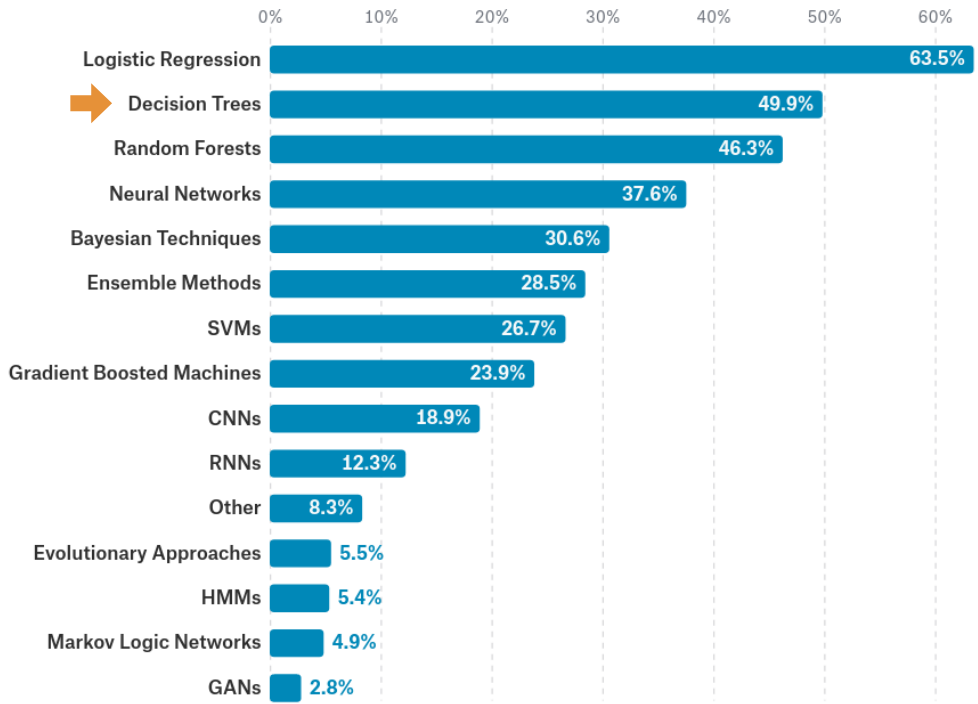
Applied Machine Learning

Decision Trees

Reihaneh Rabbany

COMP 551 (winter 2020)

Commonly used in practice



Learning objectives

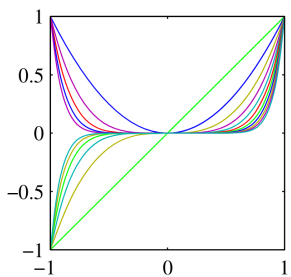
decision trees:

- model
- cost function
- how it is optimized

how to grow a tree and why you should prune it!

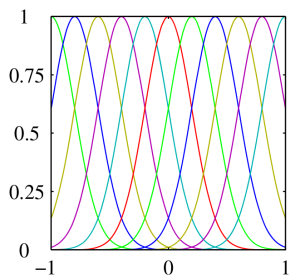
Adaptive bases

so far we assume a fixed set of bases in $f(x) = \sum_d w_d \phi_d(x)$



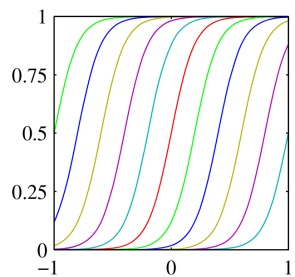
polynomial bases

$$\phi_k(x) = x^k$$



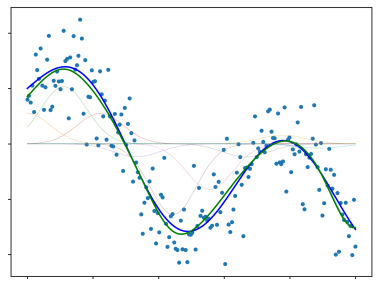
Gaussian bases

$$\phi_k(x) = e^{-\frac{(x-\mu_k)^2}{s^2}}$$



Sigmoid bases

$$\phi_k(x) = \frac{1}{1+e^{-\frac{x-\mu_k}{s}}}$$



Gaussian bases example

several methods can be classified as *learning these bases adaptively*

Adaptive bases

so far we assume a fixed set of bases in $f(x) = \sum_d w_d \phi_d(x)$

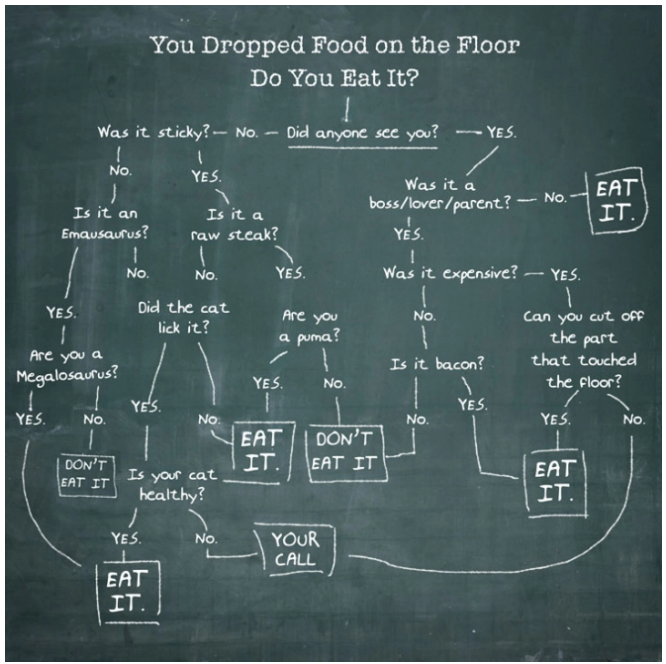
several methods can be classified as *learning these bases adaptively*

$$f(x) = \sum_d w_d \phi_d(x; v_d)$$

each basis has its own parameters

- decision trees
- generalized additive models
- boosting
- neural networks

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 they are unstable

- pruning
- random forests

image credit:https://mymodernmet.com/the-30second-rule-a-decision/

Decision trees: idea

divide the input space into regions and learn one function per region

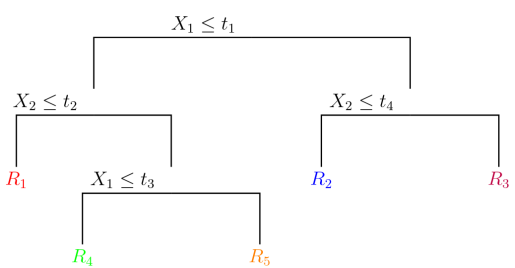
$$f(x) = \sum_k w_k \mathbb{I}(x \in \mathbb{R}_k)$$

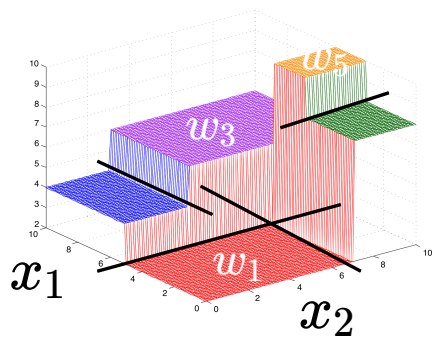
the regions are learned adaptively

more sophisticated prediction per region is also possible (e.g., one linear model per region)

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

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





Prediction per region

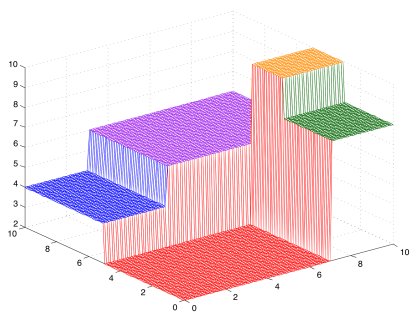
suppose we have identified the regions $\mathbb{R}_k$

what constant w_k to use for prediction in each region?

for regression

use the **mean value** of training data-points in that region

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



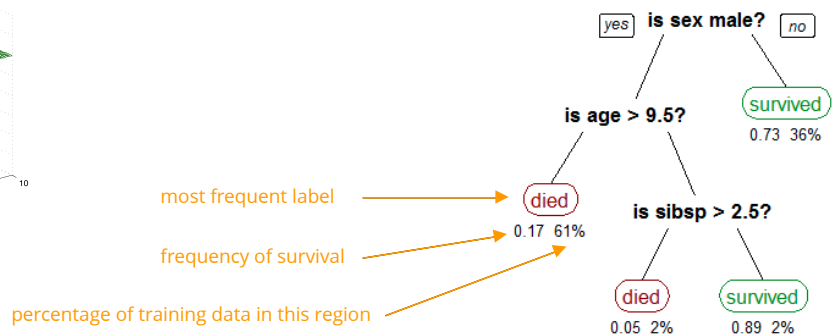
for classification

count the frequency of classes per region

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

or return probability

example: predicting survival in titanic



Feature types

given a feature what are the possible tests

continuous features - *e.g., age, height, GDP*

all the values that appear in the dataset can be used to split $\mathbb{S}_d = \{s_{d,n} = x_d^{(n)}\}$

one set of possible splits for each feature d

each split is asking $x_d > s_{d,n}$?

ordinal features - *e.g., grade, rating* $x_d \in \{1, \dots, C\}$

we can split any any value so $\mathbb{S}_d = \{s_{d,1} = 1, \dots, s_{d,C} = C\}$

each split is asking $x_d > s_{d,c}$?

categorical features -

- types, classes and categories

multi-way split $x_d = \{ \diamondsuit ? \; \heartsuit ? \; \clubsuit ? \; \spadesuit ? \}$

problem:

it could lead to sparse subsets

data fragmentation: some splits may have few/no datapoints

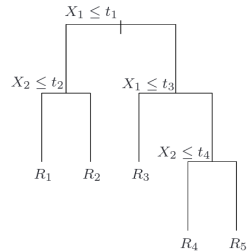
binary split

assume C binary features (one-hot coding)

instead of $x_d \in \{1, \dots, C\}$ we have

$$\begin{cases} x_{d,1} \in \{0,1\} \\ x_{d,2} \in \{0,1\} \\ \vdots \\ x_{d,C} \in \{0,1\} \end{cases} \quad \{ \clubsuit \spadesuit \heartsuit \overset{?}{=} 0 \;\; \diamondsuit \overset{?}{=} 1$$

alternative: binary splits that produce balanced subsets



Cost function

objective: find a decision tree minimizing the **cost function**

regression cost

for predicting constant $w_k \in \mathbb{R}$

cost per region (mean squared error - MSE)

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

N_k: number of instances in region k

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

classification cost

for predicting constant class $w_k \in \{1, \dots, C\}$

cost per region (misclassification rate)

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

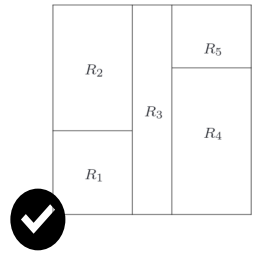
$w_k = \text{mode}(y^{(n)} | x^{(n)} \in \mathbb{R}_k)$

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

it is sometimes possible to build a tree with **zero cost**:
build a large tree with each instance having its own region (*overfitting*!)

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

Search space



K+1 regions

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

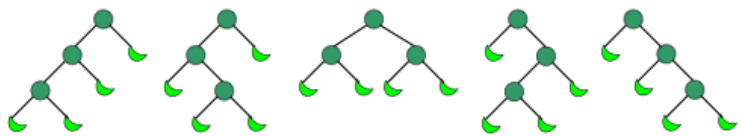
alternatively, find the smallest tree (K) that classifies all examples correctly

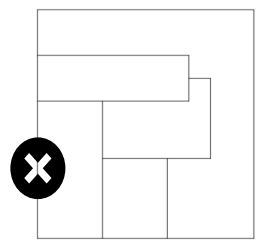
assuming D features how many different partitions of size K+1?

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

exponential in 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





not produced by a decision tree

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 $x_d \in \mathbb{S}_d$

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

Greedy heuristic

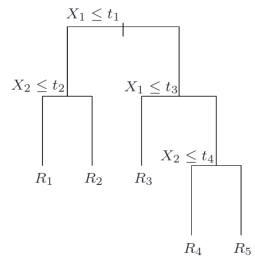
recursively split the regions based on a greedy choice of the next test

end the recursion if not worth-splitting

```
function fit-tree(ℝ_node , 𝒟 ,depth)

    ℝ_left , ℝ_right  = greedy-test ( ℝ_node , 𝒟 )
    if not worth-splitting(depth, ℝ_left , ℝ_right )
         return ℝ_node
    else
        left-set = fit-tree( ℝ_left , 𝒟, depth+1)
        right-set = fit-tree(ℝ_right , 𝒟, depth+1)
        return {left-set, right-set}
```

final decision tree in the form of nested list of regions



$\{\{\mathbb{R}_1, \mathbb{R}_2\}, \{\mathbb{R}_3, \{\mathbb{R}_4, \mathbb{R}_5\}\}$

Choosing tests

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

```
function greedy-test ( $\mathbb{R}_{\text{node}}, \mathcal{D}$ )
    best-cost = -inf
    for $d \in \{1, \ldots, D\}, s_{d,n} \in \mathbb{S}_d$
        $\mathbb{R}_{\text{left}} = \mathbb{R}_k \cup \{x_d < s_{d,n}\}$
        $\mathbb{R}_{\text{right}} = \mathbb{R}_k \cup \{x_d \geq s_{d,n}\}$
        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 $\mathbb{R}^*_{\text{left}}, \mathbb{R}^*_{\text{right}}$
```

seqarch through all single tests

creating new regions

evaluate their cost

return the split with the lowest greedy cost

Stopping the recursion

worth-splitting subroutine

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

heuristics for stopping the splitting:

- reached a desired depth
- number of examples in $\mathbb{R}_{\text{left}}$ 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

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

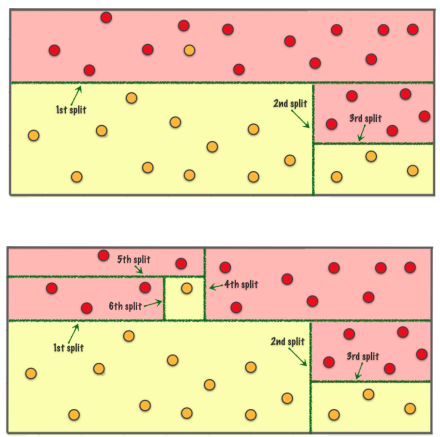


image credit: https://alanjeffares.wordpress.com/tutorials/decision-tree/

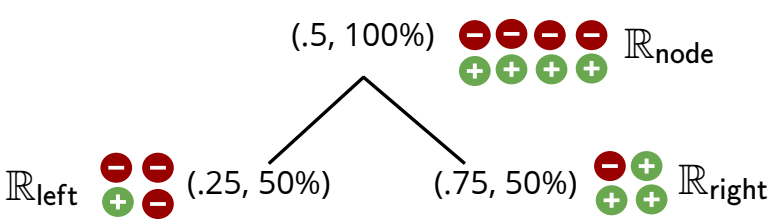
revisiting the classification cost

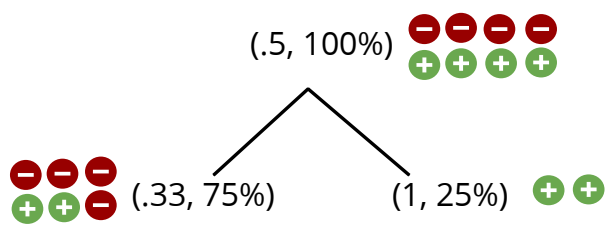
ideally we want to optimize the 0-1 loss (misclassification rate)

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

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

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





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

use a measure for homogeneity of labels in regions

Entropy a measure of the *unpredictability*

information, surprise, uncertainty of *one* particular outcome of a random variable

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

> zero information of 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 $-\log(p(c)q(d)) = -\log p(c) - \log q(d)$

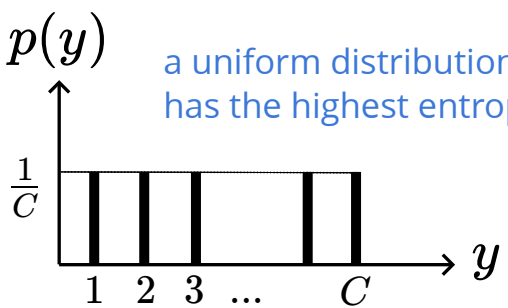
Entropy a measure of the *unpredictability*

averaged on all its possible outcomes

information, surprise, uncertainty of *one* particular outcome of a random variable

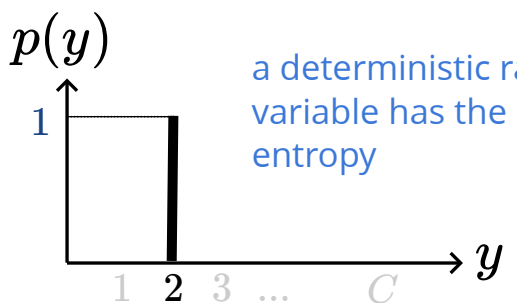
$$I(y = c) = \log(\frac{1}{p(y=c)}) = -log(p(y = c))$$

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



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

$H(y) = -1 \log(1) = 0$

Entropy

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

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

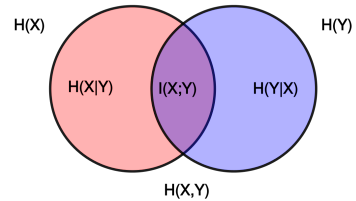
$$H(y) = -\sum_{c=1}^{C} p(y=c) \log p(y=c)$$

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

> zero information of 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

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

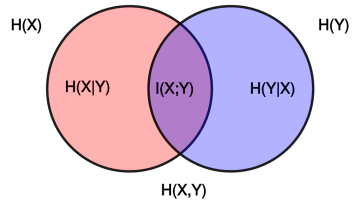
$$H(y|t) = -\sum_{l=1}^{L} p(t=l) \sum_c p(y=c|t=l) \log p(y=c|t=l) =$$
$$-\sum_{l=1}^{L} \sum_c p(t=l) \frac{p(y=c,t=l)}{p(t=l)} \log \frac{p(y=c,t=l)}{p(t=l)} = -\sum_{l=1}^{L} \sum_c p(y=c,t=l) \log \frac{p(y=c,t=l)}{p(t=l)}$$

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

$$I(y) = \sum_{c=1}^{C} \sum_{l=1}^{L} p(y=c,t=l) \left(\frac{p(y=c,t=l)}{p(t=l)} - \log p(y=c) \right)$$

$$= \sum_l \sum_c p(y=c,t=l) \log \frac{p(y=c,t=l)}{p(y=c)p(t=l)}$$

Mutual information



for two random variables t, y

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

$$= \sum_l \sum_c p(y=c, t=l) \log \frac{p(y=c,t=l)}{p(y=c)p(t=l)}$$

this is symmetric wrt **y** and **t**

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

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

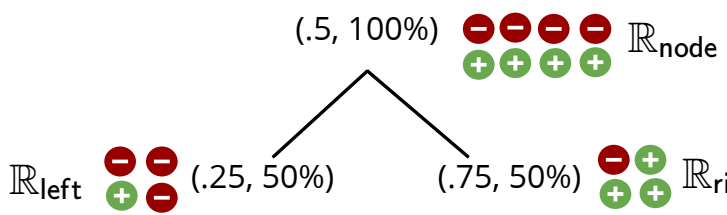
it is always positive and zero only if **y** and **t** are independent

Entropy for classification cost

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

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 $w_k = \arg\max_c p_k(c)$



$p(y = +) = \frac{1}{4}, p(y = -) = \frac{3}{4}$ $\quad$ $p(y = +) = \frac{3}{4}, p(y = -) = \frac{1}{4}$

$w_k = -$ $\quad$ $w_k = +$

misclassification cost =

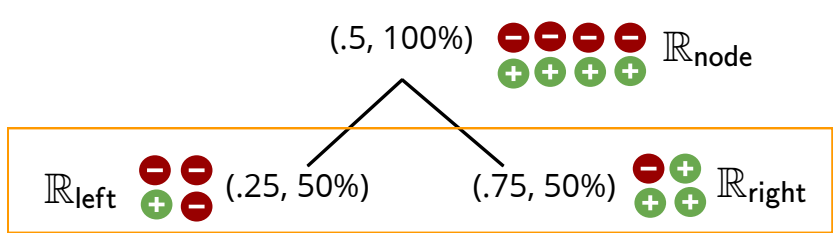
Entropy for classification cost

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

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



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

misclassification cost

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

$p(y=+) = \frac{1}{4}, p(y=-) = \frac{3}{4}$

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

Entropy for classification cost

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

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

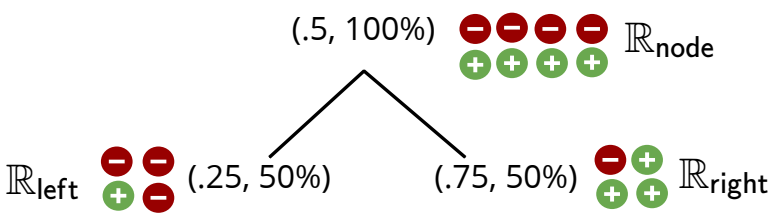
$$\text{cost}(\mathbb{R}_{\text{node}}, \mathcal{D}) - \left(\frac{N_{\text{left}}}{N_{\text{node}}} \text{cost}(\mathbb{R}_{\text{left}}, \mathcal{D}) + \frac{N_{\text{right}}}{N_{\text{node}}} \text{cost}(\mathbb{R}_{\text{right}}, \mathcal{D}) \right)$$

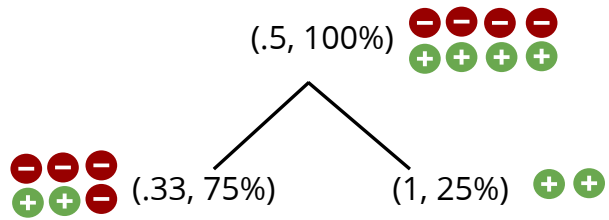
$$I(t,y) = H(y) - H(y|t)$$
$$\sum_{l=1}^{L} p(t=l) H(y|t=l)$$

$$= H(y) - \left(p(x_d \geq s_{d,n}) H(y|x_d \geq s_{d,n}) + p(x_d < s_{d,n}) H(y|x_d < s_{d,n}) \right) = I(y, x > s_{d,n})$$

choosing the test which is **maximally informative** about labels

example Entropy for classification cost





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)

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

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

lower cost split

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

C=2

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

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

$2p(1-p)$

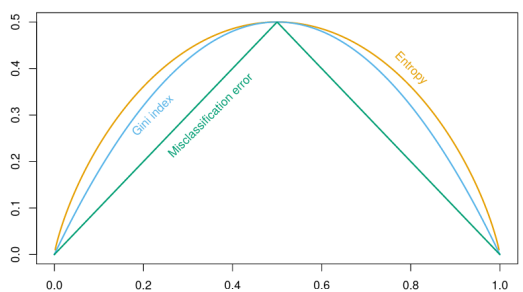
Gini index it is the expected error rate

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

probability of class c probability of error

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

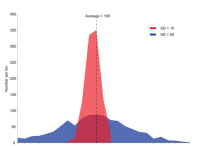
comparison of costs of a node when we have 2 classes



cost

$p(y = 1)$

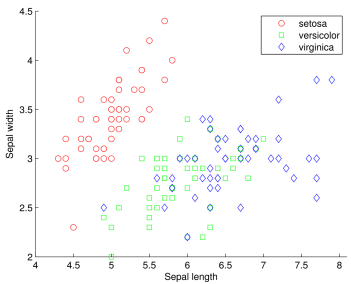
entropy & gini very similar and favor pure nodes



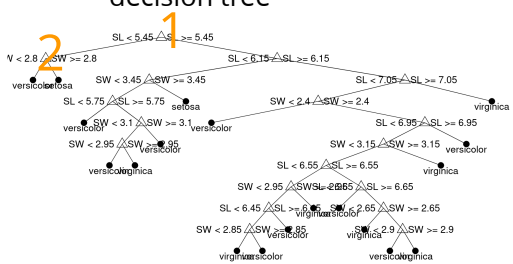
Example

decision tree for Iris dataset

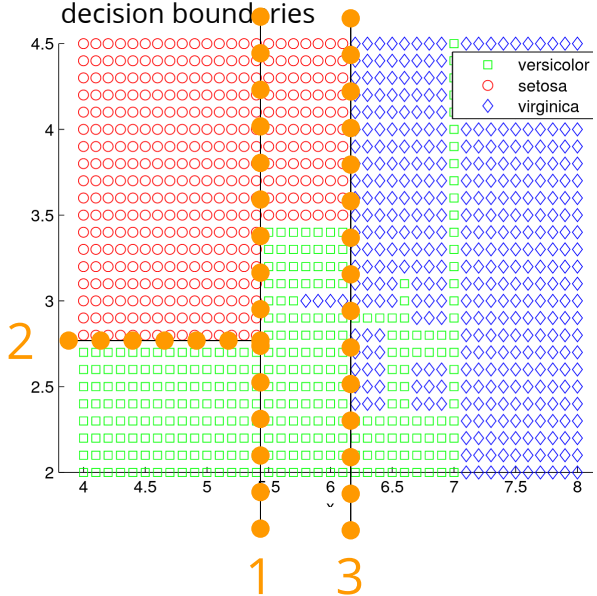
dataset (D=2)



decision tree



decision boundaries



decision boundaries suggest overfitting

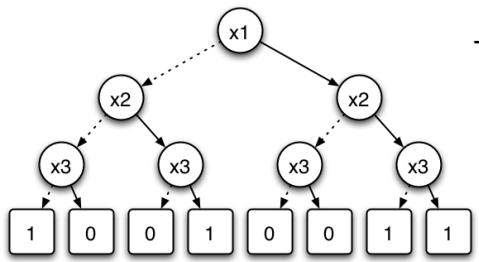
confirmed using a validation set

training accuracy	*~ 85%*
(Cross) validation accuracy	*~ 70%*

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)



x1	x2	x3	f
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

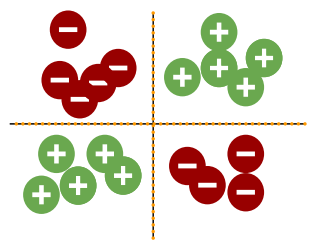
there are 2^{2^D} such functions, why?

large decision trees have a high variance - low bias *(low training error, high test error)*

idea 1. grow a small tree

substantial reduction in cost may happen after a few steps

by stopping early we cannot know this



example cost drops after the second node

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

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

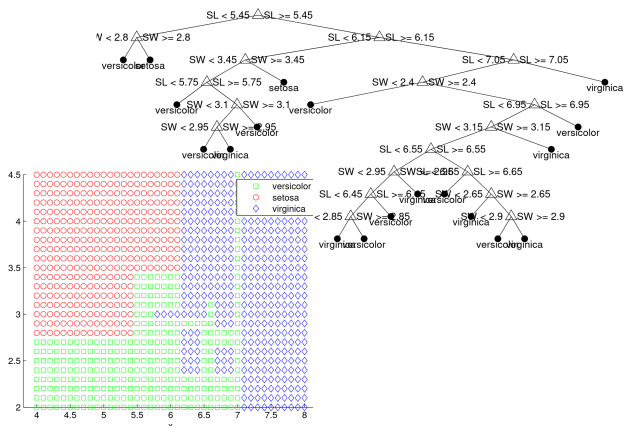
repeat this until left with the root node

pick the best among the above models using using a validation set

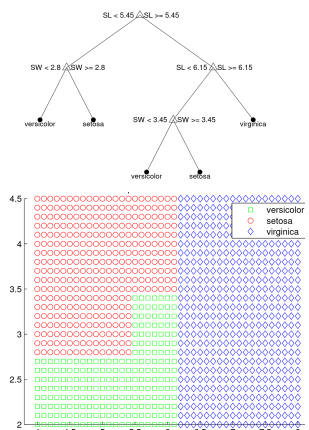
idea 3. random forests (later!)

example

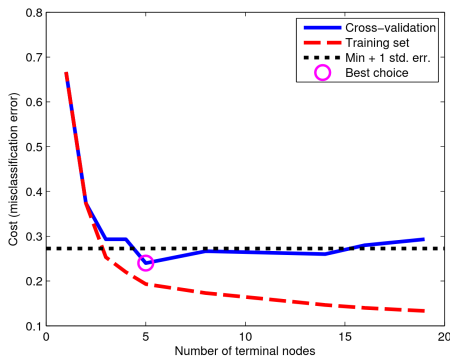
before pruning



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
- decision trees are unstable (have high variance)
  - use pruning to avoid overfitting
- there are variations on decision tree heuristics
  - what we discussed in called *Classification and Regression Trees (CART)*