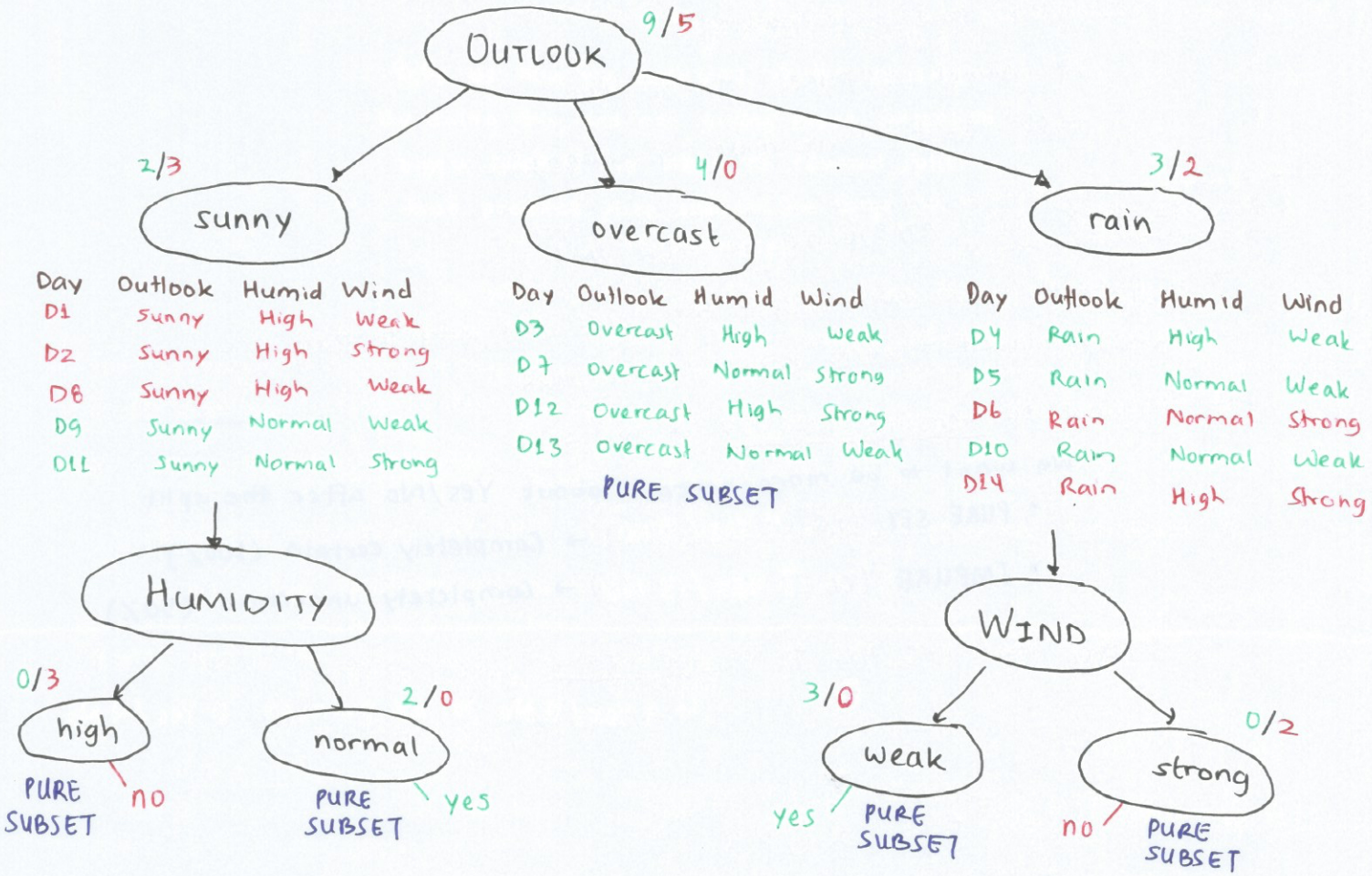
WEEK 3

DECISION TREES

→ i.e Predict if John will play tennis.

1. DISCRETE DATA

- 14 training examples → 9 yes, 5 no — Attributes are: Outlook, Humidity, Wind
- Divide-and-conquer:
  - split into subsets
  - are they pure? — YES: stop; NO: repeat



Day	Outlook	Humid	Wind
D1	Sunny	High	Weak
D2	Sunny	High	Strong
D8	Sunny	High	Weak
D9	Sunny	Normal	Weak
D11	Sunny	Normal	Strong

Day	Outlook	Humid	Wind
D3	Overcast	High	Weak
D7	Overcast	Normal	Strong
D12	Overcast	High	Strong
D13	Overcast	Normal	Weak

PURE SUBSET

Day	Outlook	Humid	Wind
D4	Rain	High	Weak
D5	Rain	Normal	Weak
D6	Rain	Normal	Strong
D10	Rain	Normal	Weak
D14	Rain	High	Strong

- We have partitioned our entire training set into small subsets based on small set of rules, and each subset is completely pure.

→ The algo. tht. builds this data struct. is called the **ID3 algorithm.**

ID3 ALGORITHM

↓ Start at root node

- Split (node, {examples}):

1. A ← the **best attribute** for splitting the {examples}
2. Decision attribute for this node ← A
3. FOR each value of A, create new child node
4. Split training {examples} to child nodes
5. FOR each child node/subset:
6. IF subset is pure, THEN stop
7. ELSE Split (child_node, {subset})

HOW TO PICK BEST ATTRIBUTE TO SPLIT ON?

→ In general, you want a split tht. gives you subsets that are heavily biased towards either (+)ve / (-)ve.

↳ Want to measure PURITY of the split.

- We want to be more certain about YES/No after the split
  - PURE SET (i.e. 4 YES / 0 No) → Completely certain (100%)
  - IMPURE (i.e. 3 YES / 3 No) → Completely uncertain (50%)

↳ This measure has to be symmetric [Can't use P("yes" | set)]

↳ 4 Yes / 0 No is as pure as 0 Yes / 4 No

ENTROPY

→ $$H(S) = - P_{(+)} \log_2 P_{(+)} - P_{(-)} \log_2 P_{(-)} \text{ bits}$$

S: subset of training examples; $P_{(+)}$: proportion of (+)ve examples in S; $P_{(-)}$: proportion of (-)ve examples in S

→ Assume item X belongs to subset S, we can interpret this as the no. of bits to overcome uncertainty. A pure subset doesn't require any bits.

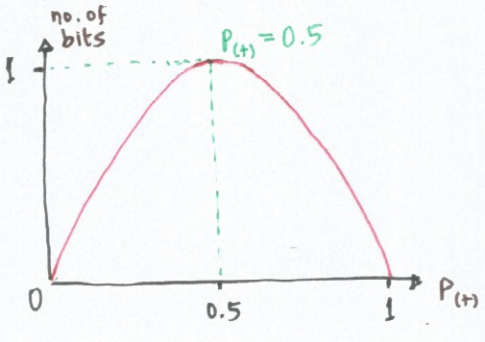
i.e. 3 YES / 3 No

$$H(S) = -\frac{3}{6} \log_2 \frac{3}{6} - \frac{3}{6} \log_2 \frac{3}{6}$$

$$= 1 \text{ bit}$$

4 YES / 0 No

$$H(S) = -\frac{4}{4} \log_2 \frac{4}{4} - \frac{0}{4} \log_2 \frac{0}{4} = 0 \text{ bit}$$



INFORMATION GAIN

→ At each node, we are going to have a bunch of children so we're going to have multiple subsets. So, we are going to take an average of the entropies for each one of the subsets.

$$Gain(S, A) = H(S) - \sum_{V \in Values(A)} \frac{|S_V|}{|S|} \cdot H(S_V)$$

- $|S_V|$: Size of the subset
- $|S|$: Total no. of examples before split
- V – possible values of attr. A
- S – set of examples $\{X\}$
- S_V – subset where $X_A = V$

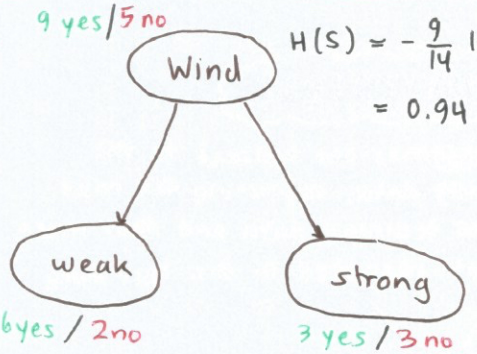
Tells us how big each one of the subsets. By doing this, we are giving more weight to bigger subsets; we want to reward splits that produce pure subset for large no. of examples.

↳ What is the avg. entropy after the split, biased towards the larger subsets?

→ The difference between the entropy before the split and entropy after the split is called **information gain.**

- In CS/Statistics, we call it **mutual information** between attribute A that we're splitting on and the class labels of S

i.e. 9 yes / 5 no



$$H(S) = -\frac{9}{14}\log_2\frac{9}{14} - \frac{5}{14}\log_2\frac{5}{14} = 0.94$$

$$H(S_{weak}) = 0.81 \qquad H(S_{strong}) = 1.0$$

$$Gain(S, Wind) = H(S) - \frac{8}{14}H(S_{weak}) - \frac{6}{14}H(S_{strong})$$

$$= 0.94 - \frac{8}{14}(0.81) - \frac{6}{14}(1.0)$$

$$= \boxed{0.049}$$

→ ~~What the gain tells you is that~~ Before the split, you were pretty uncertain on which things were (+)ve and which were (−)ve, and after the split, the gain tells you how much more certain the (+)ve & (−)ve become if you pick that attribute

→ So, you want to compute information gain for each attribute, then select the attribute w/ the highest information gain ↳ reduce uncertainty the most

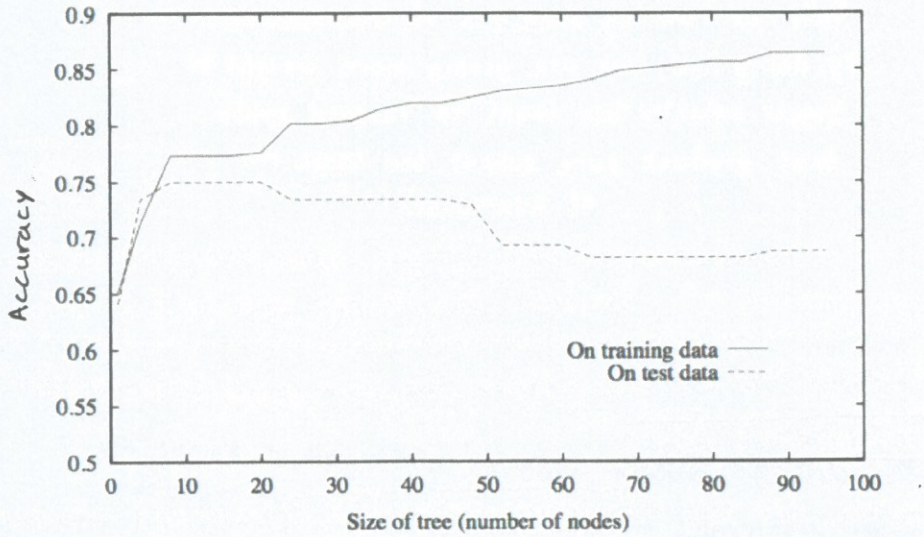
OVERFITTING IN DTs

→ The good and bad thing abt DT is if you run the algo., it will always classify training examples perfectly.

WHY? → We will keep splitting data until we end up w/ singletons. These singleton subsets are pure by defn

The problem about this is once we have split up data where we have singleton subsets, we won't have confidence in our predictions.

↳ Basically, we're saying any data pt. tht. falls into this leaf is going to be (+)ve bc. we have 1 positive example in the same branch as the data pt.



→ At first, as tree gets bigger, you get better accuracy on test data. (that's the data the algo. hasn't seen yet)

But then, accuracy starts decreasing → OVERFITTING!!

DT becomes too tailored to the training data, capturing the noise in the training data and that noise is not repeated in the test data

→ TO build an effective DT, you want to stop it before it gets too specific to the training data.

PRUNING

HOW TO AVOID OVERFITTING

1) Stop splitting when it is not statistically significant

2) A better approach is to use a validation subset for post-pruning the tree

   a) Split entire training set into 2 parts

      ↳ One part is hidden frm. the algo. → Going to be the validation set

      ↳ The other part is used to train a full decision tree on it (until you get the max. accuracy on the training data)

   b) Run the validation set through the DT and obtain accuracy

      ↳ Accuracy for test data won't be 100% bc. our tree has overfit to the training data

   c) For each node, pretend to remove node and all its children from the tree and obtain accuracy on validation set

   d) Remove node that gives highest accuracy on the validation set and repeat until further pruning is harmful

      ↳ At some pt. you will end up making your tree too small where accuracy will start decreasing

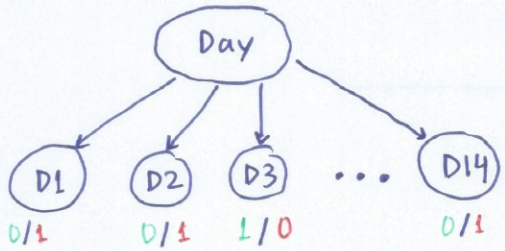
NOTE:

- This is a greedy approach; it is difficult to check all combinations to cut the tree.
- So, you always pick the sub-tree tht. gives you maximum increase in accuracy on validation set, commit to that cut and find the next cut. (Can't do it optimally → exponential no. of nodes)

PROBLEM W/ INFORMATION GAIN

→ Biased towards attributes w/ many values:

i.e. Split on attribute 'Day'



→ All subsets, by defn, are perfectly pure

→ As far as info. gain is concerned, this is a really good attribute to split on

↳ Won't work for new data (i.e. D15 Rain High Weak)

→ To resolve this, we use Information Gain Ratio:

$$SplitEntropy(S, A) = -\sum_{v \in Values(A)} \frac{|S_v|}{|S|} \log \frac{|S_v|}{|S|}$$

(A: candidate attribute)

↳ High values if end up w/ lots of tiny subsets of the data
Low values if end up w/ few, large subsets of the data

$$GainRatio(S, A) = \frac{Gain(S, A)}{SplitEntropy(S, A)}$$

↳ Penalizes attributes w/ many values

DISJUNCTIVE NORMAL FORM (DNF) FORMULAS

→ DTs are interpretable; They make you see the logic for the data to interpret (not like black-box algo. like Neural Networks)

↳ can be shown to a client

→ i.e. Rule:

(Outlook = Overcast) ∨
(Outlook = Rain ∧ Wind = Weak) ∨
(Outlook = Sunny ∧ Humidity = Normal)

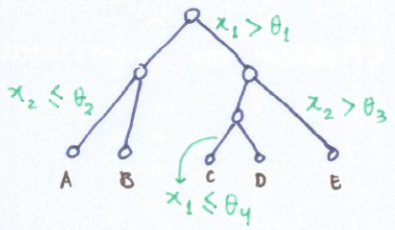
CONTINUOUS ATTRIBUTES

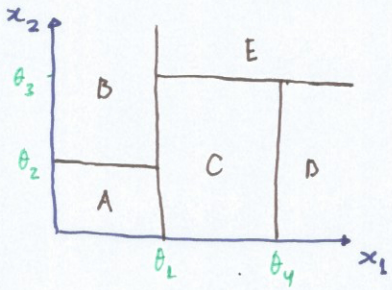
→ DT tries to pick a threshold when dealing w/ real-valued data

→ Say an attribute is height, then DT will sweep all possible values of height. Compute entropy / info. gain just like dealing w/ discrete data.

↳ Difference is we need to add a search over the possible values of a threshold

↳ Do this for ALL attributes





↳ Notice tht. you can end up using the same attribute more than once in a tree. → Not possible for categorical attributes!

2.

MULTI-CLASS CLASSIFICATION

→ For more than 2 classes, we define entropy differently:

$$H(S) = -\sum_c P_{(c)} \log_2 P_{(c)}$$

$P_{(c)}$: proportion of the subset which is labelled as class c

→ If leaf is impure, the decision is based on the most frequent class in the subset

3.

REGRESSION

→ No classes → Can't use entropy

→ Hence instead of maximizing gain, we try to find subsets such that each subset has the smallest variance

→ The predicted output is the average of the training examples in the subset

OR

We can perform a linear regression on the training examples in the subset in order to build a linear model that predicts the value.

PROS & CONS OF DT

PROS	CONS
1) Interpretable • Humans can understand decisions	1) ID3 algo. is greedy, hence it may not find best tree • Can't find optimal soln bc. there are exponentially many possible trees
2) Easily handles irrelevant attributes i.e. Gain = 0	2) Only axis-aligned splits of data (Problem for numerical data)
3) Can handle missing data	
4) Very compact • # of nodes << D after pruning	
5) Very fast at testing time • O(depth)	

RANDOM FOREST ALGORITHM

→ Instead of growing 1 tree on your dataset, we grow K different decision trees:

1) Pick a random subset S_r of training examples
2) Use it to grow a full ID3 tree T_r (No pruning!)
   → When splitting data, you pick from a subset of attributes at random
   → Compute gain based on subset S_r instead of full set
3) Repeat for $r = 1, \ldots, K$

→ Given a new data pt. X:

1) Classify X using each of the trees $T_1, \ldots, T_K$
2) Use majority votes
   ↳ take the class that was predicted most often

Take Test: Decision Trees Quiz

Test Information

Description: Self-assessment quiz for the "Decision Trees" lecture material

Instructions

Multiple Attempts: This Test allows multiple attempts.

Force Completion: This Test can be saved and resumed later.

Your answers are saved automatically.

* Question Completion Status:

QUESTION 1 — 10 points — Save Answer

Which of the following are names of Decision Tree algorithms?

- [ ] Maximum Information Gain
- [x] C4.5
- [x] CaRT
- [ ] Greedy
- [ ] Divide and Conquer
- [x] ID3

QUESTION 2 — 10 points — Save Answer

In the ID3 algorithm, when considering a node that splits on attribute A, a child node is created for (choose one):

- ( ) the child nodes which show the maximum entropy
- (x) each value of attribute A occuring in the training data
- ( ) each value of attribute A occuring in the validation data
- ( ) the child nodes which give the maximum information gain
- ( ) the child nodes which are pure

QUESTION 3 — 10 points — Save Answer

The Decision Tree algorithm builds a tree of nodes, using N training instances. If we prune the tree at level L (i.e., L levels down from the root node, where L is zero for the root node), then the total number of training instances we find at the leaves (level L, and higher if a set at a node is pure) is:

→ All the training instances are represented at every level, it is how they are divided that changes

- ( ) L
- ( ) L+N
- ( ) 2^L
- ( ) N unless there is pruning in which case some number less than N, depending on what is pruned
- (x) N
- ( ) We don't know - the particular splits that happen are data dependent, so the number allocated to each node at each level depends on the training data

QUESTION 4 — 10 points — Save Answer

When the Decision Tree algorithm is building the decision tree, it looks at a node and creates a number of child nodes by splitting on an attribute. A good algorithm tries to pick an attribute to split on where (choose one):

- ( ) the purity of the child nodes is highest
- ( ) the number of positives is highest
- ( ) child nodes are either very pure or very impure, but not in between
- ( ) the entropy of the node is the highest
- (x) the purity of the attribute is highest

QUESTION 5 — 10 points — Save Answer

For the entropy measure we use to evaluate the purity of a split, which of the following are true (select all that are true)

- [x] The maximum entropy is 1 and the minimum 0
- [ ] Entropy is high when the split is pure and low when it is impure
- [x] Entropy is low when the split is pure and high when it is impure
- [x] Entropy indicates the randomness of the subgroup
- [ ] Entropy at a node can only ever increase when we split
- [x] Entropy at a node can only ever decrease when we split

- [ ] The maximum entropy is 0 and the minimum entropy is -1
- [x] Entropy indicates the purity of the subgroup

QUESTION 6 **10 points** Save Answer

The entropy of a node which has a proportion p of positive training instances and a proportion q of negative training instances is (so $p + q = 1$)(select one):

- $-(p\log_2 p - q\log_2 q)$
- $p\log_2 p + q\log_2 q$
- [x] $-(p\log_2 p + q\log_2 q)$
- $+p\log_2 p + q\log_2 q$

QUESTION 7 **10 points** Save Answer

When we are computing purity of a group of instances, entropy is measured [in bits]

QUESTION 8 **10 points** Save Answer

In the formulation of Information Gain for node S and attribute A:

$$Gain(S,A) = H(S) - \sum_{V \in Values(A)} \frac{|S_V|}{|S|} H(S_V)$$

match up the terms and expressions with their names and equivalent values:

I [H(S)]
C [$|S_v|$]
H [$|S|$]
G [$\sum_{V \in Values(A)} \frac{|S_v|}{|S|} H(S_v)$]

A. weighted sum of entropies of child nodes
B. informaton gain of node S
C. the number of instances in the child node corresponding to value V for attribute A
D. the number of values for node A
E. the increase in entropy due to the split of S on attribute A
F. the decrease in entropy due to the split of S on attribute A

F [Gain(S,A)]

G. weighted average of entropies of child nodes
H. the number of instances in node S
I. entropy of the node S
J. the number of attributes in node S

QUESTION 9 **10 points** Save Answer

We use information gain to pick [the attribute to split the node on]. We pick the attribute with the [highest info. gain]. However, information gain tends to pick [the attr. w/ lots of values]. To avoid this, we can use [the gain ratio].

QUESTION 10 **10 points** Save Answer

The ID3 (and related) Decision Tree algorithms are prone to overfitting because

- [x] whenever you have an impure node, the agorithm will split it, until all the leaves are pure, so you will have a perfect fit
- [ ] the training data is representative of the population, but fitting it perfectly does not allow you to deal with new data
- [x] if you run the algorithm to conclusion it will fit the training data perfectly
- [ ] if you run the algorithm to conclusion it will fit the testing data perfectly
- [x] the training data is not representative of the whole population, so fitting it perfectly will bias your decisions towards its peculiarities
- [ ] if you could run the algorithm long enough it would generalise to the testing data, but that is intractable

QUESTION 11 **10 points** Save Answer

To avoid overfitting, you can do the following. First, obtain [a validation set of instances]. Then evaluate your tree on this set and remove [the node and its descendants which when removed shows the greatest ↑ in performance on this set]. Repeat this process until [no further improvement happens no matter which node is removed]

QUESTION 12 **10 points** Save Answer

If an attribute can take a continuous value, then

- ✓ It can be included in a decision tree by splitting using greater-than or less-than
- It can be included in a decision tree but only if it appears in multiple splits
- It can be included in a decision tree by discretising the values
- It can't be included in a decision tree
- It can be included in a decision tree but only if it appears in a single split

QUESTION 13 **10 points** Save Answer

Decision trees cannot be used for multiclass classification

- True
- ✓ False

QUESTION 14 **10 points** Save Answer

A multiclass Decision Tree predicts (pick one):

- based on minimising the variance in the leaf node
- ✓ the most frequent class in the leaf node reached
- using a linear fit to the instances in the leaf node reached

QUESTION 15 **10 points** Save Answer

When using a decision tree for a regression problem, we can (choose all that apply):

- ✓ use the average of the training examples in the subset as the prediction
- perform linear regression on the training examples in order to build a linear model that predicts the value
- use maximisation of variance in the subsets instead of maximisation of gain, in order to choose what to split on
- ✓ use minimisation of variance in the subsets instead of maximisation of gain, in order to choose what to split on
- ✓ perform linear regression on the training examples in the subset in order to build a linear model that predicts the value

- use the most frequent value in the training examples in the subset as the prediction

QUESTION 16 **10 points** Save Answer

In the random forest method we

- fit a linear regression to the predictions of the trees in order to get the overall prediction
- build many decision trees on all the training data, each one based on splitting on a different initial attribute
- average over the predictions of the trees in the forest
- ✓ build many decision trees, each one based on a different random subset of the training data
- ✓ take a majority vote of the trees in the forest to get the prediction
- build a forest of decision trees, each one trained on the ones that the preceding trees failed to classify correctly

Click Save and Submit to save and submit. Click Save All Answers to save all answers.