Decision Trees (Wrap-up) and Linear Models

CS771: Introduction to Machine Learning

Piyush Rai

Plan for today

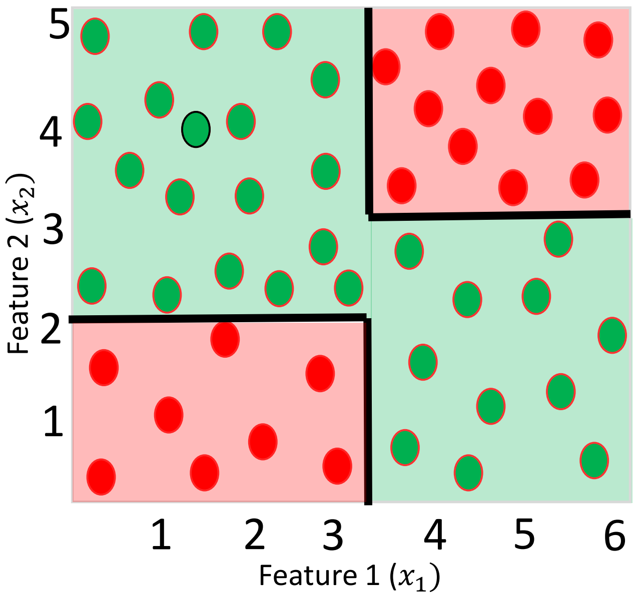
- Wrap-up the discussion of decision trees
  - How to learn decision trees from training data
- Introduction to Linear Models

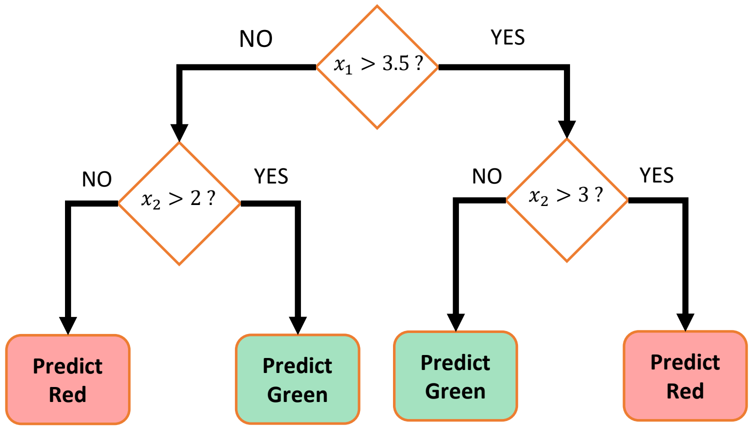
Constructing Decision Trees

Given some training data, what's the "optimal" DT?

How to decide which rules to test for and in what order?

How to assess informativeness of a rule?





In general, constructing DT is an intractable problem (NP-hard)

Often we can use some "greedy" heuristics to construct a "good" DT

The rules are organized in the DT such that most informative rules are tested first

Hmm.. So DTs are like the "20 questions" game (ask the most useful questions first)

Informativeness of a rule is of related to the extent of the purity of the split arising due to that rule. More informative rules yield more pure splits

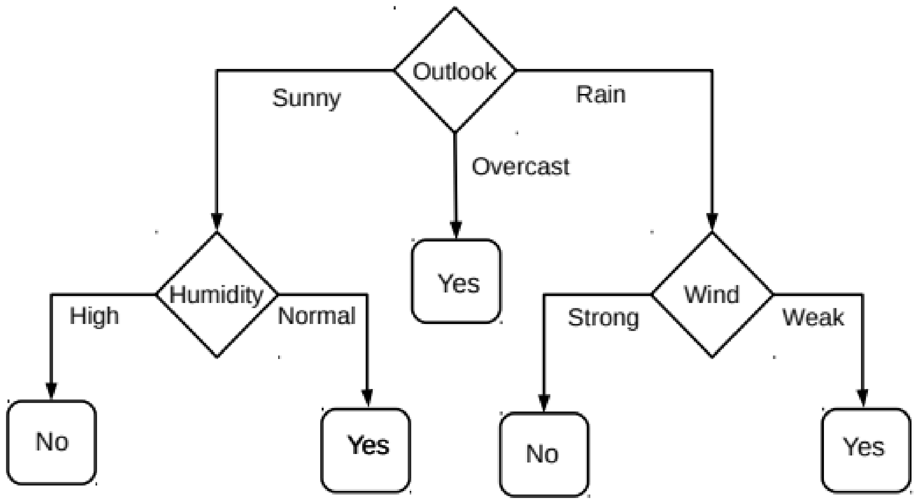
To do so, we use the training data to figure out which rules should be tested at each node

The same rules will be applied on the test inputs to route them along the tree until they reach some leaf node where the prediction is made

Decision Tree Construction: An Example

- Let's consider the playing Tennis example
- Assume each internal node will test the value of one of the features

day	outlook	temperature	humidity	wind	play
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rain	mild	high	weak	yes
5	rain	cool	normal	weak	yes
6	rain	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
8	sunny	mild	high	weak	no
9	sunny	cool	normal	weak	yes
10	rain	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes
14	rain	mild	high	strong	no

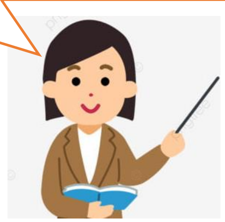


- Question: Why does it make more sense to test the feature "outlook" first?
- Answer: Of all the 4 features, it's the most informative
  - It has the highest information gain as the root node

Entropy and Information Gain

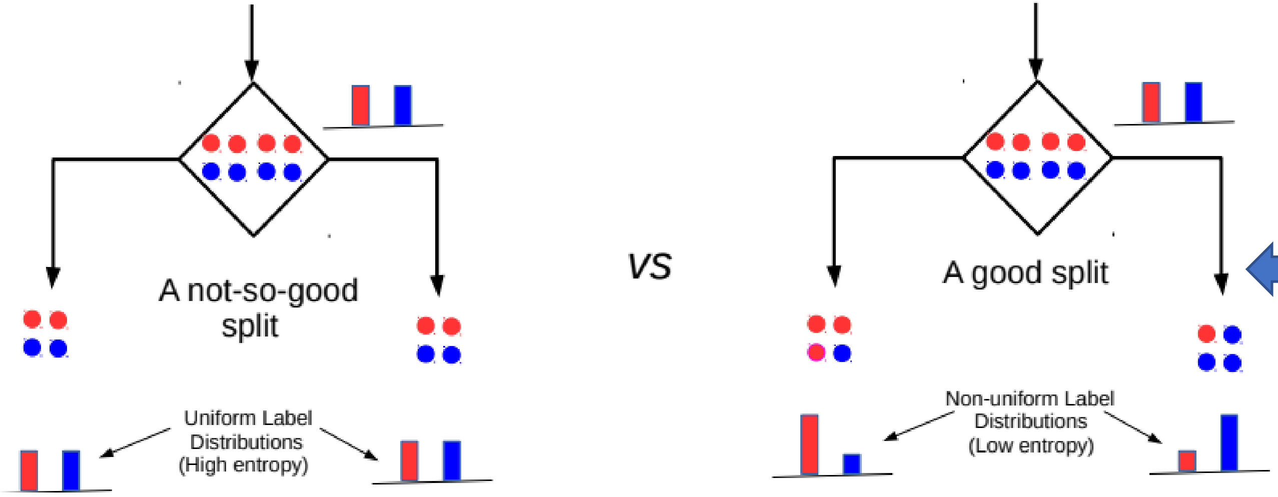
- Assume a set of labelled inputs $\boldsymbol{S}$ from C classes, p_c as fraction of class c inputs
- Entropy of the set $\boldsymbol{S}$ is defined as $\mathrm{H}(\boldsymbol{S}) = -\sum_{c \in C} p_c \log p_c$
- Suppose a rule splits $\boldsymbol{S}$ into two smaller disjoint sets $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$
- Reduction in entropy after the split is called information gain

Uniform sets (all classes roughly equally present) have high entropy; skewed sets low



$$IG = H(S) - \frac{|S_1|}{|S|} H(S_1) - \frac{|S_2|}{|S|} H(S_2)$$

This split has a low IG (in fact zero IG)



This split has higher IG

Entropy and Information Gain

- Let's use IG based criterion to construct a DT for the Tennis example
- At root node, let's compute IG of each of the 4 features
- Consider feature "wind". Root contains all examples $S = [9+,5-]$

day	outlook	temperature	humidity	wind	play
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rain	mild	high	weak	yes
5	rain	cool	normal	weak	yes
6	rain	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
8	sunny	mild	high	weak	no
9	sunny	cool	normal	weak	yes
10	rain	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes
14	rain	mild	high	strong	no

$$H(S) = -(9/14)\log_2(9/14) - (5/14)\log_2(5/14) = 0.94$$

$$S_{\text{weak}} = [6+, 2-] \Rightarrow \mathrm{H}(S_{\text{weak}}) = 0.811$$

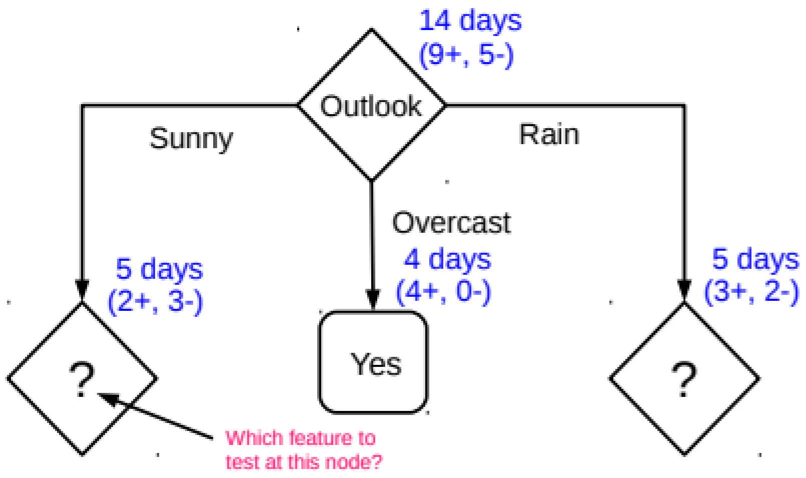
$$S_{\text{strong}} = [3+, 3-] \Rightarrow \mathrm{H}(S_{\text{strong}}) = 1$$

$$IG(S, wind) = H(S) - \frac{|S_{\text{weak}}|}{|S|} H(S_{\text{weak}}) - \frac{|S_{\text{strong}}|}{|S|} H(S_{\text{strong}}) = 0.94 - 8/14 * 0.811 - 6/14 * 1 = 0.048$$

- Likewise, at root: IG(S, outlook) = 0.246, IG(S, humidity) = 0.151, IG(S,temp) = 0.029
- Thus we choose "outlook" feature to be tested at the root node
- Now how to grow the DT, i.e., what to do at the next level? Which feature to test next?
- Rule: Iterate - for each child node, select the feature with the highest IG

Growing the tree

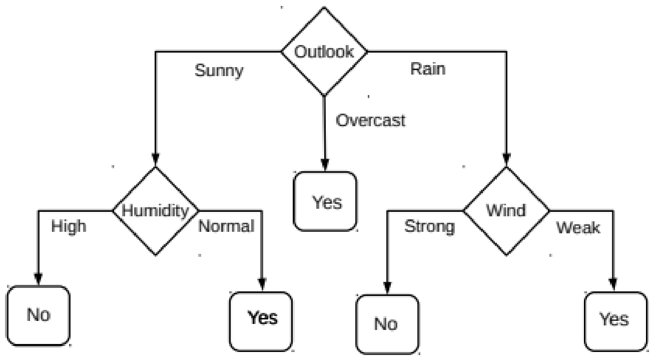
day	outlook	temperature	humidity	wind	play
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rain	mild	high	weak	yes
5	rain	cool	normal	weak	yes
6	rain	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
8	sunny	mild	high	weak	no
9	sunny	cool	normal	weak	yes
10	rain	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes
14	rain	mild	high	strong	no



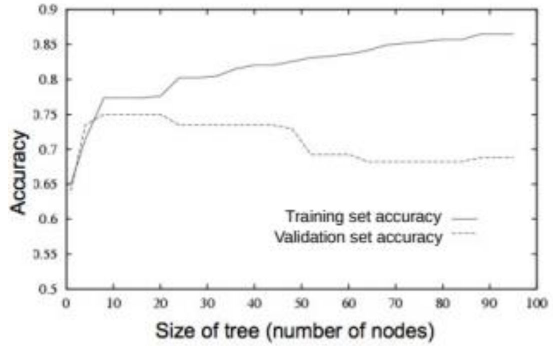
- Proceeding as before, for level 2, left node, we can verify that
  - IG(S,temp) = 0.570, IG(S, humidity) = 0.970, IG(S, wind) = 0.019
- Thus humidity chosen as the feature to be tested at level 2, left node
- No need to expand the middle node (already “pure” - all “yes” training examples ☺)
- Can also verify that wind has the largest IG for the right node
- Note: If a feature has already been tested along a path earlier, we don’t consider it again

When to stop growing the tree?

day	outlook	temperature	humidity	wind	play
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rain	mild	high	weak	yes
5	rain	cool	normal	weak	yes
6	rain	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
8	sunny	mild	high	weak	no
9	sunny	cool	normal	weak	yes
10	rain	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes
14	rain	mild	high	strong	no

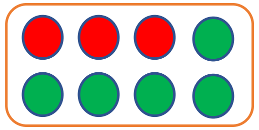


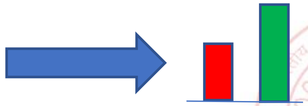
- Stop expanding a node further (i.e., make it a leaf node) when
  - It consist of all training examples having the same label (the node becomes "pure")
  - We run out of features to test along the path to that node
  - The DT starts to overfit (can be checked by monitoring the validation set accuracy)



To help prevent the tree from growing too much!

- Important: No need to obsess too much for purity
  - It is okay to have a leaf node that is not fully pure, e.g., this





OR

  - At test inputs that reach an impure leaf, can predict probability of belonging to each class (in above example, p(red) = 3/8, p(green) = 5/8), or simply predict the majority label

Avoiding Overfitting in DTs

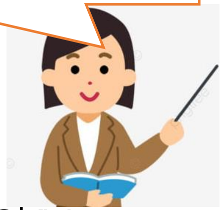
- Desired: a DT that is not too big in size, yet fits the training data reasonably
- Note: An example of a very simple DT is "decision-stump"
  - A decision-stump only tests the value of a single feature (or a simple rule)
  - Not very powerful in itself but often used in large ensembles of decision stumps
- Mainly two approaches to prune a complex DT
  - Prune while building the tree (stopping early)
  - Prune after building the tree (post-pruning)

Either can be done using a validation set

- Criteria for judging which nodes could potentially be pruned
  - Use a validation set (separate from the training set)
    - Prune each possible node that doesn't hurt the accuracy on the validation set
    - Greedily remove the node that improves the validation accuracy the most
    - Stop when the validation set accuracy starts worsening
  - Use model complexity control, such as Minimum Description Length (will see later)

Decision Trees: Some Comments

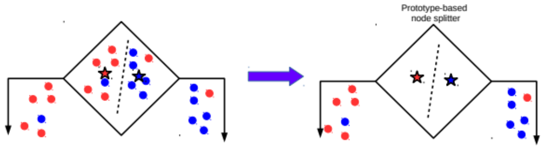
For regression, outputs are real-valued and we don't have a "set" of classes, so quantities like entropy/IG/gini etc. are undefined



- Gini-index defined as $\sum_{c=1}^{C} p_c(1 - p_c)$ can be an alternative to IG
- For DT regression[1], variance in the outputs can be used to assess purity

An illustration on the next slide

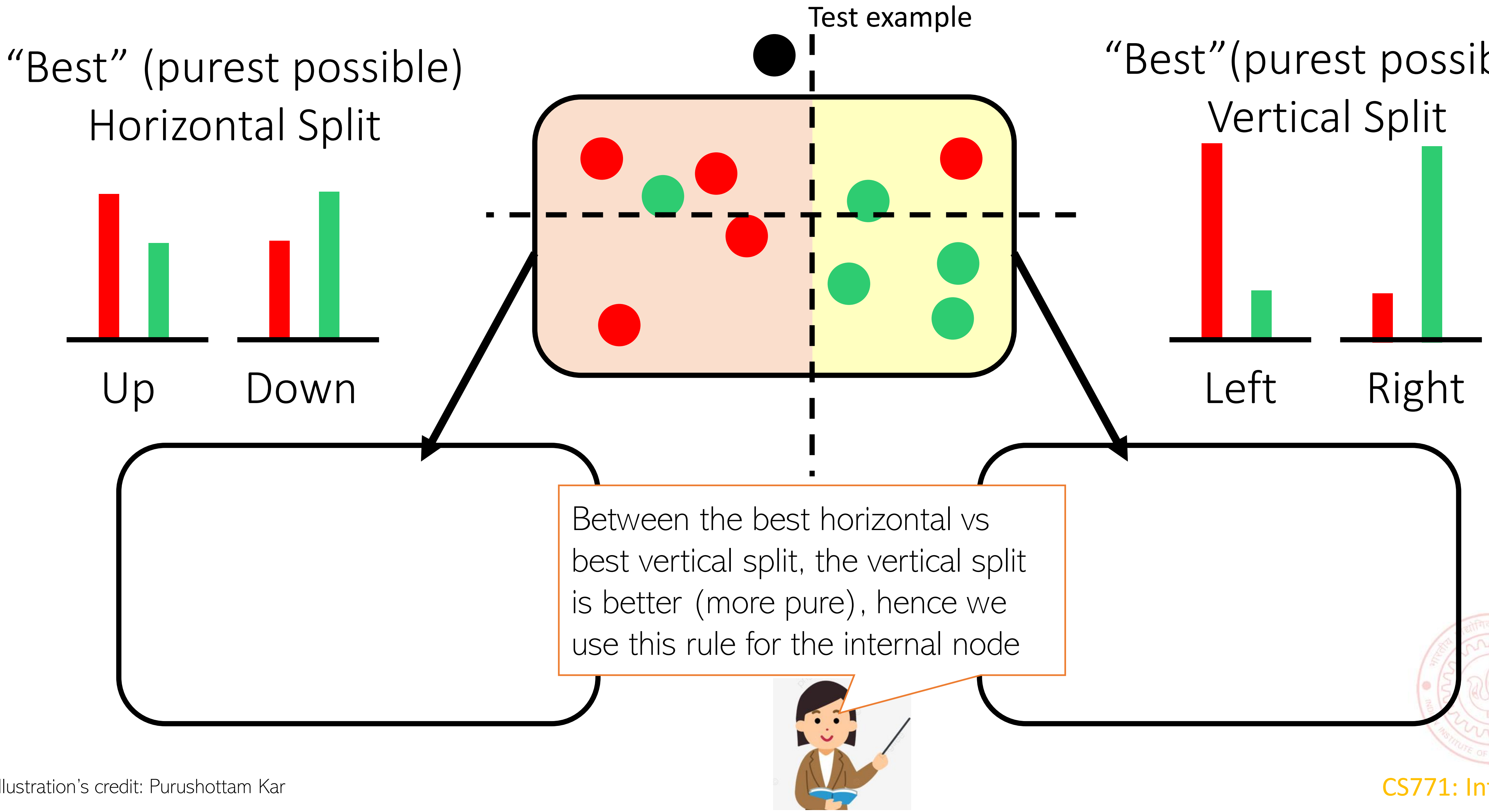
- When features are real-valued (no finite possible values to try), things are a bit more tricky
  - Can use tests based on thresholding feature values (recall our synthetic data examples)
  - Need to be careful w.r.t. number of threshold points, how fine each range is, etc.
- More sophisticated decision rules at the internal nodes can also be used



  - Basically, need some rule that splits inputs at an internal node into homogeneous groups
  - The rule can even be a machine learning classification algo (e.g., LwP or a deep learner)
  - However, in DTs, we want the tests to be fast so single feature based rules are preferred
- Need to take care handling training or test inputs that have some features missing

[1]Breiman, Leo; Friedman, J. H.; Olshen, R. A.; Stone, C. J. (1984). Classification and regression trees

An Illustration: DT with Real-Valued Features



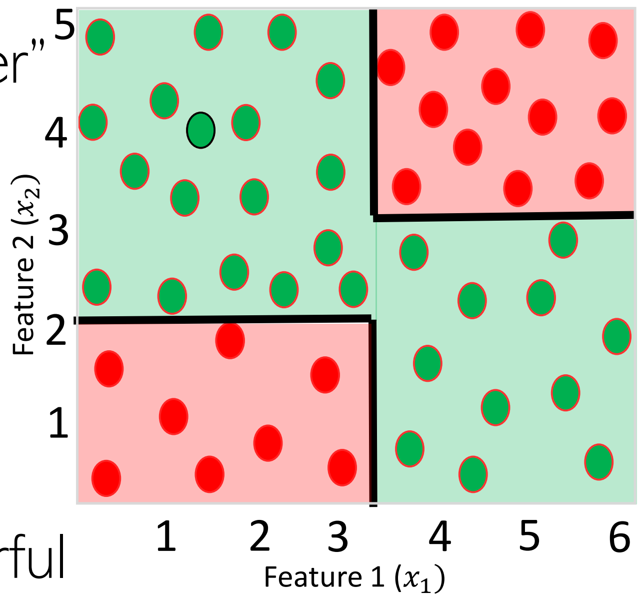
This illustration's credit: Purushottam Kar

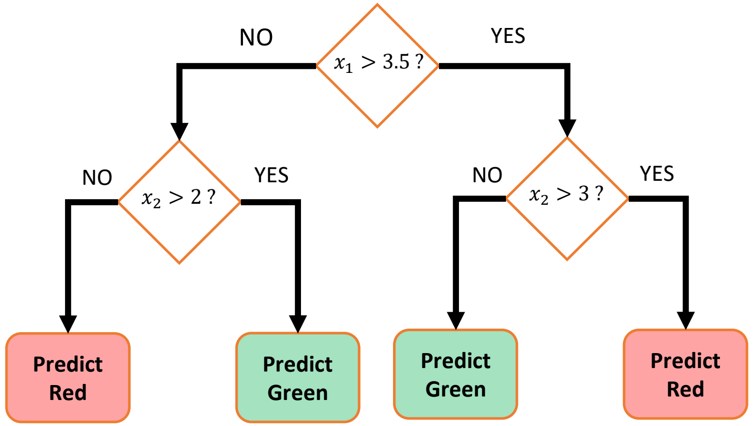
Decision Trees: A Summary

.. thus helping us learn complex rule as a combination of several simpler rules

Some key strengths:

- Simple and easy to interpret
- Nice example of "divide and conquer" paradigm in machine learning
- Easily handle different types of features (real, categorical, etc.)
- Very fast at test time
- Multiple DTs can be combined via ensemble methods: more powerful (e.g., Decision Forests; will see later)
- Used in several real-world ML applications, e.g., recommender systems, gaming (Kinect)





Human-body pose estimation

Some key weaknesses:

- Learning optimal DT is (NP-hard) intractable. Existing algos mostly greedy heuristics
- Can sometimes become very complex unless some pruning is applied