Decision Tree

Akito Sakurai

Decision tree

- Structure and construction
- Through the decision tree, the followings will be explained:
  - Over-training/learning
  - Bias vs. variance
  - Occam's razor etc.

Materials for decision tree

- Training samples/data
  - Instance or sample
  - supposed to be sampled from a population according to its (unknown) probability distribution
  - a set of independently sampled samples
- Hypothesis space (set)
- Measure between desired output and prediction
  - Error, error rate, cost, etc.
- New (unseen) samples/data
  - Used for evaluation of learned models.
  - Different but sampled likewise

Methods to learn

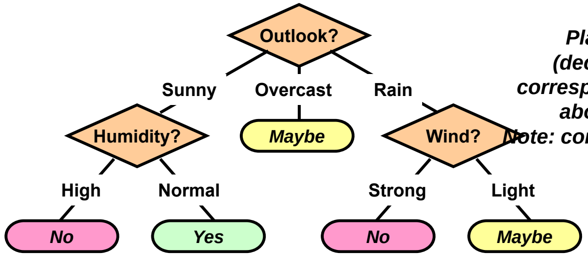
- Hypothesis set
  - A set of candidate answers (=hypothesis) of ML
    - E.g., a decision tree = a hypothesis
    - All possible decision trees = a hypothesis set
- Learning process
  - Pick up a hypothesis,
  - Check if it explains training data well,
  - Hand it as an answer if it is satisfactory, and
  - Repeat the process if it is not.

Decision Trees

Day	Outlook	Temp	Humidity	Wind	Play
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Cloudy	Hot	High	Weak	Yes
D4	Rainy	Mild	High	Weak	Yes
D5	Rainy	Cool	Normal	Weak	Yes
D6	Rainy	Cool	Normal	Strong	No
D7	Cloudy	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rainy	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Cloudy	Mild	High	Strong	Yes
D13	Cloudy	Hot	Normal	Weak	Yes
D14	Rainy	Mild	High	Strong	No

T.Mitchell, 1997

- It is a classifier
  - sample: a vector of attribute (or feature) values + label
- Internal Nodes: a test of attribute values
  - typical: if an attribute has a certain value or not (e.g., “Wind = ?”)
  - others: inequality
- Branches: True/False, values, value range, etc.
  - e.g., “Wind = Strong”, “Wind = Light”)
- Leaves: class labels, i.e. classification result

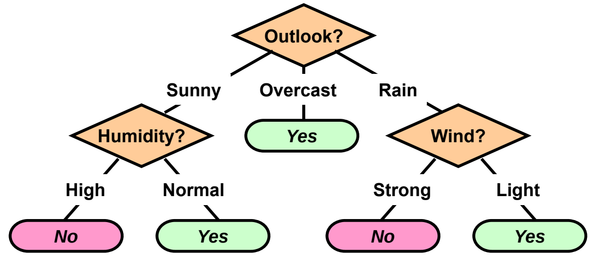


PlayTennis (decision tree corresponding to the above table)

Note: comprehensibility

Decision tree is a Boolean function

- Decision tree is a Boolean function
  - expressiveness: Any Boolean function (literal is a test on an attribute) can be expressed
  - Why?
    - Decision tree is directly interpreted as a <u>D</u>isjunctive <u>N</u>ormal <u>F</u>orm (DNF)
    - The following one: (*Sunny* ∧ *Normal-Humidity*) ∨ *Overcast* ∨ (*Rain* ∧ *Light-Wind*)



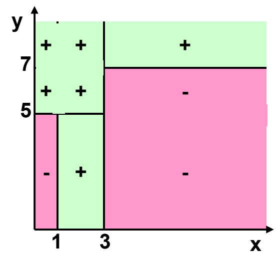
Decision boundary 1

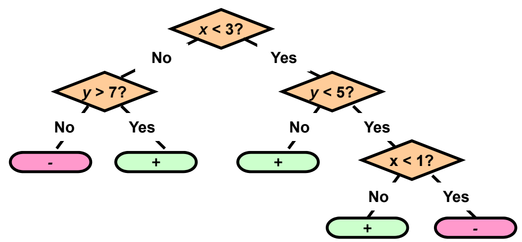
- Instances are in general expressed with discrete features.
  - Continuous values features are treated as range
  - Typical value types
    - nominal ({red, yellow, green})
    - quantized ({low, medium, high})
  - Continuous values
    - Discretization and/or <u>vector quantization</u>: divide by thresholds

ex. U. M. Fayyad and K. B. Irani, Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning, Proc. 13th IJCAI (1993).

Decision boundaries 2

- Continuous values can be used directly
  - In the case, a branch designates a value range.
  - At each node, the samples are tested with an attribute against a threshold.
    - The thresholds are determined in learning phase.
  - Decision boundaries are, therefore, piecewise linear (hyper plane)
- Sample space is divided by hyperplanes perpendicular to axes.





Learning process is hypothesis output process

- In general, learning process is a process to output hypotheses. They may be output
  - Just once,
  - Predefined times, or
  - Infinitely many times.
- When hypothesis space is finite (# of hypotheses in the space is finite)
  - Try out all the hypotheses, and output the best one, or
  - Try out a part of it, and output the best one.
    - In general, although finite, exhaustive search is impossible
- When the hypothesis space is infinite
  - Try just a part of it, and
  - Continously output tried hypothesis with the evaluation result infinitely
- In either case, the order of search is critically important

Selection order of hypothesis

- Output infinitely many times
  - Seemingly, only one hypothesis is output, because
    - When a terminal condition is satisfied, halts and outputs
- Why infinitely many times?
  - The optimum is a limit of infinite repetitions.
- Search order is critical
  - The later the better is what we require
    - If so, we can stop the sequence at any time
  - In reality, we cannot do it
- Bias
  - Any order we take, our best hypothesis has certain bias, which is called, training/learning bias

For decision trees

- Hypothesis space is finite
  - Discrete attributes only: one attribute could appear only once on a path from root to a leaf.
  - When continuous attributes exist: if we count trees once which give the same prediction to the same sample, the trees are finite.
- But exhaustive check is infeasible
  - Too many
- How to treat?

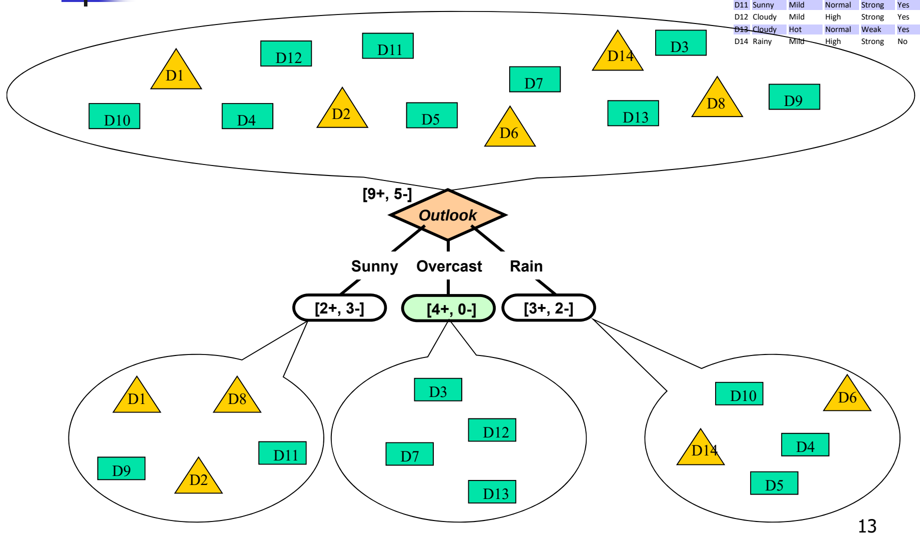
DT-learning: top-down induction (*ID3*)

- Algorithm *Build-DT* (*Examples*, *Attributes*)
  - Recursively applied to partial trees
    - Examples: a subset of training set, Attributes: a subset of all the attributes

IF the *labels* of *Examples* are the same, THEN RETURN (the leaf with the *label*)
ELSE
  IF *Attributes* is empty THEN RETURN (the leaf with the majority *label*)
  ELSE
    select a <u>best attribute *A*</u> as a root. Build trees as follows and connect them.
    FOR each value *v* of *A*
      Build branches corresponding to conditions $A = v$
      IF $\{x \in Examples \mid x.A = v\} = \emptyset$
        THEN build a leaf with majority *label*
        ELSE *Build-DT* ($\{x \in Examples \mid x.A = v\}$, Attributes – {A})

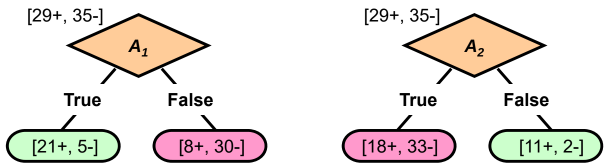
DT-learning: recursion

Day	Outlook	Temp	Humidity	Wind	Play
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Cloudy	Hot	High	Weak	Yes
D4	Rainy	Mild	High	Weak	Yes
D5	Rainy	Cool	Normal	Weak	Yes
D6	Rainy	Cool	Normal	Strong	No
D7	Cloudy	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rainy	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Cloudy	Mild	High	Strong	Yes
D13	Cloudy	Hot	Normal	Weak	Yes
D14	Rainy	Mild	High	Strong	No



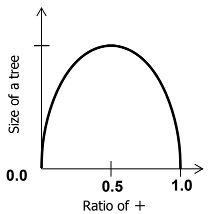
DT-learning: best attributes? (*ID3*)

- Which attribute is best?
- For example, which one is better?



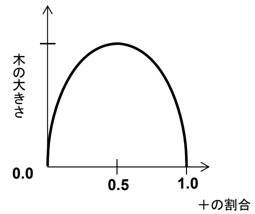
What is the best attribute?

- As a result of selecting the attribute, the smaller the built tree is, the better
  - A smaller tree is better. Why?
    - A detailed explanation will be given later. Here, a tree is smaller → paths to leaves are shorter → a fewer attributes are used for decision → closer to reality
- In case of binary classification
  - Comparing [10+,10–] to [0+,20–], which will result in smaller tree?



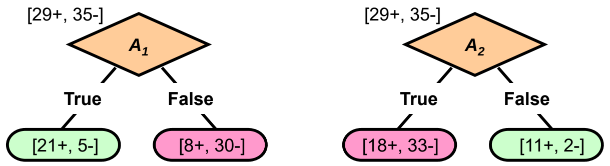
Best attribute?

- Various functions
- Suppose the x-axis represents ratio of "+" class. The peak should come at 0.5. Axial symmetric with 0.5.
- Entropy (average information content) function is the typical one
  $H(D) \equiv -p_+ \log_b(p_+) - p_- \log_b(p_-)$



Best attribute: calculation

Entropy before the selection $-\frac{29}{64}\log\frac{29}{64} - \frac{35}{64}\log\frac{35}{64} \approx 0.9936507$



0.7062741 0.7424876 0.9366674 0.6193822

26/64 38/64 51/64 13/64

0.7277758 0.8722188

Entropy after the selection of attribute

Entropy: information theoretic def.

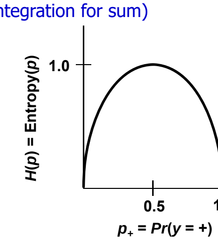
- Elements considered
  - D: a set of samples $\{<x_1, c(x_1)>, <x_2, c(x_2)>, \ldots, <x_m, c(x_m)>\}$
  - $p_+ = Pr(c(x) = +)$, $p_- = Pr(c(x) = -)$
- Definition
  - H is defined on a probability distribution p
  - For samples in D, frequency of labels + and – be expressed by p_+ and p_- respectively
  - The entropy of D is:

$$H(D) \equiv -p_+ \log_b(p_+) - p_- \log_b(p_-)$$

- Unit?
  - Depends on the basis of log (bits for $b = 2$, nats for $b = e$.)
  - 1 bit is necessary to encode a sample in its worst case ($p_+ = 0.5$)
  - If uncertainty is small (e.g., $p_+ = 0.8$), less than 1 bit is necessary

Entropy: intuitive explanations

- A measure for uncertainty/ambiguity; higher for higher uncertainty
  - Target of measure
    - purity: how the sample set is close to status being of just one label
    - impurity (disorder): how the set is close to status where the labels are not predictable
  - measure: entropy
    - Positively correlate with: impurity, uncertainty, irregularity, unpredictability
    - Negatively correlate with: purity, certainty, regularity, predactibility
- Example
  - For simplicity, suppose that H = {0, 1}, and distributed according to a distribution $Pr(y)$
    - Same as the case with (more than two) discrete labels
    - Even continuous probability distribution: differential entropy (integration for sum)
  - The most pure cases about y are one of the two:
    - $Pr(y = 0) = 1$, $Pr(y = 1) = 0$
    - $Pr(y = 1) = 1$, $Pr(y = 0) = 0$
  - The distribution with the least purity
    - $Pr(y = 0) = 0.5$, $Pr(y = 1) = 0.5$
    - The most: inpurity/uncertainty/irregularity/unpredictability
    - Entropy function: concave ("upward convex")

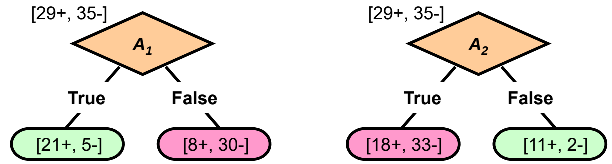


Information gain: definition

- Partition according to attribute values
  - remember: partition of D is, a set of mutually exclusive subsets whose union is D
  - target: reduction of uncertainty/impurity by partition with values of attribute A
- Definition
  - Information gain by attribute A is expected reduction of entropy by partition based on A

$$Gain(D, A) \equiv H(D) - \sum_{v \in values(A)} \left[\frac{|D_v|}{|D|} \cdot H(D_v) \right] = \frac{1}{|D|} \left(|D| \cdot H(D) - \sum_{v \in values(A)} |D_v| \cdot H(D_v) \right)$$

  where D_v is $\{x \in D \mid x.A = v\}$, i.e., a set of samples in D whose value for attribute A is v
  - Note: entropy values are adjusted according to the size of subset D_v of A
    - Because entropy value is a value per one element of the set.
- Which attribute is better?



Example

- Training sample set for a concept *PlayTennis*

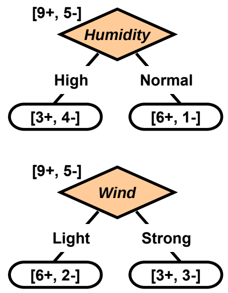
Day	Outlook	Temperature	Humidity	Wind	*PlayTennis?*
1	Sunny	Hot	High	Light	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Light	Yes
4	Rain	Mild	High	Light	Yes
5	Rain	Cool	Normal	Light	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Light	No
9	Sunny	Cool	Normal	Light	Yes
10	Rain	Mild	Normal	Light	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Light	Yes
14	Rain	Mild	High	Strong	No

- *ID3* ≡ *Build-DT* where information gain function *Gain*(•) is used
- Let us see how *ID3* works

Building decision tree for *PlayTennis* by using *ID3* (1)

- Select an attribute for the root node

Day	Outlook	Temperature	Humidity	Wind	*PlayTennis?*
1	Sunny	Hot	High	Light	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Light	Yes
4	Rain	Mild	High	Light	Yes
5	Rain	Cool	Normal	Light	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Light	No
9	Sunny	Cool	Normal	Light	Yes
10	Rain	Mild	Normal	Light	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Light	Yes
14	Rain	Mild	High	Strong	No



- Prior (unconditioned) distribution: 9+, 5-
  - *H*(*D*) = -(9/14) log (9/14) - (5/14) log (5/14) *bits* = 0.94 bits
  - *H*(*D*, *Humidity* = High) = -(3/7) log (3/7) - (4/7) log (4/7) = 0.985 bits
  - *H*(*D*, *Humidity* = Normal) = -(6/7) log (6/7) - (1/7) log (1/7) = 0.592 bits
  - *Gain*(*D*, *Humidity*) = 0.94 – ((7/14) * 0.985 + (7/14) * 0.592) = 0.151 bits
  - similarly, *Gain* (*D*, *Wind*) = 0.94 – ((8/14) * 0.811 + (6/14) * 1.0) = 0.048 bits

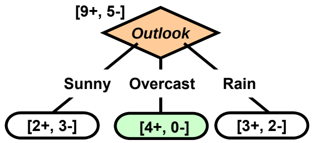
$$Gain(D, A) \equiv H(D) - \sum_{v \in values(A)} \left[\frac{|D_v|}{|D|} \bullet H(D_v) \right]$$

Building decision tree for *PlayTennis* by using *ID3* (2)

- Select an attribute for the root node

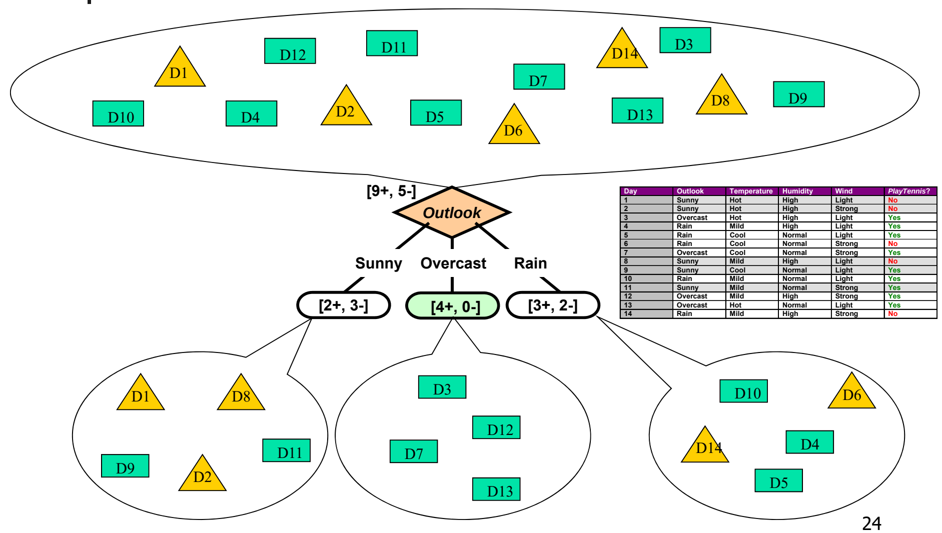
Day	Outlook	Temperature	Humidity	Wind	*PlayTennis?*
1	Sunny	Hot	High	Light	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Light	Yes
4	Rain	Mild	High	Light	Yes
5	Rain	Cool	Normal	Light	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Light	No
9	Sunny	Cool	Normal	Light	Yes
10	Rain	Mild	Normal	Light	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Light	Yes
14	Rain	Mild	High	Strong	No

  - *Gain*(*D*, *Humidity*) = 0.151 bits
  - *Gain*(*D*, *Wind*) = 0.048 bits
  - *Gain*(*D*, *Temperature*) = 0.029 bits
  - *Gain*(*D*, *Outlook*) = 0.246 bits



- Select the next attribute (root of subtree (child tree))
  - Continue until all the attributes are used (on a path to the leaf) or purity=100%
  - purity = 100% means just one label for the training samples for the leaf
  - By the way, could *Gain*(*D*, *A*) < 0 happen?

DT-learning: recursive application



Building decision tree for *PlayTennis* by using *ID3* (3)

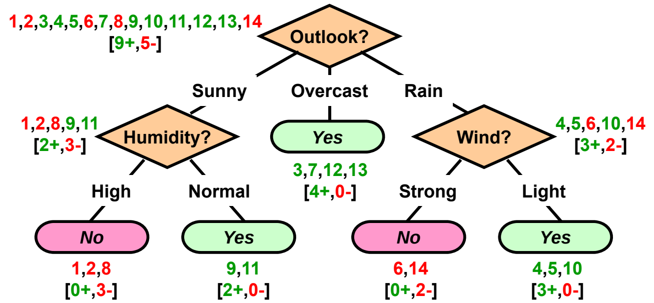
- Select an attribute for the root node (for subtree)

Day	Outlook	Temperature	Humidity	Wind	*PlayTennis?*
1	Sunny	Hot	High	Light	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Light	Yes
4	Rain	Mild	High	Light	Yes
5	Rain	Cool	Normal	Light	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Light	No
9	Sunny	Cool	Normal	Light	Yes
10	Rain	Mild	Normal	Light	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Light	Yes
14	Rain	Mild	High	Strong	No

  - convention: 0 log (0/a) = 0
  - $Gain(D_{Sunny}, Humidity)$ = 0.97 - (3/5) * 0 - (2/5) * 0 = 0.97 bits
  - $Gain(D_{Sunny}, Wind)$ = 0.97 - (2/5) * 1 - (3/5) * 0.92 = 0.02 bits
  - $Gain(D_{Sunny}, Temperature)$ = 0.57 bits
- Top-down and recursive application
  - If there are *n* discrete attributes only, O(*n*) partitions are enough for a path.
  - At each level of a tree, there exits at most one scan through a whole data

Building decision tree for *PlayTennis* by using *ID3* (4)

Day	Outlook	Temperature	Humidity	Wind	*PlayTennis?*
1	Sunny	Hot	High	Light	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Light	Yes
4	Rain	Mild	High	Light	Yes
5	Rain	Cool	Normal	Light	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Light	No
9	Sunny	Cool	Normal	Light	Yes
10	Rain	Mild	Normal	Light	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Light	Yes
14	Rain	Mild	High	Strong	No



For broader fields

- Assumptions in the algorithm ID3 and its avoidance.
  - Discrete output → continuous output
    - Continuous values can be output
    - E.g. Regression trees [Breiman *et al*, 1984]
  - Discrete input → continuous input
    - Quantization methods
    - Inequality instead of equality
- Scale-up
  - Knowledge discovery and/or data mining in very large DB (VLDB)
  - Positive: there are good algorithm to process many samples
  - Negative: too many attributes is a headaches
- Desired tolerance
  - Tolerance to noisy data (classification noise ≡ incorrect labels; attribute noise ≡ inaccurate/low frequent data).
  - Tolerance to missing data

Decision tree in R

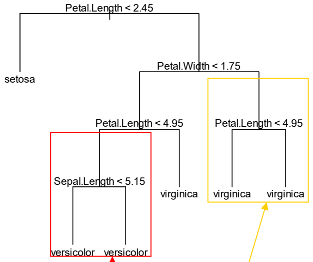
- As R language packages, we have *tree*, *rpart*, and *mvpart* which is an extension of rpart to multivariate regression trees

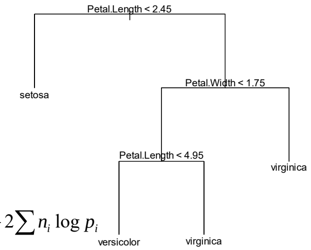
Examples: tree

```
library(tree)
```

```
data(iris)
(iris.tr<-tree(Species~.,data=iris))
plot(iris.tr,type="u"); text(iris.tr)
```

```
(iris.tr1<-snip.tree(iris.tr,nodes=c(12,7)))
plot(iris.tr1,type="u");text(iris.tr1)
```



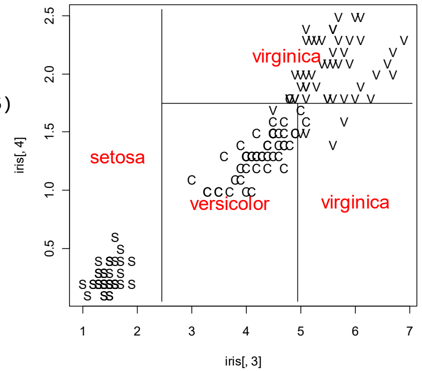


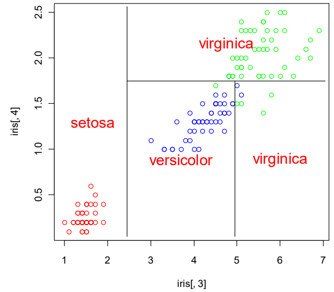
deviance= $-2\sum n_i \log p_i$

```
1) root 150 329.600 setosa ( 0.33333 0.33333 0.33333 )
   2) Petal.Length < 2.45 50   0.000 setosa ( 1.00000 0.00000 0.00000 ) *
   3) Petal.Length > 2.45 100 138.600 versicolor ( 0.00000 0.50000 0.50000 )
     6) Petal.Width < 1.75 54  33.320 versicolor ( 0.00000 0.90741 0.09259 )
      12) Petal.Length < 4.95 48   9.721 versicolor ( 0.00000 0.97917 0.02083 )
        24) Sepal.Length < 5.15 5   5.004 versicolor ( 0.00000 0.80000 0.20000 ) *
        25) Sepal.Length > 5.15 43   0.000 versicolor ( 0.00000 1.00000 0.00000 ) *
      13) Petal.Length > 4.95 6   7.638 virginica ( 0.00000 0.33333 0.66667 ) *
     7) Petal.Width > 1.75 46   9.635 virginica ( 0.00000 0.02174 0.97826 )
      14) Petal.Length < 4.95 6   5.407 virginica ( 0.00000 0.16667 0.83333 ) *
      15) Petal.Length > 4.95 40   0.000 virginica ( 0.00000 0.00000 1.00000 ) *
```


Examples: tree

```
iris.label<-c("S", "C", "V")[iris[, 5]]
plot(iris[,3],iris[,4],type="n")
text(iris[,3],iris[,4],labels=iris.label)
partition.tree(iris.tr1,add=T,col=2,cex=1.5)
```



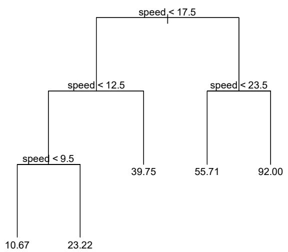



```
iris.color<-c("red","blue","green")[iris[,5]]
plot(iris[,3],iris[,4],col=iris.color)
partition.tree(iris.tr1,add=T,col=2,cex=1.5)
```


Examples: tree

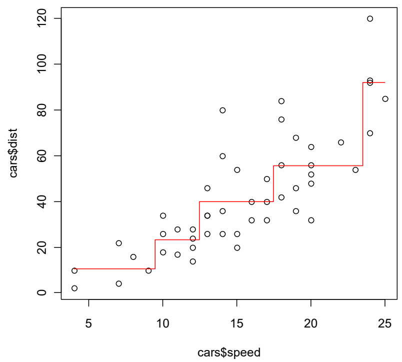
```
> library(tree)
> data(cars)
> cars.tr<-tree(dist~speed,data=cars)
> print(cars.tr)
node), split, n, deviance, yval
      * denotes terminal node

1) root 50 32540.0 42.98
  2) speed < 17.5 31  8307.0 29.32
    4) speed < 12.5 15  1176.0 18.20
      8) speed < 9.5 6   277.3 10.67 *
      9) speed > 9.5 9   331.6 23.22 *
    5) speed > 12.5 16  3535.0 39.75 *
  3) speed > 17.5 19  9016.0 65.26
    6) speed < 23.5 14  2847.0 55.71 *
    7) speed > 23.5 5  1318.0 92.00 *
> plot(cars.tr,type="u")
> text(cars.tr)
> plot(cars.tr,type="u")
> text(cars.tr)
>
```

```
library(tree)
data(cars)
cars.tr<-tree(dist~speed,data=cars)
print(cars.tr)
plot(cars.tr,type="u")
text(cars.tr)
plot(cars.tr,type="u")
text(cars.tr)
```



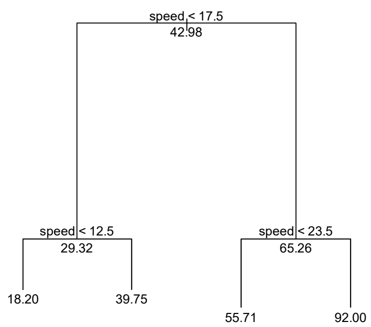
Examples: tree

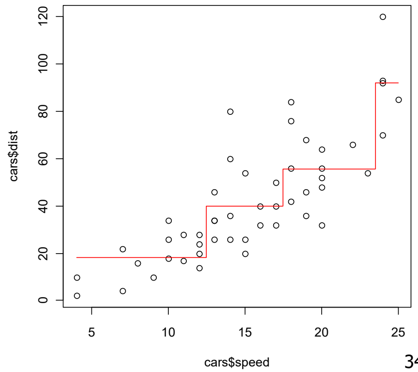
```
plot(cars$speed,cars$dist)
partition.tree(cars.tr,add=T,col=2)
```



Examples: tree

```
(cars.tr1<-prune.tree(cars.tr,best=4))
plot(cars.tr1); text(cars.tr1,all=T)

plot(cars$speed,cars$dist)
partition.tree(cars.tr1,add=T,col=2)
```

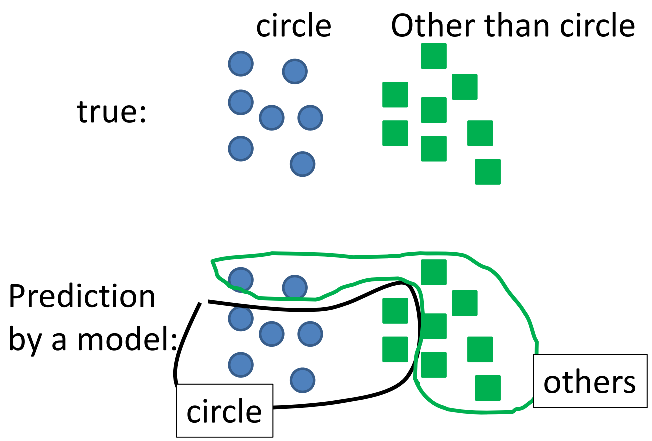




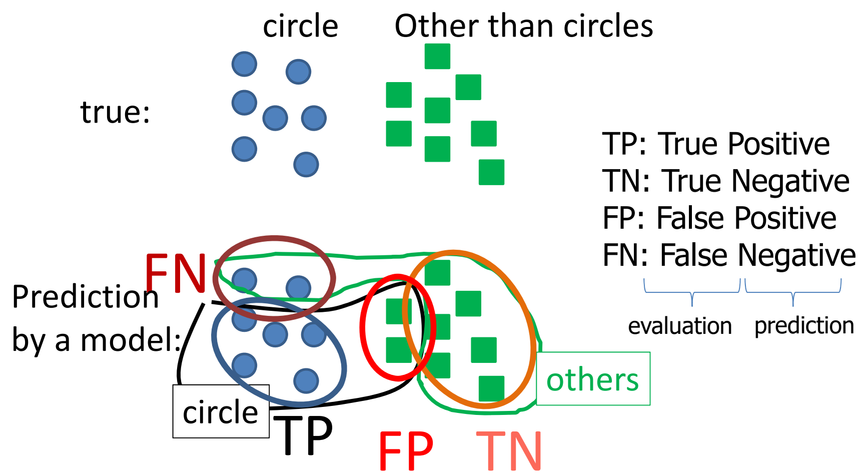
Evaluation of hypotheses

- We need to evaluate hypotheses/models
  - Hypotheses/models are something we are going to use
  - We want to use, in some reasonable sense, good, hypothesis/model with high accuracy/credibility
  - What evaluation method do we have?

Basic concepts before Precision/Recall



TP, TN, FP, FN



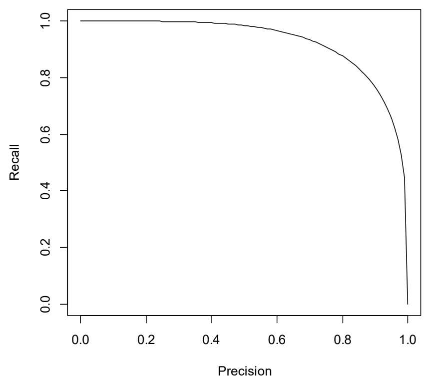
Confusion matrix

		True value P	True value N
Prediction by a model	P	**TP** (True Positive)	**FP** (False Positive)
	N	**FN** (False Negative)	**TN** (True Negative)

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN} \qquad \text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN}$$

Precision/Recall Tradeoff and F-measure



$$F = \frac{1}{\frac{1}{2}\left(\frac{1}{precision} + \frac{1}{recall}\right)}$$

Confusion matrix

1-β=sensitivity =TPR= $\frac{TP}{FN+TP}$

		True value P	True value N
Prediction	P	**TP** (True Positive)	**FP** (False Positive)
	N	**FN** (False Negative)	**TN** (True Negative)

$\alpha = \text{FPR} = \frac{FP}{FP+TN}$

Type 1 error

$\text{Precision} = \frac{TP}{TP+FP}$

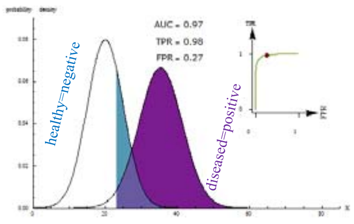
1-α=specificity =TNR= $\frac{TN}{FP+TN}$

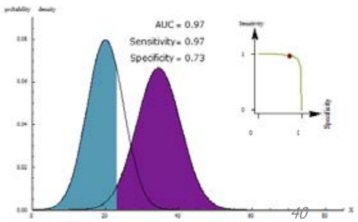
$\text{Accuracy} = \frac{TP+TN}{TP+FP+TN+FN}$

$\text{Recall} = \frac{TP}{TP+FN}$

Type 2 error

$\beta = \text{FNR} = \frac{FN}{FN+TP}$





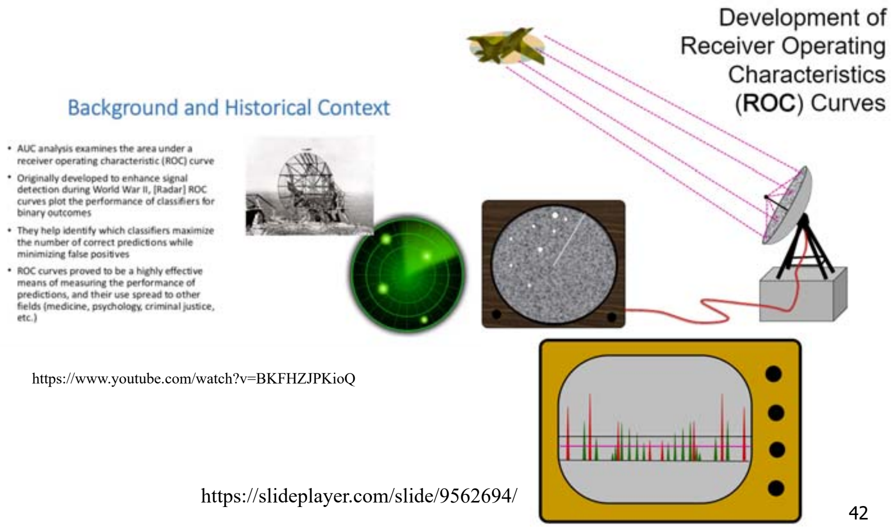
帰無仮説は、「陽性でない」
（陽性であることを示したいから）

第一種の過誤=棄却した(陽性だと言った)が、それは誤り
第二種の過誤=受理した(陰性だと言った)が、それは誤り

ROC curve

- Receiver operating characteristics
  - The ROC curve was first developed by electrical engineers and radar engineers during World War II for detecting enemy objects in battlefields and was soon introduced to psychology to account for perceptual detection of stimuli.
    - https://en.wikipedia.org/wiki/Receiver_operating_characteristic

Radar ROC



https://www.youtube.com/watch?v=BKFHZJPKioQ

https://slideplayer.com/slide/9562694/

ROC curve

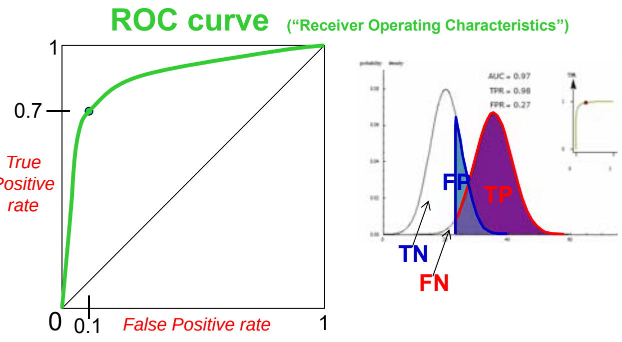
		真値	
		P	N
仮説の予測値	P	TP (True Positive)	FP (False Positive)
	N	FN (False Negative)	TN (True Negative)

True Positive rate $= \dfrac{\text{\# true positives}}{\text{\# true positives + \# false negatives}}$

$$\text{Recall} = \frac{TP}{TP+FN}$$

$$\beta = \text{FNR} = \frac{FN}{FN+TP}$$

1-β=sensitivity



False Positive rate $= \dfrac{\text{\# false positives}}{\text{\# false positives + \# true negatives}}$

$$\alpha = \text{FPR} = \frac{FP}{FP+TN}$$

1-α=specificity

ROC Curves

- Changing the threshold, count the samples
- The larger the area under the curve (AUC), the better
- Suited for comparison of different learning methods

Training vs. generalization error

- Training error: The training error is the mean of errors over the training sample $D = \langle\langle x_1, f(x_1)\rangle, \dots, \langle x_n, f(x_n)\rangle\rangle$.
  - E.g., $err_D(h) = (1/n)\sum_{i=1}^{n}(h(x_i) - f(x_i))^2$
  - Easy to calculate
- Generalization error: Expected prediction error over an independent test sample x:
  - E.g., $Err_D(h) = E_D[(h(x) - f(x))^2]$

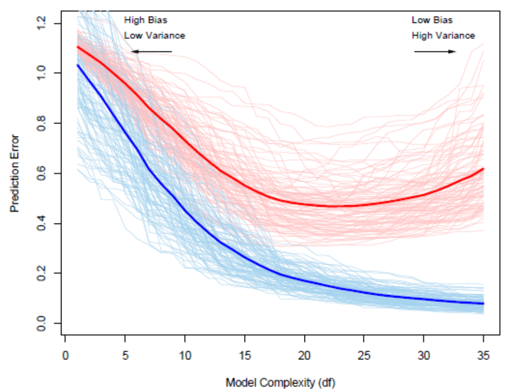
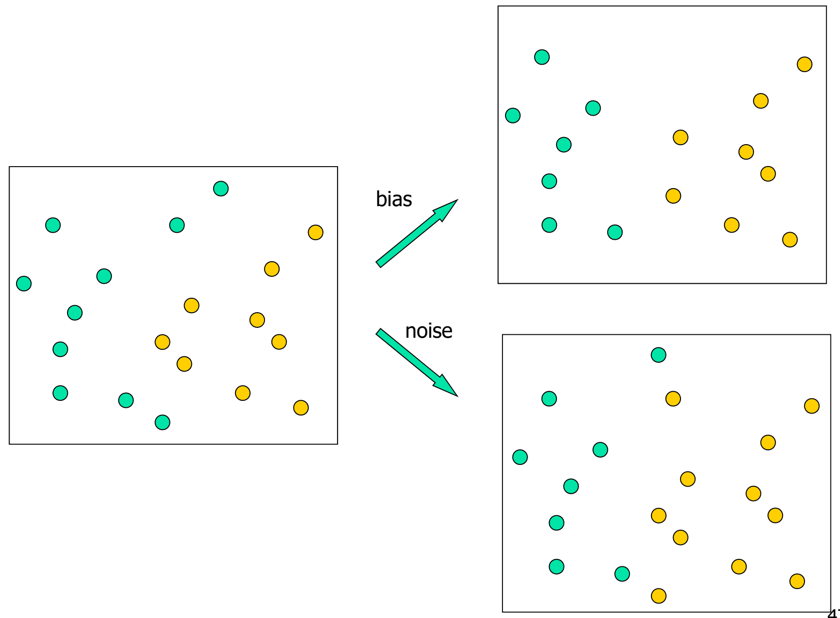


FIGURE 7.1. *Behavior of test sample and training sample error as the model complexity is varied. The light blue curves show the training error* $\overline{\text{err}}$*, while the light red curves show the conditional test error* $\text{Err}_{\mathcal{T}}$ *for 100 training sets of size 50 each, as the model complexity is increased. The solid curves show the expected test error* Err *and the expected training error* $\text{E}[\overline{\text{err}}]$.

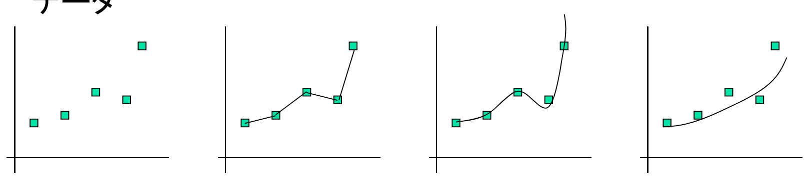
Over-learning/training/fitting

- To learn what should not be learnt
- What should not be learnt
  - Bias existing in the training data
    - Because training data is a finite subset of infinite set (population), the training data has certain bias which is not in the population.
  - Error existing in training data
    - In reality, any training set contains errors in labels
- ML tends to learn as much as it can
  - Because it has very high learning capability
    - E.g., it has a large number of adjustable parameters



Illustrative example

データ

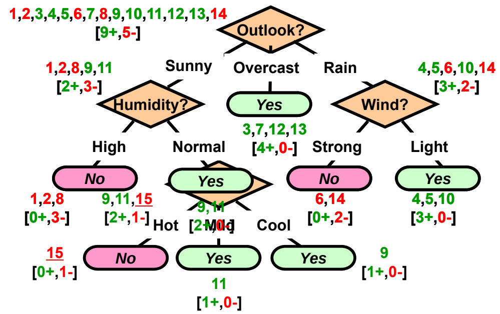


	Piecewise linear	4-th order polynomial	2^{nd} order polynomial
# parameters	2×4+3=11	5	3+degree

多分過学習？　　多分過学習

Overtraining in DT: an example

- Example: induced tree



Decision tree for *PlayTennis*

It may fit to noise or accidental regularity

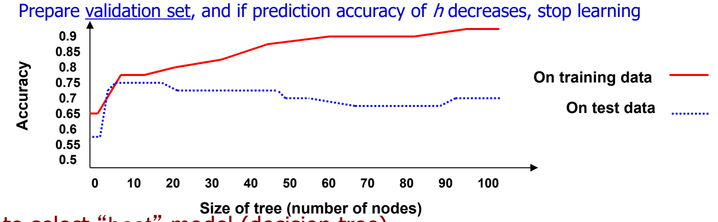
- Suppose that a noisy example exists in the training set
  - sample 15: *<Sunny, Hot, Normal, Strong, ->*
    - This is noisy . The correct label is +.
    - The tree built without it misclassifies this "new" and "noisy" sample.
  - How should the tree be updated (how should incremental learning be implemented)?
  - A new hypothesis $h' = T'$ may deteriorate the performance than $h = T$?

Overlearning in induction

- Definition
  - A hypothsis h over-learns a training set D (overfits to D) iff there exists another hypothesis h' such that $error_D(h) < error_D(h')$ and $error_{test}(h) > error_{test}(h')$ hold.
  - Possible cause: the training set is too small (decision relying on too scarce information); noisy data; simply accidental
- How to avoid?
  - prevention
    - Avoid overlearning before it occurs
    - Select only *important and/or relevant* attributes (i.e., useful for a model)
      - *Caution*: a chicken-and-egg problem; need a measure to predict relevance
  - circumvention
    - When it seems to happen, just go around it
    - Prepare a test set, when a new h behaves worse, stop to learn
  - recovery
    - Wait until it happens, detect it, and recover from it
    - Build a model, and discover and delete elements that cause over-learning (prune)

DT learning: avoiding over-learning

- How to avoid?
  - prevention
    - Select just *relevant* attributes (i.e., relevant to the decision tree)
    - Prediction of *relevance*: try and error, add and delete
  - avoidance
    - Prepare validation set, and if prediction accuracy of h decreases, stop learning

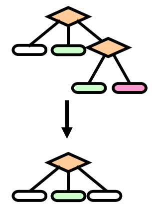


- How to select "best" model (decision tree)
  - Method described above: validation set is mutually exclusive to the learning set
  - Another method: Minimum Description Length (MDL):
    minimize: $size(h \equiv T) + size\,(misclassifications\,(h \equiv T))$

DT learning: avoiding over-learning

- Two basic approaches
  - Pre-pruning (avoidance): Stop growing the tree in the middle, i.e., when there is not enough data to select reliably the next attribute.
  - Post-pruning (recovery): Prune an over-grown tree, i.e., cut down branches whose evidence to exist is not enough
- Evaluation of subtrees to be pruned
  - Cross-validation: divide data exclusively into training and validation dataset, and repeat
  - Statistical test: test if the observed regularity is accidental or not
  - Minimum Description Length (MDL)
    - Increase of the complexity of a hypothesis T is larger/smaller than the complexity to describe exceptions of the data to be explained?
    - Tradeoff: increase of the description of the larger model versus that of increased residuals

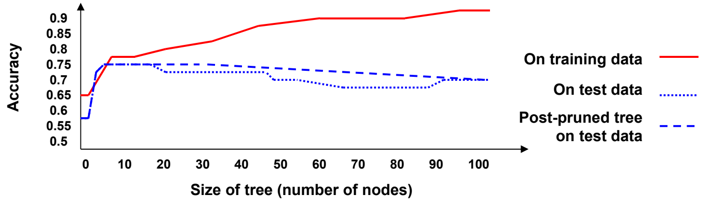
Reduced-Error Pruning



- Post-Pruning, Cross-Validation Approach
- Divide the data given into training set and validation set
- Function *Prune*(T, *node*)
  - Subtree rooted with *node* is pruned
  - Build a leaf with *node* . (its label is the majority label)
- Algorithm *Reduced-Error-Pruning* (D)
  - Divide D into D_{train} (training / "growing"), $D_{validation}$ (validation / "pruning")
  - Apply ID3 to D_{train} to build a complete tree T
  - UNTIL accuracy measured with $D_{validation}$ decreases DO
    FOR an internal node *candidate* in T
      Temp[candidate] ← Prune (T, candidate)
      Accuracy[candidate] ← Test (Temp[candidate], $D_{validation}$)
    $T \leftarrow T' \in$ *Temp* 中で *Accuracy* が最良のもの
  - RETURN (pruned) T

Effects of Reduced-Error Pruning

- Decrease of test errors by Reduced-Error Pruning



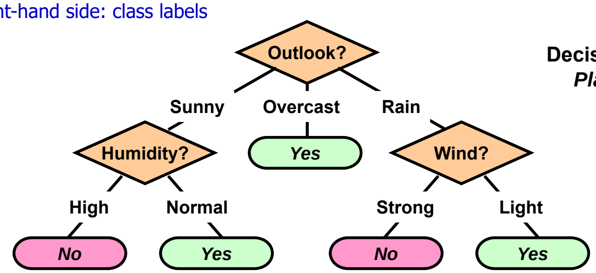
  - Pruning of nodes decrease the test error
  - *Note*: $D_{validation}$ is different from D_{train} and D_{test}
- Pros and cons
  - Pros: The smallest among the most accurate T' (a subtree of T) is obtainable
  - Cons: smaller dataset is used to build T, in case the data is scarce.
    - Could we afford $D_{validation}$?
    - If allowable data is not enough (D_{train} is not large enough), the pruning make error larger

Rule Post-Pruning

- Often used
  - Well-known countermeasure for overfitting
  - *C4.5* uses a derived version. *C4.5* is a successor of *ID3*.
- Algorithm *Rule-Post-Pruning* (D)
  - Build T from D (by *ID3*) – to adapt to D as close as possible (over-learning is allowed)
  - Convert T to an equivalent rule set (a rule is for a path from the root to a leaf)
  - Delete, independently, tests (conditions) as many as possible while estimated accuracy increases
  - Sort the pruned rules
    - Sort them according to estimated accuracy
    - In rows, apply them to D_{test}

Convert a tree to a rule set

- Syntax of the rule set
  - Left-hand side: conditions (equality tests on attributes form conjunctive formula)
  - Right-hand side: class labels

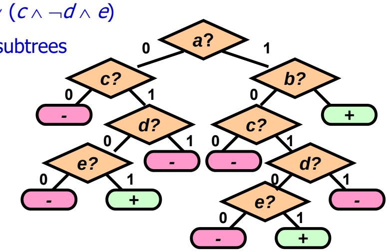


Decision tree for *PlayTennis*

- Example
  - IF (*Outlook* = *Sunny*) ∧ (*Humidity* = *High*) THEN *PlayTennis* = *No*
  - IF (*Outlook* = *Sunny*) ∧ (*Humidity* = *Normal*) THEN *PlayTennis* = *Yes*
  - …

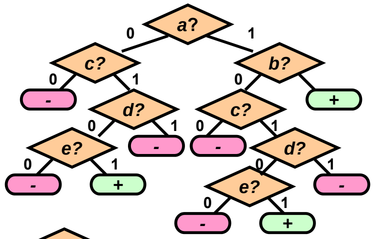
Replications in decision tree

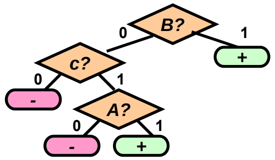
- In decision tree: a shortcoming in representation
  - Decision tree is not the simplest representation method
  - point: replication of attributes is necessary
- Example of attribute replication
  - e.g., Disjunctive Normal Form (DNF): $(a \wedge b) \vee (c \wedge \neg d \wedge e)$
  - (one of) conjunctions should be replicated as subtrees
- Partial solutions
  - Form a a new attribute
  - Alias: constructive induction (CI)
  - Ref. Chap. 10, T. Mitchell

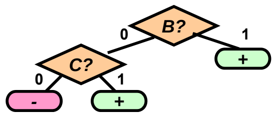


A bit of constructive induction

- Synthesize a new attribute
  - Synthesize a new attribute from the conjunction of the two attributes just before a “+ leaf”
  - Also called feature construction
- Example
  - $(a \wedge b) \vee (c \wedge \neg d \wedge e)$
  - $A = \neg d \wedge e$
  - $B = a \wedge b$
- When repeated
  - $C = A \wedge c$
  - Correctness?
  - Time complexity?







Decision tree: other topics

- Topics which common to other machine leaning methods

Continuous attributes

- Two methods to deal with continuous attributes
  - Discretization
    - Divide attribute values into ranges
    - e.g., {*high* ≡ *Temp* > 35° C, *med* ≡ 10° C < *Temp* ≤ 35° C, *low* ≡ *Temp* ≤ 10° C}
  - On internal nodes, the thresholds are used for attribute tests
    - e.g., $A \leq a$ makes two subsets $A \leq a$ and $A > a$
    - Information gain is calculated, too.
- How to find the partition that maximizes information gain
  - FOR each continuous attribute A

    Split samples $\{x \in D\}$ according to values $x.A$

    FOR each ordered pair (l, u) of values of A, which has different labels

    evaluate information gain of partition by mid-point, i.e., $D_{A \leq (l+u)/2}$, $D_{A > (l+u)/2}$
  - Example

$A \equiv$ *Length*:	10	15	21	28	32	40	50
Class:	-	+	+	-	+	+	-
Threshold candidates: *Length* ≤ 12.5?		≤ 24.5?		≤ 30?		≤ 45?	

Issues on multiple-valued attributes

- 問題
  - An attribute with multiple values is tend to be preferred by *Gain*(•)
  - E.g., image date (2019/10/10 etc.) is used as an attribute
- One approach: *GainRatio* to replze *Gain*

$$\mathbf{Gain}(D, A) \equiv H(D) - \sum_{v \in values(A)} \left[\frac{|D_v|}{|D|} \bullet H(D_v) \right]$$

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

$$\mathbf{SplitInformation}(D, A) \equiv -\sum_{v \in values(A)} \left[\frac{|D_v|}{|D|} \log \frac{|D_v|}{|D|} \right]$$

- Almost proportional to *SplitInformation*: $c = |\ values(A)\ |$
- i.e., handicapped to attribute with many values
  - e.g., example: $c_1 = c_{Date} = n$ and $c_2 = 2$
  - *SplitInformation* $(A_1) = \log(n)$, *SplitInformation* $(A_2) = 1$
  - When $Gain(D, A_1) = Gain(D, A_2)$, $GainRatio(D, A_1) << GainRatio(D, A_2)$
- i.e., *GainRatio*(•) can be used to express selection bias (to a smaller splits)

Note: Gini index

Not "Gini coefficient"

- Another partition index
  - n is the number of classes
  - Gini(D) becomes smaller when the distribution in D become more biased, i.e., inpure.

$$Gini(D) = \sum_{i \neq j} p_i p_j = 1 - \sum_{i=1}^{n} p_i^2$$

$$GiniGain(D, A) = Gini(D) - \sum_{v \in values(A)} \left[\frac{|D_v|}{|D|} \cdot Gini(D_v) \right]$$

Attributes with weights

- Weights varies in applications
  - Medical: Temperature costs 1000JPY; BloodTest 1500JPY; Biopsy 50000JPY
    - Need to consider invasiveness
    - Risk to patient (e.g., Amniocentesis)
  - Other cost
    - Sampling time: e.g., Robot sonar (range finder, etc.)
    - Risk to artifacts, organisms (what kind of information is to be gathered)
    - Related fields (e.g., tomography): noninvasive test
- How to build a consistent tree with low expected cost
  - One approach: replace gain with *Cost-Normalized-Gain*
  - Example of cost-normalization
    - [Nunez, 1988]:

$$Cost\text{-}Normalized\text{-}Gain(D, A) \equiv \frac{Gain^2(D, A)}{Cost(D, A)}$$

    - [Tan and Schlimmer, 1990]:

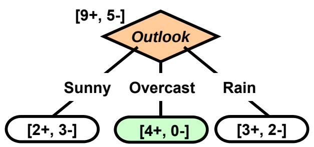
$$Cost\text{-}Normalized\text{-}Gain(D, A) \equiv \frac{2^{Gain(D,A)} - 1}{(Cost(D, A) + 1)^w} \quad w \in [0,1]$$

where w defines importance of cost

Missing value

- Problem: what if attribute *A* has no value?
  - Often, during training or test, not all values are obtained
  - Example: medical diagnosis
    - *<Fever = true, Blood-Pressure = normal, …, Blood-Test = ?, …>*
    - Value is missing or with low reliability
  - Missing value: at training versus at test
    - training: calculate *Gain* (*D*, *A*) when for some $x \in D$, the value of *A* is not givem
    - test: without knowing the value of *A* , classify a new sample
- Solution: including prediction to calculate *Gain*(*D*, *A*)

Day	Outlook	Temperature	Humidity	Wind	*PlayTennis?*
1	Sunny	Hot	High	Light	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Light	Yes
4	Rain	Mild	High	Light	Yes
5	Rain	Cool	Normal	Light	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	???	Light	No
9	Sunny	Cool	Normal	Light	Yes
10	Rain	Mild	Normal	Light	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Light	Yes
14	Rain	Mild	High	Strong	No



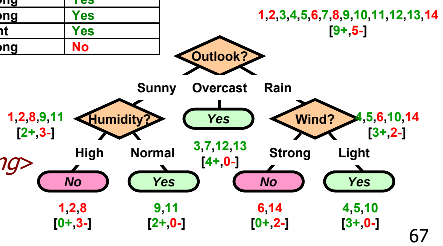
Missing values: approach

- Use training examples when going down from root to leaf
  - For every attribute to be considered, if its value is missing, make a prediction
  - The prediction is done based on the samples assigned to the node
- Predict the most likely value of *x.A*
  - Policy 1: Suppose node *n* tests attribute *A*, take majority of values of A which go through *n*
  - Policy 2 [Mingers, 1989]: Suppose at *n* , *A* is tested, take majority of values of *A*, which go through *n* with the same label as *x* .
- Disperse predictions
  - Hedging: distribute predictions according to the distribution of values
  - Assign probability p_i to possible values v_i of *x.A* [Quinlan, 1993]
    - Assign fraction p_i of *x* to each descendent of the node
    - Calculate *Gain* (*D*, *A*) or *Cost-Normalized-Gain* (*D*, *A*) based on these values.
- In any of these approaches, a new sample is classified in this way

Missing values: an example

- Predict most probable value of *x.A*
  - Plan 1: *Humidity = Normal*
  - Plan 2: *Humidity = High* (*No* examples are all *High*)
  - (Which gives the largest Gain ? *High*: *Gain* = 0.97, *Normal*: *Gain* < 0.97)

Day	Outlook	Temperature	Humidity	Wind	*PlayTennis?*
1	Sunny	Hot	High	Light	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Light	Yes
4	Rain	Mild	High	Light	Yes
5	Rain	Cool	Normal	Light	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	???	Light	No
9	Sunny	Cool	Normal	Light	Yes
10	Rain	Mild	Normal	Light	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Light	Yes
14	Rain	Mild	High	Strong	No

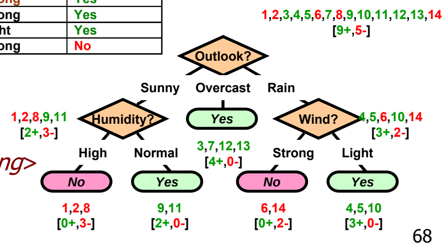


- Weigh with probability
  - *0.5 High, 0.5 Normal*
  - *Gain* < 0.97
- A test sample: *<?, Hot, Normal, Strong>*
  - 5/14 *Yes* + 4/14 *Yes* + 5/14 *No* = *Yes*

Missing values: an example

- Predict most probable value of *x.A*
  - Plan 1: *Humidity = Normal*
  - Plan 2: *Humidity = High* (*No* examples are all *High*)
  - (Which gives the largest Gain ? *High*: *Gain* = 0.97, *Normal*: *Gain* < 0.97)

Day	Outlook	Temperature	Humidity	Wind	*PlayTennis?*
1	Sunny	Hot	High	Light	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Light	Yes
4	Rain	Mild	High	Light	Yes
5	Rain	Cool	Normal	Light	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	???	Light	No
9	Sunny	Cool	Normal	Light	Yes
10	Rain	Mild	Normal	Light	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Light	Yes
14	Rain	Mild	High	Strong	No



- Weigh with probability
  - *0.5 High, 0.5 Normal*
  - *Gain* < 0.97
- A test sample: *<?, Hot, Normal, Strong>*
  - 1/3 *Yes* + 1/3 *Yes* + 1/3 *No* = *Yes*
  - 5/14 *Yes* + 4/14 *Yes* + 5/14 *No* = *Yes*

What is learning?

Induction

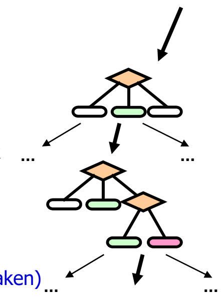
- According to OED (Oxford English Dictionary)
  - the process of inferring a general law or printciple from the observations of particular instances
  - The above one refers to inductive inference
  - inductive reasoning is: the process of reassigning a probability (or credibility) to a law or proposition from the observation of particular events

Induction

- Induction is:
  to obtain regularity implicit in data, e.g.,
  - experimental observations of free falling objects
    → Newton's law of universal gravitation
    (well, not really. The invention is combination and elaboration of Kepler's theory and others)
  - Tycho Brahe's observations
    → Johannes Kepler's laws
- How should we measure the correctness of inductions?

Search in hypothesis space by ID3

- Searching Problem
  - Target of the search is all the decision trees, which can represent any Boolean functions.
    - Pros: expressiveness; flexibility
    - Cons: heavy computation; huge, include (many) incomprehensible trees
  - Objective: to find the best one (minimal and consistent tree)
  - Obstacle: to find out this tree is NP-hard
  - Tradeoff
    - Use of heuristics (a guide of search to tell us which is the first to see)
    - Use of greedy algorithm
    - That is, hill-climbing (gradient "ascent" but discrete) without backtrack
- Statistical Learning
  - Decisions are made based on statistics p_+, p_- of D_v
  - In ID3, all the data is used
  - Robust to noisy data if some measures (simple and easy to use are taken)



Inductive bias in ID3

- Heuristics in search is an inductive bias
  - Suppose H is a power set of X (set of all the subesets)
  - ⇒ No inductive bias? There is!…
    - Preference to shorter trees (termination condition)
    - Preference to put attributes with high information gain close to the root
    - *Gain*(•): a heuristics that represent the bias of *ID3*
  - Inductive bias of *ID3*
    - Preference to some hypothesis are expressed by a heuristic function
    - Another type: restrict hypothesis space H (e.g., normal form of propositional logic: k-CNF, etc.)
- Preference for shorter trees
  - Selection of shortest tree among the ones consistent with data
  - Occam's razor bias: Entities should not be multiplied without necessity

Assumption in learning phase to suggest which hypothesis to be chosen, prioritized, or discarded. Without this, no meaningful result is obtained, because there are infinite hypotheses.

OK, but isn't it unreasonable to adopt an assumption that is not supported by data

If a bias is inevitable, which bias is better?

Learning and bias

- Bias: some order among hypotheses
  - Preference: plural of hypothesis at the same time
  - Searching: when to evaluate one by one
- Hypotheses consistent with data are, in general, many. Hence we need a bias.
  - Unless we use all the hypotheses, we need bias to select (and not to select) hypothesis

learning:

Data → hypothesis

OK, but isn't it unreasonable to adopt an assumption that is not supported by data

If a bias is inevitable, which bias is better?

Occam's razor

- How people generally say
  - Entities should not be multiplied beyond necessity.
- By Bertrand Russell
  - It is vain to do with more what can be done with fewer.
- Most common interpretation
  - Among the theories that are consistent with the observed phenomena, one should select the simplest theory.

What Isaac Newton said

- We are to admit no more causes of natural things than such as are both true and sufficient to explain the appearances. To this purpose the philosophers say that Nature does nothing in vain, and more is in vain when less will serve; for Nature is pleased with simplicity, and affects not the pomp of superfluous causes.

Occam's razor: a preference bias

- Two types of biases: preference biases and language biases
  - preference bias
    - It is implicitly incorporated in learning algorithms
    - In other words: implying searching order
  - language bias
    - It is implicitly incorporated in representation of *knowledge (hypothesis)*
    - In other words: restriction of searching space
    - Alias: restriction bias
- Occam's Razor: pros
  - Shorter hypothesis are fewer than longer hypothesis
    - E.g., bit sequences of length n are half of those of length n+1 where $n \geq 0$.
    - If a short hypothesis fits the data well, it may not be accidental.
      - Short hypothesis are scarce, phenomena explainable by them are scarce
    - If a long hypothesis fits the data well, it may possibly be accidental (Ex.: DT with 200 nodes for $|D| = 100$)
      - If long enough, one of the hypothesis fits to data surely, but which one fits is probabilistic and rare.
  - Obtained and discarded
    - Other things being equal, complex model cannot generalize as well as simple one.
    - Assuming that later more flexibility to data will not required

Occam's razor: two problems

- Occam's Razor: cons
  - Usually *size(h)* depends on *H*. For the same *h*, *size(h)* differs when *H* differs. Reaonable?
  - Is "fewer" a justification for the preference to smaller?
- Is Occam's Razor Well-Defined?
  - (Internal) knowledge representation defines which *h* is short --- arbitrary
    - A test "($Sunny \wedge Normal\text{-}Humidity$) $\vee$ $Overcast \vee$ ($Rain \wedge Light\text{-}Wind$)" is length 1 or not?
  - One answer: fix a language; At long enough side, long hypothesis is long.
    - Rebuttal: we are discussing "short" hypothesis, not "long" ones
- Why not small hypothesis space but short hypothesis?
  - Because if *H* is small, we still argue in the same way as H is "large."
  - Note first that we are thinking about infinite space.
    - If *H* is finite and practically small, its usefulness is very limited. If it is finite, we suppose it is huge.
  - Note also that any hypothesis is finite length and the hypotheses with the same *size(h)* is finite.
    - If we enumerate "small" set and "large" set in ascending order of length in parallel, we could see intuitively, that the hypothesis are not so much different.

Principle of plenitude

- Epicurus and others
  - If more than one theory is consistent with the observations, keep all theories. [M. Hutter, Universal Artificial Intelligence, 2005」

- One reason: there is no specific reason we choose one from the others

- (your personal exercise) Compare it with Bayesian approach