Tree Learning

Last revision: 21 Mar 201

Acknowledgements

Material derived from slides for the book
"Machine Learning" by T. Mitchell
McGraw-Hill (1997)
http://www-2.cs.cmu.edu/ tom/mlbook.html

Material d
http:ww

Material derived from slides by Eibe Frank
http://www.cs.waikato.ac.nz/ml/weka

Material derived from slides for the book
"Machine Learning" by P. Flach
Cambridge University Press (2012)
http://cs.bris.ac.uk/~flach/mlbook

Aims

This lecture will enable you to describe decision tree learning, the use of entropy and the problem of overfitting. Following it you should be able to:

- define the decision tree representation
- list representation properties of data and models for which decision tree
- repr
(T

- define entropy in the context of learning a Boolea
examples
- describe the inductive bias of the basic TDIDT alg

- define overfitting of a training set by a hypothesis
- describe developments of the basic TDIDT algorithm: pruning, rule generation, numerical attributes, many-valued attributes, costs, missing values
- describe regression and model trees

Brief History of Decision Tree Learning Algorithms

- late 1950's – Bruner et al. in psychology work on modelling *concept acquisition*
- earl ... *ncept Lea*

- late 1 ... ed on CLS is effic
- early 1990s – ID3 adds features, develops into C4. "default" machine learning algorithm

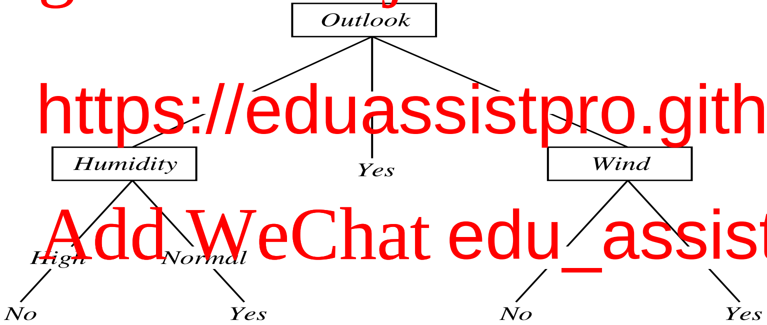
- late 1990s – C5.0, commercial version of C4.5 (av and `www.rulequest.com`)
- current – widely available and applied; influential techniques

Why use decision trees?

- Decision trees are probably the single most popular data mining tool
    -
    -
    -
    -

- There are some drawbacks, though — e.g., high v
- They do *classification*, i.e., predict a catego
  categorical and/or real inputs, or *regr*

Decision Tree for $PlayTennis$



A Tree to Predict C-Section Risk

Learned from medical records of 1000 women

Negative


```
[833+,167-] .8
Fetal_Presen
| Previous_Cse
| | Primiparous = 0: [
| | Primiparous = 1: [368+,68-] .84+ .16-
| | | Fetal_Distress = 0: [334+,47-] .88+ .12-
| | | | Birth_Weight < 3349: [201+,10.6-] .95+ .05-
| | | | Birth_Weight >= 3349: [133+,36.4-] .78+ .22-
| | | Fetal_Distress = 1: [34+,21-] .62+ .38-
| Previous_Csection = 1: [55+,35-] .61+ .39-
Fetal_Presentation = 2: [3+,29-] .11+ .89-
Fetal_Presentation = 3: [8+,22-] .27+ .73-
```

Decision Tree for Credit Rating

Decision Tree for Fisher's Iris data

Decision Trees

Decision tree representation:

- Each internal node tests an attribute
- Eac
- Eac

How would we represent the following expressions ?

- $\wedge, \vee$, XOR
- $(A \wedge B) \vee (C \wedge \neg D \wedge E)$
- M of N

Decision Trees

$X \wedge Y$

```
X = t:
| Y = t: true
| Y = f: no
X = f: no
```


$X \vee Y$

```
X = t: true
X = f:
| Y = t: true
| Y = f: no
```

Decision Trees

2 of 3

```
X = t:
| Y = t: true
| Y = f:
| | Z = t: true
| | Z = f: false
X = f:
| Y = t:
| | Z = t: true
| | Z = f: false
| Y = f: false
```


So in general decision trees represent a *dis* of constraints on the attributes values of instances.

When are Decision Trees the Right Model?

- With Boolean values for the instances $\mathbf{X}$ and class Y, the representation adopted by decision-trees allows us to represent Y as a Boo
- Giv alues for thes e subset of these, and $Y = 0$ to the rest
- Any Boolean function can be trivially represent function assigns $Y = 1$ to some subset of t f $\mathbf{X}$. So, for each combination of values with $Y = 1$, have a path from root to a leaf with $Y = 1$. All other leaves have $Y = 0$

When are Decision Trees the Right Model?

- This is nothing but a re-representation of the truth-table, and will have 2^d leaves. More compact trees may be possible, by taking into account what is common between one or more rows with the same Y valu
- But com are examples)
- In general, although possible in principle to expr function, our search and prior restrictions may n correct tree in practice.
- BUT: If you want readable models that combine logical tests with a probability-based decision, then decision trees are a good start

When to Consider Decision Trees?

- Instances described by a mix of numeric features and discrete attribute–value pairs
- Target function is discrete valued (otherwise use regression trees)
- Disj
- Pos
- Inte

Examples are extremely numerous, including:

- Equipment or medical diagnosis
- Credit risk analysis
- Modeling calendar scheduling preferences
- etc.

Top-Down Induction of Decision Trees (TDIDT)

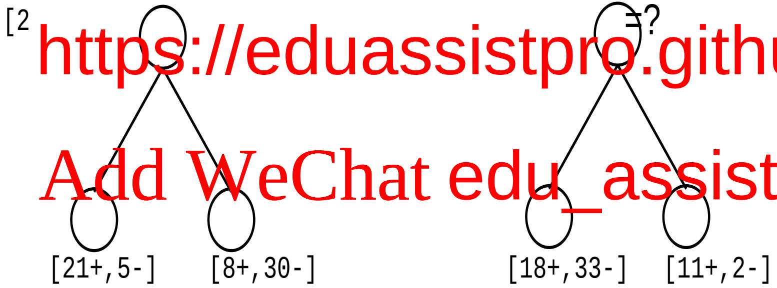
Main loop:

- A the "best" decision attribute for next $node$
- Assi
- For e

- Sort training examples to leaf nodes
- If training examples perfectly classified, Then S
  new leaf nodes

Essentially this is the "ID3" algorithm (Quinlan, 198
symbolic Machine Learning algorithm.

Which attribute is best?



Bits

You are watching a set of independent random samples of X

You obser

$P(X$	$=$	$=$	$= D) = \frac{1}{4}$

So you might see: BAACBADCDADDDA...

You transmit data over a binary serial link. You can ding with two bits (e.g. A = 00, B = 01, C = 10, D = 11)

0100001001001110110011111100...

Fewer Bits

Someon

$P(X$ [illegible]	[illegible]	[illegible]	$= D) = \frac{1}{8}$

It's possible . . .

. . . to invent a *coding* for your transmission th

average per symbol. How ?

Fewer Bits

Someone tells you that the probabilities are not equal

$P(X = A) = \frac{1}{2}$	$P(X = B) = \frac{1}{4}$	$P(X = C) = \frac{1}{8}$	$P(X = D) = \frac{1}{8}$

It's possib

. . . to inve

symbol on average. How ?

A	0
B	10
C	110
D	111

(This is just one of several ways)

Fewer Bits

Suppose there are three equally likely values

–	–	$\frac{1}{3}$

Here's a ha

A	00
B	01
C	10

Can you think of a coding that would need only 1.6 bits per symbol on average ?

Fewer Bits

Suppose there are three equally likely values

$P(X = A) = \frac{1}{3}$	$P(X = B) = \frac{1}{3}$	$P(X = C) = \frac{1}{3}$

Using the s
per symbo

A	0
B	10
C	11

This gives us, on average $\frac{1}{3} \times 1$ bit for A and $2 \times \frac{1}{3} \times 2$ bits for B and C, which equals $\frac{5}{3} \approx 1.6$ bits.

Is this the best we can do ?

Fewer Bits

Suppose t

		$\frac{1}{3}$

From information theory, the optimal number of bits to encode a symbol with probability p is $-\log_2 p$ ...

So the best we can do for this case is $-\mathrm{lo}$ $-$ C, or 1.584962500721156 bits per symbol

General Case

Suppose X can have one of m values ... $V_1, V_2, \ldots V_m$

$P(X = V\ ) = p$	$P(X = V\ ) = p$	$\ldots$	$P(X = V\ \ ) = p_m$

What's th

needed to t tion ? It's

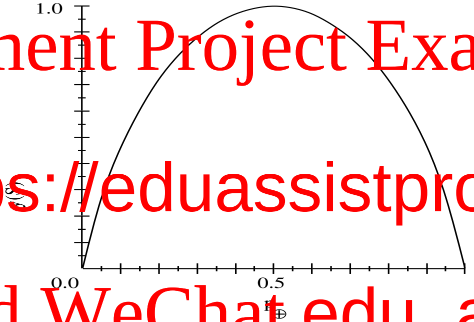
$$
\begin{aligned}
H(X) &= -p_1 \log_2 p_1 - p_2 \log \\
&= -\sum_{j=1}^{m} p_j \log_2 p_j
\end{aligned}
$$

$H(X) =$ the *entropy* of X

General Case

"High en
"Low entr

Entropy



Where:

S is a sample of training examples

$p_{\oplus}$ is the proportion of positive examples in S

$p_{\ominus}$ is the proportion of negative examples in S

Entropy

Entropy

A "pure" sample is one in which all examples are of the same

Entropy

$Entropy(S)$ = expected number of bits needed to encode class ($\oplus$ or $\ominus$) of randomly drawn member of S (under the optimal, shortest-length code)

Why ?

Informat essage having pr

So, expected number of bits to encode $\oplus$ S:

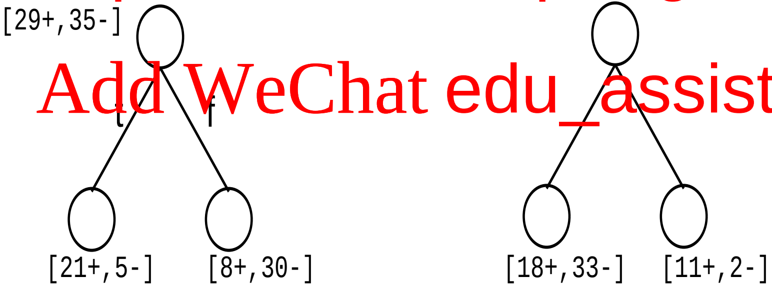
$$p_{\oplus}(-\log_2 p_{\oplus}) + p_{\ominus}($$

$$Entropy(S) \equiv -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus}$$

Information Gain

- $Gain(S, A) =$ expected reduction in entropy due to sorting on A

$$Gain(S, A) \quad Entropy(S) \quad \frac{|S_v|}{} Entropy(S_v)$$



Information Gain

$$Gain(S, \quad \frac{S_t}{} \quad \frac{S_f}{} \; ntropy(S_f) \Big)$$

$$\overline{64} \quad \overline{26} \quad {}_2 \; \overline{26} \quad \overline{} \quad {}_2 \; \overline{}$$

$$(\frac{38}{64}(-\frac{8}{38}\log_2(\frac{8}{38}) - \frac{3}{3}$$

$$= \; 0.9936 - (\; 0.2869 + 0.$$

$$= \; 0.2658$$

Information Gain

$$= \quad 0.1643$$

Information Gain

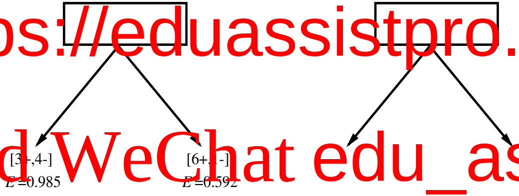
So we cho

Training Examples

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Information gain once more

Which attribute is the best classifier?



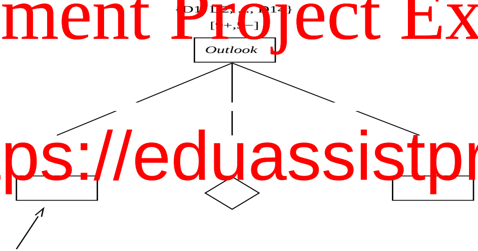
Gain (S, Humidity)

= .940 - (7/14).985 - (7/14).592
= .151

Gain (S, Wind)

= .940 - (8/14).811 - (6/14)1.0
= .048

Information gain once more



Which attribute should be tested here?

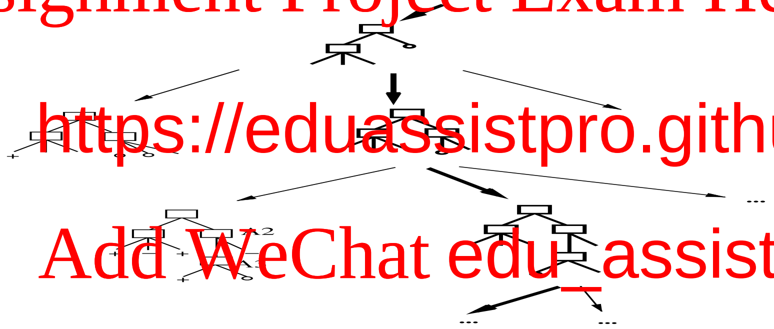
S_{sunny} = {D1,D2,D8,D9,D11}

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

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

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

Hypothesis Space Search by ID3



Hypothesis Space Search by ID3

- This can be viewed as a graph-search problem
- Each vertex in the graph is a decision tree
- Sup $_2$), and all the f

- A pai differ in just the following way: one of the leaf-node has been replaced by a non-leaf node testing a feat appeared earlier (and 2 leaves)

- This is the full space of all decision trees (is it?). We wa for a single tree or a small number of trees in this space. How should we do this?

Hypothesis Space Search by ID3

- Usual graph-search technique: greedy or beam search, starting with the vertex corresponding to the "empty tree" (single leaf node)
- Gre
  gre

  -
  - $|T')/P(D|T)$
  - Most of the calculation will cancel out: so, we will o local computation at the leaf that was convert

- RESULT: set of trees with (reasonably) high pos given D: we can now use these to answer questio $P(y' = \omega_1 | \ldots)$? or even make a *decision* or a *classification* that $y' = \omega_1$, given input data $\mathbf{x}$

Hypothesis Space Search by ID3

- Hypothesis space is complete! (contains all finite discrete-valued functions w.r.t attributes)
  -
- Out
  -

- No back tracking
  - Local minima...
- Statistically-based search choices
  - Robust to noisy data...
- Inductive bias: approx "prefer shortest tree"

Inductive Bias in ID3

Note H is the power set of instances X

$\rightarrow$Unbiased?

Not really

- Pre

attr
- Bias is a *preference* for some hypotheses *tion* of hypothesis space H

- an incomplete search of a complete hypothesis s complete search of an incomplete hypothesis space (as in learning conjunctive concepts)
- Occam's razor: prefer the shortest hypothesis that fits the data

Occam's Razor

William of Ockham (c. 1287-1347)

Entities should not be multiplied beyond necessity

Why prefe

Argument in favour:

- Fewer short hypotheses than long hypotheses

$\rightarrow$ a short hyp that fits data unlikely to be coincidence

$\rightarrow$ a long hyp that fits data might be coincidence

Occam's Razor

Argument opposed:

- The
  -
- What's so special about small sets based on *size* of hypothesis??

Look back to linear classification lecture to see how to ma

using Minimum Description Length (MDL)

Why does overfitting occur?

- Greedy search can make mistakes. We know that it can end up in local minima — so a sub-optimal choice earlier might result in a better solution later (*i.e.* pick a test whose posterior gain (or information gain) is less than the best one
- But t erro this o

  - We will see why this is the case later (lectures on Ev
  - Suppose we have two models h_1 an and e_2 and optimism o_1 and o_2. Let the true erro $+ o_1$ and $E_2 = e_2 + o_2$

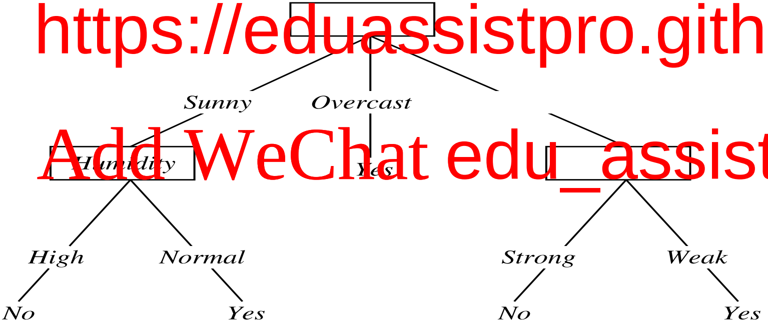
  - If $e_1 < e_2$ and $E_1 > E_2$, then we will say that h_1 has overfit then training data
- So, a search method based purely on training data estimates may end overfitting the training data

Overfitting in Decision Tree Learning

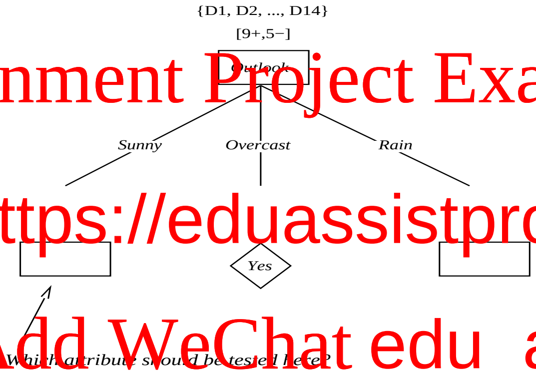
Consider adding noisy training example #15:

$$Sunny,\ Hot,\ Normal,\ Strong,\ PlayTennis = No$$

What effe



Overfitting in Decision Tree Learning



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

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

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

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

Overfitting in General

Consider error of hypothesis h over

- training data: $error_{train}(h)$
- enti

Definiti

Hypothe

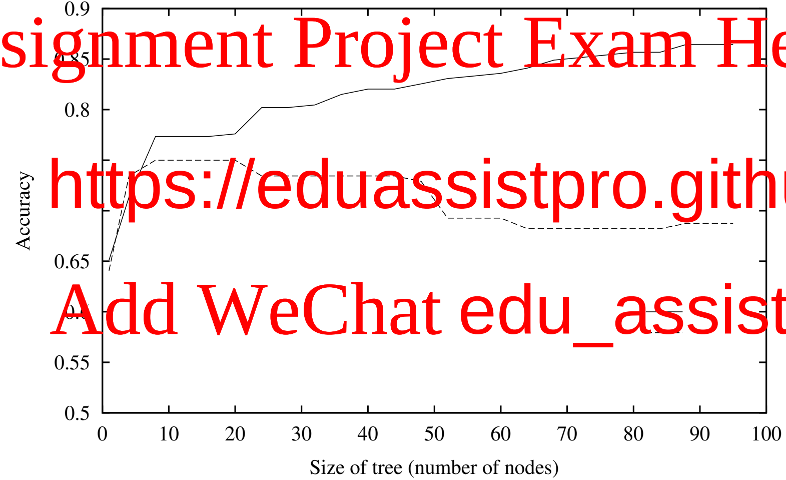
hypothesis $h' \in H$ such that

$$error_{train}(h) < erro$$

and

$$error_{\mathcal{D}}(h) > error_{\mathcal{D}}(h')$$

Overfitting in Decision Tree Learning



Avoiding Overfitting

How can we avoid overfitting? **Pruning**

- **pre-pruning** stop growing when data split not statistically significant
- **pos**
  ove

Post-pru

How to select "best" tree:

- Measure performance over training data ?
- Measure performance over separate validati
- MDL: minimize $size(tree) + size(misclassifications(tree))$?

Avoiding Overfitting

Pre-pruning

- Can
- Sto
  ass

- For example, in ID3: chi-squared test plus infor
  - only statistically significant attributes wer
    information gain procedure

Avoiding Overfitting

Pre-pruning

- Sim
  low
- In C
- In `sklearn`, this parameter is `min_sa`
- In `sklearn`, the parameter `min_impu`
  stopping when the this falls below a lower-bound

Avoiding Overfitting

Early stopping

- Pre-pruning may suffer from early stopping: may stop the growth of tree
- Cla

  - Target structure only visible in fully expande
  - Prepruning won't expand the root node

- But: XOR-type problems not common in practi
- And: pre-pruning faster than post-pruning

Avoiding Overfitting

Post-pruning

- Builds full tree first and prunes it afterwards
  - Attribute interactions are visible in fully-grown tree
- Pro
  effe

- Tw
  - Subtree replacement
  - Subtree raising

- Possible strategies: error estimation, signific
  principle
- We examine two methods: Reduced-error Pruning and Error-based Pruning

Reduced-Error Pruning

Split data i

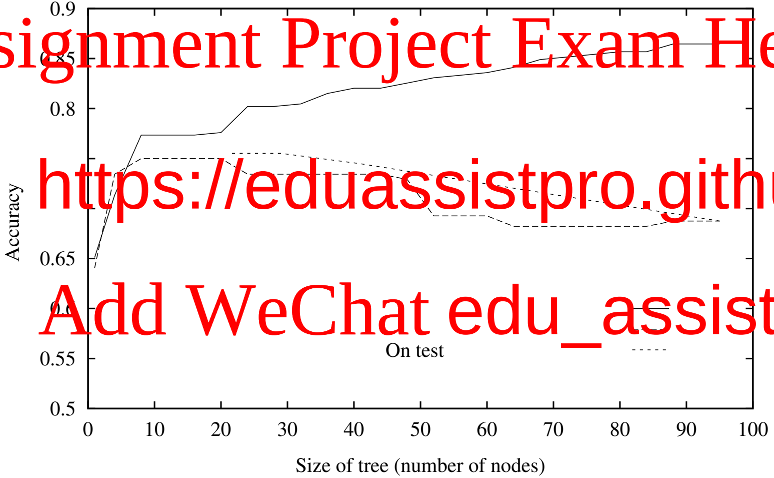
Do until fur

- Eva
  (plus those below it)
- Greedily remove the one that most improves uracy

Reduced-Error Pruning

- Go
- **Not so good** reduces effective size of training set

Effect of Reduced-Error Pruning



Error-based pruning (C4.5 / J48 / C5.0)

Quinlan (1993) describes the successor to ID3 – C4.5

- many extensions – see below
- pos
- incl

- also: pruning by converting tree to rules
- commercial version – C5.0 – is widely used
  - RuleQuest.com
  - now free

- Weka version – J48 – also widely used

Pruning operator: Sub-tree replacement

Bottom-up:
tree is considered for replacement once all its sub-trees have been considered

Error-based pruning: error estimate

Goal is to im

from train

But how ca

Make the estimate of error **pessimistic** !

Error-based pruning: error estimate

- App
- C4.
  deri

  - Standard Bernoulli-process-based method
  - **Note**: statistically motivated, but not statis
  - **However**: works well in practice !

Error-based pruning: error estimate

- The error estimate for a tree node is the weighted sum of error estimates for all its subtrees (possibly leaves).
- Upp

- f is actual (empirical) error of tree on examples a
- N is the number of examples at the tree node

- Z_c is a constant whose value depends on r c
- C4.5's default value for confidence $c = 0.25$
- If $c = 0.25$ then $Z_c = 0.69$ (from standardized normal distribution)

Error-based pruning: error estimate

- Ho
- Wh
- As

- See example on next slide (note: values not calcul
  the above formula)

Error-based pruning: error estimate

- health plan contribution: node

46

47

  - f

- sub-tr es

0.51

- sub-trees estimated to give *greater* error so prune away

Rule Post-Pruning

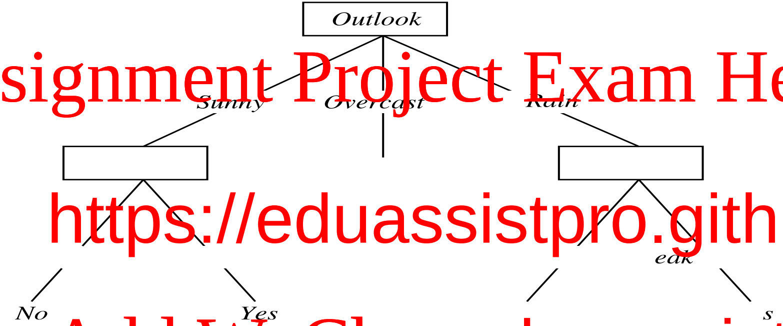
This method was introduced in Quinlan's C4.5

- Con
- Pru
- Sort final rules into desired sequence for use

For: simpler classifiers, people prefer rules to trees

Against: does not scale well, slow for large trees & dat

Converting A Tree to Rules



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

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

...

Rules from Trees (Rule Post-Pruning)

Rules can be simpler than trees but just as accurate, e.g., in C4.5Rules:

- pat
    -

- can s
    - i.e., rules can be generalized while maintaining accuracy
- greedy rule simplification algorithm
    - drop the condition giving lowest estimated er
    - continue while estimated error does not incre

Rules from Trees

Select a 'good' subset of rules within a class (C4.5Rules):

- goal: remove rules not useful in terms of accuracy
- find a
- trad
- stoc

Sets of rules can be ordered by class (C4.5Rules):

- order classes by increasing chance of making rs
- set as a default the class with the most training insta by any rule

Continuous Valued Attributes

Decision trees originated for **discrete** attributes only. Now: continuous attributes.

Can creat

- Te
- $(Te$

- Usual method: continuous attributes have a bin
- Note:

  - discrete attributes – one split exhausts all valu
  - continuous attributes – can have many splits in a tree

Continuous Valued Attributes

- Splits evaluated on all possible split points
- More computation: $n-1$ possible splits for n values of an attribute in training set
- Fay
    -
    - perature

      $\frac{(48+60)}{2}$ and $\frac{(80+90)}{2}$
- Choose best split point by info gain (or evaluation o
- Note: C4.5 uses actual values in data

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

Axis-parallel Splitting

Fitting data that is not a good “match” to the possible splits in a tree.

“Pattern Classification” Duda, Hart, and Stork, (2001)

Splitting on Linear Combinations of Features

Reduced tree size by allowing splits that are a better "match" to the data.

"Pattern Classification" Duda, Hart, and Stork, (2001)

Attributes with Many Values

Problem:

- If att
- Wh

    -
- Imagine using *Date* = March 21, 2018 as a
- High gain on training set, useless for prediction

Attributes with Many Values

One approach: use $GainRatio$ instead

c

$$SplitInformation(S, A) \equiv -$$

where S_i is subset of S for which A has val i

Attributes with Many Values

Why does this help ?

- sen
- act
    -
- therefore higher for many-valued attributes, e uniformly distributed across possible values

Attributes with Costs

Consider

- me
- rob

How to learn a consistent tree with low expected cost?

Attributes with Costs

One appr

- Exa

$$\frac{}{Cost(}$$

Preference for decision trees using lower-cost attrib

Attributes with Costs

Also: class (misclassification) costs, instance costs, . . .

SEE5 /

Can give *false positives* a different cost to *false negatives*

Forces a different tree structure to be learned to minimi ... *tric*
misclassification costs – can help if class distribution is s

Unknown Attribute Values

What if some examples missing values of A?

Use training example anyway, sort through tree. Here are 3 possible approac

- If no other exa
- assign most common value of A amo target value

- assign probability p_i to each possible value
  - assign fraction p_i of example to each desc

Note: need to classify new (unseen) examples in same fashion

Windowing

Early implementations – training sets too large for memory

As a solution ID3 implemented *windowing*:

1. select subs
2. construct d
3. use tree to cla
4. if all instances correctly classified then halt, else
5. add selected misclassified instances to the window
6. go to step 2

Windowing retained in C4.5 because it can lead to *more accurate* trees.
Related to *ensemble learning*.

Non-linear Regression with Trees

Despite some nice properties of Neural Networks, such as generalization to deal sensibly with unseen input patterns and robustness to losing neurons (predicti

problem

- Bac
  computing time; may have to be partitioned into separate modules that can be trained independently, e.g. NetTal

- Neural Networks are not very *transpa*
  representation of what has been learned

Possible solution: exploit success of tree-structured approaches in ML

Regression trees

- Differences to decision trees:
  -
  -
  -
- Can approximate piecewise constant functio
- Easy to interpret
- More sophisticated version: model trees

A Regression Tree and its Prediction Surface

"Elements of Statistical Learning" Hastie, Tibshirani & Friedman (2001)

Regression Tree on sine dataset

Regression Tree on CPU dataset

Tree learning as variance reduction

- The variance of a Boolean (i.e., Bernoulli) variable with success probability $\dot{p}$ is $\dot{p}(1-\dot{p})$, which is half the Gini index. So we could interpret the goal of tree learning as minimising the class variance (or standard deviation, in case of $\sqrt{\text{Gini}}$) in the leaves.
- In re

If a split partitions the set of target values ... ive sets $\{Y_1, \ldots, Y_l\}$, the weighted average var

$$\text{Var}(\{Y_1, \ldots, Y_l\}) = \sum_{j=1}^{l} \frac{|Y_j|}{|Y|} \text{Var}(Y_j) = \ldots = \frac{1}{|Y|} \sum_{y \in Y} y^2 - \sum_{j=1}^{l} \frac{|Y_j|}{|Y|} \overline{y}_j^2$$

The first term is constant for a given set Y and so we want to maximise the weighted average of squared means in the children.

Learning a regression tree

Imagine you are a collector of vintage Hammond tonewheel organs. You have been monitoring an online auction site, from which you collected some data about interesting transactions:

3.	A100	good	no	1051
4.	T202	good	no	270
5.	M102	good	yes	870
6.	A100	excellent	no	1770
7.	T202	fair	no	99
8.	A100	good	yes	1900
9.	E112	fair	no	77

Learning a regression tree

From this data, you want to construct a regression tree that will help you determine a reasonable price for your next purchase.

There are three features, hence three possible splits:

Model $= [\text{A100}, \text{B3}, \text{E112}, \text{M102}, \text{T202}]$

Conditio

Leslie $= [\text{yes}, \text{no}]$ $[625, 870, 1900][77, 99, 2$

The means of the first split are 1574, 4513, 77, 870 and 331, a weighted average of squared means is 3.21

split are 3142, 1023 and 267, with weighted average of sq $2.68 \cdot 10^6$; for the third split the means are 1132 and 1297, with weighted average of squared means $1.55 \cdot 10^6$. We therefore branch on Model at the top level. This gives us three single-instance leaves, as well as three A100s and three T202s.

Learning a regression tree

For the A100s we obtain the following splits:

Condition = [excellent, good, fair] $[1770][1051, 1900][]$
Leslie = [

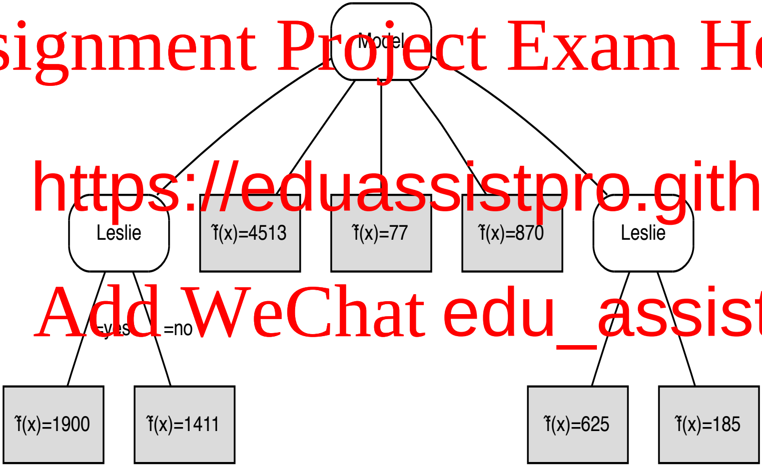
Without g

results in le
variance e
follows:

Condition = [excellent, good, fair] $[][270][99$
Leslie = [yes, no] $[625][99, 270]$

Again we see that splitting on Leslie gives tighter clusters of values. The learned regression tree is depicted on the next slide.

A regression tree



A regression tree learned from the Hammond organ dataset.

Model trees

- Like regression trees but with linear regression functions at each node
- Lin

  has b

- Onl
  - Attributes occurring in subtree (+maybe at

    to the root)

- Fast: overhead for Linear Regression (LR) not la

  only a small subset of attributes is used in tree

Two uses of features

Suppose $1 \leq x \leq 1$.

A linear ap

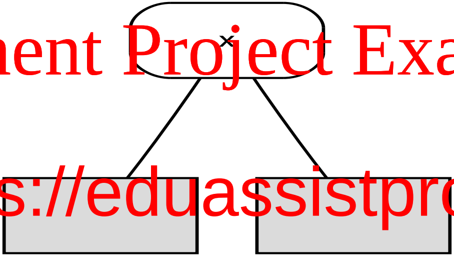
$y = 0$. Ho 0 and

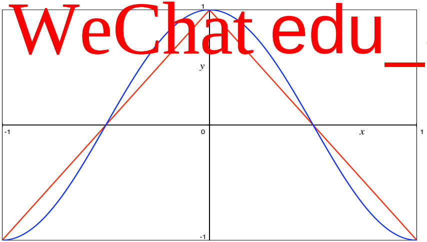
$0 \leq x \leq$

interval. We can achieve this by using x b

a regression variable (next slide).

A small model tree





Model Tree on CPU dataset

Smoothing

- Naïve prediction method – output value of LR model at corresponding leaf node
- Improve performance by *smoothing* predictions with *internal* LR mo
  -
- Sm $\overline{}$ $n+k$
  - p' prediction passed up to next higher node
  - p prediction passed to this node from below
  - q value predicted by model at this node
  - n number of instances that reach node belo
  - k smoothing constant
- Same effect can be achieved by incorporating the internal models into the leaf nodes

Building the tree

- Splitting criterion: *standard deviation reduction*

  where $T_1, T_2, \ldots$ are the sets from splits of dat
- Termination criteria (important when buildi prediction):

  - Standard deviation becomes smaller than ce training set (e.g. 5%)
  - Too few instances remain (e.g. less than four)

Pruning the tree

- Pruning is based on estimated absolute error of LR models
- Heuristic estimate:

[illegible]

  whe[illegible] v is the number of parameters in the linear model
- LR models are pruned by greedily removing term[illegible] estimated error
- Model trees allow for heavy pruning: often a single LR model can replace a whole subtree
- Pruning proceeds bottom up: error for LR model at internal node is compared to error for subtree

Discrete (nominal) attributes

- Nominal attributes converted to binary attributes and treated as nu
  -
  -
    - the ith binary attribute is 0 if an instance's value is one of the first i in the ordering, 1 otherwise
- Best binary split with original attribute provabl
  on one of the new attributes

Summary – decision trees

- Decision tree learning is a practical method for many classifier learning tasks – still a "Top 10" data mining algorithm – see `sklearn.tree.DecisionTreeClassifier`

- TDIDT family descended from ID3 searches complete hypothesis spa
- Use l, not t

- Overfitting is inevitable with an expressive hyp data, so pruning is important

- Decades of research into extensions and refine approach, e.g., for numerical prediction, logical trees
- Often the "try-first" machine learning method in applications, illustrates many general issues
- Performance can be improved with use of "ensemble" methods

Summary – regression and model trees

- Regression trees were introduced in CART — R's implementation is close to CART, but see `sklearn.tree.DecisionTreeRegressor` for a basic version
- Qui
- M5′ is bas

- Quinlan also investigated combining instanc
- CUBIST: Quinlan's rule learner for numeric pr `www.rulequest.com`

- Interesting comparison: Neural nets vs. model trees — both do *non-linear regression*
- other methods also can learn non-linear models