Decision Tree

ID3 Algorithm

C4.5 Algorithm

ID3: Introduction

- Iterative Dichotomizer: A mathematical algorithm for building the decision tree.
- Invented by J. Ross Quinlan in 1979.
- Uses Information Theory invented by Shannon in 1948.
- Builds the tree from the top down, with no backtracking.
- Information Gain is used to select the most useful attribute for classification.

Entropy

- Entropy measures the randomness/uncertainty in the data
- Let's consider a set S of examples with C many classes. Entropy of this set:

$$H(S) = -\sum_{c \in C} p_c \log_2 p_c$$

$$Entropy(t) = -\sum_j p(j \mid t) \log p(j \mid t)$$

- p_c is the probability that an element of S belongs to class c
  - .. basically, the fraction of elements of S belonging to class c
- Intuition: Entropy is a measure of the "degree of surprise"
  - Some dominant classes $\Longrightarrow$ small entropy (less uncertainty)
  - Equiprobable classes $\Longrightarrow$ high entropy (more uncertainty)
- Entropy denotes the average number of bits needed to encode S

Examples for computing Entropy

$$Entropy(t) = -\sum_j p(j \mid t) \log_2 p(j \mid t)$$

C1	**0**
C2	**6**

P(C1) = 0/6 = 0 P(C2) = 6/6 = 1

Entropy = – 0 log 0 – 1 log 1 = – 0 – 0 = 0

C1	**1**
C2	**5**

P(C1) = 1/6 P(C2) = 5/6

Entropy = – (1/6) $\log_2$ (1/6) – (5/6) $\log_2$ (1/6) = 0.65

C1	**2**
C2	**4**

P(C1) = 2/6 P(C2) = 4/6

Entropy = – (2/6) $\log_2$ (2/6) – (4/6) $\log_2$ (4/6) = 0.92

Useful Note: Calculating Entropy with Code

- Most programming languages and calculators do not have a $\log_2$ function.
- Use a conversion factor
- Take log function of 2, and divide by it.
- Example: $\log_{10}(2) = .301$
- Then divide to get $\log_2(n)$: [n=3/5]
- $\log_2\left(\frac{3}{5}\right) = \log_{10}(3/5) \,/\, .301$

- Taking $\log_{10}(0)$ produces an error.
  - Substitute 0 for (0/3) $\log_{10}(0/3)$
  - Do not try to calculate $\log_{10}(0/3)$

Splitting Based on Information-GAIN [Quality of Split]

- Information Gain:

$$GAIN_{split} = Entropy(p) - \left(\sum_{i=1}^{k} \frac{n_i}{n} Entropy(i) \right)$$

Parent Node, p is split into k partitions;

n_i is number of records in partition i

- Measures Reduction in Entropy achieved because of the split. Choose the split that achieves most reduction (maximizes GAIN)

- Disadvantage: Tends to prefer splits that result in large number of partitions, each being small but pure.

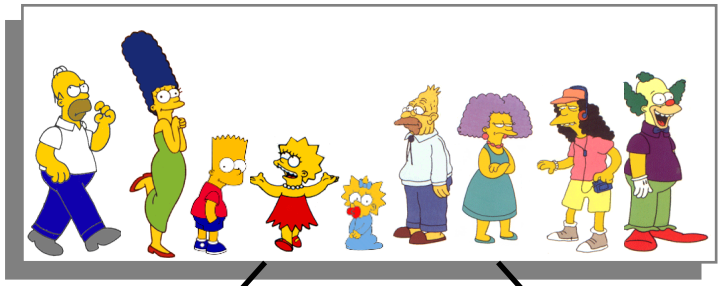
Information Gain (IG)

- The information gain is based on the decrease in entropy after a dataset is split on an attribute.
- Which attribute creates the most homogeneous branches?
- First the entropy of the total dataset is calculated.
- The dataset is then split on the different attributes.
- The entropy for each branch is calculated. Then it is added proportionally, to get total entropy for the split.
- The resulting entropy is subtracted from the entropy before the split.
- The result is the Information Gain, or decrease in entropy.
- The attribute that yields the largest IG is chosen for the decision node.

Information Gain (cont’d)

- A branch set with entropy of 0 is a leaf node.
- Otherwise, the branch needs further splitting to classify its dataset.
- The ID3 algorithm is run recursively on the non-leaf branches, until all data is classified.

Another Example



$$Entropy(S) = -\frac{p}{p+n}\log_2\left(\frac{p}{p+n}\right) - \frac{n}{p+n}\log_2\left(\frac{n}{p+n}\right)$$

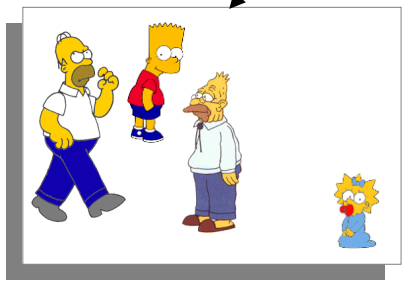
$Entropy(4\mathbf{F},5\mathbf{M}) = -(4/9)\log_2(4/9) - (5/9)\log_2(5/9)$

$= \mathbf{0.9911}$

yes

no

Hair Length <= 5?



$Entropy(1\mathbf{F},3\mathbf{M}) = -(1/4)\log_2(1/4) - (3/4)\log_2(3/4)$

$= \mathbf{0.8113}$

$Entropy(3\mathbf{F},2\mathbf{M}) = -(3/5)\log_2(3/5) - (2/5)\log_2(2/5)$

$= \mathbf{0.9710}$

$$Gain(A) = E(Current\ set) - \sum E(all\ child\ sets)$$

$Gain(\text{Hair Length} <= 5) = \mathbf{0.9911} - (4/9 * \mathbf{0.8113} + 5/9 * \mathbf{0.9710}) = \mathbf{0.0911}$

An Example of Calculating Entropy and Information-Gain

$$E(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

Play Golf	
Yes	No
9	5

Entropy(PlayGolf) = Entropy (5,9)
= Entropy (0.36, 0.64)
= - (0.36 $\log_2$ 0.36) - (0.64 $\log_2$ 0.64)
= 0.94

$$E(T, X) = \sum_{c \in X} P(c)E(c)$$

		Play Golf		
		Yes	No	
Outlook	Sunny	3	2	5
	Overcast	4	0	4
	Rainy	2	3	5
				14

E(PlayGolf, Outlook) = **P**(Sunny)***E**(3,2) + **P**(Overcast)***E**(4,0) + **P**(Rainy)***E**(2,3)
= (5/14)*0.971 + (4/14)*0.0 + (5/14)*0.971
= 0.693

Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	85	85	Weak	No
2	Sunny	80	90	Strong	No
3	Overcast	83	78	Weak	Yes
4	Rain	70	96	Weak	Yes
5	Rain	68	80	Weak	Yes
6	Rain	65	70	Strong	No
7	Overcast	64	65	Strong	Yes
8	Sunny	72	95	Weak	No
9	Sunny	69	70	Weak	Yes
10	Rain	75	80	Weak	Yes
11	Sunny	75	70	Strong	Yes
12	Overcast	72	90	Strong	Yes
13	Overcast	81	75	Weak	Yes
14	Rain	71	80	Strong	No

Information Gain(T,X) = Entropy(T) - Entropy(T, X)

IG(PlayGolf, Outlook) = E(PlayGolf) - E(PlayGolf, Outlook)
= 0.940 - 0.693
= 0.247

Lets complete a full example for ID3 algorithm

- First Select the root based on highest Gain.
- Iteratively select the feature with the highest I.G. for each child of previous node.

$$Entropy(S) = \sum - p(I) \cdot \log_2 p(I)$$

$$Gain(S, A) = Entropy(S) - \sum [\, p(S|A) \cdot Entropy(S|A) \,]$$

Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	85	85	Weak	No
2	Sunny	80	90	Strong	No
3	Overcast	83	78	Weak	Yes
4	Rain	70	96	Weak	Yes
5	Rain	68	80	Weak	Yes
6	Rain	65	70	Strong	No
7	Overcast	64	65	Strong	Yes
8	Sunny	72	95	Weak	No
9	Sunny	69	70	Weak	Yes
10	Rain	75	80	Weak	Yes
11	Sunny	75	70	Strong	Yes
12	Overcast	72	90	Strong	Yes
13	Overcast	81	75	Weak	Yes
14	Rain	71	80	Strong	No

Entropy

We need to calculate the entropy first. Decision column consists of 14 instances and includes two labels: yes and no. There are 9 decisions labeled yes, and 5 decisions labeled no.

$$\text{Entropy(Decision)} = - p(\text{Yes}) \cdot \log_2 p(\text{Yes}) - p(\text{No}) \cdot \log_2 p(\text{No})$$

$$\text{Entropy(Decision)} = - (9/14) \cdot \log_2 (9/14) - (5/14) \cdot \log_2 (5/14) = 0.940$$

Now, we need to find the most dominant factor for decisioning.
We start with **Wind.**

Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	85	85	Weak	No
2	Sunny	80	90	Strong	No
3	Overcast	83	78	Weak	Yes
4	Rain	70	96	Weak	Yes
5	Rain	68	80	Weak	Yes
6	Rain	65	70	Strong	No
7	Overcast	64	65	Strong	Yes
8	Sunny	72	95	Weak	No
9	Sunny	69	70	Weak	Yes
10	Rain	75	80	Weak	Yes
11	Sunny	75	70	Strong	Yes
12	Overcast	72	90	Strong	Yes
13	Overcast	81	75	Weak	Yes
14	Rain	71	80	Strong	No

Strong wind factor on decision

Day	Outlook	Temp.	Humidity	Wind	Decision
2	Sunny	Hot	High	Strong	No
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
14	Rain	Mild	High	Strong	No

Here, there are 6 instances for strong wind. Decision is divided into two equal parts.

1- Entropy(Decision|Wind=Strong) = – p(No) . $\log_2$p(No) – p(Yes) . $\log_2$p(Yes)

2- Entropy(Decision|Wind=Strong) = – (3/6) . $\log_2$(3/6) – (3/6) . $\log_2$(3/6) = 1

Weak wind factor on decision

Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
13	Overcast	Hot	Normal	Weak	Yes

There are 8 instances for weak wind. Decision of 2 items are no and 6 items are yes as illustrated below.

1- Entropy(Decision|Wind=Weak) = – p(No) . $\log_2$p(No) – p(Yes) . $\log_2$p(Yes)

2- Entropy(Decision|Wind=Weak) = – (2/8) . $\log_2$(2/8) – (6/8) . $\log_2$(6/8) = 0.811

Now, we can turn back to Gain(Decision, Wind) equation.

Gain(Decision, Wind) = Entropy(Decision) – [p(Decision|Wind=Weak) . Entropy(Decision|Wind=Weak)] – [p(Decision|Wind=Strong) . Entropy(Decision|Wind=Strong)] = 0.940 – [(8/14) . 0.811] – [(6/14). 1] = 0.048

Other factors on decision

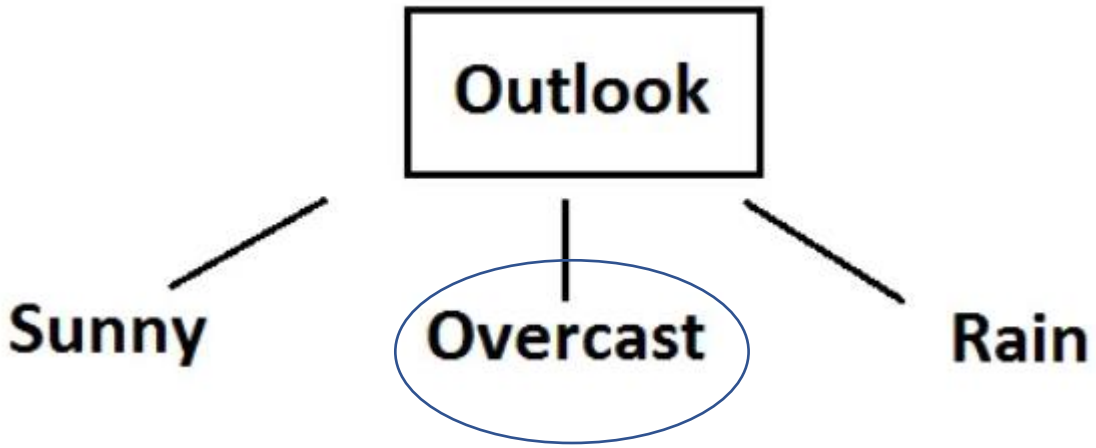
We have applied similar calculation on the other columns.

1- Gain(Decision, Outlook) = 0.246

Highest

2- Gain(Decision, Temperature) = 0.029

3- Gain(Decision, Humidity) = 0.151



Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	85	85	Weak	No
2	Sunny	80	90	Strong	No
3	Overcast	83	78	Weak	Yes
4	Rain	70	96	Weak	Yes
5	Rain	68	80	Weak	Yes
6	Rain	65	70	Strong	No
7	Overcast	64	65	Strong	Yes
8	Sunny	72	95	Weak	No
9	Sunny	69	70	Weak	Yes
10	Rain	75	80	Weak	Yes
11	Sunny	75	70	Strong	Yes
12	Overcast	72	90	Strong	Yes
13	Overcast	81	75	Weak	Yes
14	Rain	71	80	Strong	No

Overcast outlook on decision

Basically, decision will always be yes if outlook were overcast.

Day	Outlook	Temp.	Humidity	Wind	Decision
3	Overcast	Hot	High	Weak	Yes
7	Overcast	Cool	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes

Sunny outlook on decision

Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

Here, there are 5 instances for sunny outlook. Decision would be probably 3/5 percent no, 2/5 percent yes.

1- Gain(Outlook=Sunny|Temperature) = 0.570

2- Gain(Outlook=Sunny|Humidity) = 0.970

3- Gain(Outlook=Sunny|Wind) = 0.019

Now, humidity is the decision because it produces the highest score if outlook were sunny.

At this point, decision will always be no if humidity were high.

Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No

On the other hand, decision will always be yes if humidity were normal

Day	Outlook	Temp.	Humidity	Wind	Decision
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

Finally, it means that we need to check the humidity and decide if outlook were sunny.

Rain outlook on decision

Day	Outlook	Temp.	Humidity	Wind	Decision
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
10	Rain	Mild	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

1- Gain(Outlook=Rain | Temperature) = 0.01997309402197489

2- Gain(Outlook=Rain | Humidity) = 0.01997309402197489

3- Gain(Outlook=Rain | Wind) = 0.9709505944546686

Here, wind produces the highest score if outlook were rain. That's why, we need to check wind attribute in 2nd level if outlook were rain.

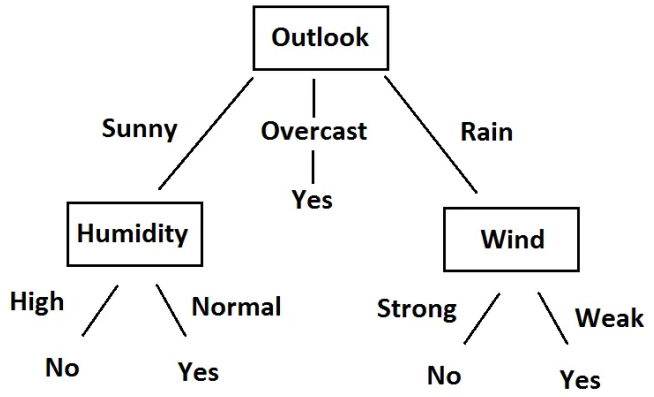
So, it is revealed that decision will always be yes if wind were weak and outlook were rain.

Day	Outlook	Temp.	Humidity	Wind	Decision
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes

What's more, decision will be always no if wind were strong and outlook were rain.

Day	Outlook	Temp.	Humidity	Wind	Decision
6	Rain	Cool	Normal	Strong	No
14	Rain	Mild	High	Strong	No

So, decision tree construction is over. We can use the following rules for decisioning.



Final version of decision tree

Advantages of using ID3

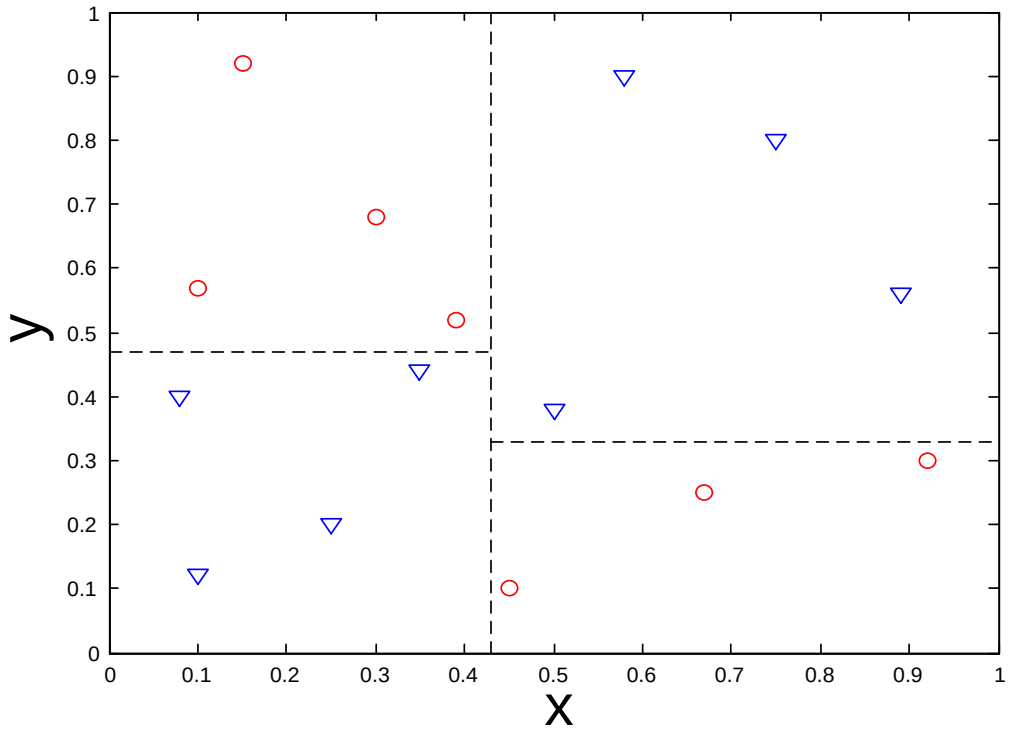
- Understandable prediction rules are created from the training data.
- Builds the fastest tree.
- Builds a short tree.
- Only need to test enough attributes until all data is classified.
- Finding leaf nodes enables test data to be pruned, reducing number of tests.
- Whole dataset is searched to create tree.

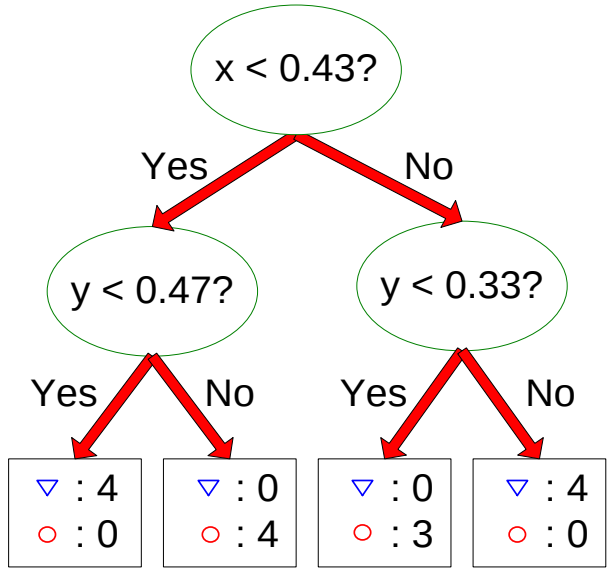
Disadvantages of using ID3

- Data may be over-fitted or over-classified, if a small sample is tested.
- Only one attribute at a time is tested for making a decision.
- Classifying continuous data may be computationally expensive, as many trees must be generated to see where to break the continuum.

C4.5 Algorithm: A successor of ID3

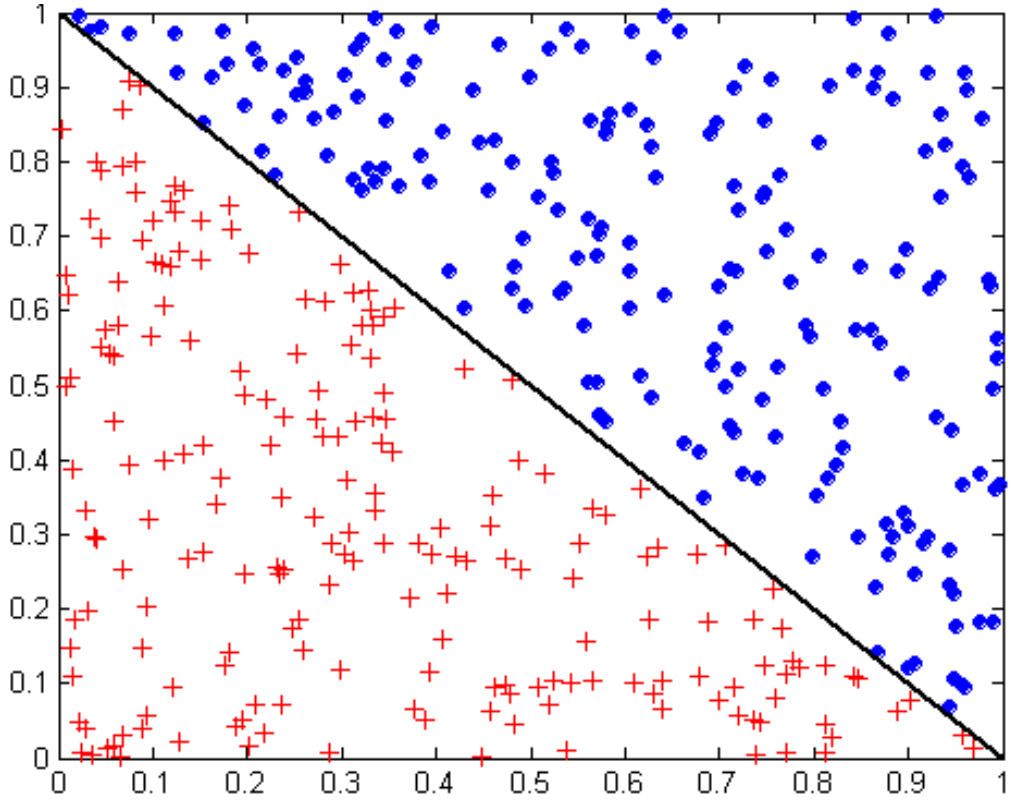
A bit about Decision Boundary

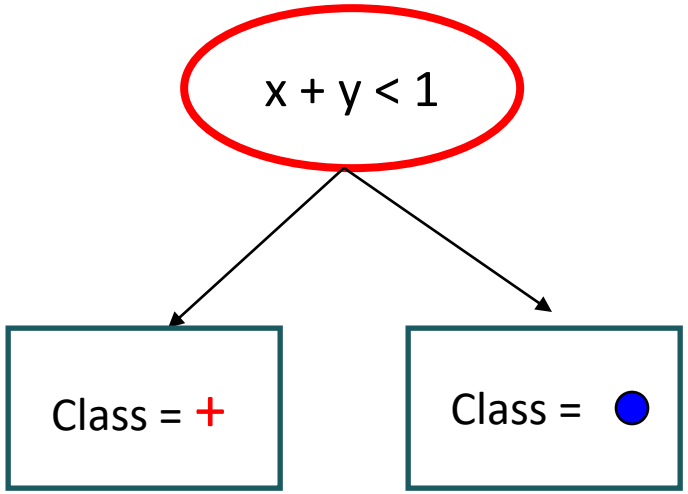




- Border line between two neighboring regions of different classes is known as decision boundary
- Decision boundary is parallel to axes because test condition involves a single attribute at-a-time

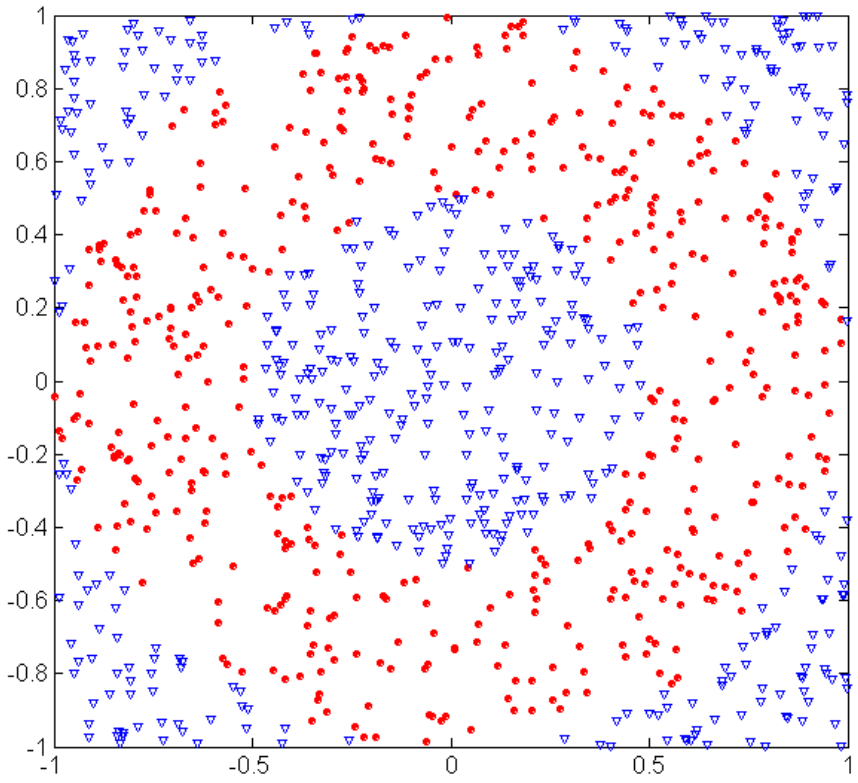
Linear Boundary





- Test condition may involve multiple attributes
- More expressive representation
- Finding optimal test condition is computationally expensive

Vertical/Horizontal Boundaries



500 circular and 500 triangular data points.

Circular points:

$0.5 \leq \text{sqrt}(x_1^2+x_2^2) \leq 1$

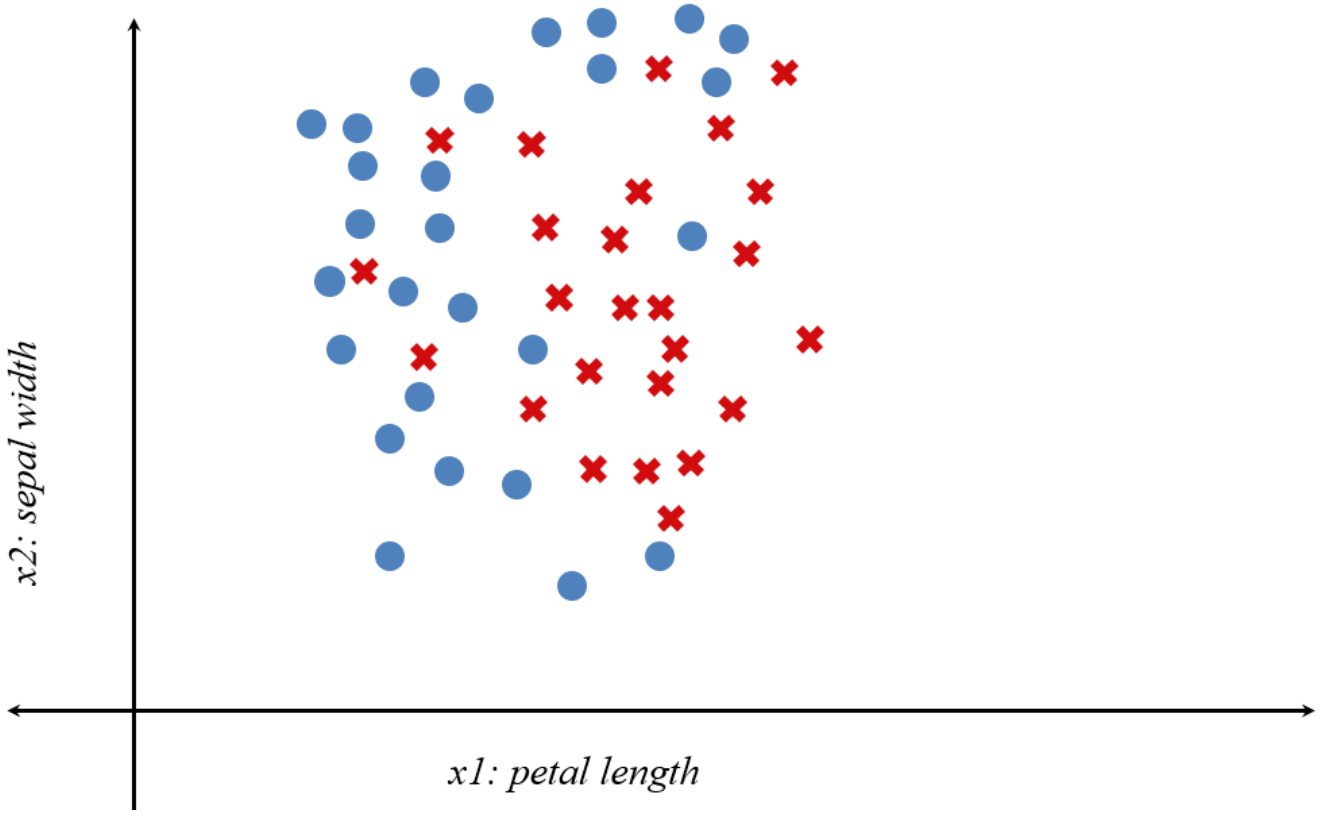
Triangular points:

$\text{sqrt}(x_1^2+x_2^2) > 0.5$ or

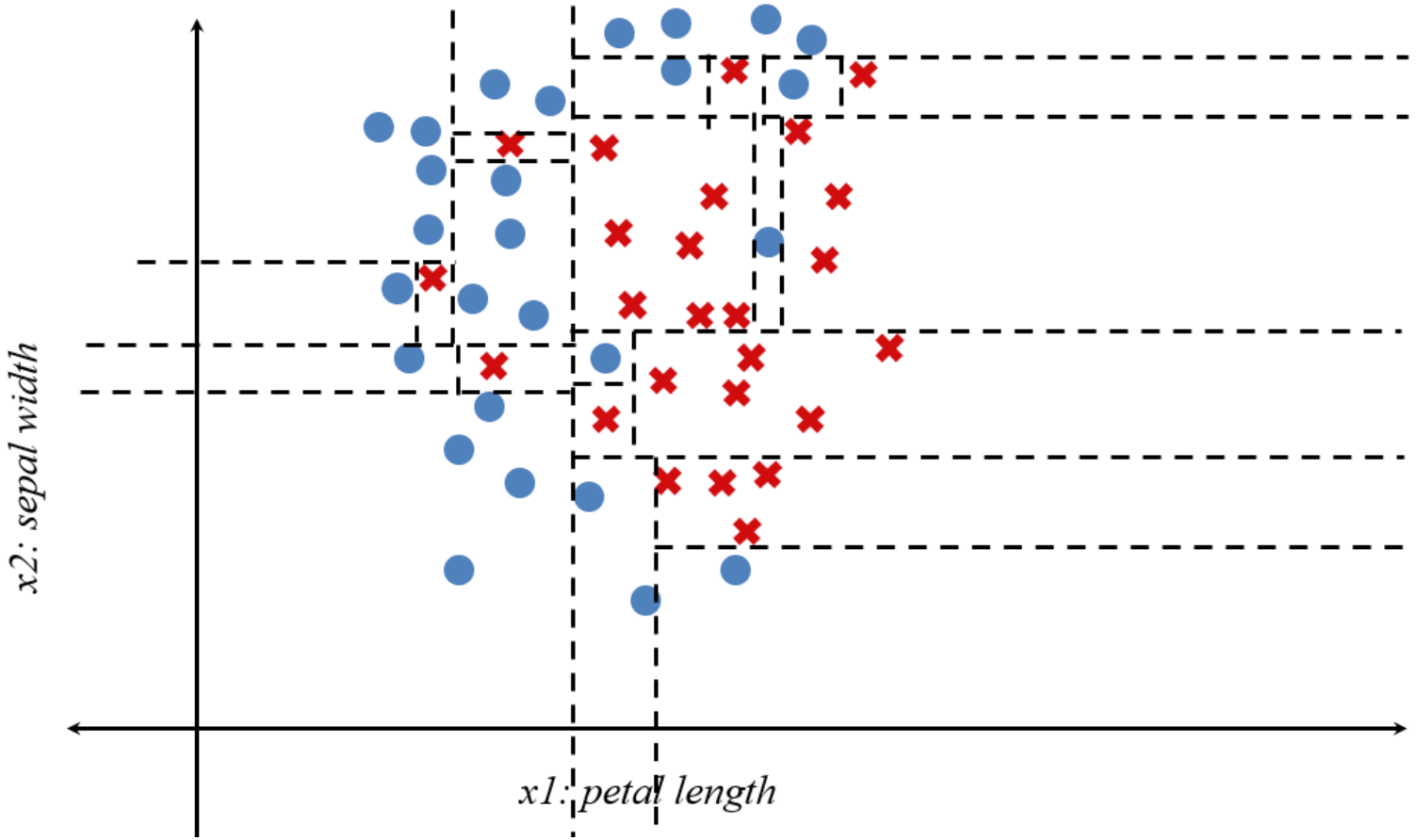
$\text{sqrt}(x_1^2+x_2^2) < 1$

DTs in practice…

Growing to purity is bad (*overfitting*)

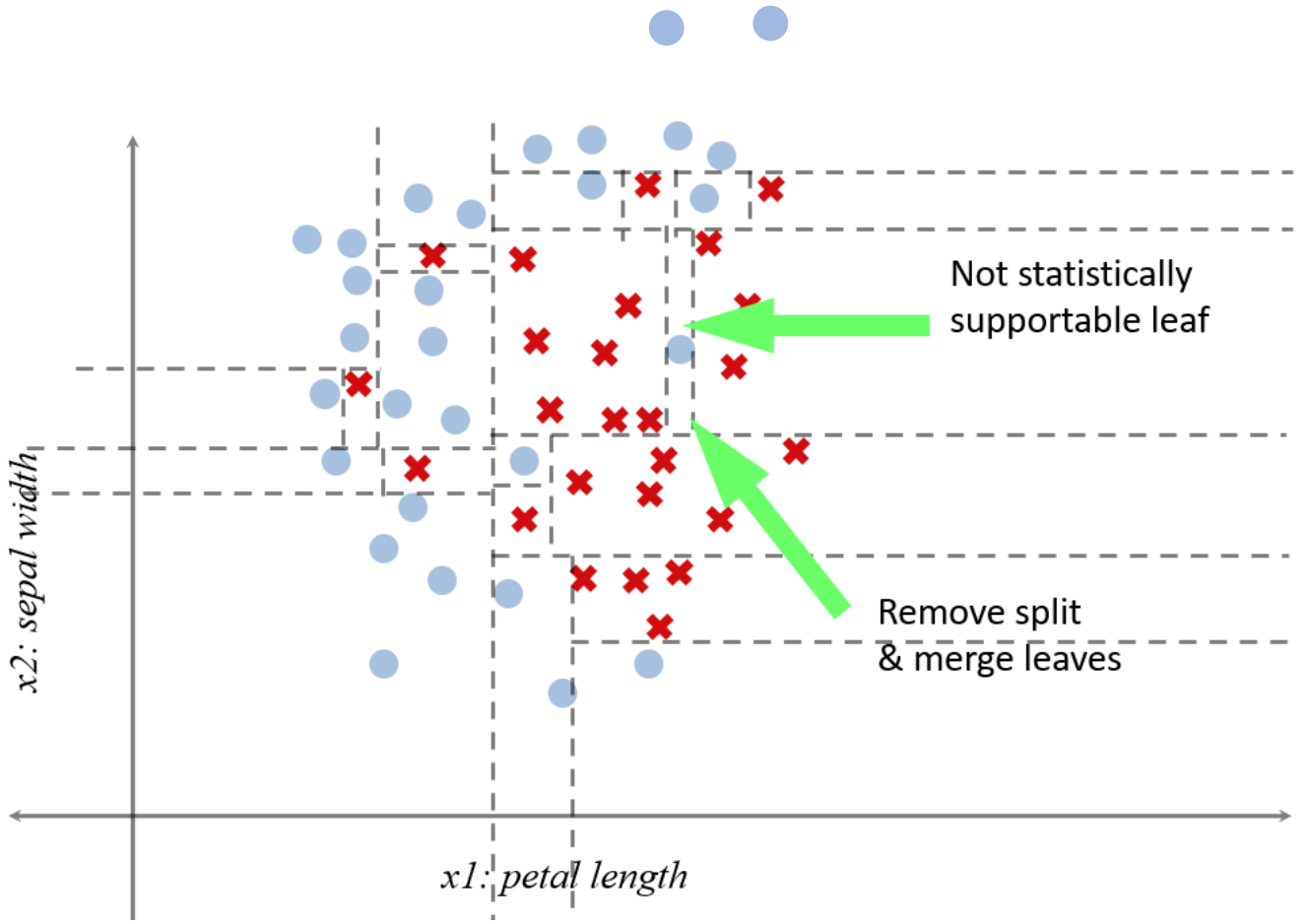


DTs in practice…



DTs in practice...

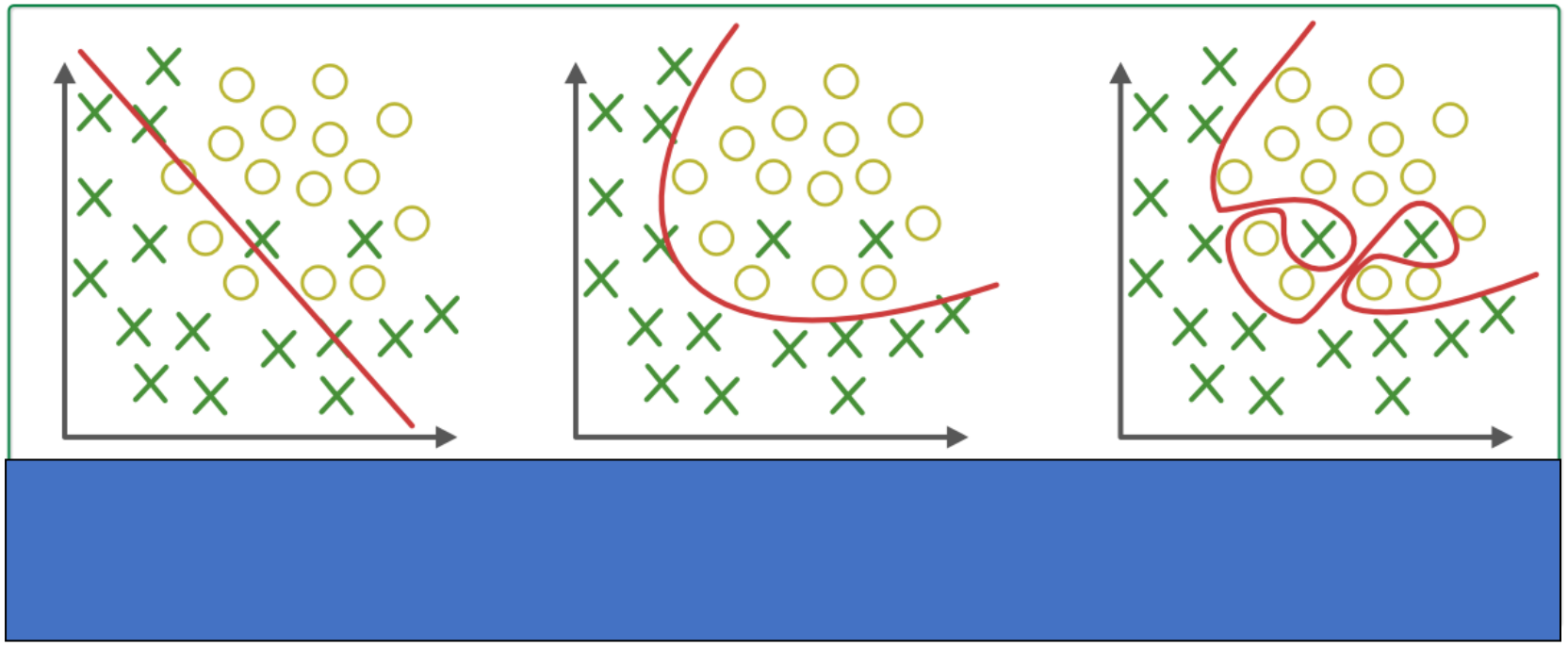
- Growing to purity is bad (*overfitting*)

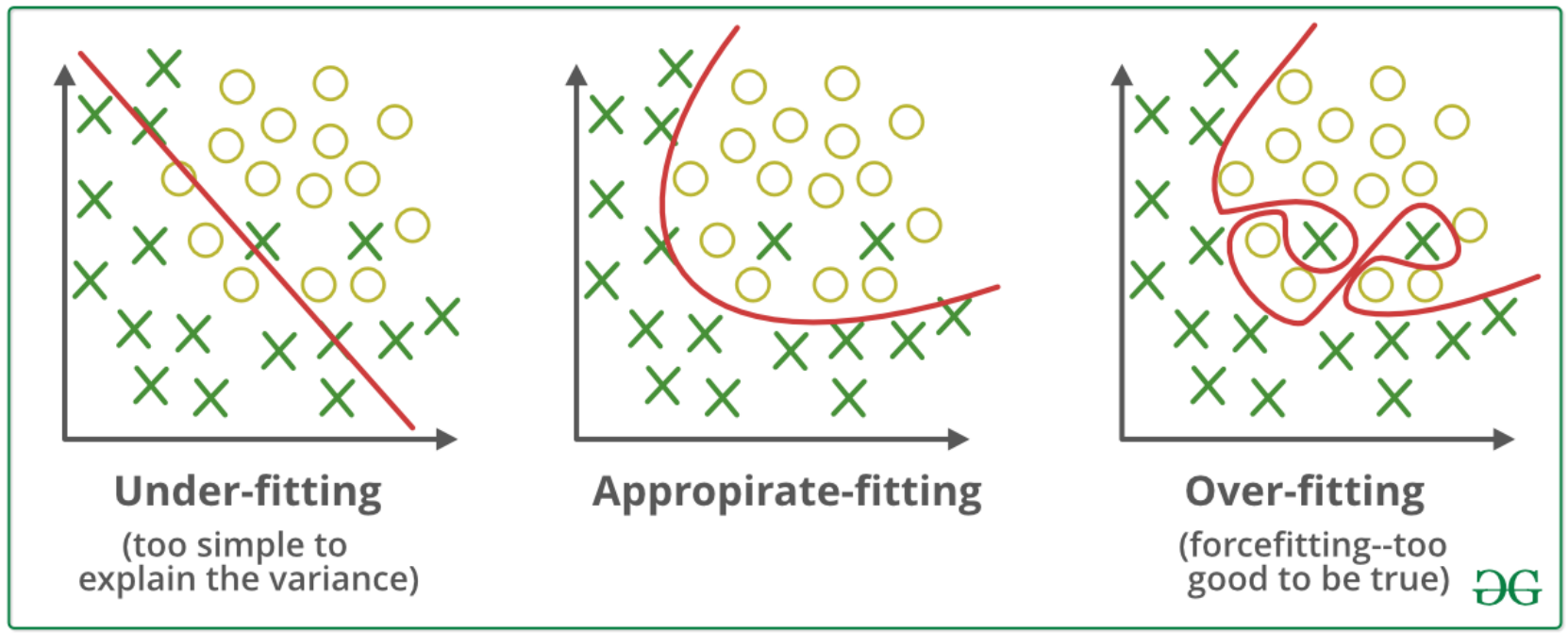


DTs in practice...

- Growing to purity is bad (*overfitting*)
  - Terminate growth early
  - Grow to purity, then *prune* back

Question: Which one of the following is good-fit/underfit/overfit?




Under-fitting
(too simple to explain the variance)
Appropirate-fitting
Over-fitting
(forcefitting--too good to be true)

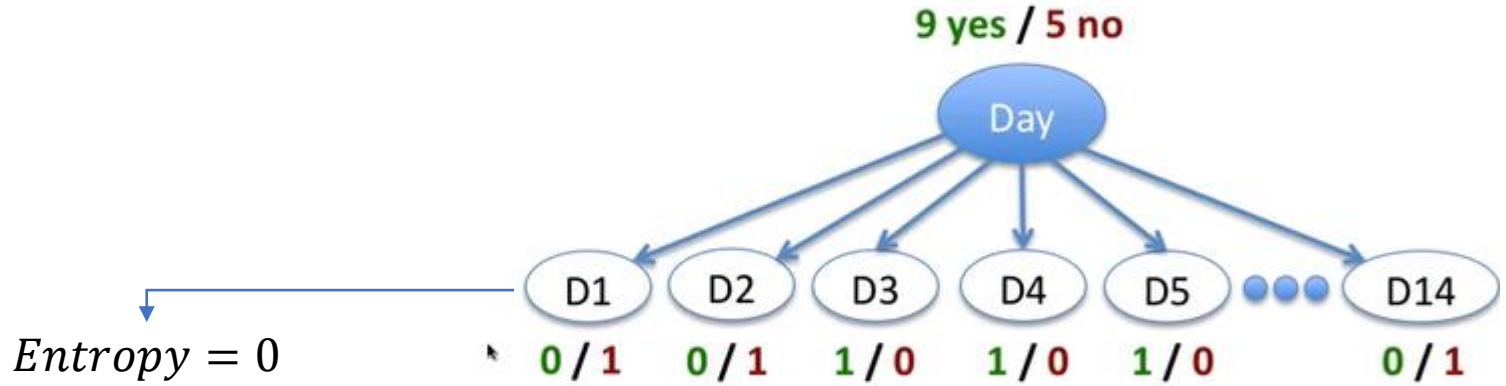
Avoid Overfitting in Decission-Tree

- The generated tree may overfit the training data
  - Too many branches, some may reflect anomalies due to noise or outliers
  - Result is in poor accuracy for unseen samples
- Two approaches to avoid overfitting
  - **Prepruning**: Halt tree construction early—do not split a node if this would result in the goodness measure falling below a threshold
    - Difficult to choose an appropriate threshold
  - **Postpruning**: Remove branches from a “fully grown” tree—get a sequence of progressively pruned trees
    - Use a set of data different from the training data to decide which is the “best pruned tree”

Algorithm C 4.5 : Introduction

- J. Ross Quinlan, a researcher in machine learning developed a decision tree induction algorithm in 1984 known as ID3 (Iterative Dichotometer 3).
- Quinlan later presented C4.5, a successor of ID3, addressing some limitations in ID3.
- ID3 uses information gain measure, which is, in fact biased towards splitting attribute having a large number of outcomes.
- For example, if an attribute has distinct values for all tuples, then it would result in a large number of partitions, each one containing just one tuple.
  - In such a case, note that each partition is pure, and hence the purity measure of the partition, that is $E_A(D) = 0$

Problems with Information Gain



$Entropy = 0$

Information Gain (I.G.)
will be maximum in this case.

ID3 is biased towards attributes with many values.

Question 1: How I.G. will treat this node?
Question 2: Will the tree size increase/decrease if choose this as the root node?
Question 3: Can we normalize the I.G. to tackle such cases?

Limitations of ID3

- Although, the previous situation is an extreme case, intuitively, we can infer that ID3 favours splitting attributes having a large number of values
  - compared to other attributes, which have a less variations in their values.
- Such a partition appears to be useless for classification.
- This type of problem is called **overfitting problem**.

Algorithm: C 4.5 : Introduction

- The overfitting problem in ID3 is due to the measurement of information gain.
- In order to reduce the effect of the use of the bias due to the use of information gain, C4.5 uses a different measure called **Gain Ratio**, denoted as β.
- Gain Ratio is a kind of normalization to information gain using a **split information**.

Algorithm: C 4.5 : Gain Ratio

$$Gain\ Ratio = \frac{Information\ Gain}{SplitInfo} = \frac{Entropy\ (before) - \sum_{j=1}^{K} Entropy(j, after)}{\sum_{j=1}^{K} w_j \log_2 w_j}$$

$$w_j = \frac{number\ of\ values\ only\ for\ subset\ 'j'}{total\ numbers\ of\ values}$$

- *SplitInfo* is also referred as ***SplitEntropy/SplitInformation*** in the literature.
- It can be interpreted like – does this attribute contains lots of tiny subsets, or not~
- It will have low value if we have few subsets, and vice-verse.

Physical Interpretation of $E_A^*(\mathrm{D})$ [SplitInfo]

Example 9.18 : Split information $E_A^*(\mathrm{D})$

- To illustrate $E_A^*(\mathrm{D})$, let us examine the case where there are 32 instances and splitting an attribute A which has a_1, a_2, a_3 and a_4 sets of distinct values.
  - Distribution 1 : Highly non-uniform distribution of attribute values

	a_1	a_2	a_3	a_4
Frequency	32	0	0	0

$$E_A^*(\mathrm{D}) = -\frac{32}{32} log_2(\frac{32}{32}) = -log_2 1 = 0$$

  - Distribution 2

	a_1	a_2	a_3	a_4
Frequency	16	16	0	0

$$E_A^*(\mathrm{D}) = -\frac{16}{32} log_2(\frac{16}{32}) - \frac{16}{32} log_2(\frac{16}{32}) = log_2 2 = 1$$

Physical Interpretation of $E_A^*(\mathrm{D})$ [SplitInfo]

- Distribution 3

	a_1	a_2	a_3	a_4
Frequency	16	8	8	0

$$E_A^*(\mathrm{D}) = -\frac{16}{32}log_2(\frac{16}{32}) - \frac{8}{32}log_2(\frac{8}{32}) - \frac{8}{32}log_2(\frac{8}{32}) = 1.5$$

- Distribution 4

	a_1	a_2	a_3	a_4
Frequency	16	8	4	4

$$E_A^*(\mathrm{D}) = 1.75$$

- Distribution 5: Uniform distribution of attribute values

	a_1	a_2	a_3	a_4
Frequency	8	8	8	8

$$E_A^*(\mathrm{D}) = (-\frac{8}{32}log_2(\frac{8}{32}))*4 = -log_2(\frac{1}{4}) = 2.0$$

Physical Interpretation of $E_A^*(D)$ [SplitInfo]

- In general, if there are m attribute values, each occurring equally frequently, then the split information is $log_2 m$.
- Based on the Example 9.18, we can summarize our observation on split information as under:
  - Split information is 0 when there is a single attribute value. It is a trivial case and implies *the minimum possible value of split information*.
  - For a given data set, when instances are uniformly distributed with respect to the attribute values, split information increases as the number of different attribute values increases.
  - The maximum value of split information occur when there are many possible attribute values, all are equally frequent.

Note:

- Split information varies between 0 and $log_2 m$ (both inclusive)

Physical Interpretation of $\beta(A, B)$ [Gain-Ratio]

- Information gain signifies how much information will be gained on partitioning the values of attribute A
  - Higher information gain means splitting of A is more desirable.
- On the other hand, split information forms the denominator in the gain ratio formula.
  - This implies that higher the value of split information is, lower the gain ratio.
  - In turns, it decreases the information gain.
- Further, information gain is large when there are many distinct attribute values.
  - When many distinct values, split information is also a large value.
  - This way split information reduces the value of gain ratio, thus resulting a balanced value for information gain.
- Like information gain (in ID3), the attribute with the maximum gain ratio is selected as the splitting attribute in C4.5.
- C4.5 follows the exact same algorithm as ID3, except using Gain-ratio instead of Information-Gain.

An Example of calculating Gain-ratio for Wind attribute

Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	85	85	Weak	No
2	Sunny	80	90	Strong	No
3	Overcast	83	78	Weak	Yes
4	Rain	70	96	Weak	Yes
5	Rain	68	80	Weak	Yes
6	Rain	65	70	Strong	No
7	Overcast	64	65	Strong	Yes
8	Sunny	72	95	Weak	No
9	Sunny	69	70	Weak	Yes
10	Rain	75	80	Weak	Yes
11	Sunny	75	70	Strong	Yes
12	Overcast	72	90	Strong	Yes
13	Overcast	81	75	Weak	Yes
14	Rain	71	80	Strong	No

$$\text{Entropy(Decision)} = \sum - p(I) \,.\, \log_2 p(I) = - p(\text{Yes}) \,.\, \log_2 p(\text{Yes}) - p(\text{No}) \,.\, \log_2 p(\text{No}) = - (9/14) \,.\, \log_2(9/14) - (5/14) \,.\, \log_2(5/14) = 0.940$$

$$\text{GainRatio}(A) = \text{Gain}(A) \,/\, \text{SplitInfo}(A)$$

$$\text{SplitInfo}(A) = -\sum |Dj|/|D| \times \log_2 |Dj|/|D|$$

Wind Attribute

Wind is a nominal attribute. Its possible values are weak and strong.

Gain(Decision, Wind) = Entropy(Decision) − ∑ (p(Decision|Wind) . Entropy(Decision|Wind))

Gain(Decision, Wind) = Entropy(Decision) − [p(Decision|Wind=Weak) . Entropy(Decision|Wind=Weak)] + [p(Decision|Wind=Strong) . Entropy(Decision|Wind=Strong)]

There are 8 weak wind instances. 2 of them are concluded as no, 6 of them are concluded as yes.

Entropy(Decision|Wind=Weak) = − p(No) . $\log_2$p(No) − p(Yes) . $\log_2$p(Yes) = − (2/8) . $\log_2$(2/8) − (6/8) . $\log_2$(6/8) = 0.811

Entropy(Decision|Wind=Strong) = − (3/6) . $\log_2$(3/6) − (3/6) . $\log_2$(3/6) = 1

Gain(Decision, Wind) = 0.940 − (8/14).(0.811) − (6/14).(1) = 0.940 − 0.463 − 0.428 = 0.049

There are 8 decisions for weak wind, and 6 decisions for strong wind.

SplitInfo(Decision, Wind) = -(8/14).$\log_2$(8/14) − (6/14).$\log_2$(6/14) = 0.461 + 0.524 = 0.985

GainRatio(Decision, Wind) = Gain(Decision, Wind) / SplitInfo(Decision, Wind) = 0.049 / 0.985 = 0.049

Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	85	85	Weak	No
2	Sunny	80	90	Strong	No
3	Overcast	83	78	Weak	Yes
4	Rain	70	96	Weak	Yes
5	Rain	68	80	Weak	Yes
6	Rain	65	70	Strong	No
7	Overcast	64	65	Strong	Yes
8	Sunny	72	95	Weak	No
9	Sunny	69	70	Weak	Yes
10	Rain	75	80	Weak	Yes
11	Sunny	75	70	Strong	Yes
12	Overcast	72	90	Strong	Yes
13	Overcast	81	75	Weak	Yes
14	Rain	71	80	Strong	No

Notes on Decision Tree Induction algorithms

1. **Optimal Decision Tree:** Finding an optimal decision tree is an NP-complete problem. Hence, decision tree induction algorithms employ a heuristic based approach to search for the best in a large search space. Majority of the algorithms follow a greedy, top-down recursive divide-and-conquer strategy to build decision trees.

2. **Missing data and noise:** Decision tree induction algorithms are quite robust to the data set with missing values and presence of noise. However, proper data pre-processing can be followed to nullify these discrepancies.

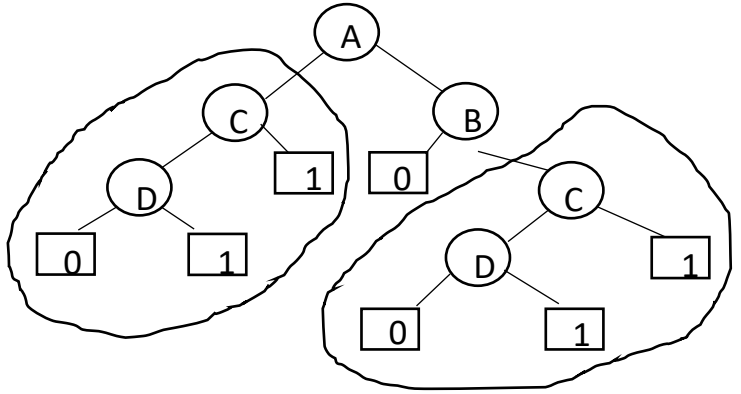
3. **Redundant Attributes:** The presence of redundant attributes does not adversely affect the accuracy of decision trees. It is observed that if an attribute is chosen for splitting, then another attribute which is redundant is unlikely to chosen for splitting.

4. **Computational complexity:** Decision tree induction algorithms are computationally inexpensive, in particular, when the sizes of training sets are large, Moreover, once a decision tree is known, classifying a test record is extremely fast, with a worst-case time complexity of O(d), where d is the maximum depth of the tree.

Notes on Decision Tree Induction algorithms

5. **Data Fragmentation Problem:** Since the decision tree induction algorithms employ a top-down, recursive partitioning approach, the number of tuples becomes smaller as we traverse down the tree. At a time, the number of tuples may be too small to make a decision about the class representation, such a problem is known as the data fragmentation. To deal with this problem, further splitting can be stopped when the number of records falls below a certain threshold.

6. **Tree Pruning:** A sub-tree can replicate two or more times in a decision tree (see figure below). This makes a decision tree unambiguous to classify a test record. To avoid such a sub-tree replication problem, all sub-trees except one can be pruned from the tree.



Notes on Decision Tree Induction algorithms

7. **Decision tree equivalence:** The different splitting criteria followed in different decision tree induction algorithms have little effect on the performance of the algorithms. This is because the different heuristic measures (such as information gain (α), Gini index (γ) and Gain ratio (β) are quite consistent with each other); also see the figure below.