

Applied Machine Learning

Decision Trees

Reihaneh Rabbany



Motivation

What we have left to cover for this course:

Classification and regression trees

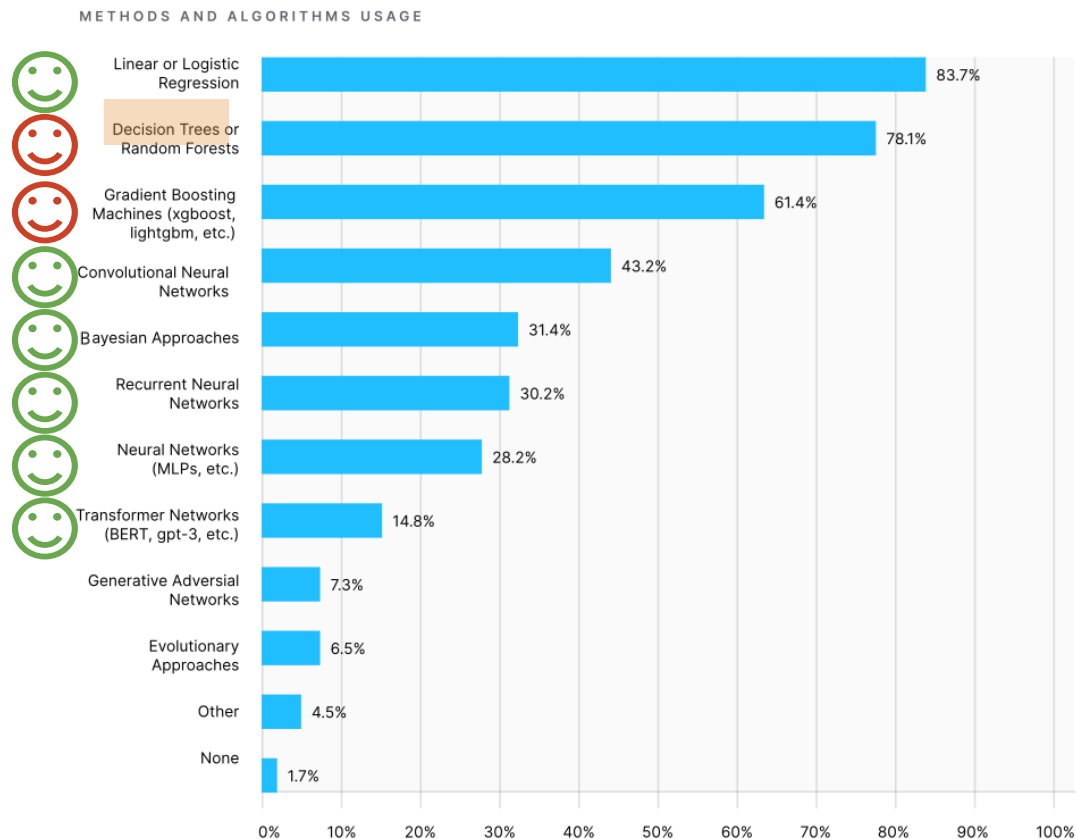
Linear support vector machines

Bagging & boosting

Unsupervised learning

Dimensionality reduction

from 2020 Kaggle's survey on the state of Machine Learning and Data Science, you can read the full version [here](#)

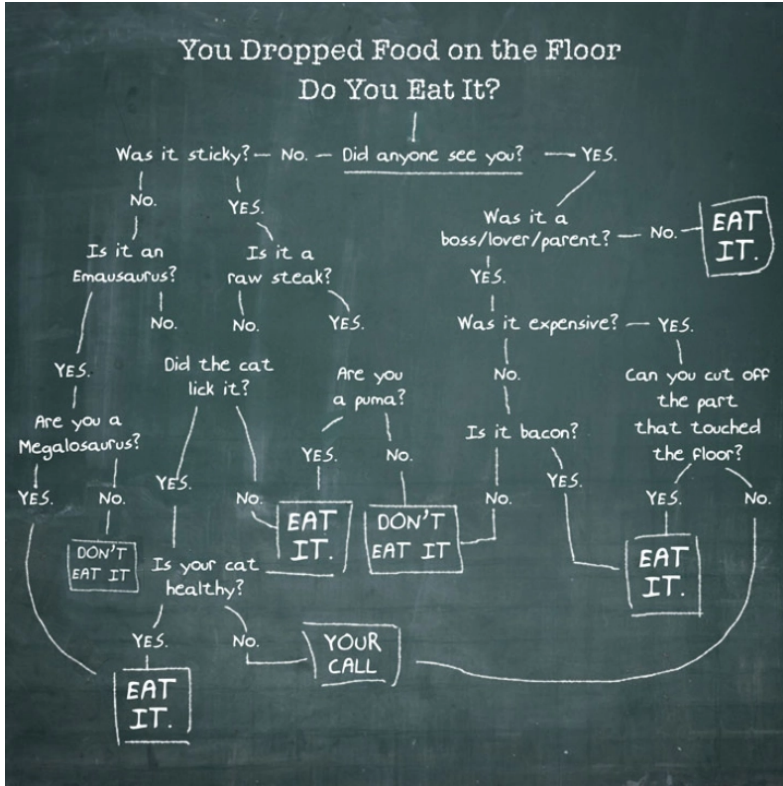


Learning objectives

Decision trees:

- how does it model the data?
- how to specify the best model using a cost function
- how the cost function is optimized

Decision trees: motivation



pros.

- decision trees are interpretable
- they are not very sensitive to outliers
- do not need data normalization

cons.

- they could easily overfit and are unstable to small changes in input data

Notation overview

	<tumorsize, texture, perimeter>	,	<cancer>		
$x^{(1)}$	<18.2,	27.6,	117.5>	, < No >	$y^{(1)}$
$x^{(2)}$	<17.9,	10.3,	122.8>	, < No >	$y^{(2)}$
$x^{(3)}$	<20.2,	14.3,	111.2>	, < Yes >	$y^{(3)}$
\vdots			\vdots		\vdots
$x^{(N)}$	<15.5,	15.2,	135.5>	, < No >	$y^{(N)}$

Our datasets consists of N pairs of input vector and corresponding target or label

$$\mathcal{D} = \{(x^{(1)}, y^{(1)}), \dots, (x^{(N)}, y^{(N)})\}$$

we use N to denote the size of the dataset and n for indexing a pair of input, label

x, y denote an input and label pair where

$$x \text{ is a } D\text{-dimensional vector: } x = [x_1, x_2, \dots, x_D]$$

we use D to denote the number of *features* (dimensionality of the input space)

$y \in \{1, \dots, C\}$ for classification problems, we use C for number of classes

$x_d^{(n)}$ $n \in [1 \dots N]$ indexes an instance, row index, e.g. which patient
 $d \in [1 \dots D]$ indexes a feature, column index, e.g. which measurement

e.g. $x_2^{(3)}$

Decision trees: idea

- divide the input space into regions $\mathbb{R}_1, \dots, \mathbb{R}_K$ using a tree structure
- assign a prediction **label** to each region

for classification this is the class label

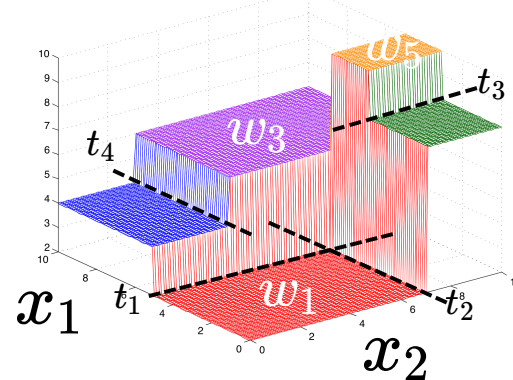
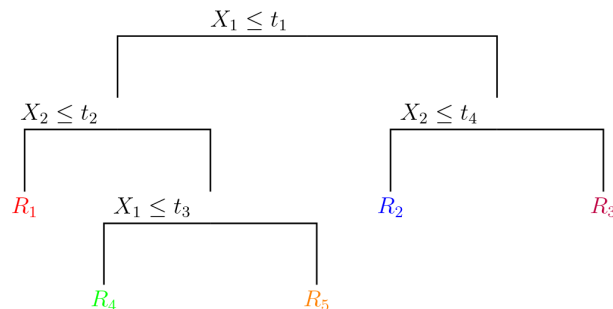
for regression, this is a real scalar or vector

$$f(x) = \sum_k w_k \mathbb{I}(x \in \mathbb{R}_k)$$

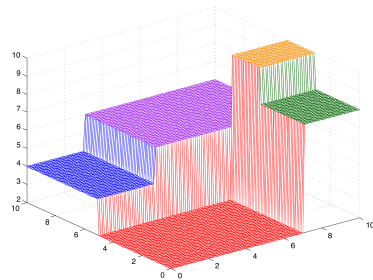
how to build the regions and the tree?

split regions successively based on the value of a single variable called test

each region is a set of conditions $\mathbb{R}_2 = \{x_1 \geq t_1, x_2 \leq t_4\}$



Prediction per region



What constant w_k should we use for prediction in each region \mathbb{R}_k ?

Classification

count the frequency of classes per region,
predict the most frequent label or return
probability $w_k = \text{mode}(y^{(n)} | x^{(n)} \in \mathbb{R}_k)$

Regression

use the mean value of training data-points in
that region $w_k = \text{mean}(y^{(n)} | x^{(n)} \in \mathbb{R}_k)$

example: predicting survival in titanic

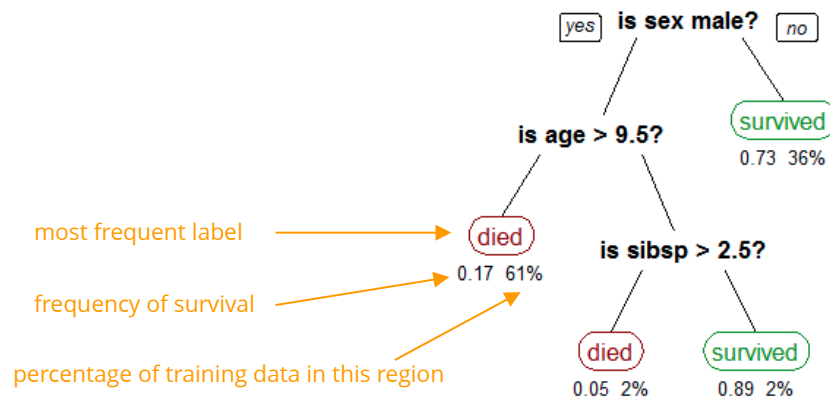


figure from [wiki](#)

Possible tests

next questions: what are all the possible tests? which test do we choose next?



Continuous features

all the values that appear in the dataset can be used to split



Categorical features

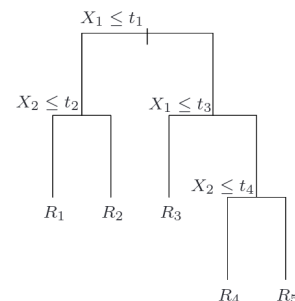
if a feature can take C values $x_i \in \{1, \dots, C\}$

convert that feature into C binary features (**one-hot coding**) $x_{i,1}, \dots, x_{i,C} \in \{0, 1\}$

split based on the value of a binary feature

alternatives:

- **multi-way** split: can lead to regions with few datapoints
- binary splits that produce balanced subsets



Food Name	Categorical #	Calories
Apple	1	95
Chicken	2	231
Broccoli	3	50

Apple	Chicken	Broccoli	Calories
1	0	0	95
0	1	0	231
0	0	1	50

one-hot example from [here](#)

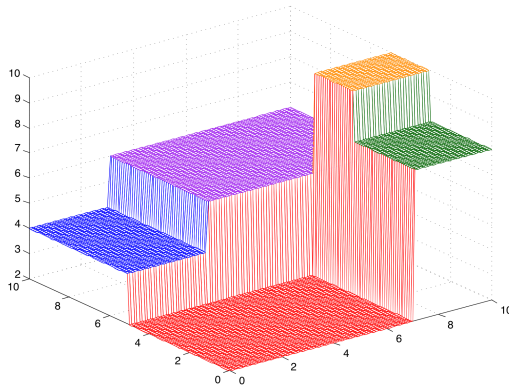
Cost function

find a **decision tree** minimizing the following cost function, this cost function specifies "what is a good decision or regression tree?"

regression cost

first calculate cost per region \mathbb{R}_k

Similar to other ML algorithms minimize a cost function or maximize an objective function



we predict $w_k = \text{mean}(y^{(n)} | x^{(n)} \in \mathbb{R}_k)$ for region \mathbb{R}_k

mean squared error (MSE)

$$\text{cost}(\mathbb{R}_k, \mathcal{D}) = \frac{1}{N_k} \sum_{x^{(n)} \in \mathbb{R}_k} (y^{(n)} - w_k)^2$$

number of instances in region k truth prediction

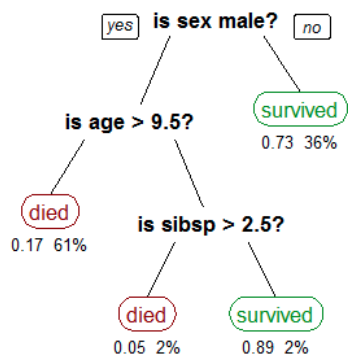
total cost is the normalized sum over all regions $\text{cost}(\mathcal{D}) = \sum_k \frac{N_k}{N} \text{cost}(\mathbb{R}_k, \mathcal{D})$

Cost function

find a **decision tree** minimizing the following cost function, this cost function specifies "what is a good decision or regression tree?"

classification cost

again, calculate the cost per region \mathbb{R}_k



for each region we predict the most frequent label $w_k = \text{mode}(y^{(n)} | x^{(n)} \in \mathbb{R}_k)$

misclassification rate

$$\text{cost}(\mathbb{R}_k, \mathcal{D}) = \frac{1}{N_k} \sum_{x^{(n)} \in \mathbb{R}_k} \mathbb{I}(y^{(n)} \neq w_k)$$

number of instances in region k truth prediction

total cost is the normalized sum $\text{cost}(\mathcal{D}) = \sum_k \frac{N_k}{N} \text{cost}(\mathbb{R}_k, \mathcal{D})$

Cost function

find a **decision tree** minimizing the following cost function, this cost function specifies "what is a good decision or regression tree?"

total cost is the normalized sum $\text{cost}(\mathcal{D}) = \sum_k \frac{N_k}{N} \text{cost}(\mathbb{R}_k, \mathcal{D})$

problem

it is sometimes possible to build a tree with **zero cost**:

build a large tree with each instance having its own region (*overfitting!*)

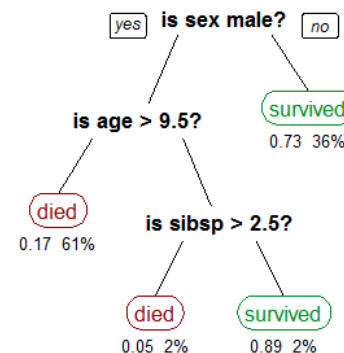
example

use features such as height, eye color etc, to make perfect prediction on training data

solution

find a decision tree with at most **K tests** minimizing the cost function

K tests = K internal node in our binary tree = K+1 leaves (regions)



Search space

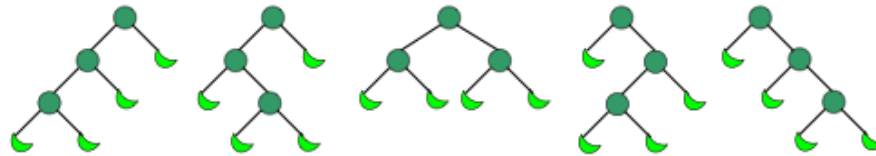
K+1 regions

objective: find a decision tree with **K tests** minimizing the cost function

the number of full binary trees with K+1 leaves (regions \mathbb{R}_k) is the **Catalan number** $\frac{1}{K+1} \binom{2K}{K}$

1, 1, 2, **5**, 14, 42, 132, 429, 1430, 4862, 16796, 58786, 208012, 742900, 2674440, 9694845, 35357670, 129644790, 477638700, 1767263190, 6564120420, 24466267020, 91482563640, 343059613650, 1289904147324, 4861946401452

exponential in K



we also have a choice of feature x_d for each of K internal node D^K

moreover, for each feature different choices of splitting

bottom line: finding optimal decision tree is an **NP-hard** combinatorial optimization problem

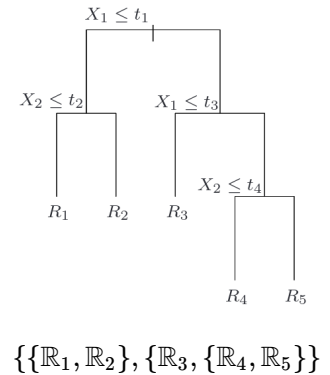
Greedy heuristic

finding the **optimal** tree is too difficult, instead use a greedy heuristic to find a **good** tree
recursively split the regions based on a **greedy choice of the next test**
end the recursion if not **worth-splitting**

```
function fit-tree( $\mathbb{R}_{\text{node}}$ ,  $\mathcal{D}$ , depth)

     $\mathbb{R}_{\text{left}}, \mathbb{R}_{\text{right}}$  = greedy-test ( $\mathbb{R}_{\text{node}}$ ,  $\mathcal{D}$  )
    if not worth-splitting(depth,  $\mathbb{R}_{\text{left}}$ ,  $\mathbb{R}_{\text{right}}$  )
        return  $\mathbb{R}_{\text{node}}$ 
    else
        left-set = fit-tree( $\mathbb{R}_{\text{left}}$ ,  $\mathcal{D}$ , depth+1)
        right-set = fit-tree( $\mathbb{R}_{\text{right}}$ ,  $\mathcal{D}$ , depth+1)
        return {left-set, right-set}
```

final decision tree in the form of nested list of regions



Choosing tests

the split is **greedy** because it looks one step ahead
this may not lead to the lowest overall cost

```
function greedy-test (  $\mathbb{R}_{\text{node}}, \mathcal{D}$  )  
    best-cost = inf  
    for each feature  $d \in \{1, \dots, D\}$  and each possible test  
        split  $\mathbb{R}_{\text{node}}$  into  $\mathbb{R}_{\text{left}}, \mathbb{R}_{\text{right}}$  based on the test  
        split-cost =  $\frac{N_{\text{left}}}{N_{\text{node}}} \text{cost}(\mathbb{R}_{\text{left}}, \mathcal{D}) + \frac{N_{\text{right}}}{N_{\text{node}}} \text{cost}(\mathbb{R}_{\text{right}}, \mathcal{D})$   
        if split-cost < best-cost:  
            best-cost = split-cost  
             $\mathbb{R}_{\text{left}}^* = \mathbb{R}_{\text{left}}$   
             $\mathbb{R}_{\text{right}}^* = \mathbb{R}_{\text{right}}$   
  
    return  $\mathbb{R}_{\text{left}}^*, \mathbb{R}_{\text{right}}^*$ 
```

Stopping the recursion

worth-splitting subroutine

if we stop when \mathbb{R}_{node} has zero cost, we may overfit

heuristics for stopping the splitting:

- reached a desired depth
- number of examples in \mathbb{R}_{left} or $\mathbb{R}_{\text{right}}$ is too small
- w_k is a good approximation, the cost is small enough
- reduction in cost by splitting is small

$$\text{cost}(\mathbb{R}_{\text{node}}, \mathcal{D}) - \left(\frac{N_{\text{left}}}{N_{\text{node}}} \text{cost}(\mathbb{R}_{\text{left}}, \mathcal{D}) + \frac{N_{\text{right}}}{N_{\text{node}}} \text{cost}(\mathbb{R}_{\text{right}}, \mathcal{D}) \right)$$

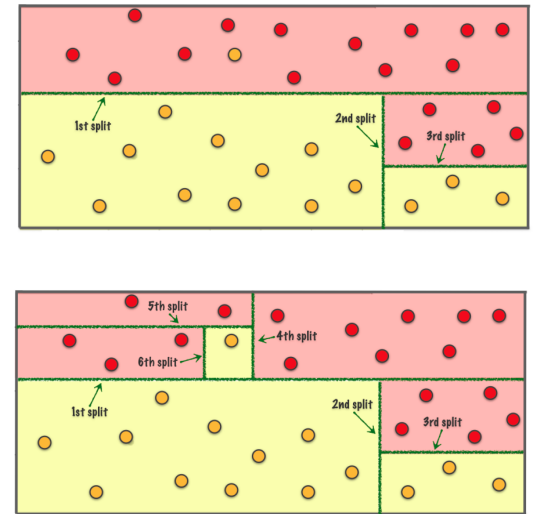


image credit: <https://alanjeffares.wordpress.com/tutorials/decision-tree/>

revisiting the classification cost

ideally we want to optimize the misclassification rate

$$\text{cost}(\mathbb{R}_k, \mathcal{D}) = \frac{1}{N_k} \sum_{x^{(n)} \in \mathbb{R}_k} \mathbb{I}(y^{(n)} \neq w_k)$$

this may not be the optimal cost for *each step of greedy heuristic*

example

both splits have the same misclassification rate (2/8)

$$\text{cost}(\mathcal{D}) = \frac{1}{N} \sum_k \sum_{x^{(n)} \in \mathbb{R}_k} \mathbb{I}(y^{(n)} \neq w_k)$$



however the second split *may be* preferable because one region does not need further splitting

idea: use a measure for homogeneity of labels in regions

Entropy: a measure of the *unpredictability*

The most basic concept in the field of **Information Theory**

entropy is the **expected amount of information** in observing a random variable

$$H(y) = - \sum_{c=1}^C p(y=c) \log p(y=c)$$

averaged on all its possible outcomes

$-\log p(y=c)$ is the amount of **information** in observing value c

$$I(y=c) = \log\left(\frac{1}{p(y=c)}\right) = -\log(p(y=c))$$

note that it is common to use capital letters for random variables (here for consistency we use lower-case)

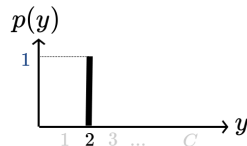
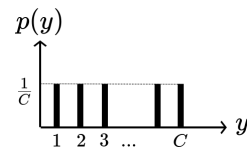
zero information if $p(c)=1$

less probable events are more informative $p(c) < p(c') \Rightarrow -\log p(c) > -\log p(c')$

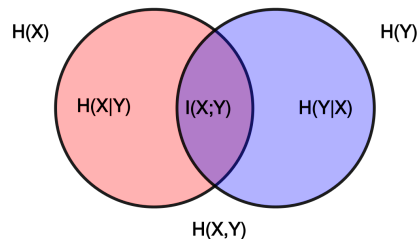
information from two independent events is additive $-\log(p(c)q(d)) = -\log p(c) - \log q(d)$

a uniform distribution has the highest entropy $H(y) = - \sum_{c=1}^C \frac{1}{C} \log \frac{1}{C} = \log C$

a deterministic random variable has the lowest entropy $H(y) = -1 \log(1) = 0$



Mutual information



for two random variables t, y , their **mutual information** is

the amount of information t conveys about y

change in the entropy of y after observing the value of t

how much knowing t reduces uncertainty about y

$$I(t, y) = H(y) - H(y|t)$$

conditional entropy $\sum_{l=1}^L p(t=l)H(y|t=l)$ uncertainty we have in y after seeing t

$$= \sum_l \sum_c p(y=c, t=l) \log \frac{p(y=c, t=l)}{p(y=c)p(t=l)} \quad \text{this is symmetric wrt } y \text{ and } t$$

$$= H(t) - H(t|y) = I(y, t)$$

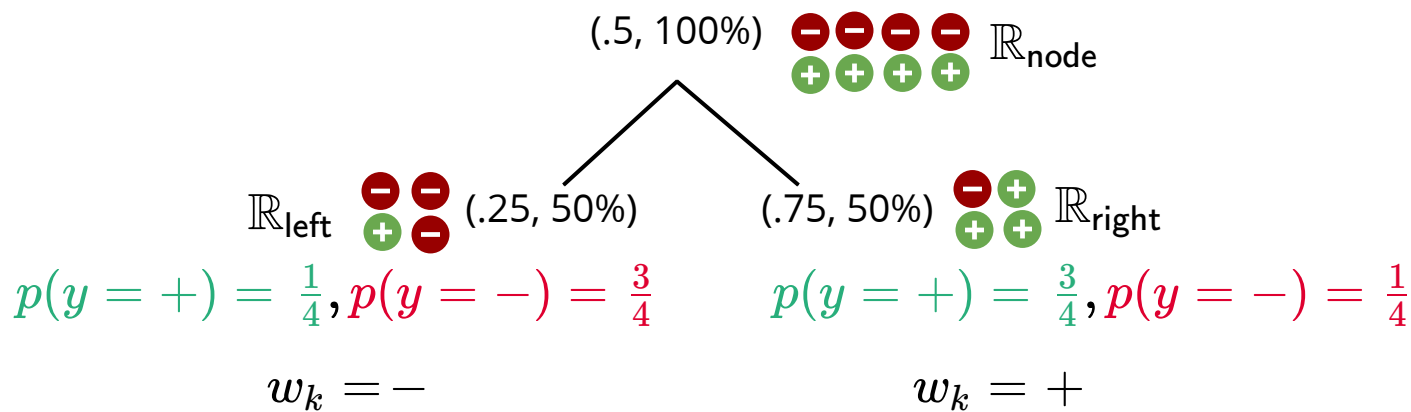
mutual information is always positive and zero only if y and t are independent

Classification cost: **example**

we care about the empirical distribution of labels in each region $p_k(y = c) = \frac{\sum_{x^{(n)} \in \mathbb{R}_k} \mathbb{I}(y^{(n)} = c)}{N_k}$

misclassification cost $\text{cost}(\mathbb{R}_k, \mathcal{D}) = \frac{1}{N_k} \sum_{x^{(n)} \in \mathbb{R}_k} \mathbb{I}(y^{(n)} \neq w_k) = 1 - p_k(w_k)$

the most probable class $w_k = \arg \max_c p_k(c)$



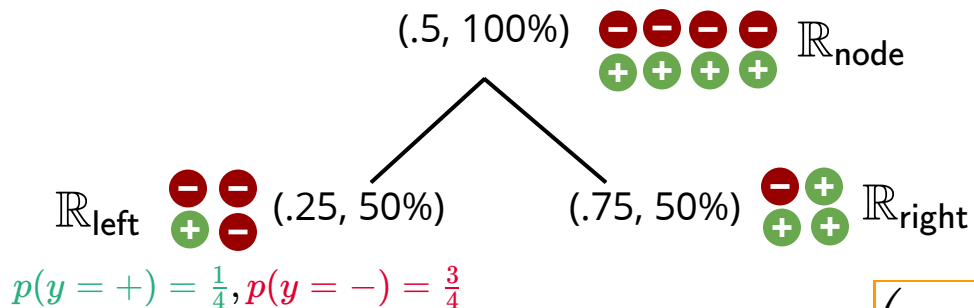
misclassification cost = $\frac{4}{8} \cdot \frac{1}{4} + \frac{4}{8} \cdot \frac{1}{4} = \frac{1}{4}$

Entropy for classification cost: **example**

we care about the empirical distribution of labels in each region $p_k(y = c) = \frac{\sum_{x^{(n)} \in \mathbb{R}_k} \mathbb{I}(y^{(n)} = c)}{N_k}$

misclassification cost $\text{cost}(\mathbb{R}_k, \mathcal{D}) = \frac{1}{N_k} \sum_{x^{(n)} \in \mathbb{R}_k} \mathbb{I}(y^{(n)} \neq w_k) = 1 - p_k(w_k)$
 the most probable class $w_k = \arg \max_c p_k(c)$

entropy cost $\text{cost}(\mathbb{R}_k, \mathcal{D}) = H(y) = - \sum_{c=1}^C p(y = c) \log p(y = c)$ choose the split with the lowest entropy



misclassification cost

$$\frac{4}{8} \cdot \frac{1}{4} + \frac{4}{8} \cdot \frac{1}{4} = \frac{1}{4}$$

entropy cost

$$\frac{4}{8} \left(-\frac{1}{4} \log\left(\frac{1}{4}\right) - \frac{3}{4} \log\left(\frac{3}{4}\right) \right) + \frac{4}{8} \left(-\frac{1}{4} \log\left(\frac{1}{4}\right) - \frac{3}{4} \log\left(\frac{3}{4}\right) \right)$$

$$\text{cost}(\mathcal{D}) = \sum_k \frac{N_k}{N} \text{cost}(\mathbb{R}_k, \mathcal{D})$$

Entropy for classification cost: example



misclassification cost

$$\frac{4}{8} \cdot \frac{1}{4} + \frac{4}{8} \cdot \frac{1}{4} = \frac{1}{4}$$

the same costs

$$\frac{6}{8} \cdot \frac{1}{3} + \frac{2}{8} \cdot \frac{0}{2} = \frac{1}{4}$$

entropy cost (using base 2 logarithm)

$$\frac{4}{8} \left(-\frac{1}{4} \log\left(\frac{1}{4}\right) - \frac{3}{4} \log\left(\frac{3}{4}\right) \right) + \frac{4}{8} \left(-\frac{1}{4} \log\left(\frac{1}{4}\right) - \frac{3}{4} \log\left(\frac{3}{4}\right) \right) \approx .81$$



$$\frac{6}{8} \left(-\frac{1}{3} \log\left(\frac{1}{3}\right) - \frac{2}{3} \log\left(\frac{2}{3}\right) \right) + \frac{2}{8} \cdot 0 \approx .68$$

lower cost split

Entropy for classification cost

we care about the empirical distribution of labels in each region $p_k(y = c) = \frac{\sum_{x^{(n)} \in \mathbb{R}_k} \mathbb{I}(y^{(n)} = c)}{N_k}$

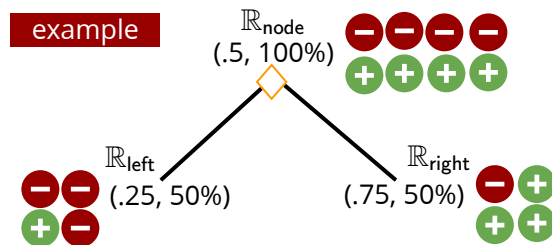
entropy cost $\text{cost}(\mathbb{R}_k, \mathcal{D}) = H(y)$ choose the split with the lowest entropy

change in the cost becomes the **mutual information** between the test and labels

$$\begin{aligned} \text{cost}(\mathbb{R}_{\text{node}}, \mathcal{D}) &= \left(\frac{N_{\text{left}}}{N_{\text{node}}} \text{cost}(\mathbb{R}_{\text{left}}, \mathcal{D}) + \frac{N_{\text{right}}}{N_{\text{node}}} \text{cost}(\mathbb{R}_{\text{right}}, \mathcal{D}) \right) \\ &= H(y) - \left(\underbrace{p(x_d \geq t)H(y|x_d \geq t)} + \underbrace{p(x_d < t)H(y|x_d < t)} \right) = I(y, x > t) \end{aligned}$$

$$I(t, y) = H(y) - H(y|t) = H(y) - \sum_{l=1}^L p(t=l)H(y|t=l)$$

this means by using entropy as our cost, we are choosing the test which is **maximally informative** about labels



$$\begin{aligned} I(y, \diamond) &= H(y) - \left(p(\diamond)H(y|\diamond) + p(!\diamond)H(y|!\diamond) \right) \\ &= 1 - 0.81 \end{aligned}$$

Gini index

another cost for selecting the *test* in classification

misclassification (error) rate

$$\text{cost}(\mathbb{R}_k, \mathcal{D}) = \frac{1}{N_k} \sum_{x^{(n)} \in \mathbb{R}_k} \mathbb{I}(y^{(n)} \neq w_k) = 1 - p(w_k)$$

entropy

$$\text{cost}(\mathbb{R}_k, \mathcal{D}) = H(y)$$

Gini index

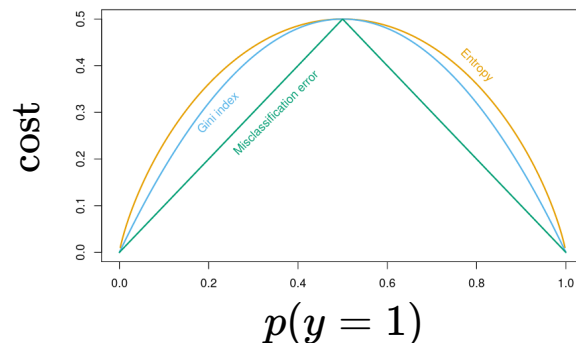
it is the expected error rate

$$\text{cost}(\mathbb{R}_k, \mathcal{D}) = \sum_{c=1}^C p(c)(1 - p(c))$$

probability of class c probability of error

$$= \sum_{c=1}^C p(c) - \sum_{c=1}^C p(c)^2 = 1 - \sum_{c=1}^C p(c)^2$$

comparison of costs of a node when we have 2 classes



C=2

$$1 - \max(p, 1 - p)$$

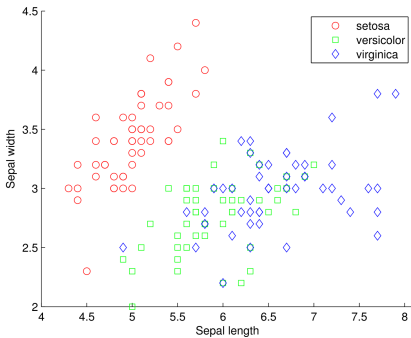
$$-p \log p - (1 - p) \log(1 - p)$$

$$2p(1 - p)$$

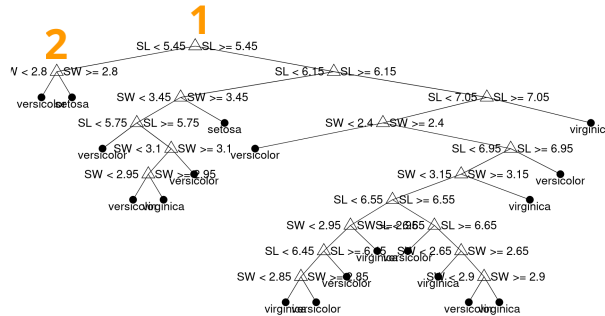
entropy & gini very similar and both favour pure nodes

Decision tree example decision tree for Iris dataset

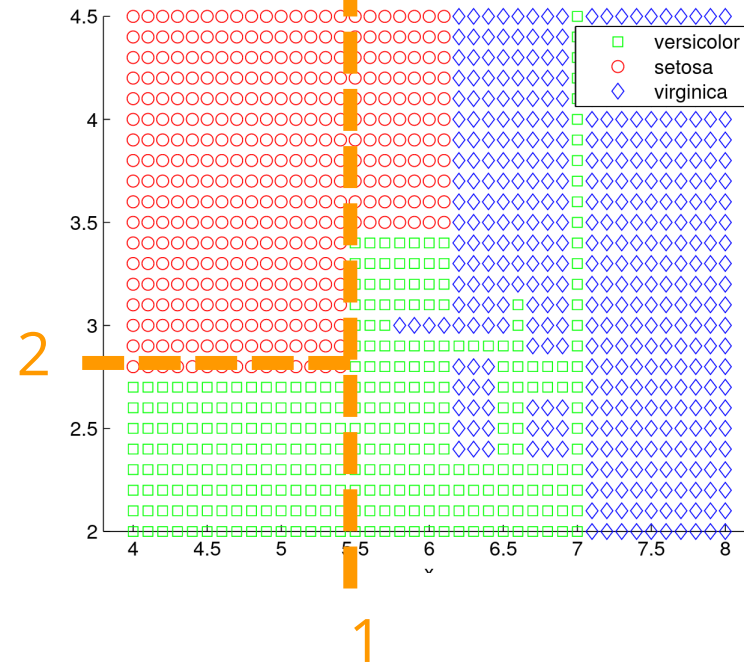
dataset (D=2)



decision tree



decision boundaries



decision boundaries suggest overfitting

confirmed using a validation set

training accuracy ~ 85%

validation accuracy ~ 70%

Decision tree: **overfitting**

a decision tree can fit any **Boolean function** (binary classification with binary features)

example: of decision tree representation of a boolean function (D=3)

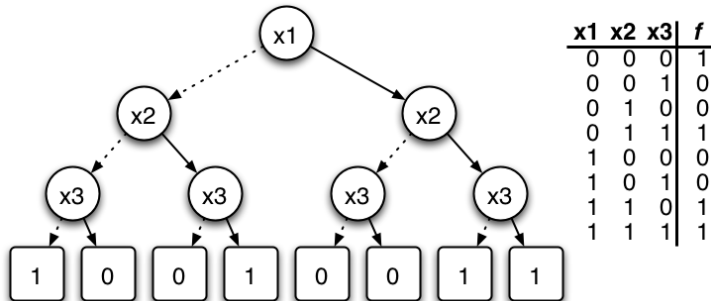


image from: https://www.wikiwand.com/en/Binary_decision_diagram

there are 2^{2^D} such functions, why?

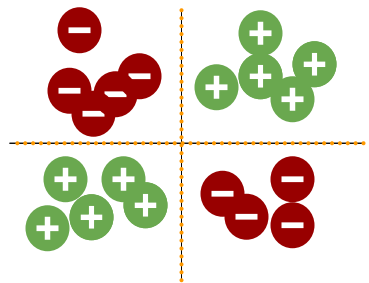
decision tree can perfectly fit our training data

How to solve the problem of overfitting in large decision trees?

idea 1. grow a small tree



problem: substantial reduction in cost may happen after a few steps by stopping early we cannot know this



example cost drops after the second node

Decision tree: overfitting & pruning

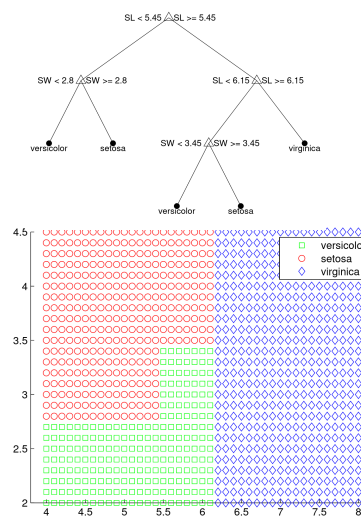
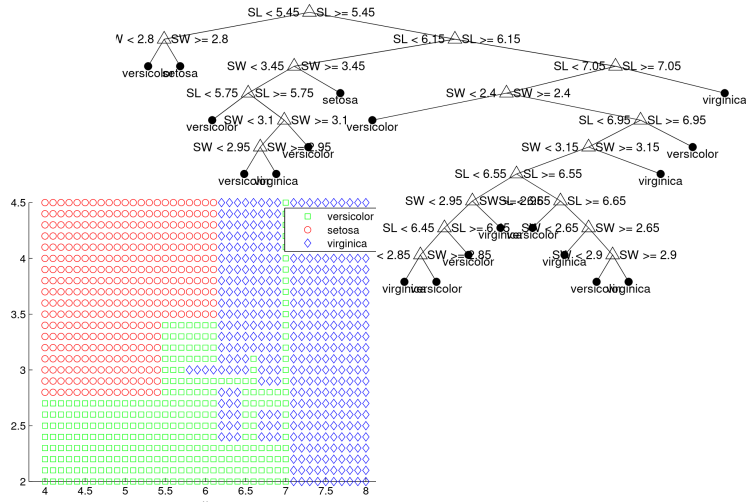
idea 2. grow a large tree and then prune it
 greedily turn an internal node into a leaf node
 choice is based on the lowest increase in the cost
 repeat this until left with the root node
 pick the best among the above models using a validation set

idea 3. random forests (later!)

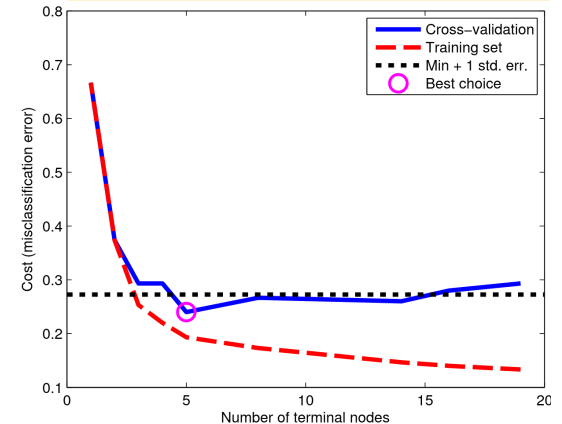
example

before pruning

after pruning



cross-validation is used to pick the best size



Summary

- model: divide the input into axis-aligned regions
- cost: for regression and classification
- optimization:
 - NP-hard
 - use greedy heuristic
- adjust the cost for the heuristic
 - using entropy (relation to mutual information maximization)
 - using Gini index
- there are variations on decision tree heuristics
 - what we discussed in called *Classification and Regression Trees (CART)*
- Compared to KNN, robust to scaling and noise, fast predictions, more interpretable