innovate achieve lead

Machine Learning
DSECL ZG565

Dr. Chetana Gavankar, Ph.D,
IIT Bombay-Monash University Australia
Chetana.gavankar@pilani.bits-pilani.ac.in

BITS Pilani
Pilani Campus

innovate achieve lead

BITS Pilani

Pilani Campus

Lecture No. – 7 | Decision Tree and Random Forest

Date – 01/12/2019

Time – 2:00 PM – 4:00 PM

Session Content

(Tom Mitchell Chapter 3 page 67)

- Decision Tree
- Handling overfitting
- Continuous values
- Missing Values
- Random Forest

Decision trees

- Decision Trees is one of the most widely used and practical methods of classification
- Method for approximating discrete-valued functions
- Learned functions are represented as decision trees (or if-then-else rules)
- Expressive hypotheses space

Decision Tree

- Advantages:

– Inexpensive to construct

– Extremely fast at classifying unknown records

– Easy to interpret for small-sized trees

– Can easily handle redundant or irrelevant attributes (unless the attributes are interacting)

- Disadvantages:

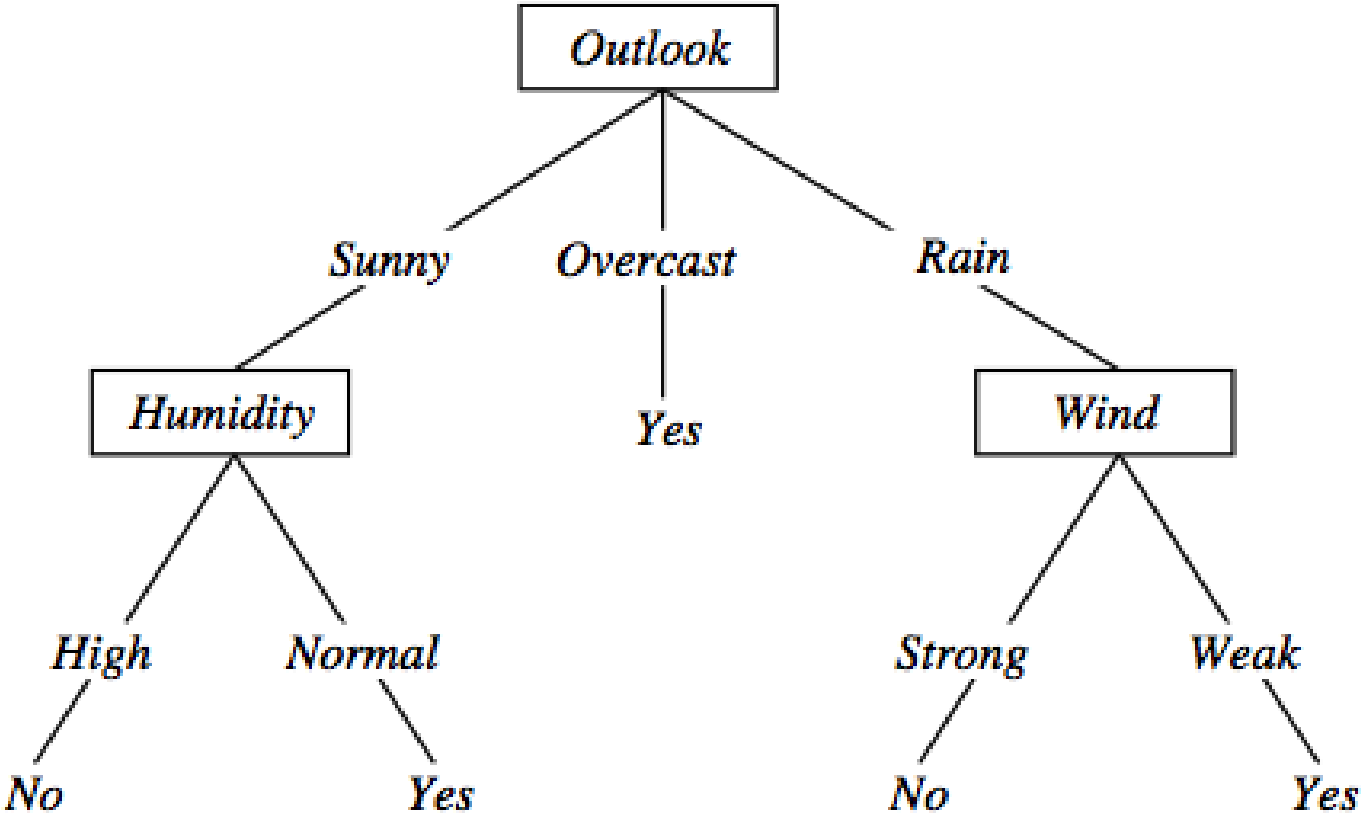
– Space of possible decision trees is exponentially large.

Greedy approaches are often unable to find the best tree.

– Does not take into account interactions between attributes

– Each decision boundary involves only a single attribute

Decision tree representation (PlayTennis)

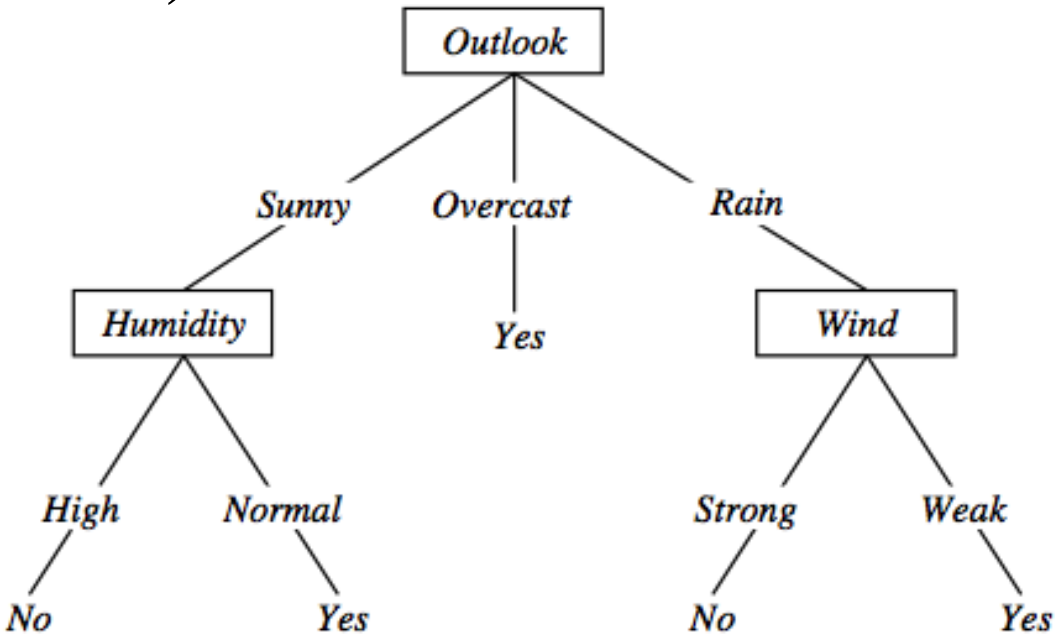


⟨*Outlook=Sunny, Temp=Hot, Humidity=High, Wind=Strong*⟩ No

Decision trees expressivity

- Decision trees represent a disjunction of conjunctions on constraints on the value of attributes:

$$(Outlook = Sunny \wedge Humidity = Normal) \vee$$
$$(Outlook = Overcast) \vee$$
$$(Outlook = Rain \wedge Wind = Weak)$$



Measure of Information

- The amount of information (surprise element) conveyed by a message is inversely proportional to its probability of occurrence. That is

$$I_k \alpha \frac{1}{p_k}$$

- The mathematical operator satisfies above properties is the logarithmic operator.

$$I_k = \log_r \frac{1}{p_k} \; units$$

Entropy

- Entropy of discrete random variable X={x_1, x_2…x_n}

$$\mathrm{H}(X) = \mathrm{E}[\mathrm{I}(X)] = \mathrm{E}[-\log(\mathrm{P}(X))].$$

; since: $\log_2(1/P(event)) = -\log_2 P(event)$

- As uncertainty increases, entropy increases
- Entropy across all values

$$\mathrm{H}(X) = -\sum_{i=1}^{n} \mathrm{P}(x_i) \log_b \mathrm{P}(x_i)$$

Entropy in general

- Entropy measures the amount of information in a random variable

$$H(X) = -p_+ \log_2 p_+ - p_- \log_2 p_- \qquad X = \{+, -\}$$

for binary classification [two-valued random variable]

$$H(X) = -\sum_{i=1}^{c} p_i \log_2 p_i = \sum_{i=1}^{c} p_i \log_2 1/p_i \qquad X = \{i, \ldots, c\}$$

for classification in *c* classes

Entropy in binary classification

- Entropy measures the *impurity* of a collection of examples. It depends from the distribution of the random variable *p.*
  - S is a collection of training examples
  - p_+ the proportion of positive examples in S
  - p_- the proportion of negative examples in S

$$Entropy\,(S) \equiv -p_+ \log_2 p_+ - p_- \log_2 p_- \quad [0 \log_2 0 = 0]$$

$$Entropy\,([14+, 0-]) = -14/14 \log_2 (14/14) - 0 \log_2 (0) = 0$$

$$Entropy\,([9+, 5-]) = -9/14 \log_2 (9/14) - 5/14 \log_2 (5/14) = 0.94$$

$$Entropy\,([7+, 7-]) = -7/14 \log_2 (7/14) - 7/14 \log_2 (7/14) =$$

$$= 1/2 + 1/2 = 1 \qquad [\log_2 1/2 = -1]$$

Note: the log of a number < 1 is negative, $0 \le p \le 1$, $0 \le entropy \le 1$

- https://www.easycalculation.com/log-base2-calculator.php

Information gain as entropy reduction

- *Information gain* is the *expected* reduction in entropy caused by partitioning the examples on an attribute.
- The higher the information gain the more effective the attribute in classifying training data.
- Expected reduction in entropy knowing *A*

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|Sv|}{|S|} Entropy(S_v)$$

$Values(A)$ possible values for A
S_v subset of S for which A has value v

Example

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

Example: Information gain

- Let
  - $Values(Wind) = \{Weak, Strong\}$
  - $S = [9+, 5-]$
  - $S_{Weak} = [6+, 2-]$
  - $S_{Strong} = [3+, 3-]$
- Information gain due to knowing *Wind*:

$$Gain(S, Wind) = Entropy(S) - 8/14\ Entropy(S_{Weak}) - 6/14\ Entropy(S_{Strong})$$

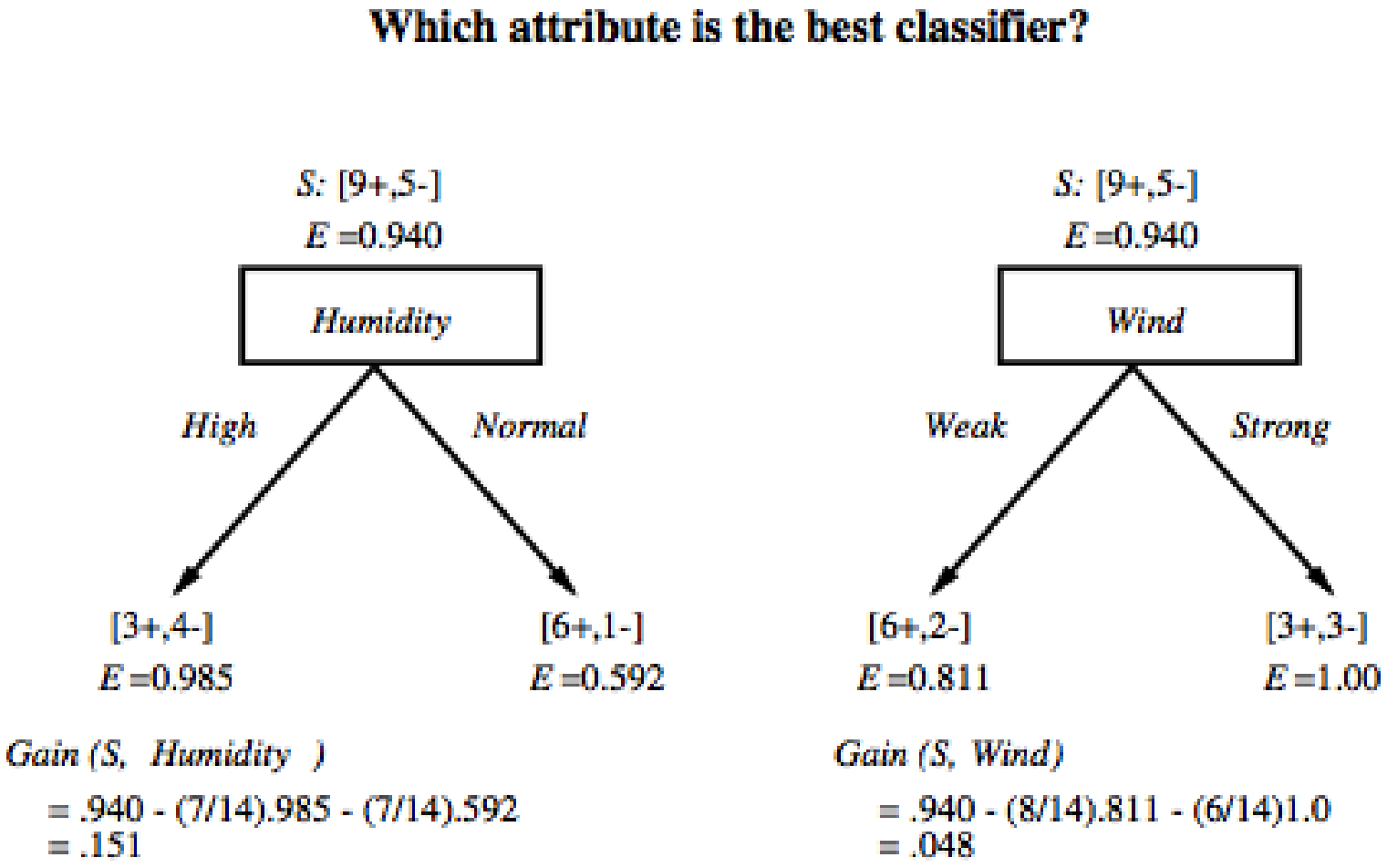
$$= 0.94 - 8/14 \times 0.811 - 6/14 \times 1.00$$

$$= 0.048$$

Example

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

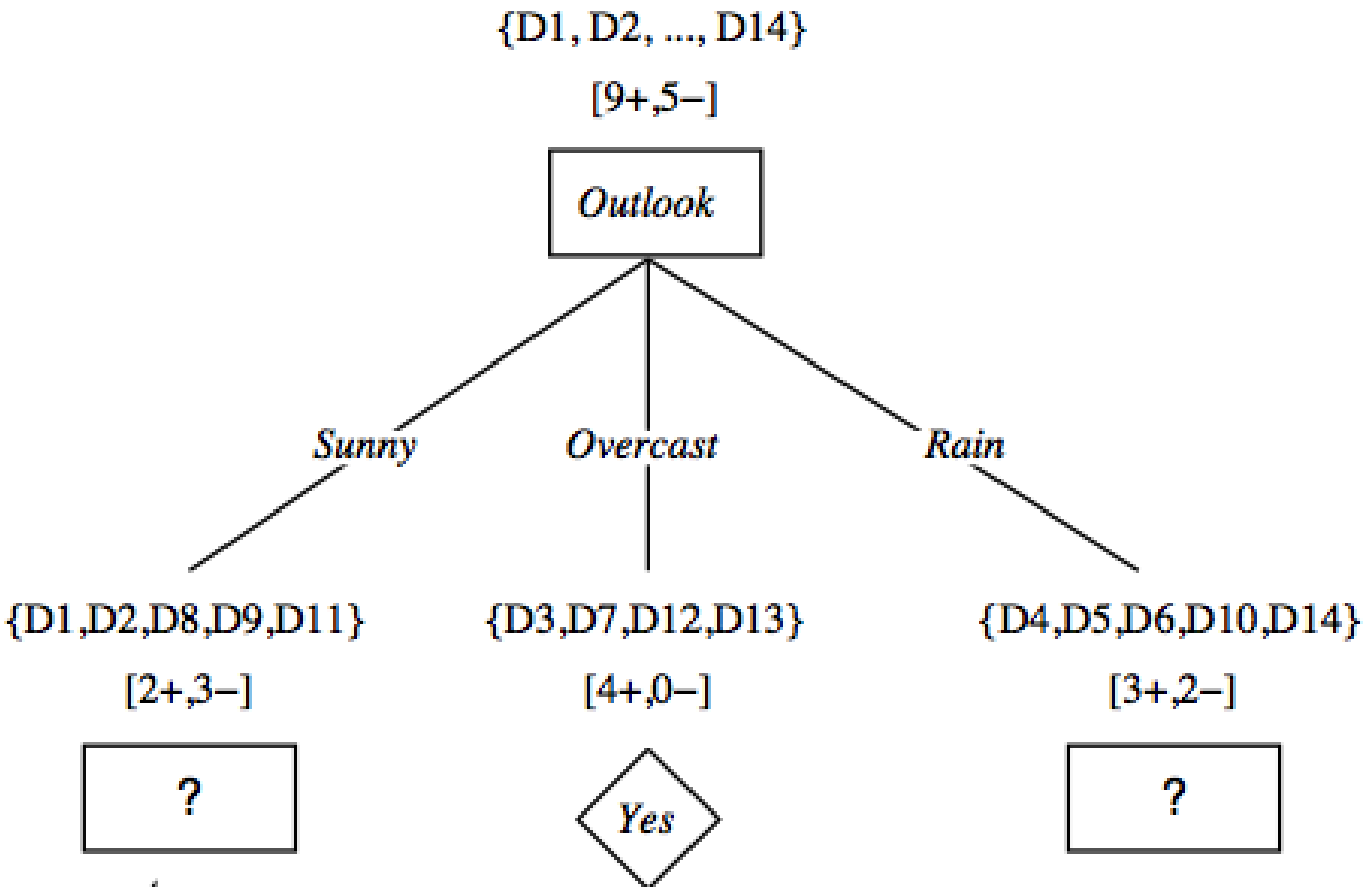
Which attribute is the best classifier?



First step: which attribute to test at the root?

- Which attribute should be tested at the root?
  - $Gain(S, Outlook) = 0.246$
  - $Gain(S, Humidity) = 0.151$
  - $Gain(S, Wind) = 0.084$
  - $Gain(S, Temperature) = 0.029$
- *Outlook* provides the best prediction for the target
- Lets grow the tree:
  - add to the tree a successor for each possible value of *Outlook*
  - partition the training samples according to the value of *Outlook*

After first step



Second step

- Working on *Outlook=Sunny* node:

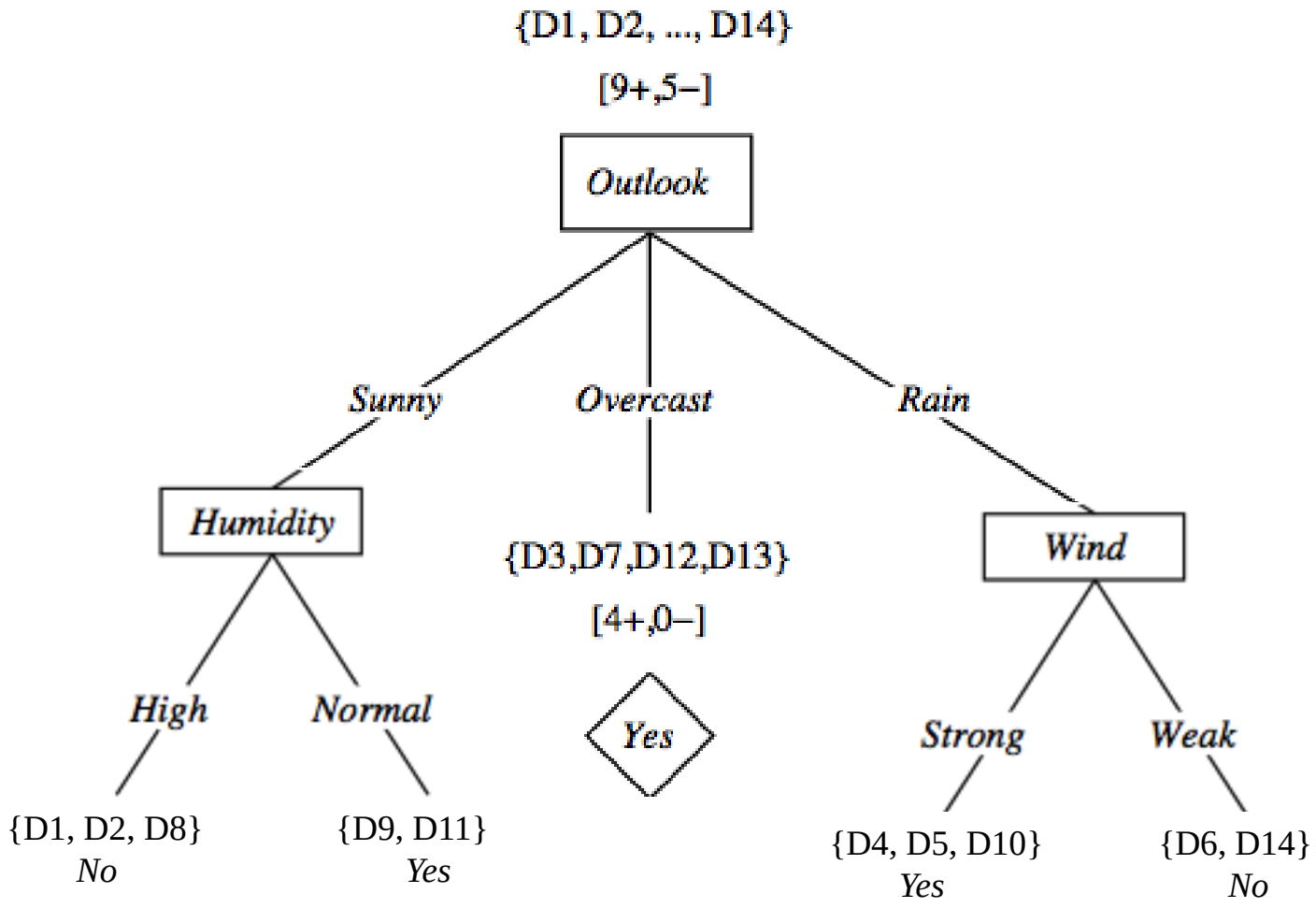
  $Gain(S_{Sunny}, Humidity) = 0.970 - 3/5 \times 0.0 - 2/5 \times 0.0 = 0.970$

  $Gain(S_{Sunny}, Wind) = 0.970 - 2/5 \times 1.0 - 3.5 \times 0.918 = 0\ .019$

  $Gain(S_{Sunny}, Temp.) = 0.970 - 2/5 \times 0.0 - 2/5 \times 1.0 - 1/5 \times 0.0 = 0.570$

- *Humidity* provides the best prediction for the target
- Lets grow the tree:
  - add to the tree a successor for each possible value of *Humidity*
  - partition the training samples according to the value of *Humidity*

Second and third steps



ID3: algorithm

ID3(*X, T, Attrs*) *X*: training examples:
T: target attribute (e.g. $PlayTennis$),
Attrs: other attributes, initially all attributes

Create *Root* node
If all X's are +, *return Root* with class +
If all X's are –, *return Root* with class –
If Attrs is empty *return Root* with class most common value of *T* in *X*
else
$A \leftarrow$ best attribute; decision attribute for Root $\leftarrow A$
For each possible value v_i of *A:*
- add a new branch below *Root,* for test $A = v_i$
- $X_i \leftarrow$ subset of *X* with $A = v_i$
- *If* X_i is empty *then* add a new leaf with class the most common value of *T* in *X*
else add the subtree generated by ID3(X_i, *T, Attrs* – {*A*})

return Root

Prefer shorter hypotheses: Occam's razor

- Why prefer shorter hypotheses?
- Arguments in favor:
  - There are fewer short hypotheses than long ones
  - If a short hypothesis fits data unlikely to be a coincidence
  - Elegance and aesthetics
- Arguments against:
  - Not every short hypothesis is a reasonable one.
- Occam's razor says that when presented with competing hypotheses that make the **same** predictions, one should select the solution which is simple"

Issues in decision trees learning

- Overfitting
  - Reduced error pruning
  - Rule post-pruning
- Extensions
  - Continuous valued attributes
  - Handling training examples with missing attribute values

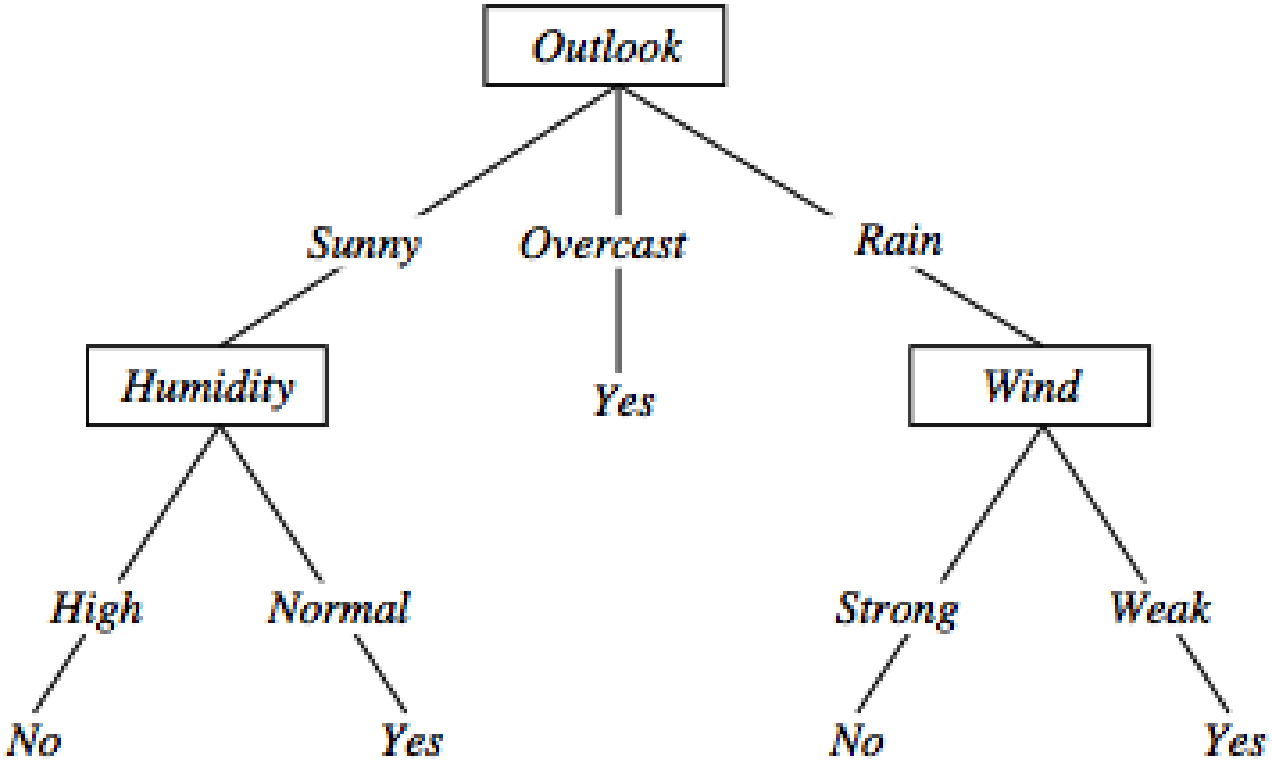
Overfitting: definition

- **overfitting** is "the production of an analysis that corresponds too closely or exactly to a particular set of data
- Building trees that “adapt too much” to the training examples may lead to “overfitting”.
- May therefore fail to fit additional data or predict future observations reliably
- **overfitted model** is a statistical model that contains more parameters than can be justified by the data

Example

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
D15	Sunny	Hot	Normal	Strong	No

Overfitting in decision trees



$\langle Outlook{=}Sunny, Temp{=}Hot, Humidity{=}Normal, Wind{=}Strong, PlayTennis{=}No \rangle$

New noisy example causes splitting of second leaf node.

Avoid overfitting in Decision Trees

- Two strategies:
  1. Stop growing the tree earlier the tree, before perfect classification
  2. Allow the tree to *overfit* the data, and then *post-prune* the tree
- Training and validation set
  - split the training in two parts (training and validation) and use validation to assess the utility of *post-pruning*
    - *Reduced error pruning*
    - *Rule post pruning*

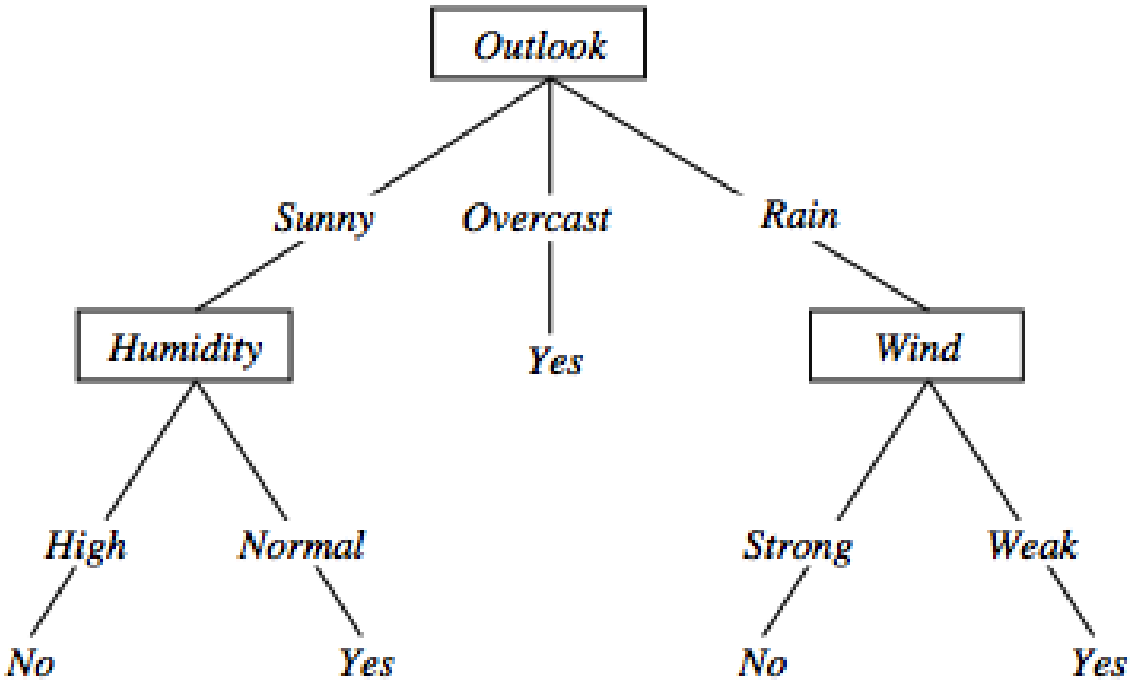
Reduced-error pruning

- Each node is a candidate for pruning
- *Pruning* consists in removing a subtree rooted in a node: the node becomes a leaf and is assigned the most common classification
- Nodes are removed only if the resulting tree performs no worse on the validation set.
- Nodes are pruned iteratively: at each iteration the node whose removal most increases accuracy on the validation set is pruned.
- Pruning stops when no pruning increases accuracy

Rule post-pruning

1. Create the decision tree from the training set
2. Convert the tree into an equivalent set of rules
   - Each path corresponds to a rule
   - Each node along a path corresponds to a pre-condition
   - Each leaf classification to the post-condition
3. Prune (generalize) each rule by removing those preconditions whose removal improves accuracy …
   - … over validation set
4. Sort the rules in estimated order of accuracy, and consider them in sequence when classifying new instances

Converting to rules



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

Rule Post-Pruning

- Convert tree to rules (one for each path from root to a leaf)
- For each antecedent in a rule, remove it if error rate on validation set does not decrease
- Sort final rule set by accuracy

```
Outlook=sunny ^ humidity=high -> No
Outlook=sunny ^ humidity=normal -> Yes
Outlook=overcast -> Yes
Outlook=rain ^ wind=strong -> No
Outlook=rain ^ wind=weak -> Yes
```

Compare first rule to:

```
Outlook=sunny->No
Humidity=high->No
```

Calculate accuracy of 3 rules based on validation set and pick best version.

Why converting to rules?

- Each distinct path produces a different rule: a condition removal may be based on a local (contextual) criterion. Node pruning is global and affects all the rules
- Provides flexibility of not removing entire node
- In rule form, tests are not ordered and there is no book-keeping involved when conditions (nodes) are removed
- Converting to rules improves readability for humans

Dealing with continuous-valued attributes

- Given a continuous-valued attribute A, dynamically create a new attribute A_c

$$A_c = \text{True } if\ A < c, \text{ False } otherwise$$

- How to determine threshold value c ?
- Example. *Temperature* in the *PlayTennis* example
  - Sort the examples according to *Temperature*

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

  - Determine candidate thresholds by averaging consecutive values where there is a change in classification: (48+60)/2=54 and (80+90)/2=85

Problems with information gain

- Natural bias of information gain: it favors attributes with many possible values.
- Consider the attribute *Date* in the *PlayTennis* example.
  - *Date* would have the highest information gain since it perfectly separates the training data.
  - It would be selected at the root resulting in a very broad tree
  - Very good on the training, this tree would perform poorly in predicting unknown instances. Overfitting.
- The problem is that the partition is too specific, too many small classes are generated.
- We need to look at alternative measures …

An alternative measure: gain ratio

$$SplitInformation(S, A) \equiv -\sum_{i=1}^{c} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}$$

- S_i are the sets obtained by partitioning on value i of A
- *SplitInformation* measures the entropy of S with respect to the values of A. The more uniformly dispersed the data the higher it is.

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

- *GainRatio* penalizes attributes that split examples in many small classes such as *Date*. Let $|S|=n$, *Date* splits examples in n classes
  - $SplitInformation(S, Date) = -[(1/n \log_2 1/n) + \ldots + (1/n \log_2 1/n)] = -\log_2 1/n = \log_2 n$
- Compare with A, which splits data in two even classes:
  - $SplitInformation(S, A) = -[(1/2 \log_2 1/2) + (1/2 \log_2 1/2)] = -[-1/2 - 1/2] = 1$

Handling missing values training data

- How to cope with the problem that the value of some attribute may be missing?
- The strategy: use other examples to guess attribute
  1. Assign the value that is most common among the training examples at the node
  2. Assign a probability to each value, based on frequencies, and assign values to missing attribute, according to this probability distribution

Applications

Suited for following classification problems:

- Applications whose Instances are represented by attribute-value pairs.
- The target function has discrete output values
- Disjunctive descriptions may be required
- The training data may contain missing attribute values

Real world applications

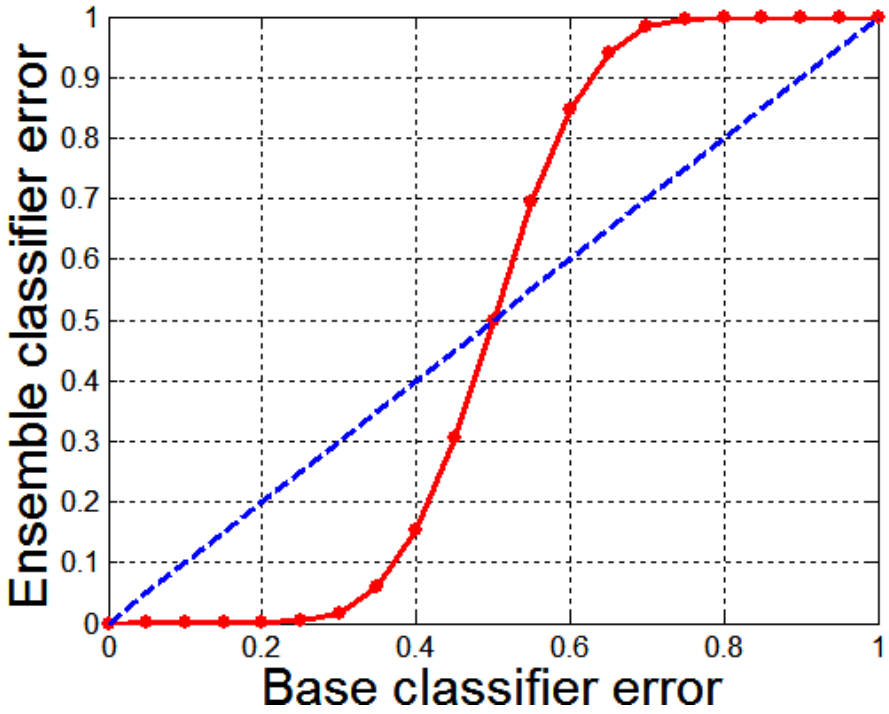
- Biomedical applications
- Manufacturing
- Banking sector
- Make-Buy decisions

Ensemble Methods

- **Ensemble methods** use multiple learning algorithms to obtain better predictive performance than could be obtained from any of the constituent learning algorithms alone
- Construct a set of classifiers from the training data
- Predict class label of test records by combining the predictions made by multiple classifiers
- Tend to reduce problems related to over-fitting of the training data.

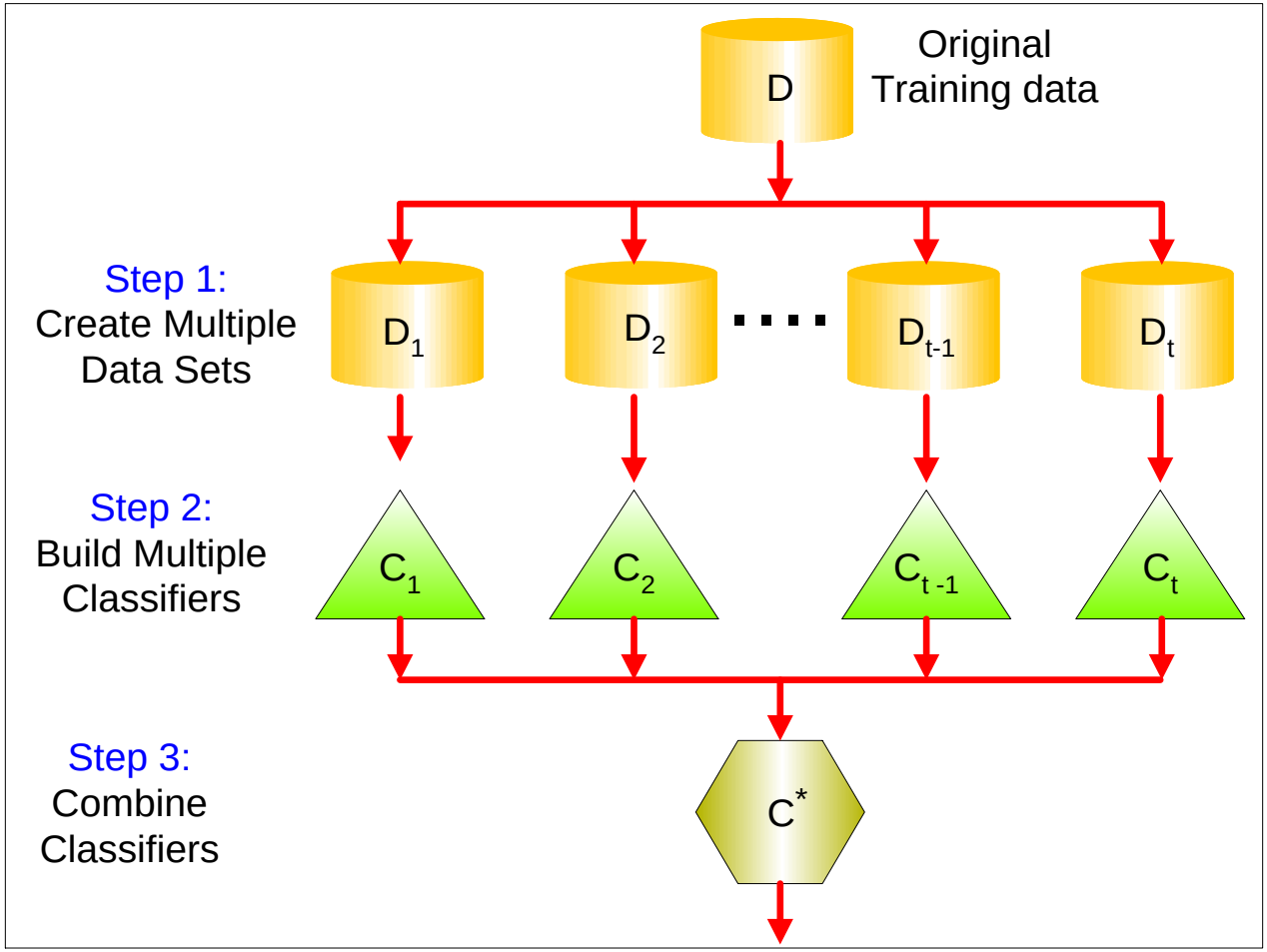
Why Ensemble Methods work?

- 25 base classifiers
- Each classifier has error rate, $\varepsilon = 0.35$
- If base classifiers are identical, then the ensemble will misclassify the same examples predicted incorrectly by the base classifiers depicted by dotted line
- Assume errors made by classifiers are uncorrelated
- ensemble makes a wrong prediction only if more than half of the base classifiers predict incorrectly
- Probability that the ensemble classifier makes a wrong prediction:



$$P(X \geq 13) = \sum_{i=13}^{25} \binom{25}{i} \varepsilon^i (1-\varepsilon)^{25-i} = 0.06$$

General Approach



Simple Ensemble Techniques

innovate achieve lead

Max Voting

Ex: Movie rating

The result of max voting would be something like this:

- Rating by 5 friends: 5 4 5 4 4

Averaging- (5+4+5+4+4)/5 = 4.4 Final rating

Weighted Average

Weight-0.23 0.23 0.18 0.18 0.18

The result is calculated as [(5*0.23) + (4*0.23) + (5*0.18) + (4*0.18) + (4*0.18)] = **4.41.**

Types of Ensemble Methods

Manipulate data distribution
- Example: bagging

Manipulate input features
- Example: random forests

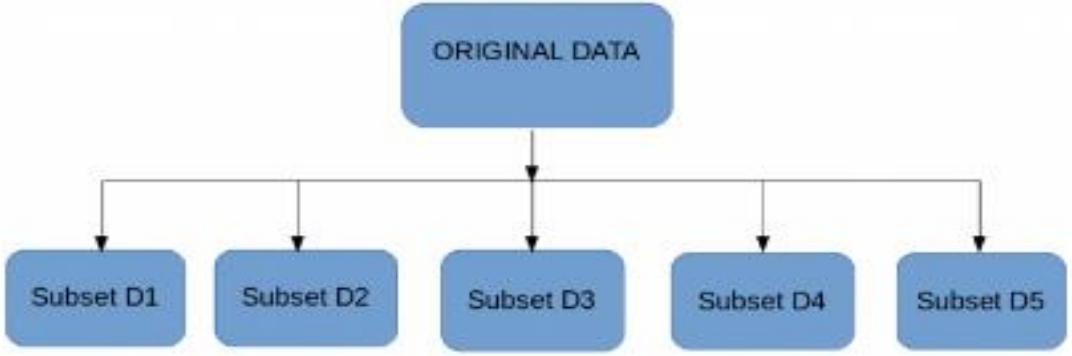
When does Ensemble work?

- Ensemble classifier performs better than the base classifiers when error is smaller than 0.5
- Necessary conditions for an ensemble classifier to perform better than a single classifier:
  - Base classifiers should be independent of each other
  - Base classifiers should do better than a classifier that performs random guessing

Bagging (Bootstrap Aggregating)

- Technique uses these subsets (bags) to get a fair idea of the distribution (complete set).
- The size of subsets created for bagging may be less than the original set.
- Bootstrapping is a sampling technique in which we create subsets of observations from the original dataset, **with replacement**.
- When you sample with replacement, items are independent. One item does not affect the outcome of the other. You have 1/7 chance of choosing the first item and a 1/7 chance of choosing the second item.
- If the two items are **dependent**, or linked to each other. When you choose the first item, you have a 1/7 probability of picking a item. Assuming you don't replace the item, you only have six items to pick from. That gives you a 1/6 chance of choosing a second item.

Bagging



- Multiple subsets are created from the original dataset, selecting observations with replacement.
- A base model (weak model) is created on each of these subsets.
- The models run in parallel and are independent of each other.
- The final predictions are determined by combining the predictions from all the models.

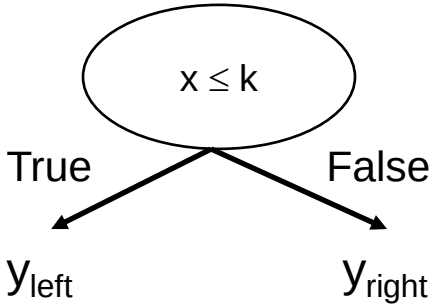
Bagging Example

- Consider 1-dimensional data set:

Original Data:

x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
y	1	1	1	-1	-1	-1	-1	1	1	1

- Classifier is a decision stump
  - Decision rule: $x \leq k$ versus $x > k$
  - Split point k is chosen based on entropy



Bagging Example

Bagging Round 1:

x	0.1	0.2	0.2	0.3	0.4	0.4	0.5	0.6	0.9	0.9
y	1	1	1	1	-1	-1	-1	-1	1	1

x <= 0.35 ➔ y = 1
x > 0.35 ➔ y = -1

Bagging Round 2:

x	0.1	0.2	0.3	0.4	0.5	0.5	0.9	1	1	1
y	1	1	1	-1	-1	-1	1	1	1	1

x <= 0.7 ➔ y = 1
x > 0.7 ➔ y = 1

Bagging Round 3:

x	0.1	0.2	0.3	0.4	0.4	0.5	0.7	0.7	0.8	0.9
y	1	1	1	-1	-1	-1	-1	-1	1	1

x <= 0.35 ➔ y = 1
x > 0.35 ➔ y = -1

Bagging Round 4:

x	0.1	0.1	0.2	0.4	0.4	0.5	0.5	0.7	0.8	0.9
y	1	1	1	-1	-1	-1	-1	-1	1	1

x <= 0.3 ➔ y = 1
x > 0.3 ➔ y = -1

Bagging Round 5:

x	0.1	0.1	0.2	0.5	0.6	0.6	0.6	1	1	1
y	1	1	1	-1	-1	-1	-1	1	1	1

x <= 0.35 ➔ y = 1
x > 0.35 ➔ y = -1

Bagging Example

Bagging Round 6:

x	0.2	0.4	0.5	0.6	0.7	0.7	0.7	0.8	0.9	1
y	1	-1	-1	-1	-1	-1	-1	1	1	1

x <= 0.75 ➔ y = -1
x > 0.75 ➔ y = 1

Bagging Round 7:

x	0.1	0.4	0.4	0.6	0.7	0.8	0.9	0.9	0.9	1
y	1	-1	-1	-1	-1	1	1	1	1	1

x <= 0.75 ➔ y = -1
x > 0.75 ➔ y = 1

Bagging Round 8:

x	0.1	0.2	0.5	0.5	0.5	0.7	0.7	0.8	0.9	1
y	1	1	-1	-1	-1	-1	-1	1	1	1

x <= 0.75 ➔ y = -1
x > 0.75 ➔ y = 1

Bagging Round 9:

x	0.1	0.3	0.4	0.4	0.6	0.7	0.7	0.8	1	1
y	1	1	-1	-1	-1	-1	-1	1	1	1

x <= 0.75 ➔ y = -1
x > 0.75 ➔ y = 1

Bagging Round 10:

x	0.1	0.1	0.1	0.1	0.3	0.3	0.8	0.8	0.9	0.9
y	1	1	1	1	1	1	1	1	1	1

x <= 0.05 ➔ y = 1
x > 0.05 ➔ y = 1

Bagging Example

- Summary of Training sets:

Round	Split Point	Left Class	Right Class
1	0.35	1	-1
2	0.7	1	1
3	0.35	1	-1
4	0.3	1	-1
5	0.35	1	-1
6	0.75	-1	1
7	0.75	-1	1
8	0.75	-1	1
9	0.75	-1	1
10	0.05	1	1

Bagging Example

- Assume test set is the same as the original data
- Use majority vote to determine class of ensemble classifier

Round	x=0.1	x=0.2	x=0.3	x=0.4	x=0.5	x=0.6	x=0.7	x=0.8	x=0.9	x=1.0
1	1	1	1	-1	-1	-1	-1	-1	-1	-1
2	1	1	1	1	1	1	1	1	1	1
3	1	1	1	-1	-1	-1	-1	-1	-1	-1
4	1	1	1	-1	-1	-1	-1	-1	-1	-1
5	1	1	1	-1	-1	-1	-1	-1	-1	-1
6	-1	-1	-1	-1	-1	-1	-1	1	1	1
7	-1	-1	-1	-1	-1	-1	-1	1	1	1
8	-1	-1	-1	-1	-1	-1	-1	1	1	1
9	-1	-1	-1	-1	-1	-1	-1	1	1	1
10	1	1	1	1	1	1	1	1	1	1
Sum	2	2	2	-6	-6	-6	-6	2	2	2
Sign	1	1	1	-1	-1	-1	-1	1	1	1

Predicted Class

Bagging Algorithm

Algorithm 5.6 Bagging Algorithm

1: Let k be the number of bootstrap samples.
2: for $i = 1$ to k do
3: Create a bootstrap sample of size n, D_i.
4: Train a base classifier C_i on the bootstrap sample D_i.
5: end for
6: $C^*(x) = \arg\max_y \sum_i \delta\big(C_i(x) = y\big)$, $\{\delta(\cdot) = 1$ if its argument is true, and 0 otherwise.$\}$

Random Forest

- Random Forest is another ensemble machine learning algorithm that follows the bagging technique.
- The base estimators in random forest are decision trees.
- Random forest randomly selects a set of features which are used to decide the best split at each node of the decision tree.

Random Forest

- As in bagging, we build a number of decision trees on bootstrapped training samples each time a split in a tree is considered, a random sample of m predictors is chosen as split candidates from the full set of p predictors.
- Note that if m = p, then this is bagging.

Random Forest

innovate achieve lead

- Random subsets are created from the original dataset (bootstrapping).
- At each node in the decision tree, only a random set of features are considered to decide the best split.
- A decision tree model is fitted on each of the subsets.
- The final prediction is calculated by averaging the predictions from all decision trees.


All Data
random subset
random subset
random subset
random subset
tree
tree
tree
tree
tree
At each node:
choose some ballsubset of variables at random
find a variable (and a value for that variable) which optimizes the split

Random Forests Algorithm

- For b = 1 to B:
  - (a) Draw a bootstrap sample $Z*$ of size N from the training data.
  - (b) Grow a random-forest tree to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size nmin is reached.
    - i. Select m variables at random from the p variables.
    - ii. Pick the best variable/split-point among the m.
    - iii. Split the node into two daughter nodes. Output the ensemble of trees.
- To make a prediction at a new point x we do:
  - For regression: average the results
  - For classification: majority vote

Random Forests Tuning

- The inventors make the following recommendations:
  - For classification, the default value for m is √p and the minimum node size is one.
  - For regression, the default value for m is p/3 and the minimum node size is five.
- In practice the best values for these parameters will depend on the problem, and they should be treated as tuning parameters.

Example

- 4,718 genes measured on tissue samples from 349 patients.
- Each gene has different expression
- Each of the patient samples has a qualitative label with 15 different levels: either normal or 1 of 14 different types of cancer.
- Use random forests to predict cancer type based on the 500 genes that have the largest variance in the training set.

Random Forests Issues

- When the number of variables is large, but the fraction of relevant variables is small, random forests are likely to perform poorly when m is small

Why?

- Because: At each split the chance can be small that the relevant variables will be selected
- For example, with 3 relevant and 100 not so relevant variables the probability of any of the relevant variables being selected at any split is ~0.25

Advantages of Random Forest

- Algorithm can solve both type of problems i.e. classification and regression
- Power of handle large data set with higher dimensionality.
- It can handle thousands of input variables and identify most significant variables so it is considered as one of the dimensionality reduction methods.
- Model outputs **Importance of variable,** which can be a very handy feature (on some random data set).

Disadvantages of Random Forest

- May over-fit data sets that are particularly noisy.
- Random Forest can feel like a black box approach for statistical modelers – you have very little control on what the model does. You can at best – try different parameters and random seeds!

Good References

Decision Tree

- https://www.youtube.com/watch?v=eKD5gxPPeY0&list=PLBv09BD7ez_4temBw7vLA19p3tdQH6FYO&index=1

Overfitting

- https://www.youtube.com/watch?time_continue=1&v=t56Nid85Thg
- https://www.youtube.com/watch?v=y6SpA2Wuyt8

Random Forest

- https://www.stat.berkeley.edu/~breiman/RandomForests/