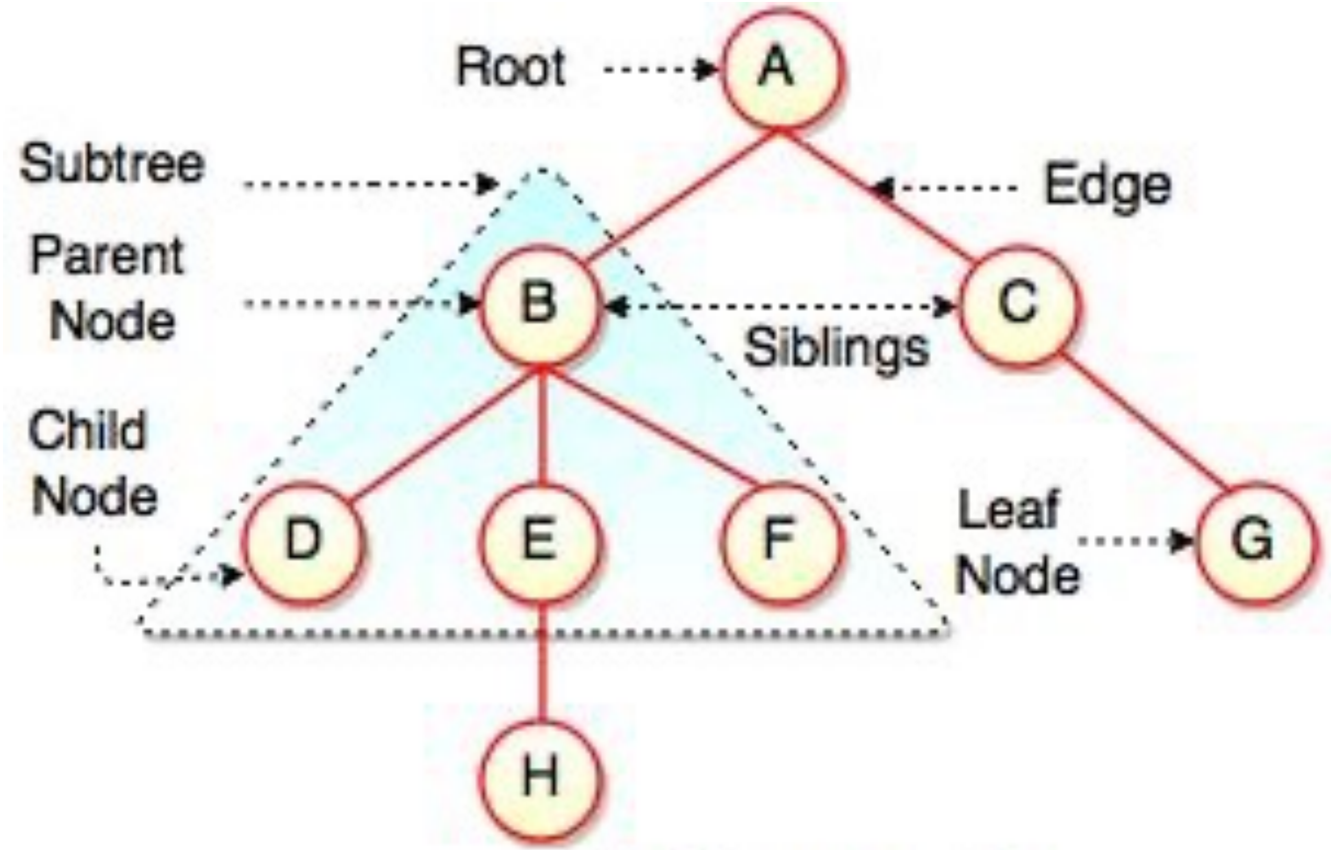
Decision Trees

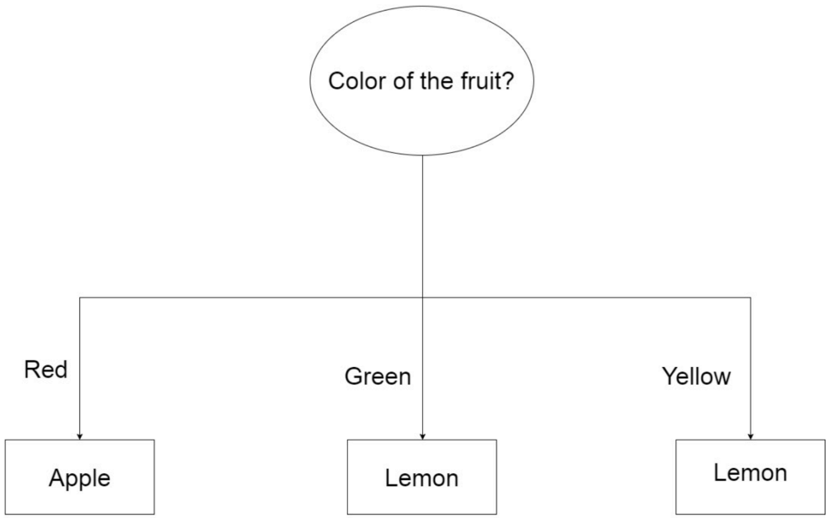
Slides are prepared from several information sources on the web and books

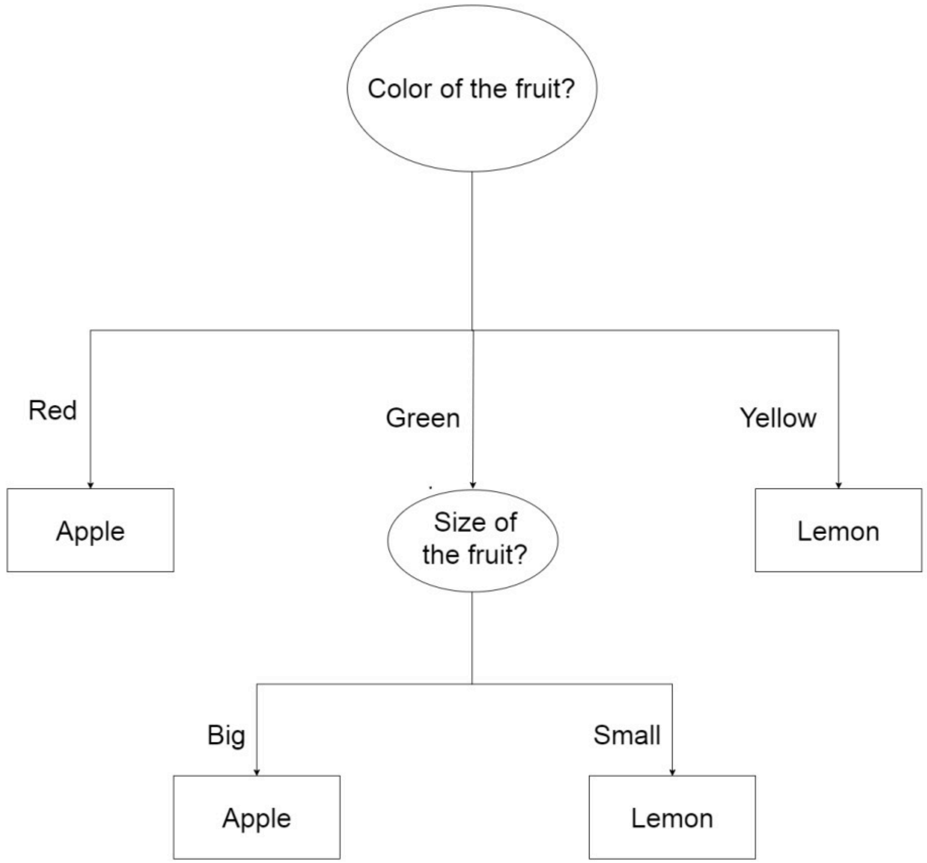
What are trees?



Decision Trees

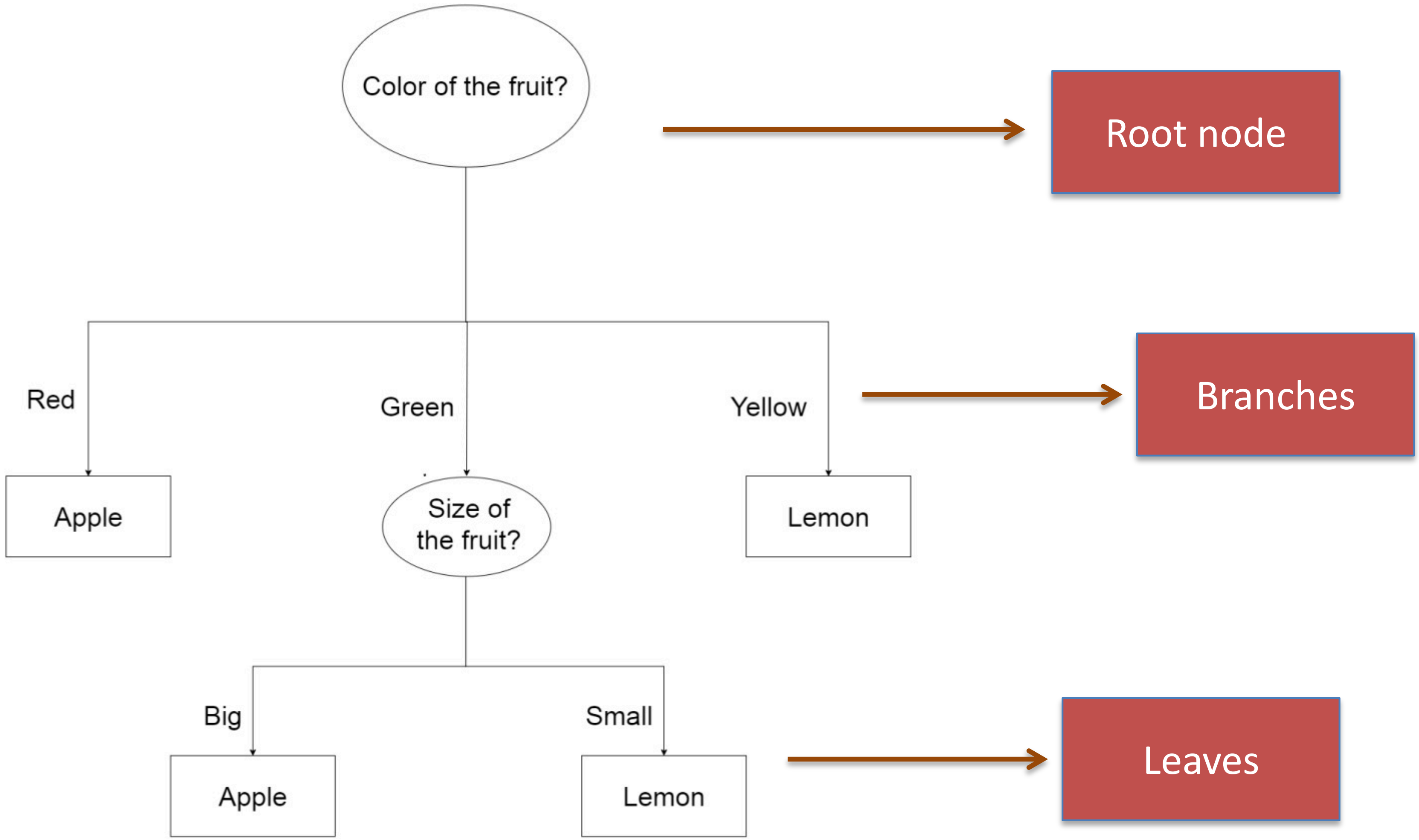
- Classify between lemon and apples





Images from https://machinelearning.school.blog/2017/01/12/introduction-to-decision-trees/

Decision Trees

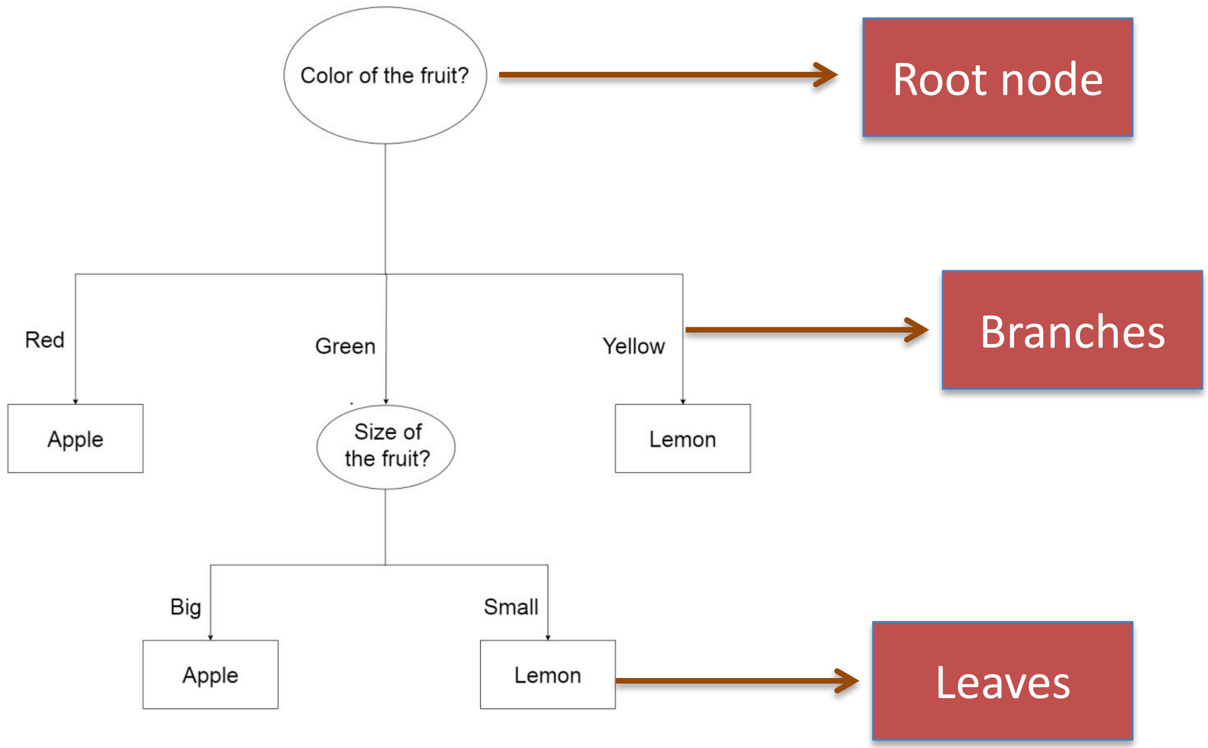


Images from https://machinelearning.school.blog/2017/01/12/introduction-to-decision-trees/

Rules for classifying data using attributes

- The tree consists of decision nodes and leaf nodes.
- A decision node has two or more branches, each representing values for the attribute tested.
- A leaf node attribute produces a homogeneous result (all in one class), which does not require additional classification testing

$\mathcal{F}$ – Decision Trees $\qquad f(X_1, X_2, X_3) \in \mathcal{F}$



Each internal node: tests one feature Xi

Each branch from a node: selects one value for Xi

Each leaf node: prediction for Y

Features can be discrete, continuous or categorical

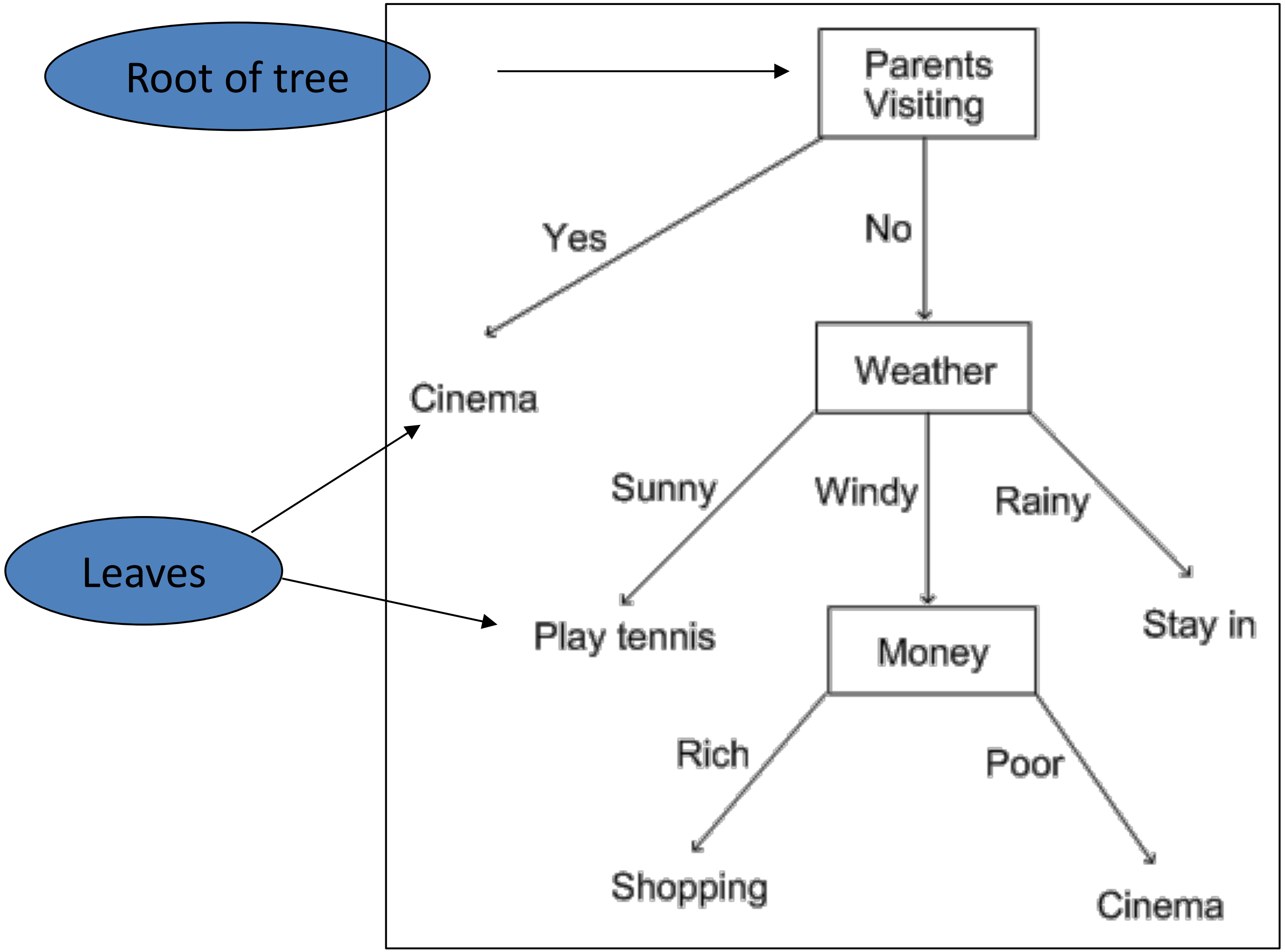
Images from https://machinelearning.school.blog/2017/01/12/introduction-to-decision-trees/

- Features can be discrete, continuous or categorical
- Each internal node: test some set of features {Xi}
- Each branch from a node: selects a set of value for {Xi}
- Each leaf node: prediction for Y

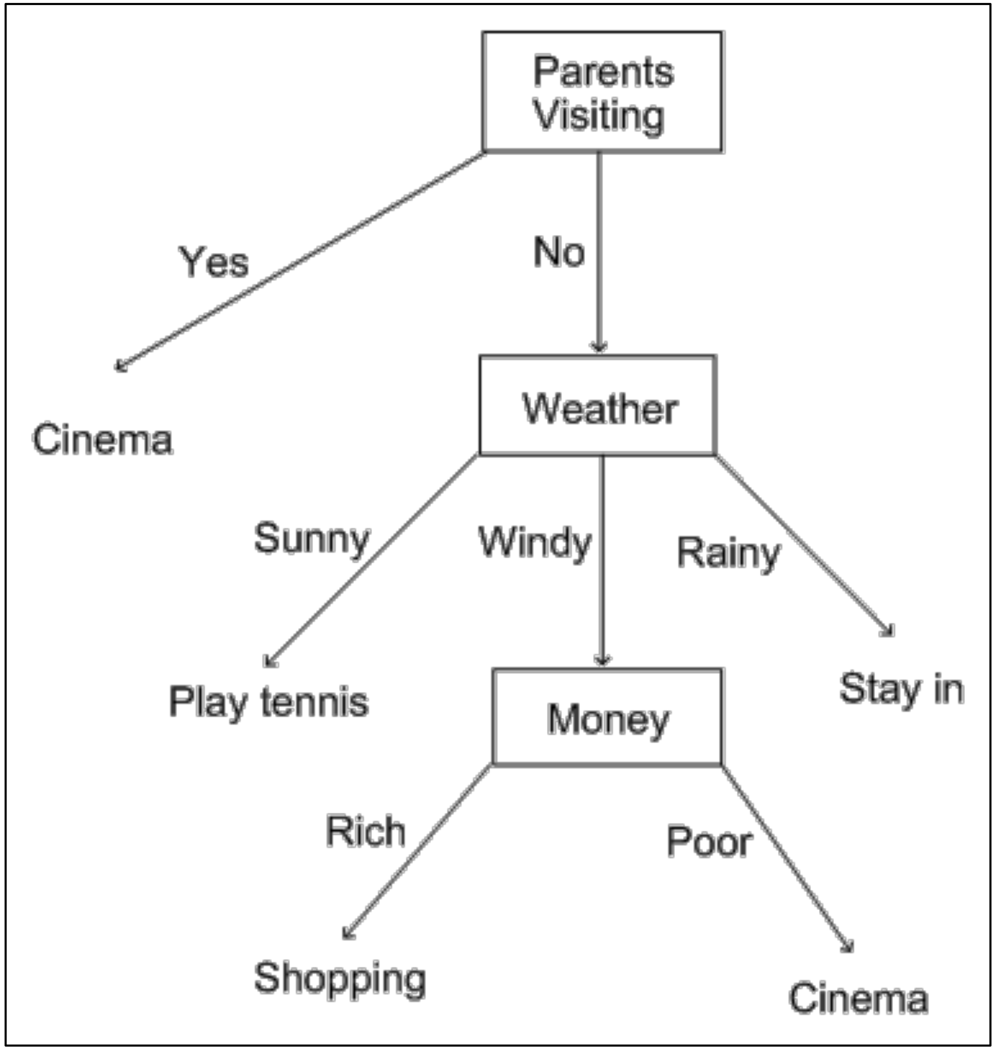
Example: What to do this Weekend?

- If my parents are visiting
  - We'll go to the cinema
- If not
  - Then, if it's sunny I'll play tennis
  - But if it's windy and I'm rich, I'll go shopping
  - If it's windy and I'm poor, I'll go to the cinema
  - If it's rainy, I'll stay in

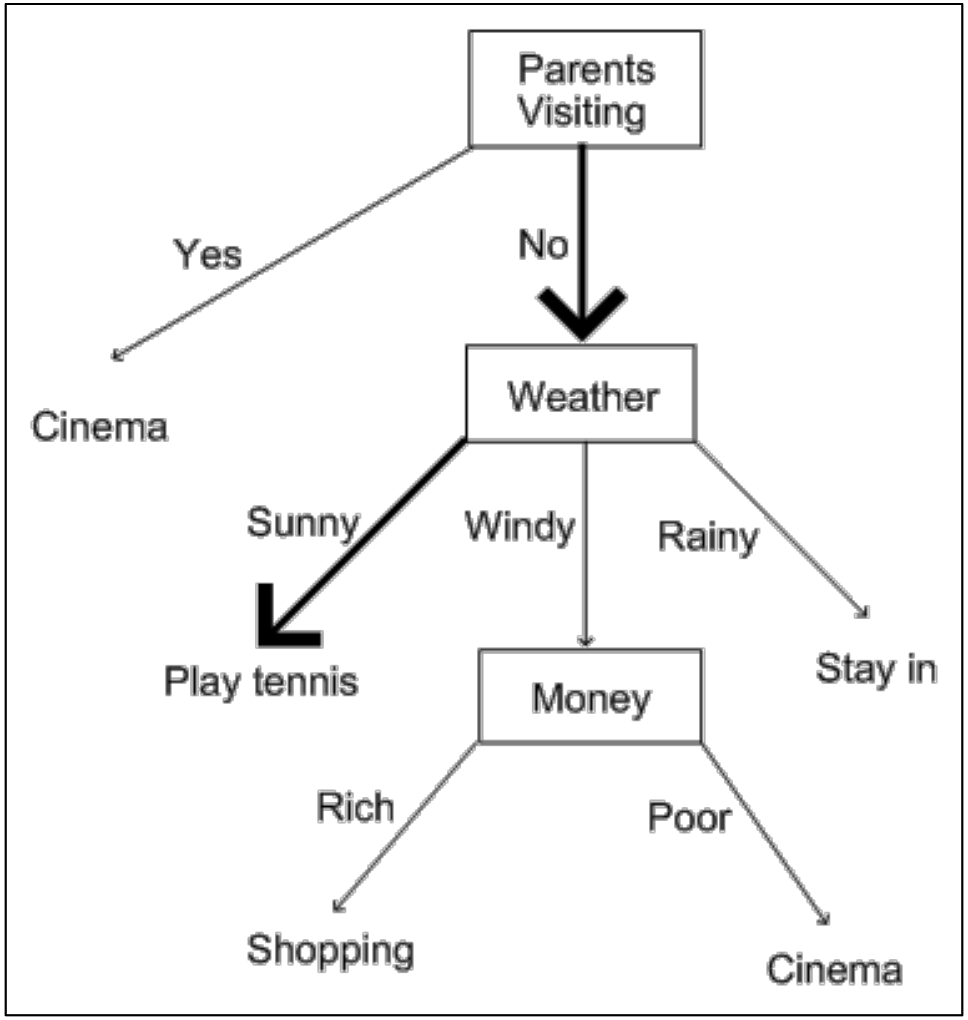
Written as a Decision Tree



Using the Decision Tree (No parents on a Sunny Day)



Using the Decision Tree (No parents on a Sunny Day)



From Decision Trees to Logic

- Read from the root to every tip
  - If this and this and this … and this, then do this

- In our example:
  - If no_parents and sunny_day, then play_tennis
  - no_parents $\wedge$ sunny_day ➔ play_tennis

How to design a decision tree

- Decision tree can be seen as rules for performing a categorisation
  - E.g., “what kind of weekend will this be?”

- Remember that we’re learning from examples
  - Not turning thought processes into decision trees

- The major question in decision tree learning is
  - Which nodes to put in which positions
  - Including the root node and the leaf nodes

What do you think: how should we compute which nodes to put in which positions?

The ID3 Algorithm

- Invented by J. Ross Quinlan in 1979
- ID3 uses a measure called Information Gain
  - Used to choose which node to put next
- Node with the highest information gain is chosen
  - When there are no choices, a leaf node is put on
- Builds the tree from the top down, with no backtracking
- Information Gain is used to select the most useful attribute for classification

Entropy – General Idea

- From Tom Mitchell's book:
  - "In order to define information gain precisely, we begin by defining a measure commonly used in information theory, called entropy that characterizes the (im)purity of an arbitrary collection of examples"

- A notion of impurity in data
- A formula to calculate the homogeneity of a sample
- A completely homogeneous sample has entropy of 0
- An equally divided sample has entropy of 1

Entropy - Formulae

- Given a set of examples, S
- For example, in a binary categorization
  - Where p_+ is the proportion of positives
  - And p_- is the proportion of negatives

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

- For examples belonging to classes c_1 to c_n
  - Where p_n is the proportion of examples in c_n

$$Entropy(S) \equiv \sum_{i=1}^{n} -p_i \log_2 p_i$$

Entropy Example

PlayTennis: training examples

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

Entropy Example

Entropy(S) =

- (9/14) Log2 (9/14) - (5/14) Log2 (5/14)

= 0.940

Information Gain (IG)

- 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 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 the data is classified.

Information Gain (cont'd)

- Calculate Gain(S,A)
  - Estimate the reduction in entropy we obtain if we know the value of attribute A for the examples in S

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

An Example Calculation of Information Gain

- Suppose we have a set of examples
  - $S = \{s_1, s_2, s_3, s_4\}$
  - In a binary categorization
    - With one positive example and three negative examples
    - The positive example is s_1
- And Attribute A
  - Which takes values v_1, v_2, v_3
- S_1 takes value v_2 for A, S_2 takes value v_2 for A

  S_3 takes value v_3 for A, S_4 takes value v_1 for A

First Calculate Entropy(S)

- Recall that

  $Entropy(S) = -p_+\log_2(p_+) – p_-\log_2(p_-)$

- From binary categorisation, we know that

  $p_+ = ¼$ and $p_- = ¾$

- Hence, $Entropy(S) = -(1/4)\log_2(1/4) – (3/4)\log_2(3/4)$

  $= 0.811$

Calculate Gain for each Value of A

- Remember that

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

- And that S_v = {set of example with value V for A}
  - So, $S_{v1} = \{s_4\}$, $S_{v2} = \{s_1, s_2\}$, $S_{v3}=\{s_3\}$

- Now, $(|S_{v1}|/|S|) * \text{Entropy}(S_{v1})$
  $= (1/4) * (-(0/1)*\log_2(0/1)-(1/1)\log_2(1/1))$
  $= (1/4) * (0 - (1)\log_2(1)) = (1/4)(0-0) = 0$

- Similarly, $(|S_{v2}|/|S|) = 0.5$ and $(|S_{v3}|/|S|) = 0$

Final Calculation

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

- So, we add up the three calculations and take them from the overall entropy of S:

- Final answer for information gain:
  - Gain(S,A) = 0.811 – (0.25*0 +1/2*1 + 0*0.25) = 0.311

A Worked Example

Weekend	Weather	Parents	Money	Decision (Category)
W1	Sunny	Yes	Rich	Cinema
W2	Sunny	No	Rich	Tennis
W3	Windy	Yes	Rich	Cinema
W4	Rainy	Yes	Poor	Cinema
W5	Rainy	No	Rich	Stay in
W6	Rainy	Yes	Poor	Cinema
W7	Windy	No	Poor	Cinema
W8	Windy	No	Rich	Shopping
W9	Windy	Yes	Rich	Cinema
W10	Sunny	No	Rich	Tennis

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

Information Gain for All of S

- S = {W1,W2,…,W10}
- Firstly, we need to calculate:
  - Entropy(S) = … = 1.571
- Next, we need to calculate information gain
  - For all the attributes we currently have available
    - (which is all of them at the moment)
  - Gain(S, weather) = 0.7
  - Gain(S, parents) = 0.61
  - Gain(S, money) = 0.2816

The ID3 Algorithm

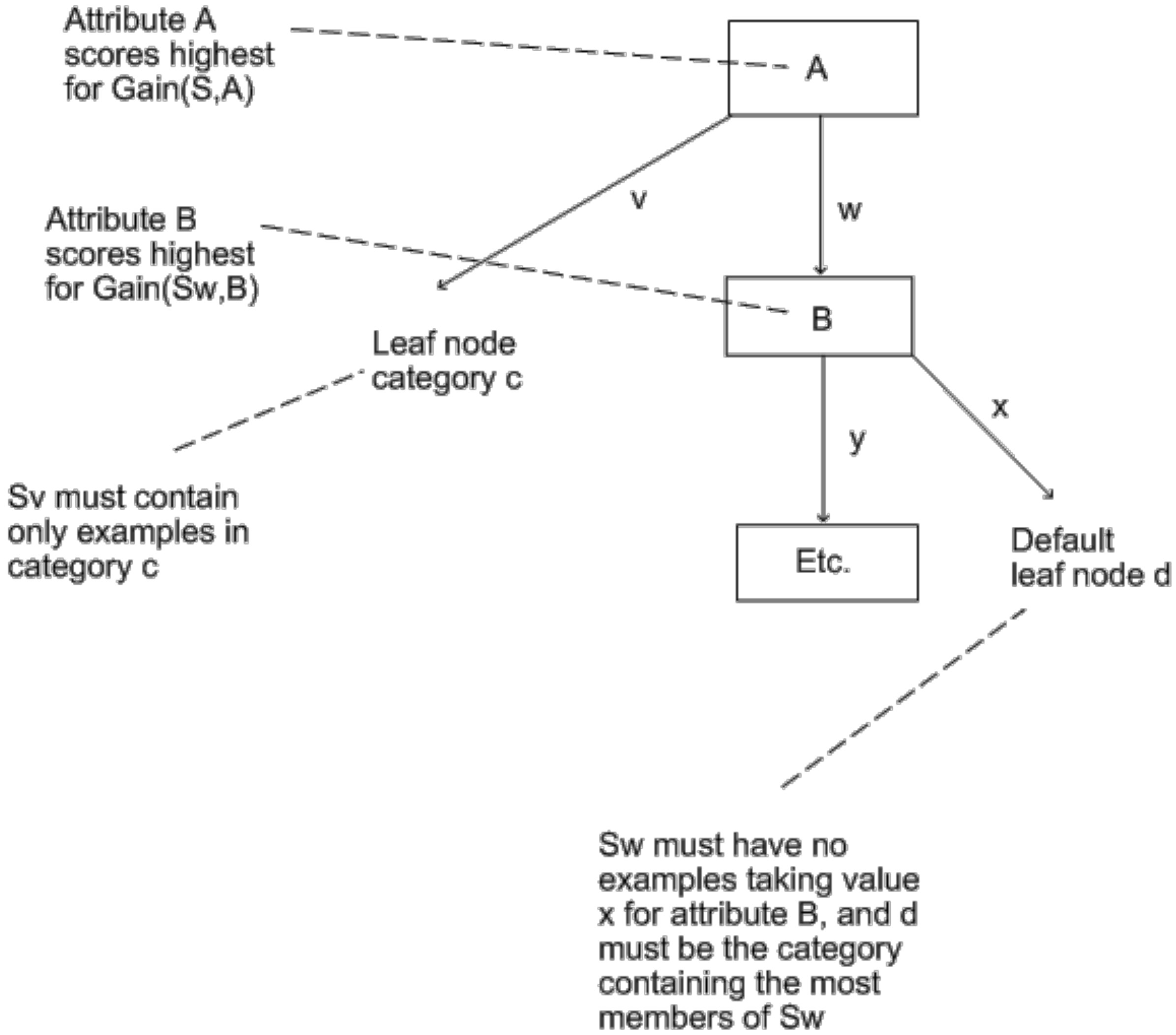
- Given a set of examples, S
  - Described by a set of attributes A_i
  - Categorised into categories c_j

1. Choose the root node to be attribute A
   - Such that A scores highest for information gain
     - Relative to S, i.e., gain(S,A) is the highest over all attributes
2. For each value v that A can take
   - Draw a branch and label each with corresponding v

The ID3 Algorithm

- For each branch you've just drawn (for value v)
  - If S_v only contains examples in category c
    - Then put that category as a leaf node in the tree
  - If S_v is empty
    - Then find the default category (which contains the most examples from S)
      - Put this default category as a leaf node in the tree
  - Otherwise
    - Remove A from attributes which can be put into nodes
    - Replace S with S_v
    - Find new attribute A scoring best for Gain(S, A)
    - Start again at part 2
- Make sure you replace S with S_v

Explanatory Diagram



A Worked Example

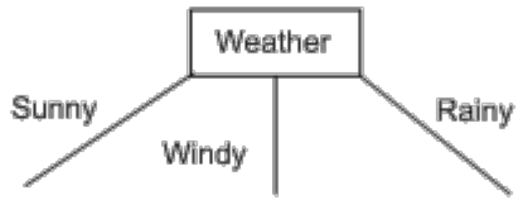
Weekend	Weather	Parents	Money	Decision (Category)
W1	Sunny	Yes	Rich	Cinema
W2	Sunny	No	Rich	Tennis
W3	Windy	Yes	Rich	Cinema
W4	Rainy	Yes	Poor	Cinema
W5	Rainy	No	Rich	Stay in
W6	Rainy	Yes	Poor	Cinema
W7	Windy	No	Poor	Cinema
W8	Windy	No	Rich	Shopping
W9	Windy	Yes	Rich	Cinema
W10	Sunny	No	Rich	Tennis

Information Gain for All of S

- S = {W1,W2,…,W10}
- Firstly, we need to calculate:
  - Entropy(S) = … = 1.571
- Next, we need to calculate information gain
  - For all the attributes we currently have available
    - (which is all of them at the moment)
  - Gain(S, weather) = … = 0.7
  - Gain(S, parents) = … = 0.61
  - Gain(S, money) = … = 0.2816
- Hence, the weather is the first attribute to split on
  - Because this gives us the biggest information gain

Top of the Tree

- So, this is the top of our tree:
- Now, we look at each branch in turn
  - In particular, we look at the examples with the attribute prescribed by the branch
- S_{sunny} = {W1,W2,W10}
  - Categorisations are cinema, tennis and tennis for W1,W2 and W10
  - What does the algorithm say?
    - Set is neither empty, nor a single category
    - So we have to replace S by S_{sunny} and start again



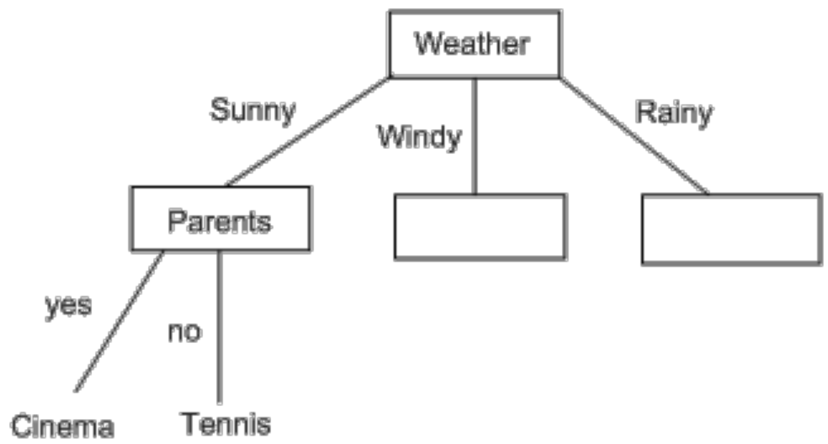
Working with S_{sunny}

Weekend	Weather	Parents	Money	Decision
W1	Sunny	Yes	Rich	Cinema
W2	Sunny	No	Rich	Tennis
W10	Sunny	No	Rich	Tennis

- Need to choose a new attribute to split on
  - Cannot be weather, of course – we've already had that
- So, calculate information gain again:
  - Gain(S_{sunny}, parents) = … = 0.918
  - Gain(S_{sunny}, money) = … = 0
- Hence we choose to split on parents

Getting to the leaf nodes

- If it’s sunny and the parents have turned up
  - Then, looking at the table in previous slide
    - There’s only one answer: go to cinema
- If it’s sunny and the parents haven’t turned up
  - Then, again, there’s only one answer: play tennis
- Hence our decision tree looks like this:

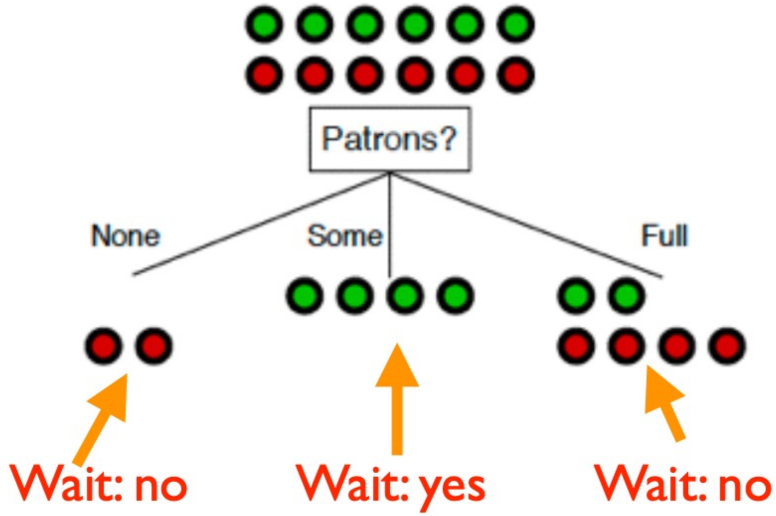


What is the optimal Tree Depth?

- We need to be careful to pick an appropriate tree depth.
- If the tree is too deep, we can overfit.
- If the tree is too shallow, we underfit
- Max depth is a hyper-parameter that should be tuned by the data. Alternative strategy is to create a very deep tree, and then to prune it.

Control the size of the tree

- If we stop early, not all training samples would be classified correctly.
- How do we classify a new instance:
  - We label the leaves of this smaller tree with the majority of training samples' labels



Summary of learning classification trees

- Advantages:
  - Easily interpretable by human (as long as the tree is not too big)
  - Computationally efficient
  - Handles both numerical and categorical data
  - It is parametric thus compact: unlike Nearest Neighborhood Classification, we do not have to carry our training instances around Building block for various ensemble methods (more on this later)
- Disadvantages
  - Heuristic training techniques
  - Finding partition of space that minimizes empirical error is NP-hard.
  - We resort to greedy approaches with limited theoretical underpinning.

Feature Space

- Suppose that we have p explanatory variables X1, . . . , Xp and n observations.

  – a numeric variable: n − 1 possible splits

  – an ordered factor: k − 1 possible splits

  – an unordered factor: −→ 2(k−1) − 1 possible splits.

Measures of Impurity

- At each node *i* of a classification tree, we have a probability distribution p_{ik} over *k* classes.

$$\hat{p}_{ik} = \frac{n_{ik}}{n_{i.}}$$

- Deviance: $D = \sum D_i$, where $D_i = -2\sum_k n_{ik} \log(p_{ik})$.
- Entropy: $\sum p_{ik} \log(p_{ik})$.
- Gini index: $\sum_{j \neq k} p_{ij} p_{ik} = 1 - \sum_k p_{ik}^2$.
- Residual sum of squares

$$D = \sum_{\text{cases } j} \left(y_j - \mu_{[j]}\right)^2$$

where $\mu_{[j]}$ is the mean of the values in the node that case j belongs to.

Pruning Rules

- Stop when one instance in each leaf (regression problem)
- Stop when all the instance in each leaf have the same label (classification problem)
- Stop when the number of leaves is less than the threshold
- Stop when the leaf's error is less than the threshold
- Stop when the number of instances in each leaf is less than the threshold
- Stop when the p-value between two divided leaves is larger than the certain threshold (e.g. 0.05 or 0.01) based on chosen statistical tests.

Thanks.

Machine Learning by Tom Mitchell