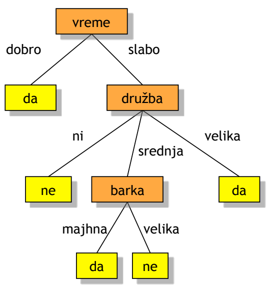
Klasifikacijska drevesa

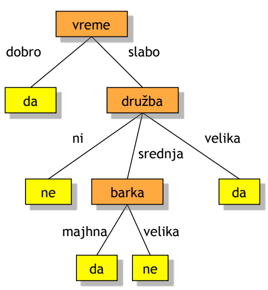
Klasifikacijsko drevo

	atributi			razred
#	vreme	družba	barka	jadranje?
1	dobro	velika	majhna	da
2	dobro	srednja	majhna	da
3	dobro	srednja	velika	da
4	dobro	ni	majhna	da
5	dobro	velika	velika	da
6	slabo	ni	majhna	ne
7	slabo	srednja	majhna	da
8	slabo	velika	velika	da
9	slabo	ni	velika	ne
10	slabo	srednja	velika	ne



Klasifikacija nerazvrščenih primerov

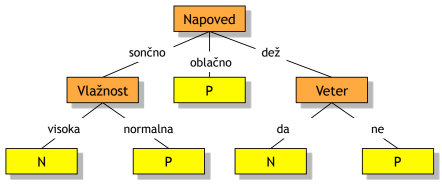
	atributi			razred
#	vreme	družba	barka	jadranje
1	dobro	ni	velika	?
2	slabo	velika	majhna	?



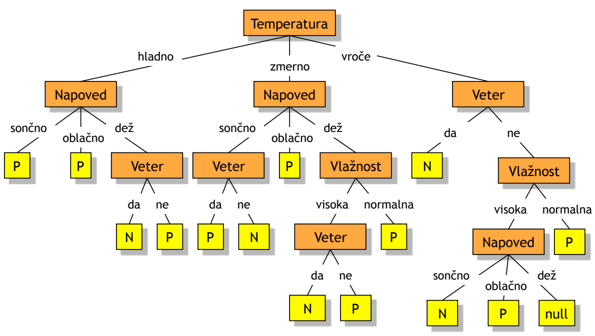
Še en primer

#	Atribut				Razred
	Napoved	Temp	Vlažnost	Veter	
1	sončno	vroče	visoka	ne	-
2	sončno	vroče	visoka	da	-
3	oblačno	vroče	visoka	ne	+
4	dež	zmerno	visoka	ne	+
5	dež	hladno	normalna	ne	+
6	dež	hladno	normalna	da	-
7	oblačno	hladno	normalna	da	+
8	sončno	zmerno	visoka	ne	-
9	sončno	hladno	normalna	ne	+
10	dež	zmerno	normalna	ne	+
11	sončno	zmerno	normalna	da	+
12	oblačno	zmerno	visoka	da	+
13	oblačno	vroče	normalna	ne	+
14	dež	zmerno	visoka	da	-

Enostavno drevo



Kompleksno drevo



Rajši imamo enostavnejša drevesa!

- Bolj razumljiva
- Bolj natančna pri napovedovanju razreda za nove primere

- Gradnja: ID3 algoritem [Quinlan, 1986]

Gradnja klasifikacijskih dreves

1. V kolikor je množica primerov razvrščena v en sam razred, jo predstavi z listom in končaj
2. Sicer
   - izberi atribut A, ki razdeli učno množico na "najbolj čiste" podmnožice
   - uporabi A za razbitje množice primerov na podmnožice; vsake podmnožica vsebuje primere, ki imajo določeno vrednost atributa a
   - ponovi postopek na nastalih podmnožicah

ID3 algoritem

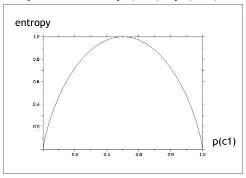
```
funkcija drevo(Primeri)
  if (vsi Primeri so razvščeni v razred R)
    return (list R)
  else
    poišči "najboljši" atribut A
    Vozlišče := zgradi odločitveno vozlišče A
    for every vrednost V atributa A do
      PV so Primeri z vrednostjo A enako V
      naslednik(A) := drevo(PV)
    enddo
    return (Vozlišče)
  endif

Ob zagonu programa:
  KlasifikacijskoDrevo := drevo(Učni primeri)
```

Entropija

- Mera nečistoče. Za diskretno spremenljivko ocenimo njeno entropijo kot:

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

- Za dvorazredni problem, p(c1)+p(c2)=1:



Residualna entropija

- Po uporabi atributa A množico primerov *S* razdelimo na podmnožice v skladu z vrednostmi, ki jih zavzame ta atribut
- *Ires* je residualna entropija, to je pričakovana entropija, ki še ostane po delitvi na podmnožice

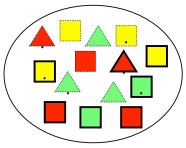
$$I_{res} = -\sum_v p(v) \sum_c p(c|v) \log_2 p(c|v)$$

Primer trikotnikov in kvadratov

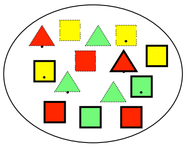
#	Atribut			Razred
	Barva	Rob	Pika	Oblika
1	zelena	črtast	ne	trikotnik
2	zelena	črtast	da	trikotnik
3	rumena	črtast	ne	kvadrat
4	rdeča	črtast	ne	kvadrat
5	rdeča	poln	ne	kvadrat
6	rdeča	črtast	da	trikotnik
7	zelena	poln	da	kvadrat
8	zelena	črtast	ne	trikotnik
9	rumena	poln	ne	kvadrat
10	rdeča	poln	ne	kvadrat
11	rumena	črtast	da	kvadrat
12	zelena	poln	ne	kvadrat
13	rumena	poln	da	kvadrat
14	rdeča	poln	da	trikotnik

Primer trikotnikov in kvadratov

#	Atribut			Oblika
	Barva	Rob	Pika	
1	zelena	črtast	ne	trikotnik
2	zelena	črtast	da	trikotnik
3	rumena	črtast	ne	kvadrat
4	rdeča	črtast	ne	kvadrat
5	rdeča	poln	ne	kvadrat
6	rdeča	črtast	da	trikotnik
7	zelena	poln	da	kvadrat
8	zelena	črtast	ne	trikotnik
9	rumena	poln	ne	kvadrat
10	rdeča	poln	ne	kvadrat
11	rumena	črtast	da	kvadrat
12	zelena	poln	ne	kvadrat
13	rumena	poln	da	kvadrat
14	rdeča	poln	da	trikotnik



Ocena entropije iz podatkov



- 5 trikotnikov
- 9 kvadratov
- verjetnosti razredov

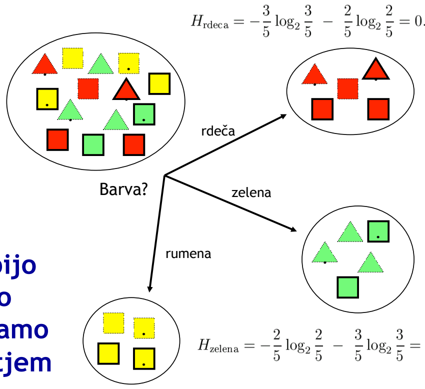
$$P(\square) = \frac{9}{14}$$

$$P(\triangle) = \frac{5}{14}$$

- entropija

$$H(X) = -\frac{9}{14}\log_2\frac{9}{14} - \frac{5}{14}\log_2\frac{5}{14} = 0.940 \text{ bits}$$

$$H_{\text{rdeca}} = -\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5} = 0.971 \text{ bits}$$

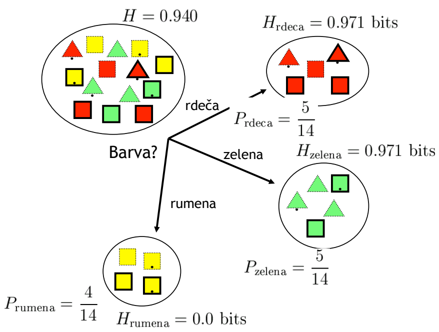


$$H_{\text{zelena}} = -\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5} = 0.971 \text{ bits}$$

$$H_{\text{rumena}} = -\frac{4}{4}\log_2\frac{4}{4} - \frac{0}{4}\log_2\frac{0}{4} = 0.0 \text{ bits}$$

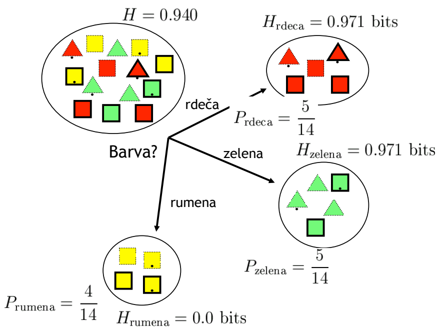
Entropijo lahko zmanjšamo z razbitjem učne množice

Entropija vrednosti atributa



$$H(\text{Barva}) = \sum P_i H_i = \frac{5}{14}0.971 + \frac{5}{14}0.971 + \frac{4}{14}0.0 = 0.694\ bits$$

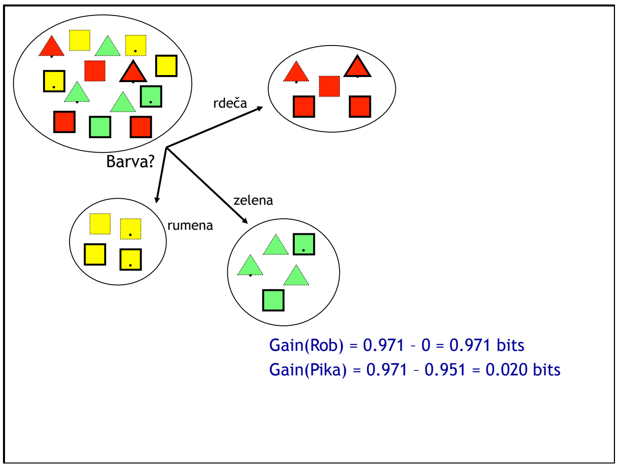
Informacijski prispevek atributa

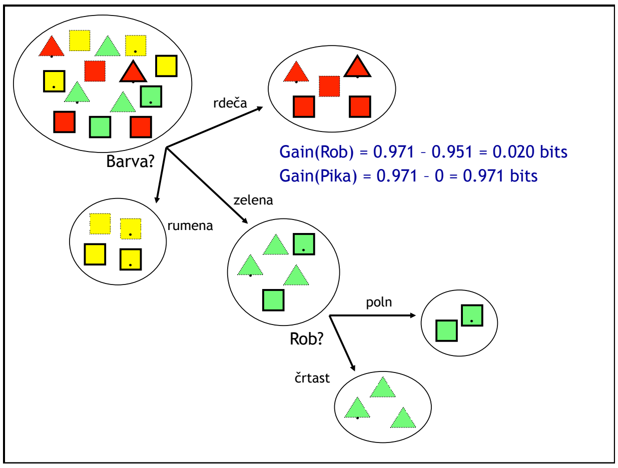


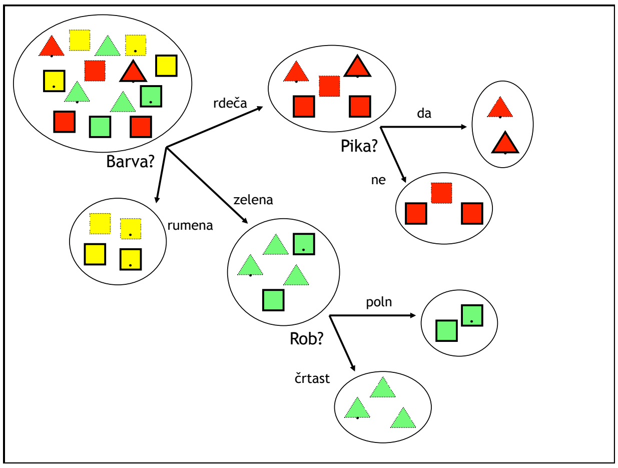
$$Gain(\text{Barva}) = H - H(\text{Barva}) = 0.940 - 0.694 = 0.246\ bits$$

Informacijski prispevki atributa

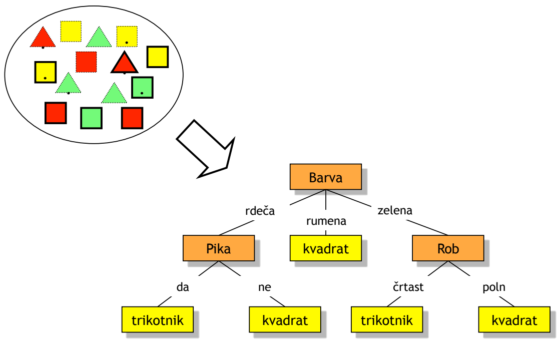
- Za vse tri atribute
  - Gain(Barva) = 0.246
  - Gain(Rob) = 0.151
  - Gain(Pika) = 0.048
- Hevristika: za razbitje množice izberemo atribut z največjim informacijskim prispevkom
- Razbitje nadaljujemo na preostalih množicah
  - odtod druga imena metodi:
    - recursive partitioning method
    - top-down induction of decision trees


rdeča
Barva?
zelena
rumena
Gain(Rob) = 0.971 - 0 = 0.971 bits
Gain(Pika) = 0.971 - 0.951 = 0.020 bits


rdeča
Barva?
Gain(Rob) = 0.971 - 0.951 = 0.020 bits
Gain(Pika) = 0.971 - 0 = 0.971 bits
zelena
rumena
poln
Rob?
črtast


rdeča
da
Pika?
Barva?
ne
zelena
rumena
poln
Rob?
črtast

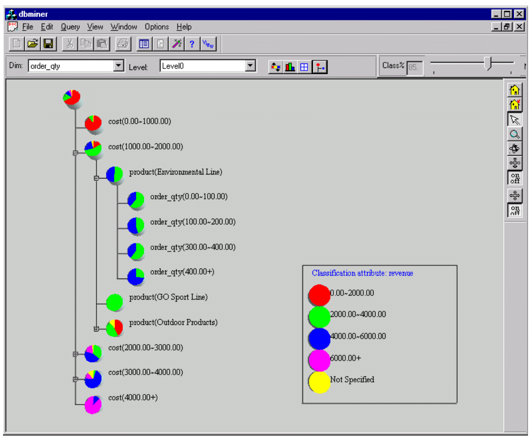
Klasifikacijsko drevo



Nekaj orodij …

- ID3 (Quinlan 79)
- CART (Brieman et al. 84)
- Assistant (Cestnik et al. 87)
- C4.5 (Quinlan 93)
- See5 (Quinlan 97)
- ...
- Orange (Demšar, Zupan 98)

DB Miner: visualization



Kleiberg et al.: Botanical Visualization of Huge Hierarchies

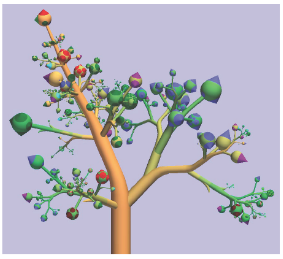


Figure 12. Unix home-directory.

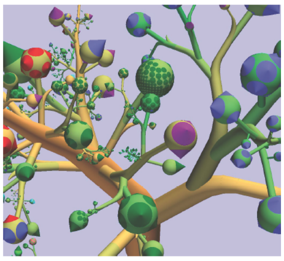


Figure 13. Detail of figure 12.

Barlow, Neville: A Comparison of 2-D Visualizations of Hierarchies

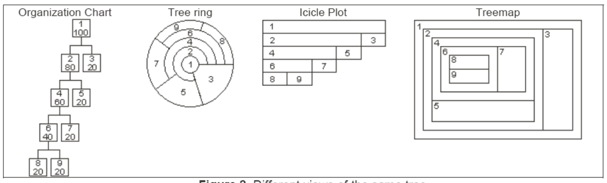
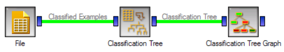
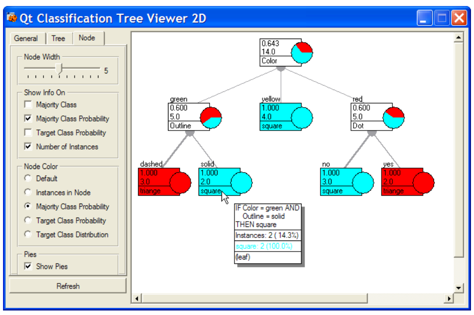


Figure 2. Different views of the same tree

Klasifikacijska drevesa v Orange





Tehnike in izboljšave

- Obravnava šuma: rezanje dreves
  - rezanje ob gradnji
  - rezanje po gradnji
- Obravnava zveznih atributov
  - vnaprejšnja diskretizacija
  - iskanje najboljšega reza
- Binarna drevesa
  - vrednosti atributa razbijemo na dve množici
  - lahko dobimo manjša in bolj točna drevesa
- Alternativni kriteriji izbora atributov v vozliščih

Klasifikacijska drevesa ostale mere nečistoče

A Defect of *Ires*

- *Ires* favors attributes with many values
- Such attribute splits S to many subsets, and if these are small, they will tend to be pure anyway
- One way to rectify this is through a corrected measure of **information gain ratio**.

Information Gain Ratio

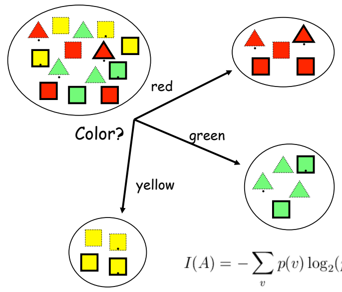
- I(A) is amount of information needed to determine the value of an attribute A

$$I(A) = -\sum_v p(v)\log_2(p(v))$$

- Information gain ratio

$$GainRatio(A) = \frac{Gain(A)}{I(A)} = \frac{I - I_{res}(A)}{I(A)}$$

Information Gain Ratio



$$I(A) = -\sum_{v} p(v)\log_2(p(v))$$

$$I(\text{Color}) = -\frac{5}{14}\log_2\frac{5}{14} - \frac{5}{14}\log_2\frac{5}{14} - \frac{4}{14}\log_2\frac{4}{14} = 1.58 \text{ bits}$$

$$\mathit{GainRatio}(\text{Color}) = \frac{\mathit{Gain}(\text{Color})}{I(\text{Color})} = \frac{0.940 - 0.694}{1.58} = 0.156$$

Information Gain and Information Gain Ratio

A	|v(A)|	I(A)	Gain(A)	GainRatio(A)
Color	3	1.58	0.247	0.156
Outline	2	1	0.152	0.152
Dot	2	0.98	0.048	0.049

Gini Index

- Another sensible measure of impurity (i and j are classes)

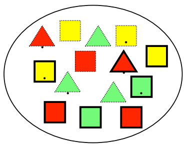
$$Gini = \sum_{i \neq j} p(i)p(j)$$

- After applying attribute A, the resulting Gini index is

$$Gini(A) = \sum_{v} p(v) \sum_{i \neq j} p(i|v)p(j|v)$$

- Gini can be interpreted as expected error rate

Gini Index



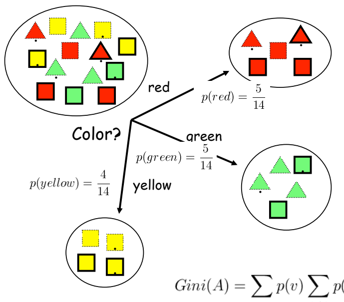
$$p(\square) = \frac{9}{14}$$

$$p(\triangle) = \frac{5}{14}$$

$$Gini = \sum_{i \neq j} p(i)p(j)$$

$$Gini = \frac{9}{14} \times \frac{5}{14} = 0.230$$

Gini Index for Color



$$Gini(A) = \sum_v p(v) \sum_{i \neq j} p(i|v)p(j|v)$$

$$Gini(\text{Color}) = \frac{5}{14} \times (\frac{3}{5} \times \frac{2}{5}) + \frac{5}{14} \times (\frac{2}{5} \times \frac{3}{5}) + \frac{4}{14} \times (\frac{4}{4} \times \frac{0}{4}) = 0.171$$

Gain of Gini Index

$$Gini = \frac{9}{14} \times \frac{5}{14} = 0.230$$

$$Gini(\text{Color}) = \frac{5}{14} \times (\frac{3}{5} \times \frac{2}{5}) + \frac{5}{14} \times (\frac{2}{5} \times \frac{3}{5}) + \frac{4}{14} \times (\frac{4}{4} \times \frac{0}{4}) = 0.171$$

$$\boxed{GiniGain(\text{Color}) = 0.230 - 0.171 = 0.058}$$

Three Impurity Measures

A	Gain(A)	GainRatio(A)	GiniGain(A)
Color	0.247	0.156	0.058
Outline	0.152	0.152	0.046
Dot	0.048	0.049	0.015

These impurity measures assess the effect of a single attribute

Criterion "most informative" that they define is local (and "myopic")

It does not reliably predict the effect of several attributes applied jointly

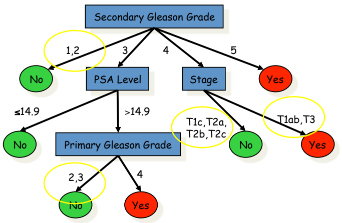
Selected other techniques for induction of decision trees

Binarization

- Induction of binary trees
  - for nominal attributes, attempts to find two sets of values that would yield minimal residual impurity
  - for continuous attributes, tries to find most appropriate cut-off points
- Effects
  - a way to attack over-fragmentation
  - (usually) increases accuracy on test sets
  - may reveal interesting attribute value groupings

Subsetting

- Similar to binarization, except attribute values may be split to several subsets rather than 2
- Optimization problem
- Used in C4.5 through simulated annealing

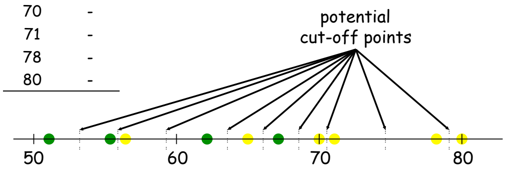


Numerical Attributes

- For every numerical attribute Ai in the node v
  - find a cut-off point X that splits the set of examples in v to subsets
    - Ai < c
    - Ai >= c
  - c that minimizes the impurity of these sets should be found
- The impurity of Ai with cut-off point at c is then assigned to Ai as its impurity

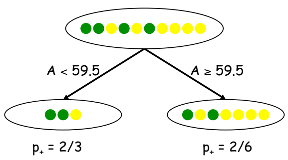
Example

A	*Class*
52	+
56	+
57	-
62	+
65	-
67	+
70	-
71	-
78	-
80	-

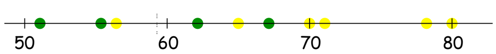


Example

A	*Class*
52	+
56	+
57	-
62	+
65	-
67	+
70	-
71	-
78	-
80	-



$$I_{res} = -\sum_v p(v) \sum_c p(c|v) \log_2 p(c|v)$$

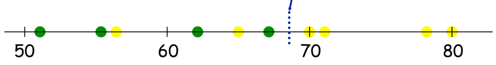


Example

A	Class
52	+
56	+
57	-
62	+
65	-
67	+
70	-
71	-
78	-
80	-

Cut-Off	Ires
54.0	0.83
56.5	0.65
59.5	0.88
63.5	0.71
66.0	0.85
68.5	0.55
70.5	0.69
74.5	0.80
79.0	0.89

best cut-off point



Treatment of Missing Values

- Causes
  - data was not collected (frequent in medicine, ...)
  - changes in experimental design during collection
  - collation of similar data sets
- Treat them as special values
  - only if missing value has some special significance

Treatment of Missing Values Through Multiplication of Example

- Multiplication of an example
  - to weighted examples with no missing values
- Procedure
  - given an example E, let missing(E) be all attributes which have their values missing
  - for E, construct examples with all possible imputation. Let $p(A_{i,j})$ be a probability that attribute Ai has j-th value. Then, the weight of the constructed examples is computed as

$$w_E = \prod_{A_i \in \text{missing(E)}} p(A_{i,j})$$

Treatment of Missing Values Through Multiplication of Example

- May be done implicitly, else data set may grow too large
  - especially if examples have few or more attributes with missing values
- Learning algorithms have then be able to handle weighted examples
  - the information gain and gain ratio can be applied to "partial" examples
  - instead of having integer counts, the weights are used when computing both gain figures

Noise Handling
in Tree Induction
(Introduction to Pruning)

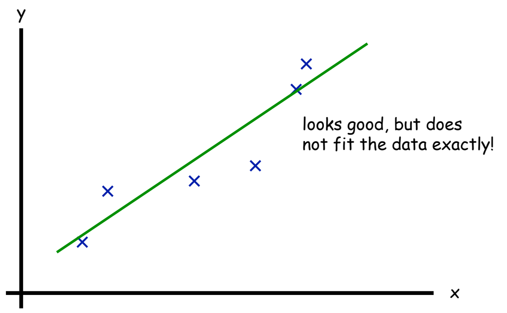
Overview

- Learning from noisy data
- Idea of tree pruning
- How to prune optimally
- Methods for tree pruning
- (Estimating probabilities)

Learning from Noisy Data

- Sources of "noise"
  - Errors in measurements, errors in data encoding, errors in examples, missing values
- "Clashes" between examples
  - same attribute vector, different class
- Problems
  - Complex hypothesis
  - Poor comprehensibility
  - **Overfitting**: hypothesis overfits the data
  - Low classification accuracy on new data

Example of Overfitting



What is the relation between x and y, y = y(x)?
How can we predict y from x?

Example of Overfitting

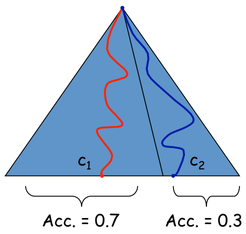
What is the relation between x and y, y = y(x)?
How can we predict y from x?

Overfitting in Extreme

- Let **default accuracy** be the probability of majority class
- Overfitting may result in accuracy lower then default
- Example
  - Data set with attributes in no correlation with class (i.e., 100% noise)
  - Two classes: c_1, c_2
  - Class probabilities: $p(c_1) = 0.7$, $p(c_2) = 0.3$
  - Default accuracy = 0.7

Overfitting in Extreme

Decision tree with one example per leaf

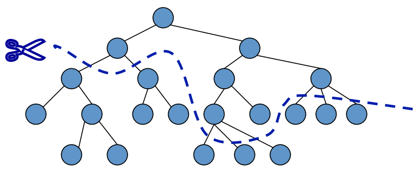


Expected accuracy = 0.7 x 0.7 + 0.3 x 0.3 = 0.58

0.58 < 0.7

Pruning of Decision Trees

- Means of handling noise in tree learning



- After pruning, the accuracy on previously unseen examples may increase

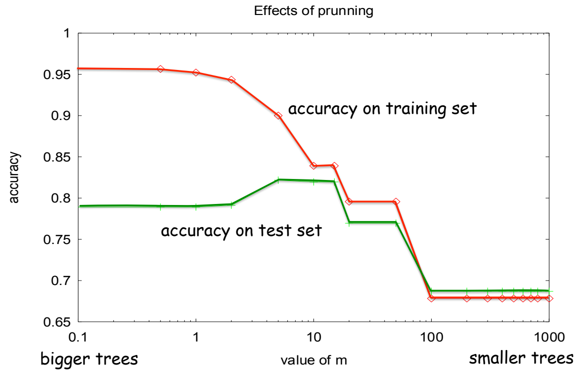
Typical Example from Practice: Locating Primary Tumor

- Data set
  - 20 classes
  - Default classifier 24.7%

	# nodes	accuracy
Overfitted tree	150	41%
Prunned tree	15	45%
Experts	-	42%

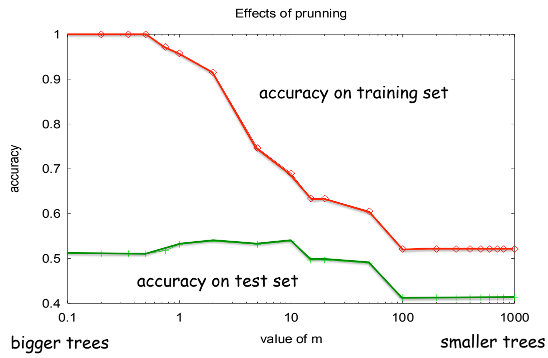
[credit]

Effects of Pruning



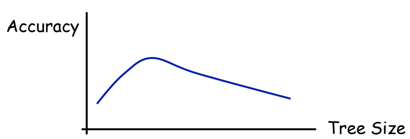
[glass]

Effects of Pruning



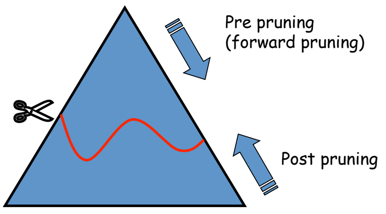
How to Prune Optimally?

- Main questions
  - How much pruning?
  - Where to prune?
  - Large number of candidate pruned trees!
- Typical relation btw tree size and accuracy on the new data



- Main difficulty in pruning: this curve is not known!

Two Kinds of Pruning



Forward Pruning

- Stop expanding trees if benefits of potential sub-trees seem dubious
  - Information gain low
  - Number of examples very small
  - Example set statistically insignificant
  - Etc.

Forward Pruning in Orange

```
learner = orangeTree.TreeLearner(\
   minExamples=5,  #1
   maxMajority = 0.7,  #2
   minSubset = 2)  #3
```

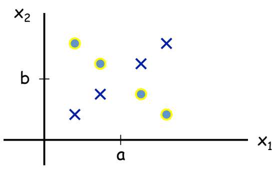
#1 Do not grow a sub-tree if there is less than 5 example in the node (make such node terminal)

#2 Stop growing tree once a frequency of majority class is above 70%

#3 Do not split a node if this would result in a leaf with less then 2 examples

Forward Pruning Inferior

- Myopic
- Depends on parameters which are hard (impossible?) to guess
- Example:



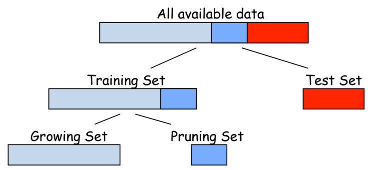
Pre and Post Pruning

- Forward pruning considered inferior and myopic
- Post pruning makes use of sub-trees and in this way reduces the complexity

Post-Pruning of Decision Trees

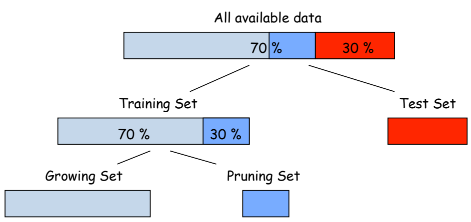
Partitioning Data in Tree Induction

- Estimating accuracy of a tree on new data: “Test Set”
- Some post pruning methods need an independent data set: “Pruning Set”
- Terminology by Esposito et al. 96



- To evaluate the classification technique, experiment with repeated random splits of data

Typical Proportions



Problem with using “Pruning Set”: less data for “Growing Set”

Overview of Selected Post-Pruning Methods

- Reduced error pruning (REP)
  Quinlan 87, Mingers 87, Esposito *et al.* 96
- Minimal error pruning (MEP)
  Niblett & Bratko 86, Cestnik & Bratko 91
- Pessimistic error pruning (PEP)
  Quinlan 87
- Error-Based Pruning (EBP)
  Quinlan 87
- Cost-Complexity Pruning (CCP)
  Brieman *et al.* 84

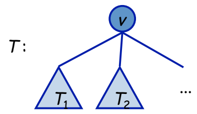
Minimal Error Pruning

- Niblett & Bratko 1986
- Cestnik & Bratko 1991

Minimal Error Pruning (MEP)

- Does not require a pruning set for estimating error
- Estimates error on new data directly from "growing set"
- Uses Bayesian method for probability estimation (either Laplace or *m*-estimate)
- Main principle
  - prune from bottom-up
  - prune so that estimated classification error is minimal

Minimal Error Pruning



$E(T)$: error of optimally pruned tree T

- Deciding about pruning at node v with sub-trees T_1, T_2, ...
- Define: static error at v : $$e(v) = p(\text{class} \neq C|v)$$
  where C is the most likely (preferred) class at v
- If T pruned at v then $$E(T) = e(v)$$
- If T not pruned at v than its (backed-up) error is
$$E(T) = p_1 E(T_1) + p_2 E(T_2) + \ldots$$

Minimal Error Pruning

- Decision about pruning
  - prune if static error ≤ backed-up error
  - error of (sub)tree T

$$E(T) = \min(\underbrace{e(v)}_{\text{static error}}, \underbrace{\sum_i p_i E(T_i)}_{\text{backed-up error}})$$

- Main question
  - How to estimate static errors $e(v)$?
  - Use Laplace or m -estimate of probability

Laplace Estimate

- At node v
  - N examples
  - n_C majority class examples
- Laplace estimate (k is number of classes)
$$p_C = \frac{n_c + 1}{N + k}$$
- Problems with Laplace
  - Assumes all classes a priory equally likely
  - Degree of pruning depends on number of classes

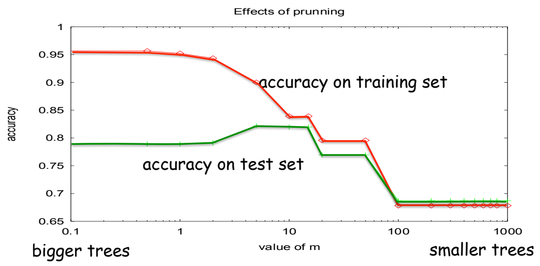
m-estimate

$$p_C = \frac{p_{Ca} \times m + n_C}{N + m}$$

- where
  - p_{Ca} = *a priori* probability of class *C*
  - *m* is non-negative parameter (tuned by expert)
- Important points
  - Takes into account prior probabilities
  - Pruning not sensitive to number of classes
  - Varying m: series of different pruned trees
  - Choice of m depends on confidence in the data

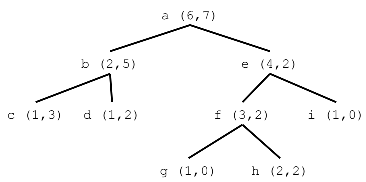
Choice of *m*

- Low noise → low *m* → little pruning
- High noise → high *m* → much pruning

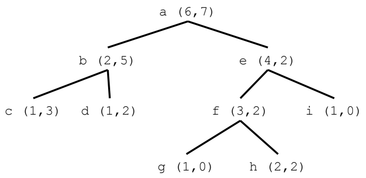


Problem

Consider decision tree below induced from 13 examples for a 2-class learning problem (classes A and B) The pairs of numbers in the nodes are the numbers of examples of class A and B in the nodes. E.g., (6,7) at the root says that there are 6 examples of class A and 7 of class B. a .. i are node labels.



Problem



- Estimate the accuracy (on new cases) of this tree using Laplace estimate!
- Prune this tree by the minimal-error pruning algorithm, use Laplace estimate!
- Assume prior probabilities of A and B to be equal to the relative frequencies at the root. Consider minimal-error pruning using m-estimates. What values of parameter m result in full size tree after pruning (i.e., no pruning occurs)?

MEP in Orange

```
import orange, orngTree, orngTest, orngStat
data = orange.ExampleTable('voting')

ms = [0, 0.2, 0.5, 1, 2, 5, 10, 100]

selection = orange.MakeRandomIndices2(data, 0.5)
train = data.select(selection, 0)
test = data.select(selection, 1)
tree = orngTree.TreeLearner(train, mForPruning=0, sameMajorityPruning=1)

print '%7s %10s %10s %10s' % ('m', 'AccTrain', 'AccTest', 'TreeSize')
for m in ms:
    pruned = orange.TreePruner_m(tree, m=m)
    resTrain = orngTest.testOnData([pruned], train)
    resTest = orngTest.testOnData([pruned], test)
    print '%7.1f %10.3f %10.3f %10d' % (m, orngStat.CA(resTrain)[0],
      orngStat.CA(resTest)[0], orngTree.countNodes(pruned))
```

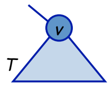
Other Pruning Techniques

Reduced Error Pruning

- Quinlan 1978
- Mingers 1978
- Esposito et al. 1996
- Elomaa & Kaariainen 2001

Reduced Error Pruning (REP)

- Use pruning set to estimate accuracy of sub-trees and accuracy at individual nodes
- Let *T* be a sub-tree rooted at node *v*



- Define:

$$\text{Gain from prunning at } v = \#\text{misclassification in } T - \#\text{misclassification at } v$$

- Repeat: prune at node with largest gain until until only negative gain nodes remain
- "Bottom-up restriction": *T* can only be pruned if it does not contain a sub-tree with lower error than *T*

Property of REP

- Esposito *et al.* 1997:
  REP with bottom-up restriction finds the smallest most accurate sub-tree w.r.t. pruning set
- I.e., from the set of all possible pruned trees, there is no pruned tree that would be more accurate than w.r.t. pruning set than the one found by REP with bottom-up restriction

Some (Original) Definitions

Quinlan's (1987, p. 225–226) original description of REP does not clearly specify the pruning algorithm and leaves room for interpretation. It includes, e.g., the following characterizations.

> "For every non-leaf subtree S of T we examine the change in misclassifications over the test set that would occur if S were replaced by the best possible leaf. If the new tree would give an equal or fewer number of errors and S contains no subtree with the same property, S is replaced by the leaf. The process continues until any further replacements would increase the number of errors over the test set.

Different Prunning Approaches

- Bottom-Up (Esposito et al., Elomaa & Kaariainen)

Nodes are pruned in a single bottom-up sweep through the decision tree, pruning each node is considered as it is encountered. The nodes are processed in postorder.

- Best-First (Mingers)

Nodes are pruned iteratively, always choosing the node whose removal most increases the decision tree accuracy over the pruning set. The process continues until further pruning is harmful.

- Comment on Mingers

However, this algorithm appears to be incorrect. Esposito et al. (1993, 1997) have shown that a tree produced by this algorithm does not meet the objective of being the most accurate subtree with respect to the pruning set. Moreover, this algorithm overlooks the explicit requirement of checking whether a subtree would lead to reduction of the classification error.

Another Ambiguity: Replacement Leaf

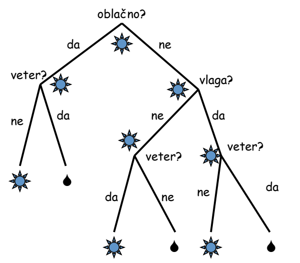
Another source of confusion in Quinlan's (1987) description of REP is that it is not clearly specified how to choose the labels for the leaves that are introduced to the tree through pruning. Oates and Jensen (1999) interpreted that the intended algorithm would label the new leaves according to the majority class of the *training* examples, but themselves analyzed a version of the algorithm where the new leaves obtain as their labels the majority of the *pruning* examples. Oates and Jensen motivated their choice by the empirical observation that in practice there is very little difference between choosing the leaf labels in either way. However, choosing the labels of pruned leaves according to the majority of pruning examples will set such leaves into a different status than the original leaves, which have as their label the majority class of training examples.

On the "Safe Side"

- Prunned bottom-up
  - in a single step prunes only internal nodes with only leaves as direct successors
  - in case of ties, prune
- Use training set to determine the class of the replacement leaf
  - classify to majority class

oblačno?	veter?	vlaga?	napoved
ne	da	ne	sonce
da	da	ne	dež
da	ne	da	dež
ne	ne	da	sonce
da	da	da	dež
da	da	ne	dež
da	da	da	sonce
da	da	ne	sonce
ne	ne	da	dež
ne	da	da	sonce
da	ne	da	sonce
ne	ne	da	sonce
da	ne	ne	sonce
ne	da	da	sonce
ne	ne	ne	sonce
ne	da	ne	sonce
ne	ne	da	sonce
da	ne	ne	sonce
da	ne	da	dež
ne	ne	da	dež
da	ne	ne	dež
da	ne	da	sonce
da	ne	ne	sonce

Primer



(večinski razred za vozlišča je označen in dobljen na osnovi učnih primerov)

Pessimistic Error Pruning (PEP)

- Does not need pruning set; uses growing set to estimate error on new data
- Error estimate (relative frequency with continuity correction)
  - probability of error (apparent error rate)

$$q = \frac{N - n_C + 0.5}{N}$$

  - where
    - N = #examples
    - n_C = #examples in majority class

Pessimistic Error Pruning (PEP)

- Error of a node (if pruned)

$$q(v) = \frac{N_v - n_{C,v} + 0.5}{N_v}$$

- Error of a subtree

$$q(T) = \frac{\sum_{l \in \text{leafs}(T)} N_l - N_{C,l} + 0.5}{\sum_{l \in \text{leafs}(T)} N_l}$$

- Prune if $q(v) \leq q(T)$
- Prunes in top-down fashion
  - fast
  - considered a weakness (on accuracy)

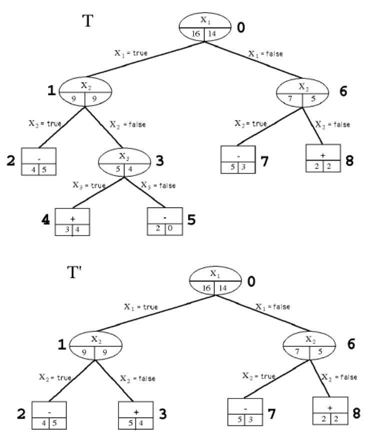
Error-Based Pruning (EBP)

- Improvement of PEP
- PEP's error estimate is in EBP made further pessimistic by the 1-SE pruning rule (1 standard error)
- Let T be a sub-tree rooted at v. Prune if

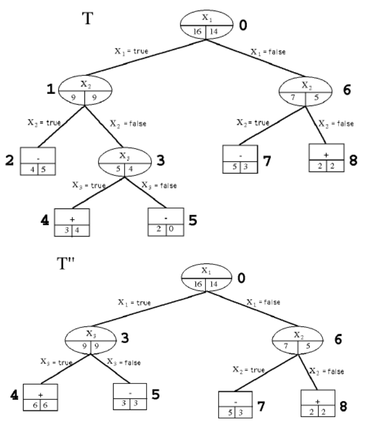
$$q(v) \leq q(T) + SE(q(T))$$

- Prunes in bottom-up fashion (as PEP)
- In addition to pruning allows grafting
  - An internal node of decision tree is removed and replaced by one of its sub trees

Example of Pruning



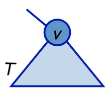
Example of Grafting



Cost-Complexity Pruning (CCP)

- Considers
  - Error rate on growing set
  - Size of tree
  - Error rate on pruning set
  - Minimize error **and** complexity
    - find best compromise between error and size
- Pruning method in CART

Cost-Complexity Pruning (CCP)



$R(v) = \#\text{errors on growing set at node } v$

$R(T) = \#\text{errors on growing set of tree } T$

$N_T = \#\text{leaves in } T$

$$\text{Total cost} = \text{Error cost} + \text{Complexity cost}$$

$$\text{Total cost} = R + \alpha N$$

$$\alpha = \text{complexity cost per leaf}$$

$$\text{Cost of } T \ (T \text{ unpruned}) = R(T) + \alpha N_T$$

$$\text{Cost of } v \ (T \text{ pruned at } v) = R(v) + \alpha$$

- When costs of T and v are equal
  - α = reduction of error per leaf

Cost-Complexity Pruning (CCP)

- Compute α for each node in unpruned tree
- Repeat
  - prune sub tree with smallest α
  - until only root is left
- This gives a series of increasingly pruned trees; estimate their accuracy
- Finally, select the "best" tree from series
  - select the smallest tree with accuracy within 1 standard error of minimum error (1-SE rule)

$$\text{Standard error} = \sqrt{\frac{R_{min}(1 - R_{min})}{\#\text{examples}}}$$

Comments on CCP

- CCP limits selection to a subset of all possible pruned trees
  - Consequence: best pruned tree may be missed
- Two ways of estimating error on new data
  - using pruning set
  - using cross-validation
    - rather complicated
    - based on debatable assumptions
- 1-SE rule tends to overprune
  - simply choosing min. error tree ("0-SE rule") performs better in experiments

Analysis of Pruning Techniques

Empirical Comparison of Pruning Methods

- Esposito, Malerba & Semeraro 1996
- 14 data sets from UCI repository
- Each data set randomly split
  - growing set: 49%
  - pruning set: 21%
  - test set: 30%
- Methods that do not require pruning set use union of growing and pruning set to grow tree
- Random splits repeated 25 times
- Measured: accuracy, over pruning, under pruning (w.r.t. optimally pruned trees)

Data Sets

database	No. Classes	No. Attributes	Real	Multi	Null Values	% Base Error	Noise Level	Uniform Distrib.
Iris	3	4	4	0	no	66.67	low	yes
Glass	7	9	9	0	no	64.49	low	no
Led	10	7	0	0	no	90	10%	yes
Hypo	4	29	7	1	yes	7.7	none	no
P.-gene	2	57	0	57	no	50	none	yes
Hepatitis	2	19	6	0	yes	20.65	none	no
Cleveland	2	14	5	5	yes	45.21	low	approx.
Hungary	2	14	5	5	yes	36.05	low	no
Switzerland	2	14	5	5	yes	6.5	low	no
Long Beach	2	14	5	5	yes	25.5	low	no
Heart	2	14	5	5	yes	44.67	low	approx.
Blocks	5	10	10	0	no	10.2	low	no
Pima	2	8	8	0	no	34.9	?	no
Australian	2	14	6	4	yes	44.5	?	approx.

Results of the Tests on Error Rates

database	REP	MEP	CVP	OSE	1SE	PEP	CV 0SE	CV 1SE	EBP	Total +/-
Iris	0	0	0	0	0	0	0	-	0	0/1
Glass	-	0	0	0	-	0	0	-	0	0/3
Led-1000	-	-	-	0	-	0	0	-	0	0/5
Led-200	-	-	-	-	-	0	0	+	0	1/5
Hypo	+	+	0	+	0	+	+	0	+	6/0
P. gene	0	0	0	0	0	0	0	0	0	0/0
Hepatitis	0	0	0	0	0	0	0	0	0	0/0
Cleveland	0	0	0	0	0	0	0	0	0	0/0
Hungary	+	+	0	+	+	+	+	+	+	8/0
Switzerland	+	0	+	+	+	+	+	+	+	8/0
Long Beach	+	+	+	+	+	+	+	+	+	9/0
Heart	0	0	0	0	0	+	0	0	+	2/1
Blocks	+	+	0	+	+	+	+	0	+	7/0
Pima	+	+	+	+	+	+	+	+	+	9/0
Australian	+	+	0	+	+	+	+	+	+	8/0
Total +/-	7/3	6/2	3/2	7/1	6/3	8/0	7/0	6/4	8/0	

Results of the Tests on Tree Size

database	REP	MEP	CVP	OSE	1SE	PEP	CV 0SE	CV 1SE	EBP
Iris	-	-	u	-	o	-	-	o	u
Glass	-	u	u	-	o	u	u	o	u
Led-1000	-	u	u	u	o	o	u	o	u
Led-200	o	-	u	-	o	o	-	-	-
Hypo	o	u	u	-	o	-	-	o	u
P-gene	o	u	u	-	o	o	-	o	u
Hepatitis	-	u	-	-	o	-	-	o	u
Cleveland	-	-	u	-	o	u	-	o	u
Hungary	o	-	u	o	o	-	-	o	u
Switzerland	-	u	-	-	-	-	-	-	-
Long Beach	-	u	-	-	o	-	-	o	u
Heart	-	-	u	-	o	o	o	o	-
Blocks	-	u	u	-	o	u	o	o	u
Pima	-	u	u	o	o	u	o	o	u
Australian	o	o	u	o	o	o	o	o	u

Experimental Results

- Does pruning improve accuracy?
  - Generally yes
  - But the effects depend on domain
- No indication that methods using pruning set perform better than those that do not