Machine Learning

CSE 142

Xin (Eric) Wang

Monday, October 18, 2021

Today

- Decision trees, Ch. 5

Notes

- HW #1 due tonight
  - Done individually
  - Can only be discussed at a general level, e.g., concept underlying the question, what lectures or materials may be relevant
    - Do NOT ask for particular answers or how to answer it
- Late submission due by 10/22 (Friday) 11:59pm PT
  - The time will be deducted from your four penalty-free late days
  - Late days are counted by days, not hours or minutes
  - If you have run out of four late days, then your late submission will be penalized with 50% discount of credits
  - Submissions later than 10/22 11:59pm PT will receive zero credits by any means
- Don’t discuss answers with anyone before the late submission deadline

Notes

- **HW #2 out, due by November 3, 11:59pm PT**
- HW #2 problem 6—programming problem
  - Constructing the discriminant functions (binary classifiers): see Fig. 1.1 in the textbook
    - Built from scratch. Do NOT call any third-party classifiers
  - Submit your Python code to CodaLab
  - Describe your solution in detail in your HW solutions
  - **Attend the discussion session tomorrow for more information**

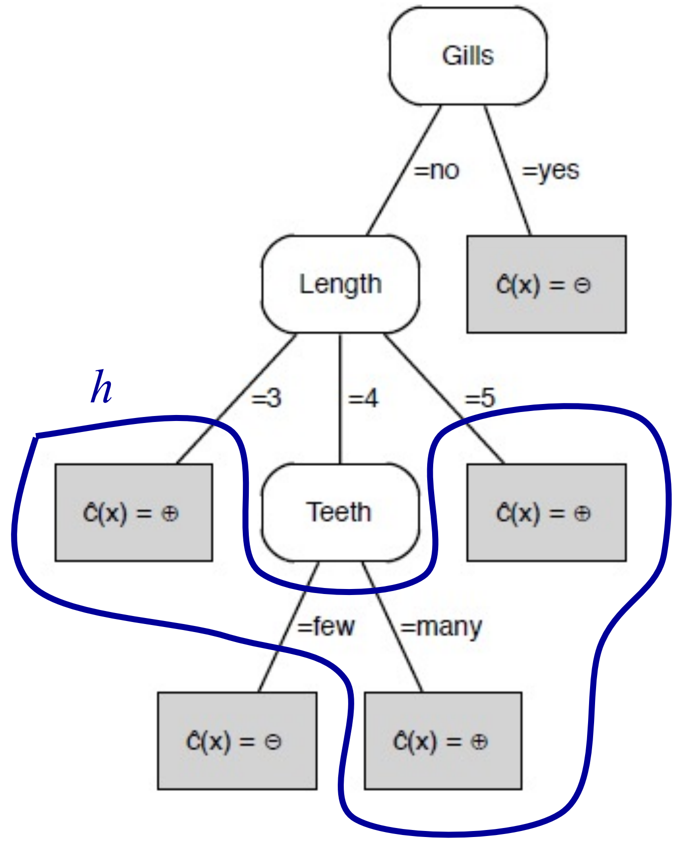
Decision Trees

Chapter 5 in the textbook

Tree models and decision trees

Decision trees

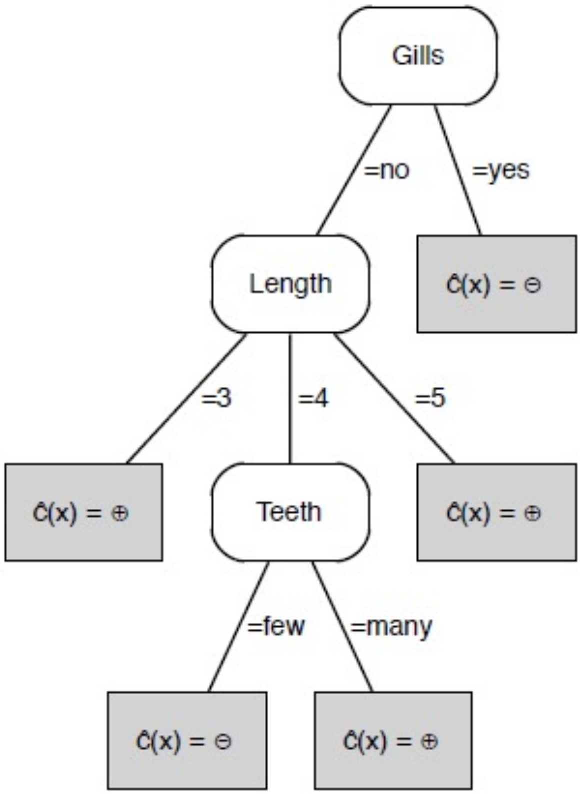
- A decision tree partitions the instance space by branching on feature values (literals), with leaves representing hypotheses
- Each leaf represents a conjunction of literals on its path
- The learned concept is the disjunction of the positive leaves
  - $L_1 \vee L_2 \vee L_3 \vee \ldots$



Quiz: can you write down the learned concept *h* of this decision tree?

Decision trees

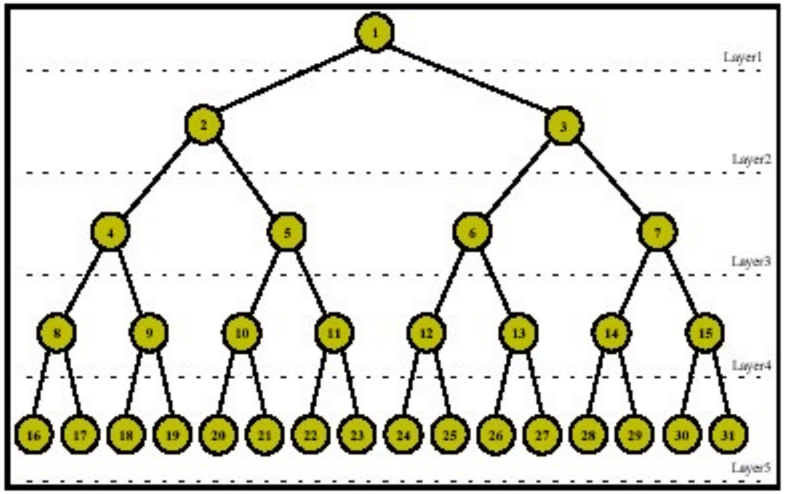
- Decision trees are maximally expressive – they can separate any consistently labeled data
  - Thus more powerful than the conjunctive hypothesis space we just discussed (which, for example, can't handle OR)
- Ideally, each leaf contains only positives or only negatives from the training data.
  - But not in practice



Decision trees

- The drawback of this is that they may not generalize well – i.e., overfitting can be a problem
  - So we have to employ mechanisms to enforce generalization beyond the examples and avoid overfitting
  - These are referred to as the inductive bias of the learning algorithm

Overfitting?



- A typical inductive bias is towards less complex hypotheses
  - A linear discriminant in classification, a hyperplane for a regression function, a restrictive hypothesis language for concept learning, etc.
  - What's the inductive bias in decision trees? (We'll see….)

Decision trees

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

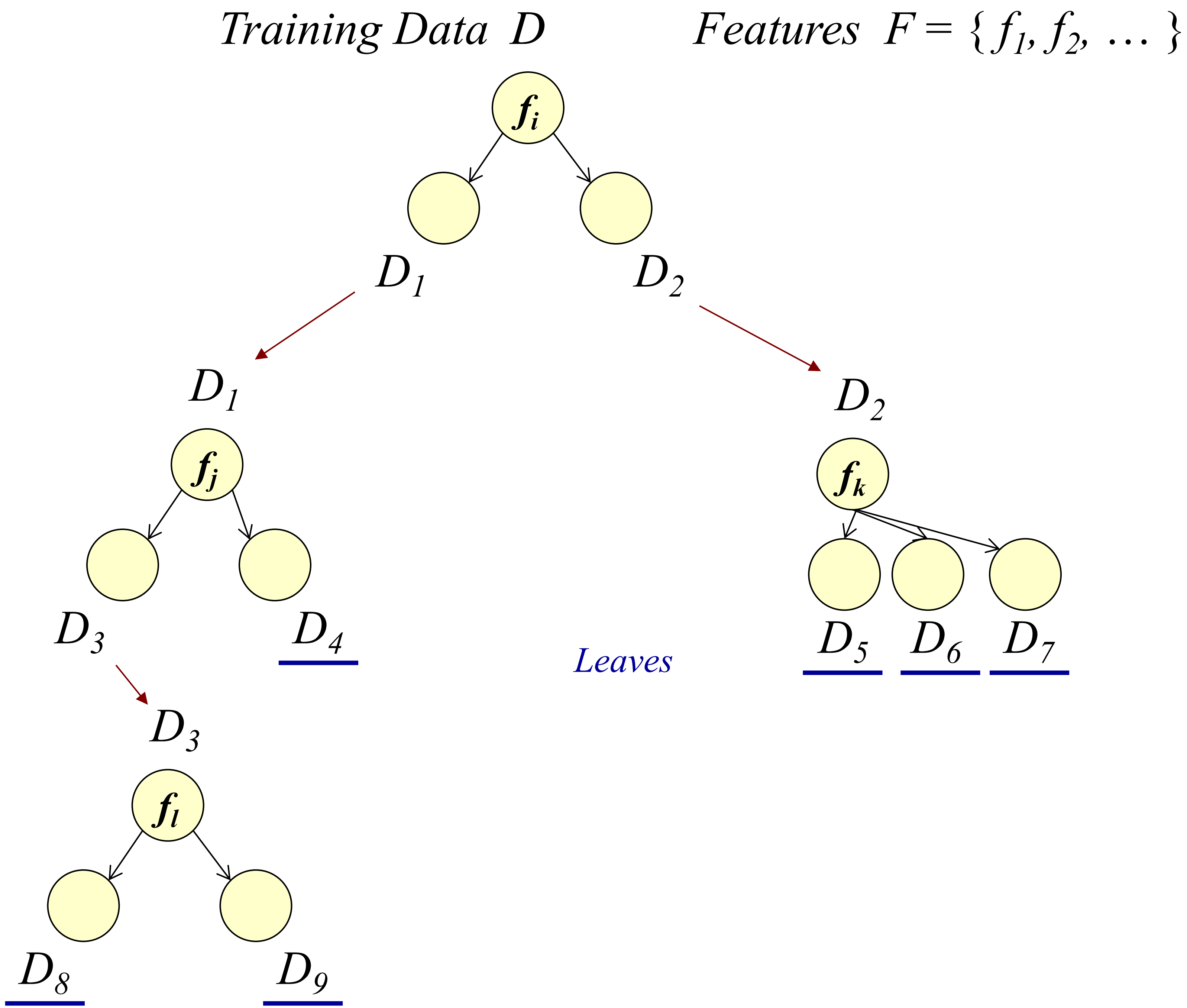
100%?
99%?
80%?

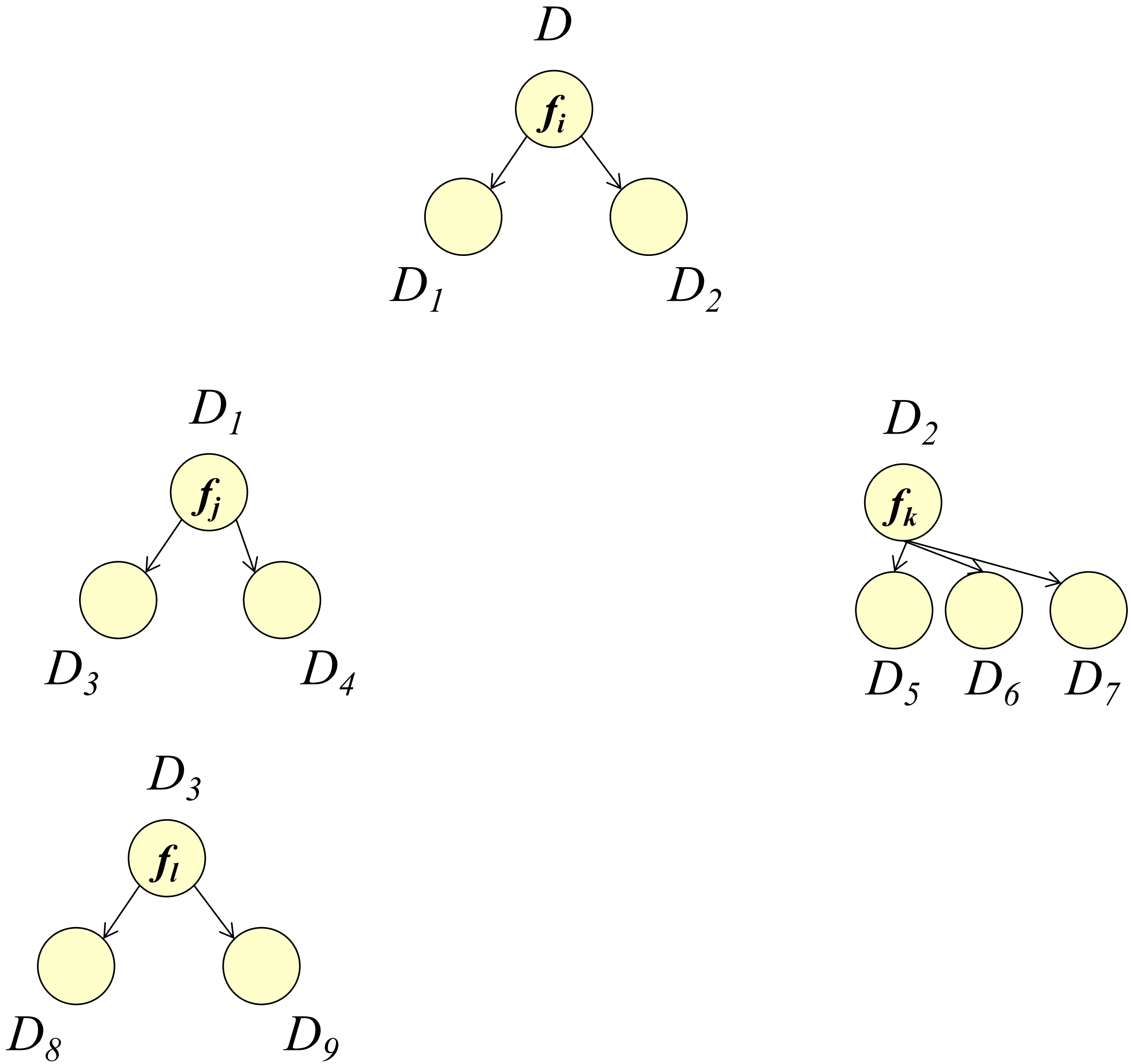
Most useful feature

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

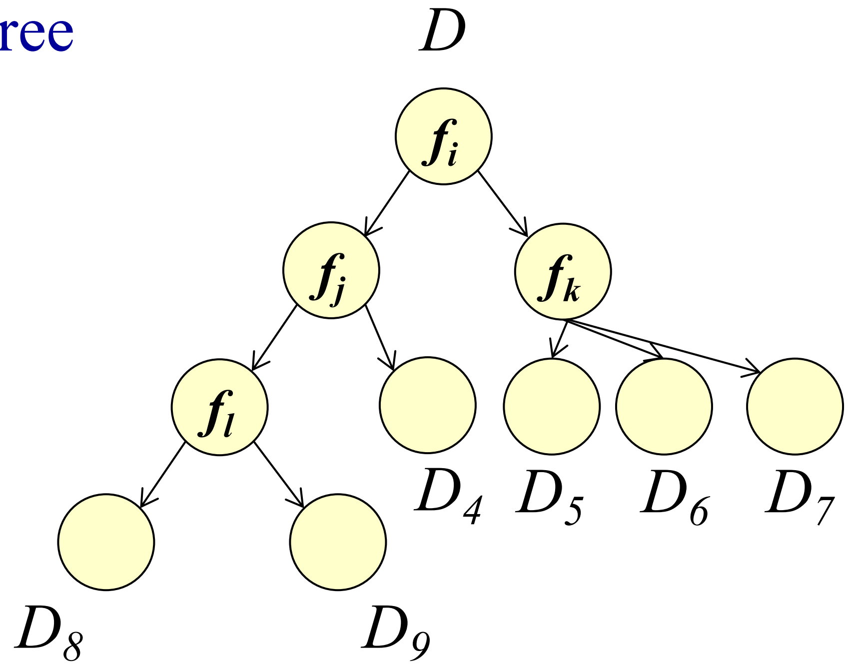
Input : data D; set of features F.
Output : feature tree T with labelled leaves.
if Homogeneous(D) **then return** Label(D);
$S \leftarrow$ BestSplit(D, F) ; // e.g., BestSplit-Class (Algorithm 5.2)
split D into subsets D_i according to the literals in S;
for each i **do**
    if $D_i \neq \emptyset$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


Training Data D
Features F = { f1, f2, … }
fi
D1
D2
D1
D2
fj
fk
D3
D4
Leaves
D5
D6
D7
D3
fl
D8
D9


D
fi
D1
D2
D1
fj
D3
D4
D2
fk
D5
D6
D7
D3
fl
D8
D9

Feature Tree



Decision trees

- 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

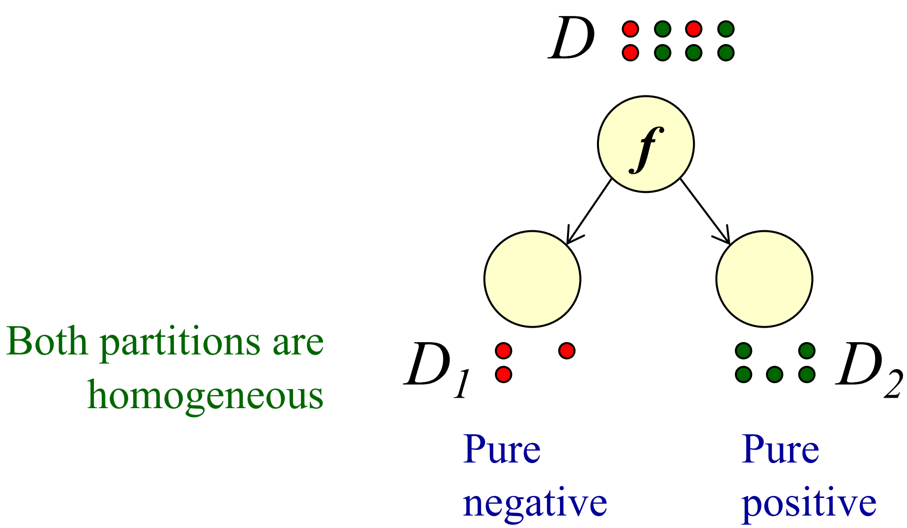
Input : data D; set of features F.
Output : feature tree T with labelled leaves.
if Homogeneous(D) **then return** Label(D);
$S \leftarrow$ $\boxed{\text{BestSplit}(D, F)\ ;}$ // e.g., BestSplit-Class (Algorithm 5.2)
split D into subsets D_i according to the literals in S;
for each i **do**
  if $D_i \neq \emptyset$ **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

- 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



Which feature f is best for this?

How to measure partition purity if not completely homogeneous?

Impurity

- In the binary case, we have P positives and N negatives in the data
  - The best split would be a feature that divides the data D into two pure partitions: D_1 with the P positives and D_2 with the N negatives
- So a measure of partition impurity should be <u>minimum</u> when the data are 100% positives or negatives, and <u>maximum</u> when 50/50
- Like with empirical probabilities, we can estimate impurity by counting. We define the proportion of positives in D_i as:

$$\dot{p} = \frac{P}{P + N}$$

- Impurity is a function of $\dot{p}$
  - Should be zero when $\dot{p} = 0$ or 1, and maximum when $\dot{p} = 0.5$
  - We can write impurity as Imp(D) or Imp($\dot{p}$)

Impurity functions

Minority class

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

Gini index

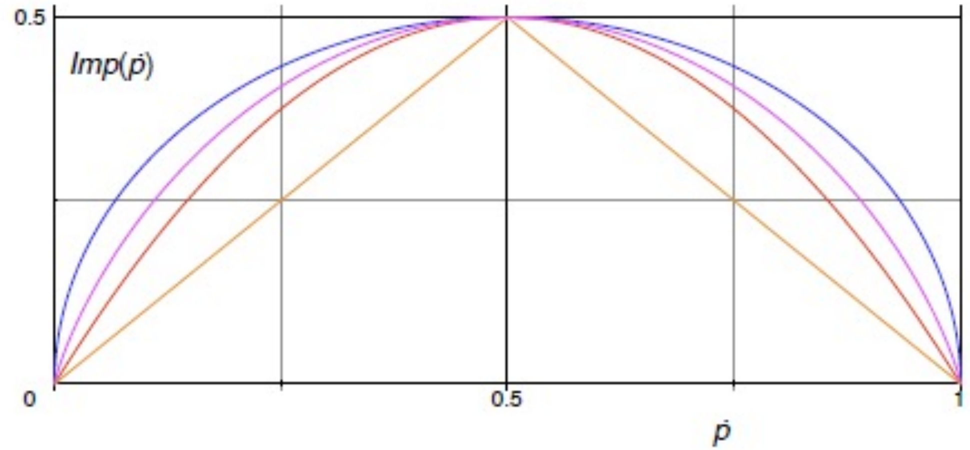
$$\mathrm{Imp}(\dot{p}) = 2\dot{p}(1-\dot{p})$$

Entropy

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

√Gini index

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



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

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

Impurity functions for $k > 2$

For more than two classes, the impurity functions are defined by the sum of the per-class impurities based on "one versus rest"

k-class Entropy

$$\mathrm{Imp}(\dot{p}_1,\ldots,\dot{p}_k) = \sum_{i=1}^{k} -\dot{p}_i \log_2(\dot{p}_i) \qquad \text{where} \quad \dot{p}_i = \frac{C_i}{\sum_{i=1}^{k} C_i}$$

k-class Gini index

$$\mathrm{Imp}(\dot{p}) = \sum_{i=1}^{k} \dot{p}_i (1 - \dot{p}_i)$$

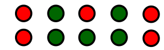
To split a parent node D into children $D_1,\ldots, D_L$ we can consider the purity gain = $\mathrm{Imp}(D) - \mathrm{Imp}(\{D_1,\ldots, D_L\})$

Reminder: What is k? What is L?

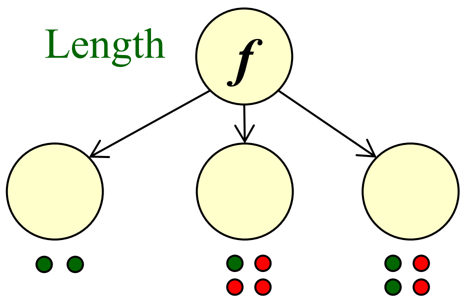
The number of classes

The number of values (literals) for a given feature F_i

Impurity example

D 

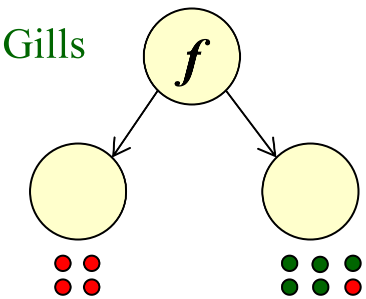
Length



0.0 .81 1.0

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

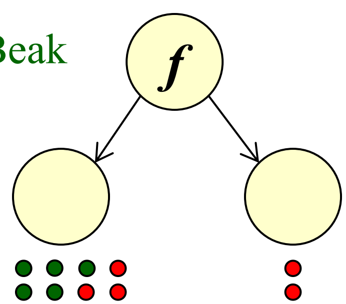
Gills



0.0 .65

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

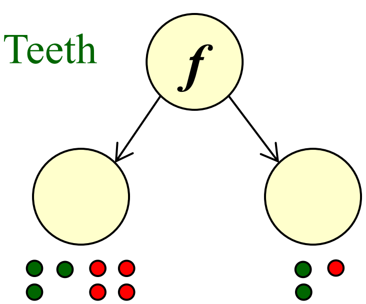
Beak



.95 0.0

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

Teeth



.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