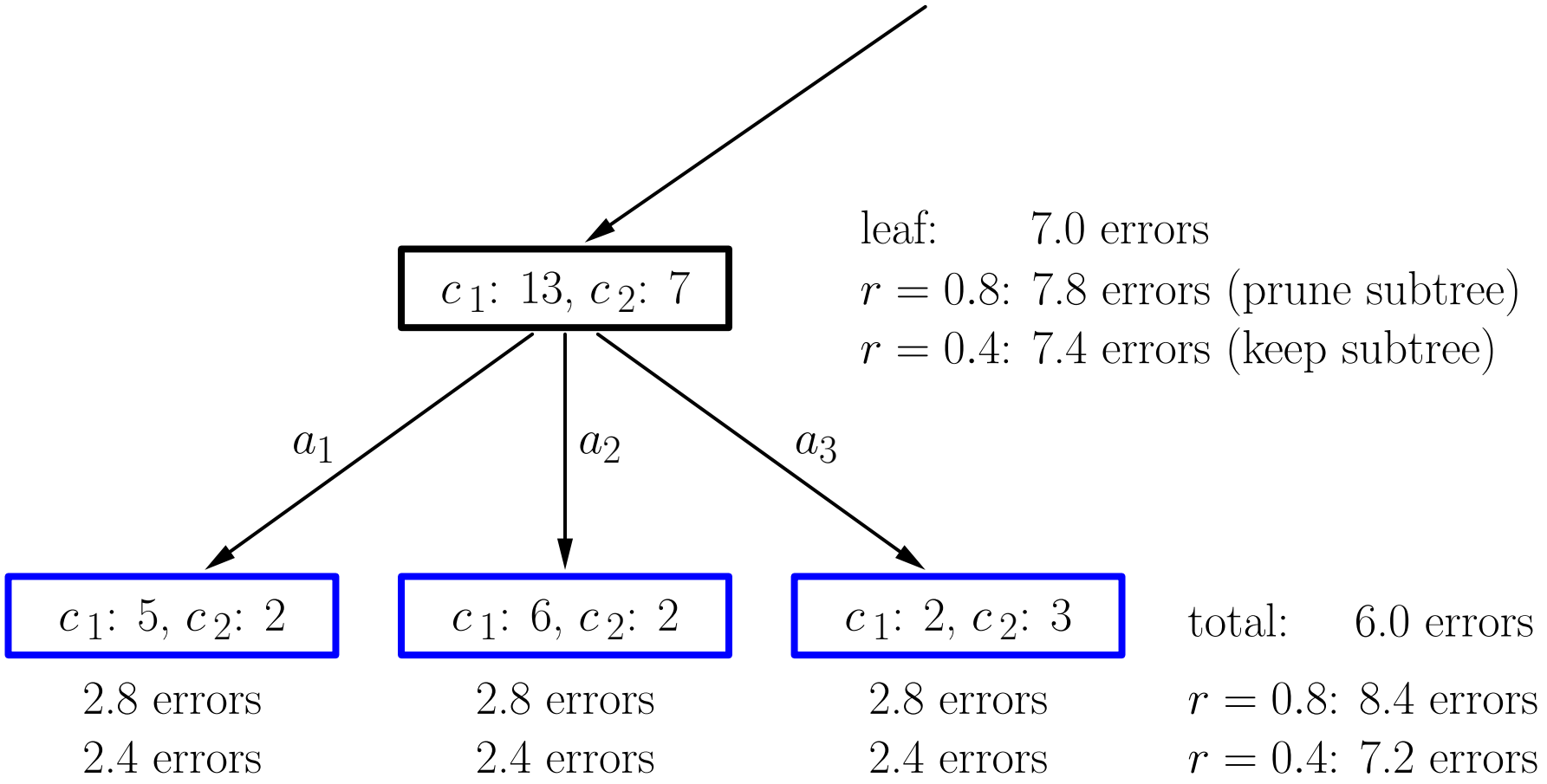
- Like pessimistic pruning, but the number of errors is computed as follows:
  - See classification in a leaf as a Bernoulli experiment (error / no error).
  - Estimate an interval for the error probability based on a user-specified confidence level α.
    (use approximation of the binomial distribution by a normal distribution)
  - Increase error number to the upper level of the confidence interval
    times the number of cases assigned to the leaf.
  - Formal problem: Classification is not a random experiment.

Advantage: No additional example cases needed, good pruning.

Disadvantage: Statistically dubious foundation.

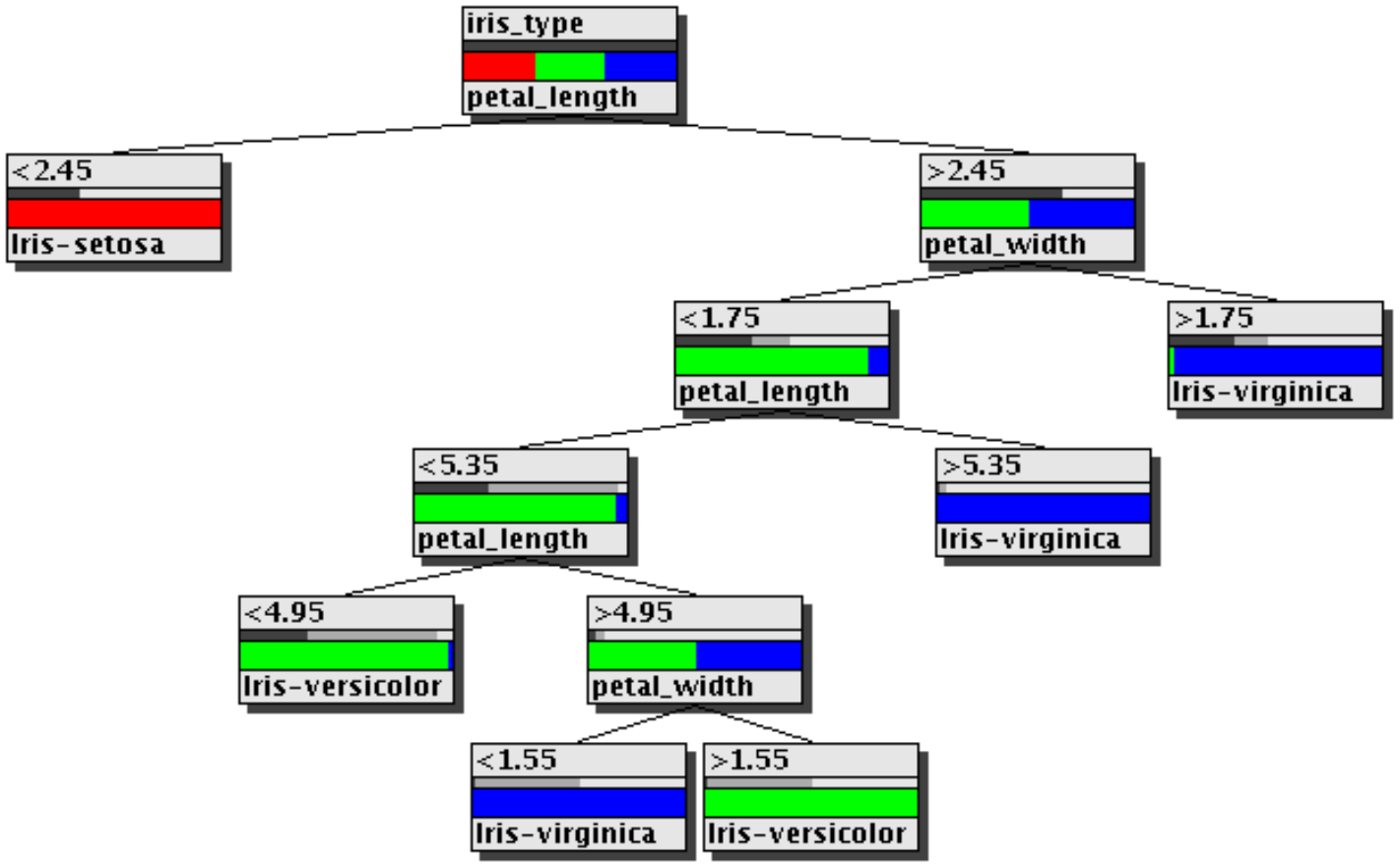
Pruning a Decision Tree: A Simple Example

Pessimistic Pruning with $r = 0.8$ and $r = 0.4$:



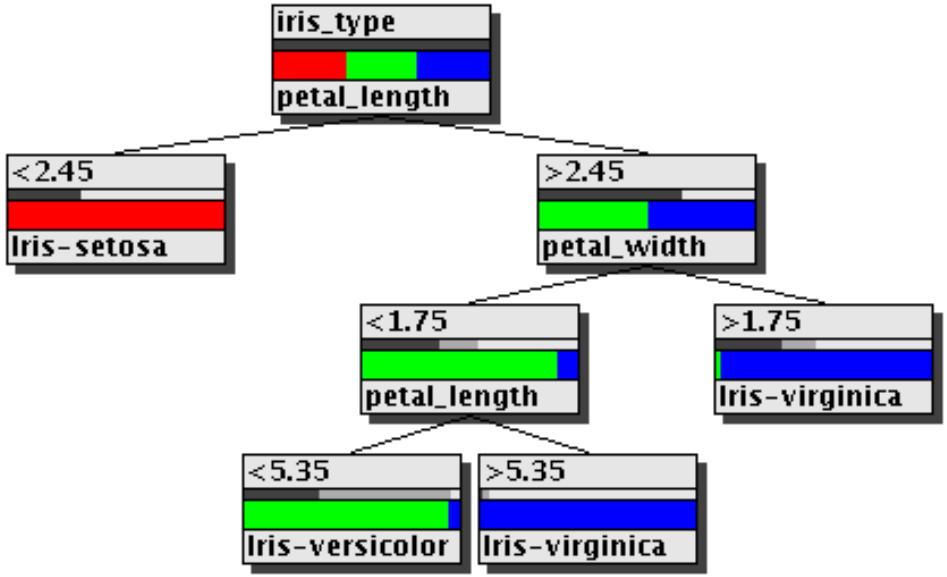
Decision Trees: An Example

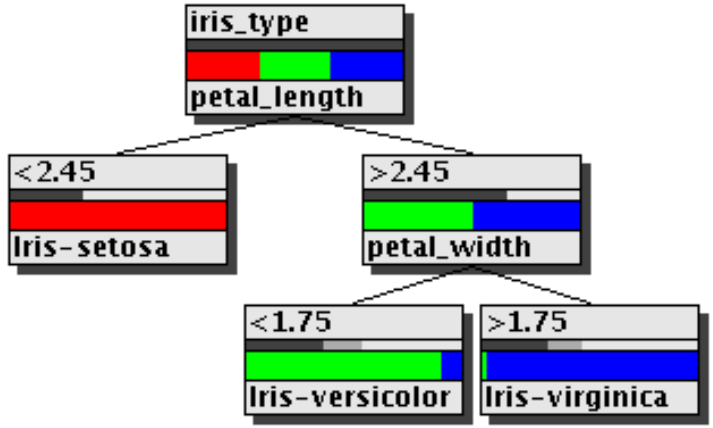
A decision tree for the Iris data
(induced with information gain ratio, unpruned)



Decision Trees: An Example

A decision tree for the Iris data
(pruned with confidence level pruning, $\alpha = 0.8$, and pessimistic pruning, $r = 2$)

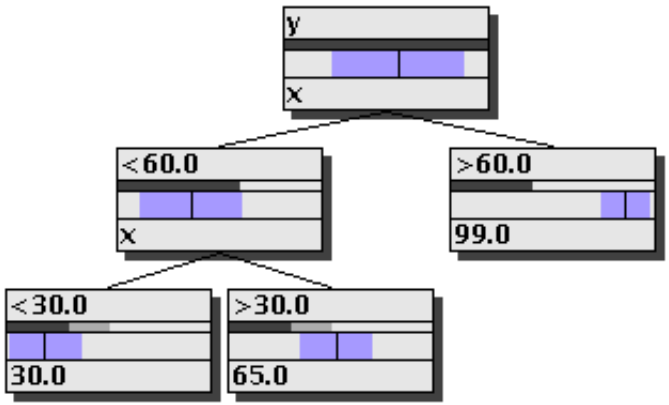




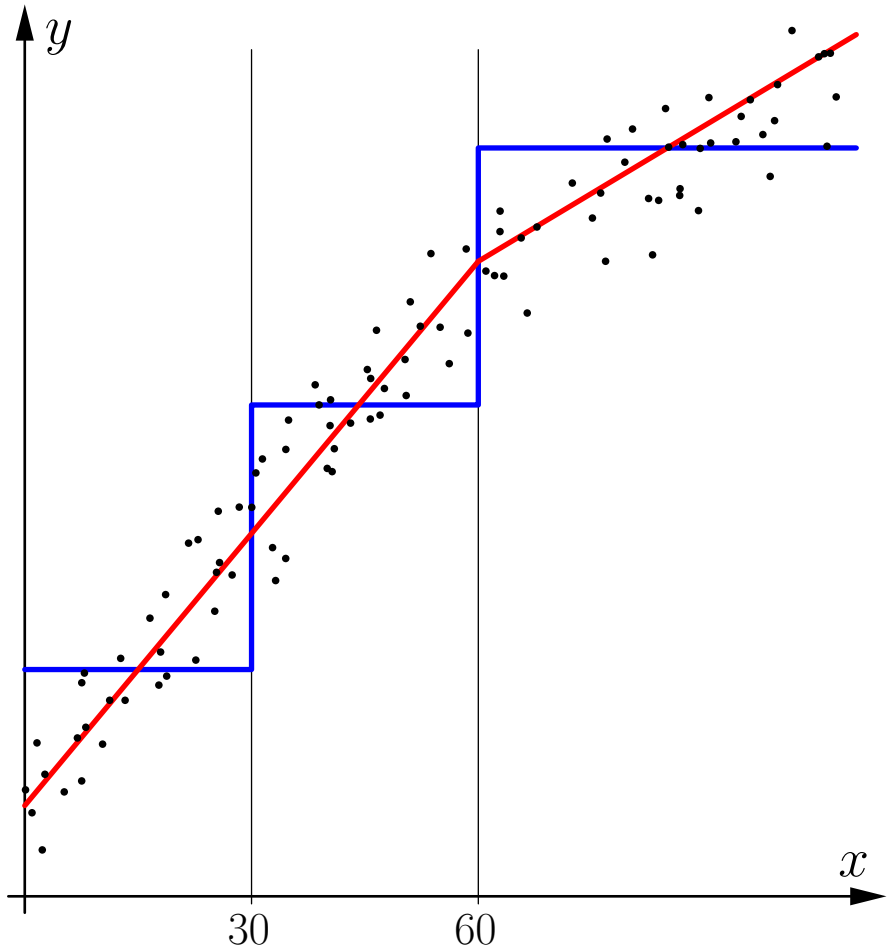
- Left: 7 instead of 11 nodes, 4 instead of 2 misclassifications.
- Right: 5 instead of 11 nodes, 6 instead of 2 misclassifications.
- The right tree is "minimal" for the three classes.

Regression Trees

- Target variable is not a class, but a numeric quantity.
- Simple regression trees: predict constant values in leaves. (blue lines)

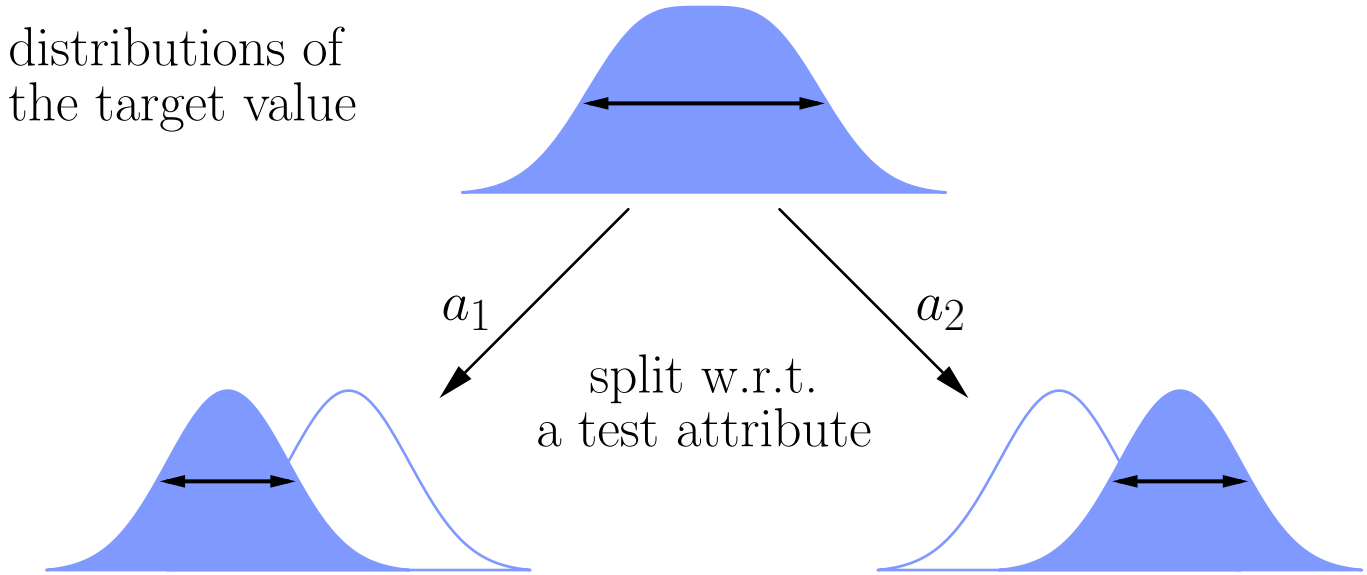


- More complex regression trees: predict linear functions in leaves. (red line)



x: input variable, y: target variable

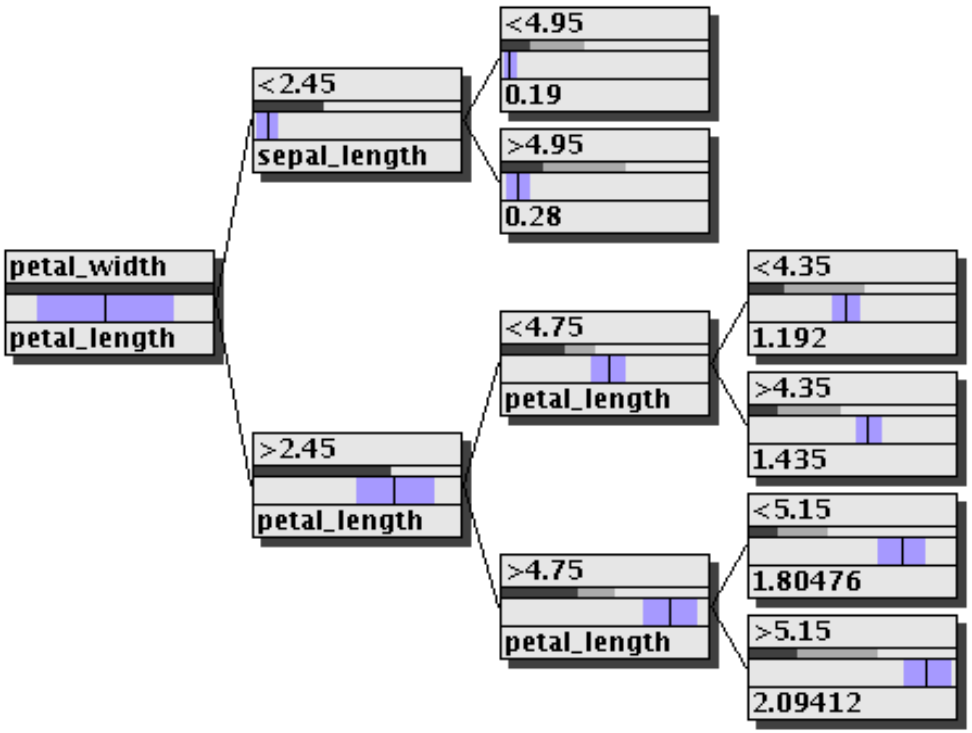
Regression Trees: Attribute Selection

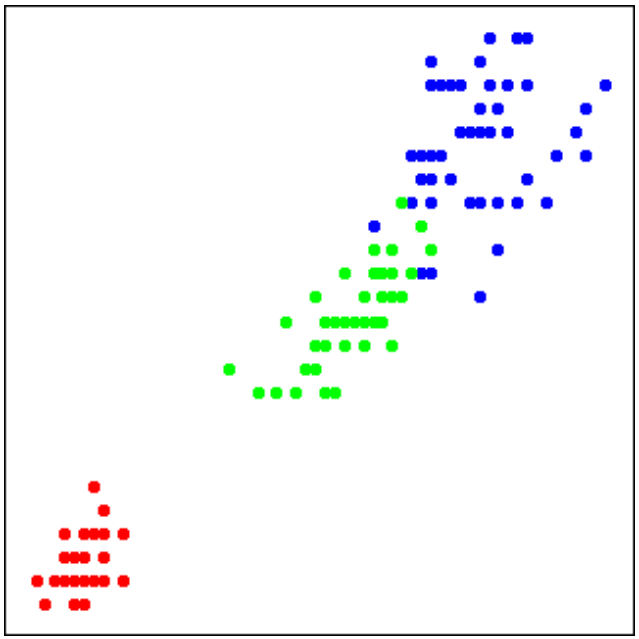


- The variance / standard deviation is compared to the variance / standard deviation in the branches.
- The attribute that yields the highest reduction is selected.

Regression Trees: An Example

A regression tree for the Iris data (petal width)
(induced with reduction of sum of squared errors)





Summary Decision and Regression Trees

- **Decision Trees are Classifiers with Tree Structure**
  - Inner node: Test of a descriptive attribute
  - Leaf node: Assignment of a class
- **Induction of Decision Trees from Data**
  (Top-Down Induction of Decision Trees, TDIDT)
  - *Divide and conquer* approach / *recursive descent*
  - *Greedy* selection of the test attributes
  - Attributes are selected based on an *evaluation measure*, e.g. information gain, χ^2 measure
  - Recommended: *Pruning* of the decision tree
- **Numeric Target: Regression Trees**

Classification Evaluation: Cross Validation

- General method to evaluate / predict the performance of classifiers.
- Serves the purpose to estimate the error rate on new example cases.
- Procedure of cross validation:
  - Split the given data set into n so-called *folds* of equal size (n-fold cross validation).
  - Combine $n - 1$ folds into a training data set, build a classifier, and test it on the n-th fold.
  - Do this for all n possible selections of $n - 1$ folds and average the error rates.
- Special case: Leave-1-out cross validation. (use as many folds as there are example cases)
- Final classifier is learned from the full data set.