Machine Learning

CSE 142

Xin (Eric) Wang

Wednesday, October 20, 2021

T
o
d
a
y

- Decision trees, Ch. 5

Decision trees

Review

- Tree models can be used for classification, ranking, probability estimation, regression, and clustering
- Recursive generic tree learning procedure:

Algorithm GrowTree(D, F) – grow a feature tree from training data.

100%?
99%?
80%?

Most useful feature

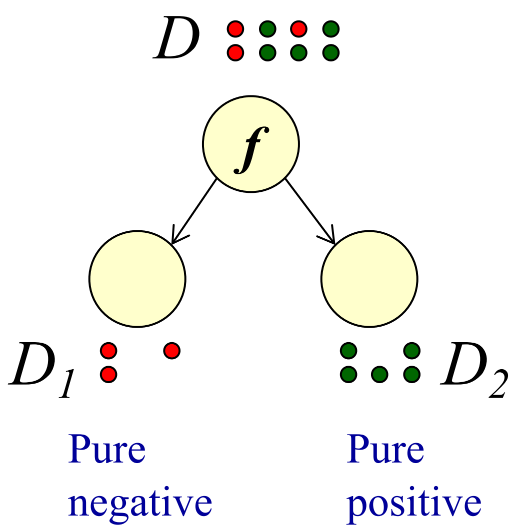
Input : data D; set of features F.
Output : feature tree T with labelled leaves.
if Homogeneous(D) **then return** Label(D);
$S \leftarrow$BestSplit(D, F) ; // e.g., BestSplit-Class (Algorithm 5.2)
split D into subsets D_i according to the literals in S;
for each i **do**
  if $D_i \neq \emptyset$ **then** $T_i \leftarrow$GrowTree(D_i, F) ;
  else T_i is a leaf labelled with Label(D);
end
return a tree whose root is labelled with S and whose children are T_i

Divide-and-conquer approach: build a tree for each subset of the data, then merge into a single tree

BestSplit

Review

- BestSplit(D, F) – what feature $f \in F$ will produce the best split (partitioning) of the training data $D = \{ D_i \}$?
- What's a good split/partitioning?
  - One that produces **pure** partitions D_i, each of which contains only instances from a single class
  - E.g., in a binary classification problem, D_1 has only positive examples and D_2 has only negative examples



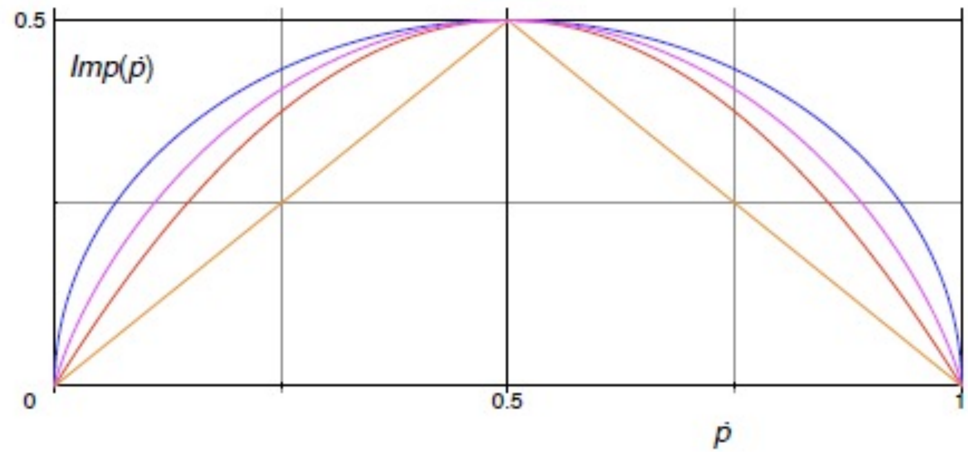
Both partitions are homogeneous

Which feature f is best for this?

How to measure partition purity if not completely homogeneous?

Impurity functions

Review



Minority class

$$\text{Imp}(\dot{p}) = \min(\dot{p}, 1-\dot{p})$$

Gini index

$$\text{Imp}(\dot{p}) = 2\dot{p}(1-\dot{p})$$

Entropy

$$\text{Imp}(\dot{p}) = -\dot{p}\log_2(\dot{p}) - (1-\dot{p})\log_2(1-\dot{p})$$

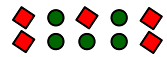
√Gini index

$$\text{Imp}(\dot{p}) = \sqrt{2\dot{p}(1-\dot{p})}$$

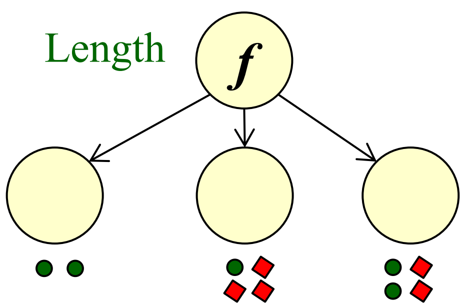
The total impurity for a data partitioning is just the weighted sum of each partition's impurity

$$\text{Imp}(\{D_1, \dots, D_l\}) = \sum_{i=1}^{l} \frac{|D_i|}{|D|} Imp(D_i)$$

Impurity example

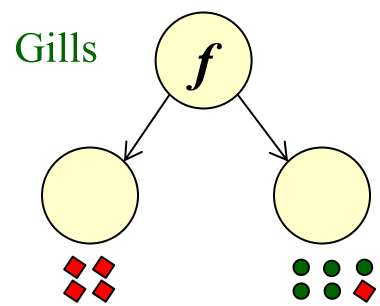
D 

Review



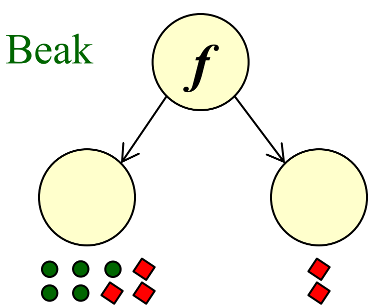
0.0 .81 1.0

$$\frac{2(0.0)+4(.81)+4(1.0)}{10} = 0.72$$



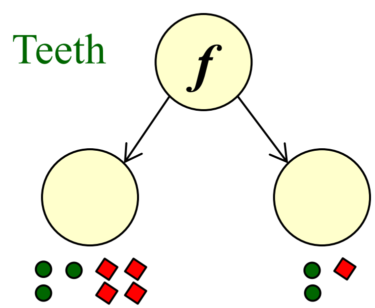
0.0 .65

$$\frac{4(0.0)+6(.65)}{10} = 0.39$$



.95 0.0

$$\frac{8(.95)+2(0.0)}{10} = 0.76$$



.99 .92

$$\frac{7(.99)+3(.92)}{10} = 0.97$$

Using the entropy measure

$$\mathrm{Imp}(\dot{p}) = -\dot{p}\log_2(\dot{p}) - (1-\dot{p})\log_2(1-\dot{p})$$

$$\mathrm{Imp}(\{D_1, \ldots, D_l\}) = \sum_{i=1}^{l} \frac{|D_i|}{|D|} Imp(D_i)$$

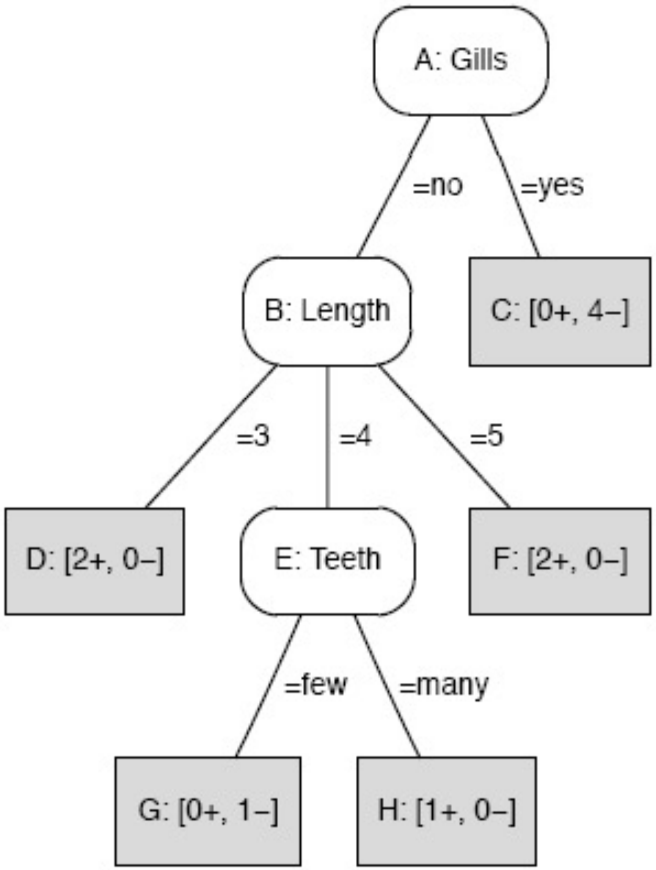
Which of these is the best feature to use?

This is the Gills feature in our dolphin example

DT for dolphin example

Training data (1-5: positive, 6-10: negative):

1. Length = 3 ∧ Gills = no ∧ Beak = yes ∧ Teeth = many
2. Length = 4 ∧ Gills = no ∧ Beak = yes ∧ Teeth = many
3. Length = 3 ∧ Gills = no ∧ Beak = yes ∧ Teeth = few
4. Length = 5 ∧ Gills = no ∧ Beak = yes ∧ Teeth = many
5. Length = 5 ∧ Gills = no ∧ Beak = yes ∧ Teeth = few
6. Length = 5 ∧ Gills = yes ∧ Beak = yes ∧ Teeth = many
7. Length = 4 ∧ Gills = yes ∧ Beak = yes ∧ Teeth = many
8. Length = 5 ∧ Gills = yes ∧ Beak = no ∧ Teeth = many
9. Length = 4 ∧ Gills = yes ∧ Beak = no ∧ Teeth = many
10. Length = 4 ∧ Gills = no ∧ Beak = yes ∧ Teeth = few



Choose the best feature based on minimizing impurity of the remaining data

$$\mathrm{Imp}(\dot{p}) = -\,\dot{p}\log_2(\dot{p}) - (1-\dot{p})\log_2(1-\dot{p})$$

Find the best DT split

Algorithm 5.2: BestSplit-Class(D, F) – find the best split for a decision tree.

Input : data D; set of features F.

Output : feature f to split on.

$I_{\text{min}} \leftarrow 1$;
for each $f \in F$ **do**
    split D into subsets $D_1, \ldots, D_l$ according to the values v_j of f;
    if $\text{Imp}(\{D_1, \ldots, D_l\}) < I_{\text{min}}$ **then**
        $I_{\text{min}} \leftarrow \text{Imp}(\{D_1, \ldots, D_l\})$;
        $f_{\text{best}} \leftarrow f$;
    end
end
return f_{best}

DT approach

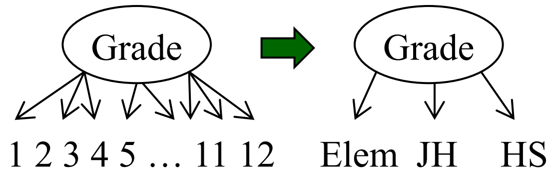
- We've described a greedy algorithm – it maximizes each individual choice, but it does not guarantee a global maximum
  - For this we would need the ability to backtrack and reconsider choices based on the total impurity

$$\mathrm{Imp}(\{D_1, \dots, D_m\}) = \sum_{i=1}^{m} \frac{|D_i|}{|D|} Imp(D_i)$$

- However, it works rather well in practice!
- We can modify the strategy slightly to deal with "messy" (non-separable) data and to limit the size of the tree
  - By not requiring a homogeneous data partition before stopping and assigning a label – i.e., the Homogeneous(D) function
  - E.g., if we have a feature separates as {1000+, 3–}, that may be good enough – no need to keep checking additional features

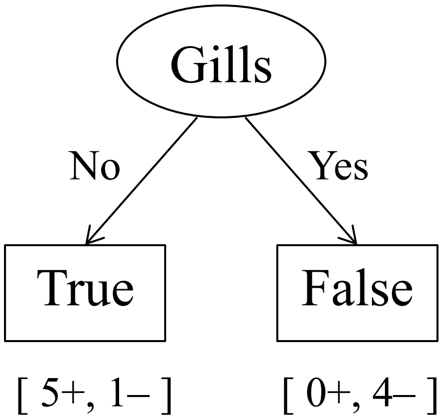
Simplifying decision trees

- Some ways to create simpler decision trees:
  - Merge feature labels and test the difference
  - Enforce a depth limit (maximum depth of ***d***)
  - Enforce a purity threshold – e.g., if the impurity of a node is $< \varepsilon$, turn the node into a leaf (don't expand it further)
  - Enforce a purity increment threshold – e.g., if a node expansion increases purity by less than δ, delete the expansion and turn the node into a leaf
  - Build the complete tree and then iteratively merge leaves based on lowest purity decrease up to a number of leaves N or a purity δ
  - Combinations of depth and purity measures



Simplifying decision trees

We could simplify the dolphin decision tree to this:



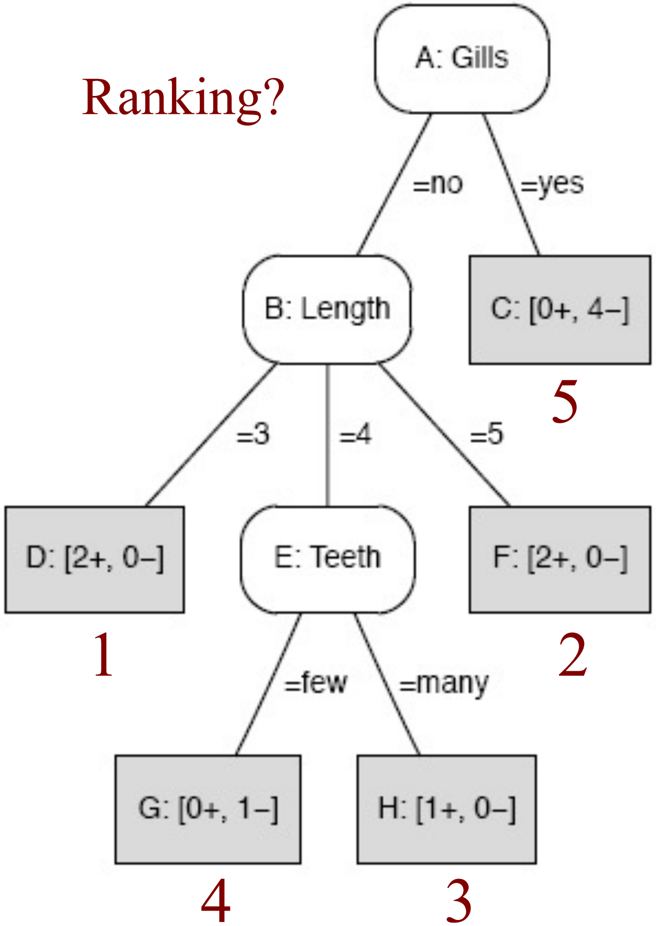
Does this generalize well?

Ranking trees

Note: We can't tell the ranking from the tree structure; only from leaves and their data

- Since a decision tree divides the instance space into segments (leaves) and we have data for each segment, we can turn the DT into a ranking model by evaluating and ordering the segments
- As before, we use empirical probabilities $\dot{p}$ – for segments i and j, order $i > j$ if $\dot{p}_i > \dot{p}_j$
  - May use Laplace correction or m-estimate for smoothing
  - (We'll need a way to decide ties…)
- As before, we can compute ranking error rate and accuracy

Ranking?



(This would be a better example if it had more data and non-homogeneous leaves!)

Ranking trees

Dolphin Ranking: D F H G C
(from previous slide)

- Ranking is with respect to a particular class (e.g., *dolphin*)
  - On a set of m instances $X = \{ x_1, \ldots, x_m \}$
- A decision tree with N leaves will have N different ranks
  - Each instance will have one of those ranks, 1..N
  - So there are likely to be many ties if N is small or m is large

		Leaf
1.	Length = 3 ∧ Gills = no ∧ Beak = yes ∧ Teeth = many	**D**
2.	Length = 4 ∧ Gills = no ∧ Beak = yes ∧ Teeth = many	**H**
3.	Length = 3 ∧ Gills = no ∧ Beak = yes ∧ Teeth = few	**D**
4.	Length = 5 ∧ Gills = no ∧ Beak = yes ∧ Teeth = many	**F**
5.	Length = 5 ∧ Gills = no ∧ Beak = yes ∧ Teeth = few	**F**
6.	Length = 5 ∧ Gills = yes ∧ Beak = yes ∧ Teeth = many	**C**
7.	Length = 4 ∧ Gills = yes ∧ Beak = yes ∧ Teeth = many	**C**
8.	Length = 5 ∧ Gills = yes ∧ Beak = no ∧ Teeth = many	**C**
9.	Length = 4 ∧ Gills = yes ∧ Beak = no ∧ Teeth = many	**C**
10.	Length = 4 ∧ Gills = no ∧ Beak = yes ∧ Teeth = few	**G**

From the leaf rankings on the previous slide, the ranking of the 10 instances is:

Highest | 1, 3 | 4, 5 | 2 | 10 | 6, 7, 8, 9 | Lowest

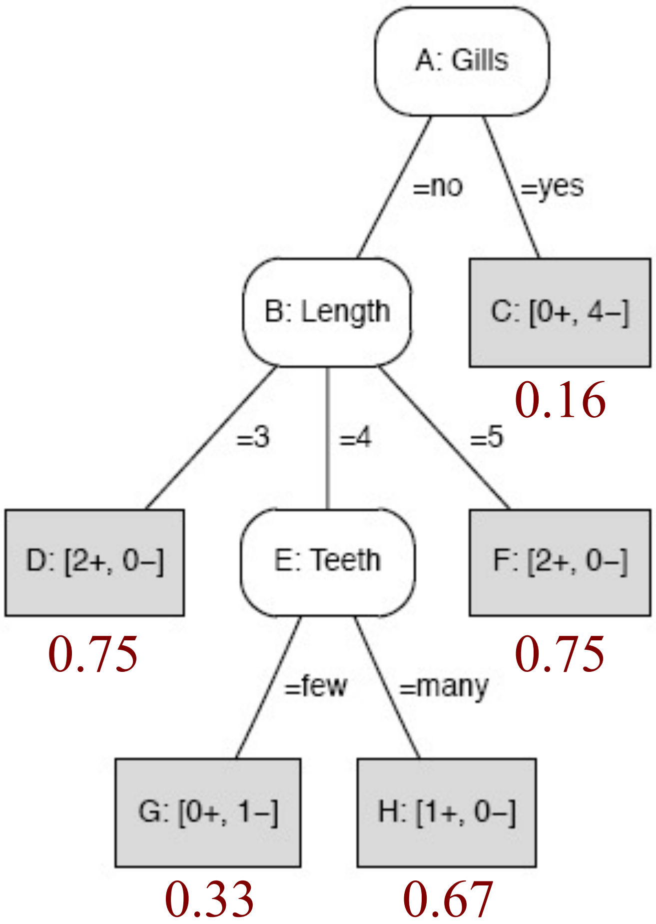
Ties

For this, accuracy = 100%

Probability estimation trees

- We can use the empirical probabilities (for each class) calculated for every leaf to create a probability estimation tree
  - A probability classifier, a.k.a. probability estimation
  - Since it's a 2-class problem, we can just show the probability for *dolphin*
- Using Laplace correction, what are the leaf probability estimates?

$$\text{Laplace correction} = \frac{N_i + 1}{|S| + k}$$



P(dolphin) = { 0.75, 0.75, 0.67, 0.33, 0.16 }

Actually showing P(dolphin=true | leaf) or P(hypothesis | data)

P(¬dolphin) = { 0.25, 0.25, 0.33, 0.67, 0.84 }

Logical models – summary

- In concept learning and decision trees, we've mostly been discussing logical models, based on simple predicate (or first-order) logic
- Pros:
  - English-readable data
  - Intuitive representation and models for people to comprehend
  - Good for explaining the decision-making process
  - Works with both numerical and categorical data
  - Some errors are obvious, easily found and debugged
- Cons:
  - Not a good fit for massive amounts of data, for purely numerical data, for subtle concepts, for things that are difficult to articulate in language
    - I.e., for many of the most important problems ML is being applied to these days!

Where we're going from here

- There are many uses of logical models, especially decision trees, in machine learning applications
  - DTs are used in many current, practical machine learning systems
- But the focus for some time has moved toward methods that can crunch large amounts of numbers – more and more to statistical and probabilistic models and methods
- From logical models, we'll now head back to geometry models and then on to probabilistic models
- We'll continue to focus on classification and regression, as well as clustering, and we'll address these problems with a variety of "modern ML" tools and techniques
- Building a linear classifier may seem like a long way from creating intelligent machines that learn and think – but it's not as far as you may think!

Linear Learning Models

Chapter 7 in the textbook

And SVMs, kernel methods, perceptrons…