The question of whether computers can think is like the question of whether submarines can swim.

- E. W. Dijsktra

CSE455 & CSE 552 Machine Learning

Spring 2025

Decision Trees

1

Supervised Machine Learning

of experiences

observed outcome

observed input

$$\arg\max_{f \in F} \sum_{i=1}^{n} g(y_i, f(x_i))$$

hypothesis/program space

goodness/performance measure

selected hypothesis

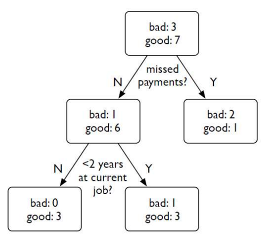
2

Decision Trees

- Classifying from a set of attributes

Predicting credit risk

<2 years at current job?	missed payments?	defaulted?
N	N	N
Y	N	Y
N	N	N
N	N	N
N	Y	Y
Y	N	N
N	Y	N
N	Y	Y
Y	N	N
Y	N	N



- Each level splits the data according to different attributes
- **Goal**: achieve perfect classification with minimal number of decisions
  - not always possible due to noise or inconsistencies in the data

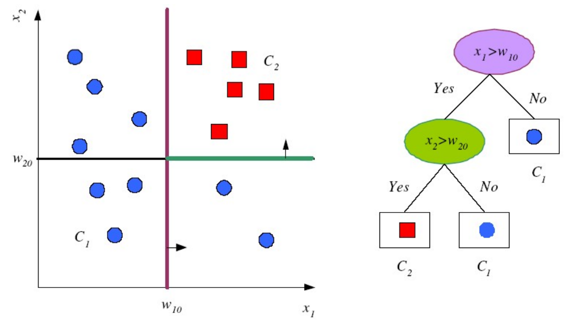
3

Observations

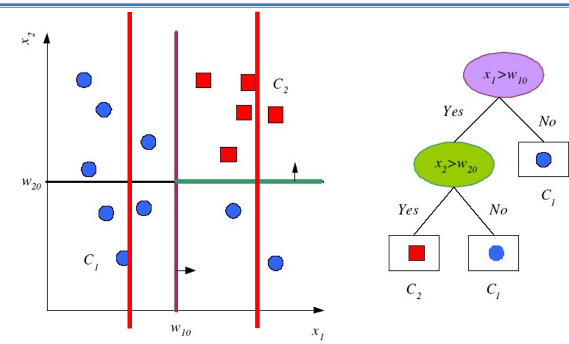
- Any boolean function can be represented by a decision tree
- Not good for all functions, e.g.:
  - parity function: return 1 iff an even number of inputs are 1
  - majority function: return 1 if more than half inputs are 1
- Best when a small number of attributes provide a lot of information
- Note: finding optimal tree for arbitrary data is NP-hard

4

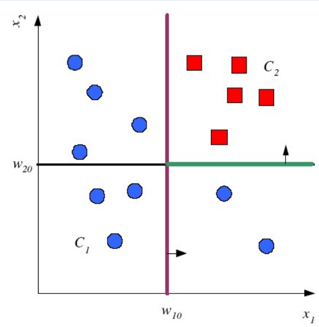
Tree Uses Nodes, and Leaves



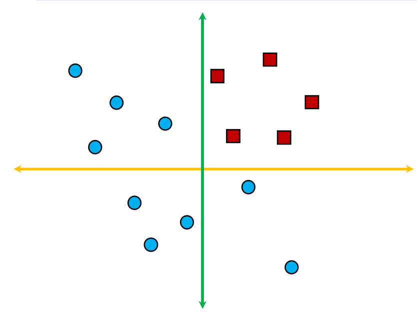
Tree Uses Nodes, and Leaves



Tree Uses Nodes, and Leaves

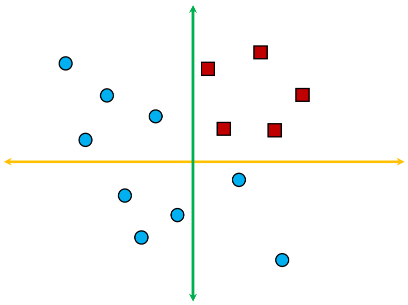


How to build a DT?



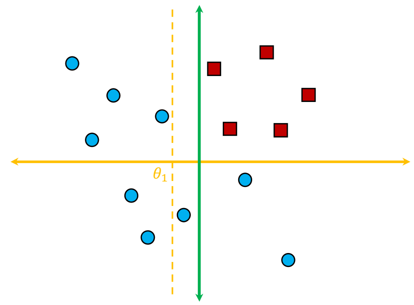
Training data: $\vec{x}_i = (x_1, x_2) \in \mathcal{R}^2$ for i $= 1, \ldots, n$

Possible Decisions



9

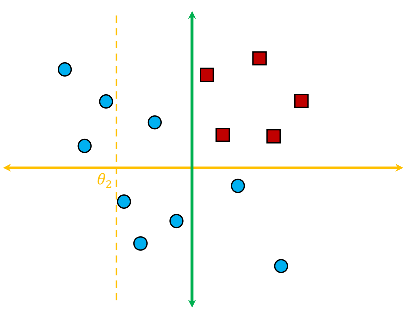
Possible Decisions



Decision 1: $x_1 > \theta_1$

10

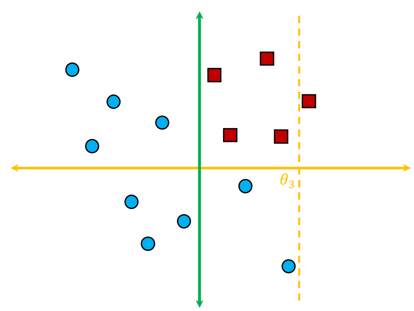
Possible Decisions



Decision 1: $x_1 > \theta_2$

11

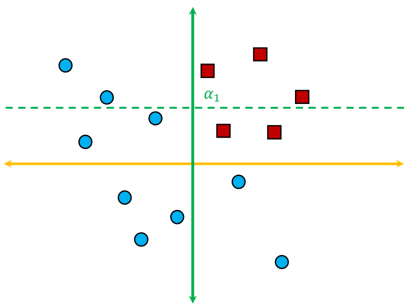
Possible Decisions



Decision 1: $x_1 > \theta_3$

12

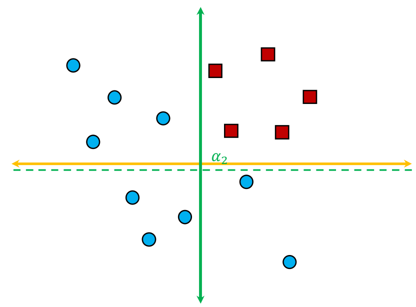
Possible Decisions



Decision 2: $x_2 > \alpha_1$

13

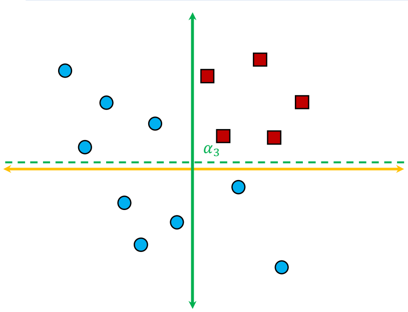
Possible Decisions



Decision 2: $x_2 > \alpha_2$

14

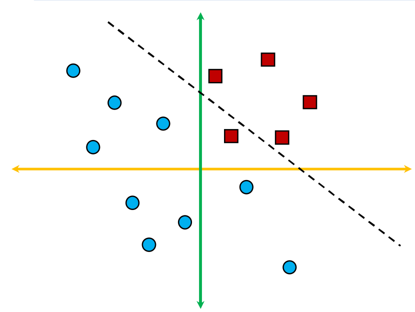
Possible Decisions



Decision 2: $x_2 > \alpha_3$

15

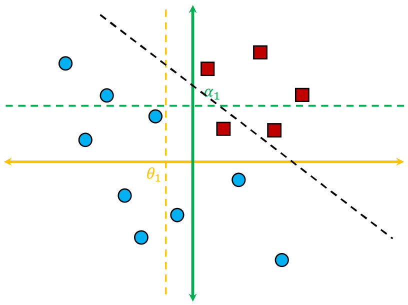
Possible Decisions



Decision 3: $a_1 x_1 + a_2 x_2 + a_3 > 0$

16

Possible Decisions



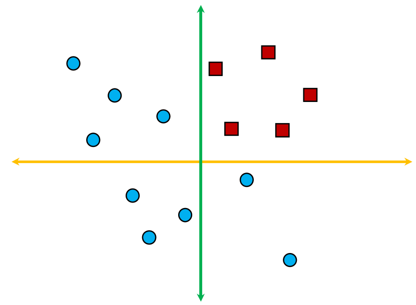
Decision 1: $x_1 > \theta_1$

Decision 2: $x_2 > \alpha_1$

Decision 3: $a_1x_1 + a_2x_2 + a_3 > 0$

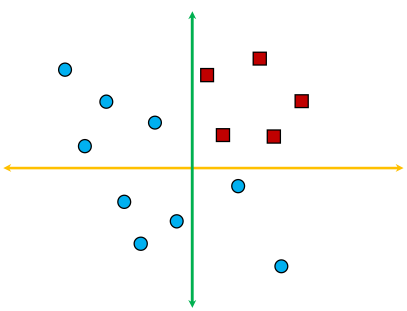
17

Search Algorithm



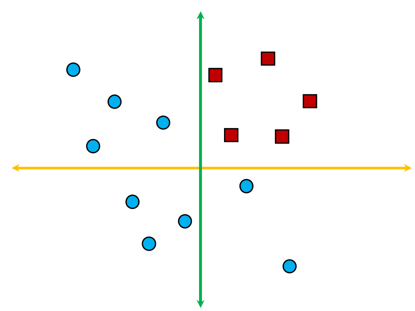
18

Search Algorithm



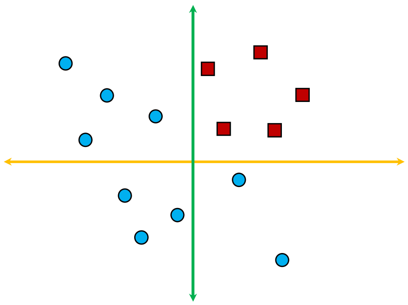
19

Search Algorithm

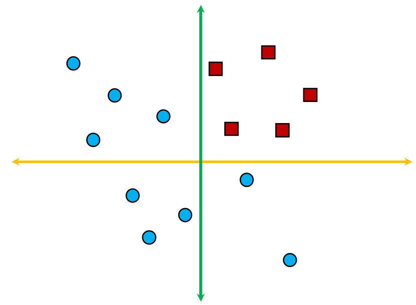


20

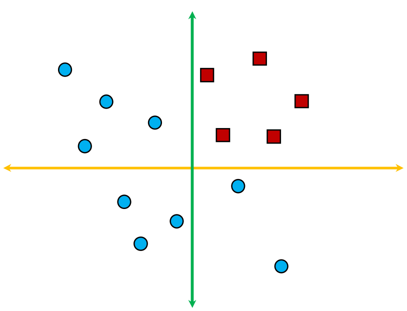
Search Algorithm



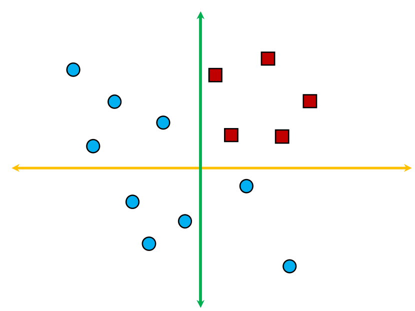
Search Algorithm



Search Algorithm



Search Algorithm



Divide and Conquer

- Internal decision nodes
  - Univariate: Uses a single attribute, x_i
  - Numeric x_i: Binary split : $x_i > w_m$
    - Discrete x_i: *n*-way split for *n* possible values
  - Multivariate: Uses all attributes, $\vec{x}$
- Leaves
  - Classification: Class labels, or proportions
  - Regression: Numeric; *r* average, or local fit
- Learning is **greedy**; find the best split recursively (Breiman et al, 1984; Quinlan, 1986, 1993)

Classification Trees (ID3, CART, C4.5)

- For node m, N_m instances reach m, ${N_m}^i$ belong to C_i

$$\hat{P}(C_i \mid \mathbf{x}, m) \equiv p_m^i = \frac{N_m^i}{N_m}$$

- Node m is **pure** if $p^i{}_m$ is 0 or 1
- Measure of **impurity** is **entropy**

$$\mathcal{H}_m = -\sum_{i=1}^{K} p_m^i \log_2 p_m^i$$

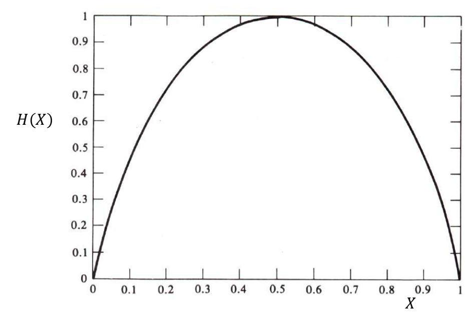
Information and Entropy

- For a random variable X with probability P(x), the entropy is the average (or expected) amount of information obtained by observing x:

$$H(X) = \sum_x P(x) I(x) = -\sum_x P(x) \log_2 P(x)$$

- Information: $I(x) = -\log_2 P(x)$
- $H(X)$ depends only on the probability, not the value

Entropy of a Binary Random Variable



Credit Risk – Entropy

- How many bits does it take to specify the attribute of 'defaulted?'
  - P(defaulted = Y) = 3/10
  - P(defaulted = N) = 7/10

$$H(X) = -\sum_{i=Y,N} P(Y = y_i)\log_2 P(Y = y_i)$$
$$= -0.3\log_2 0.3 - 0.7\log_2 0.7$$
$$= 0.8813$$

- How much can we reduce the entropy (or uncertainty) of 'defaulted' by knowing the other attributes?
  - Ideally, we could reduce it to zero, in which case we classify perfectly

Predicting credit risk

<2 years at current job?	missed payments?	defaulted?
N	N	N
Y	N	Y
N	N	N
N	N	N
N	Y	Y
Y	N	N
N	Y	N
N	Y	Y
Y	N	N
Y	N	N

Best Split

- If node m is pure, generate a leaf and stop, otherwise split and continue recursively
- Impurity after split: N_{mj} of N_m take branch j. N^i_{mj} belong to C_i

$$\hat{P}(C_i \mid \boldsymbol{x}, m, j) \equiv p^i_{mj} = \frac{N^i_{mj}}{N_{mj}}$$
$$\mathcal{H}'_m = -\sum_{j=1}^{n} \frac{N_{mj}}{N_m} \sum_{i=1}^{K} p^i_{mj} \log_2 p^i_{mj}$$

- Find the variable and split that min impurity (among all variables – and split positions for numeric variables)

GenerateTree($\mathcal{X}$)
  If NodeEntropy($\mathcal{X}$)< θ_I /* eq. 9.3
    Create leaf labelled by majority class in $\mathcal{X}$
    Return
  $i \leftarrow$ SplitAttribute($\mathcal{X}$)
  For each branch of $\boldsymbol{x}_i$
    Find $\mathcal{X}_i$ falling in branch
    GenerateTree($\mathcal{X}_i$)

SplitAttribute($\mathcal{X}$)
  MinEnt← MAX
  For all attributes $i = 1, \ldots, d$
    If $\boldsymbol{x}_i$ is discrete with n values
      Split $\mathcal{X}$ into $\mathcal{X}_1, \ldots, \mathcal{X}_n$ by $\boldsymbol{x}_i$
      e ← SplitEntropy($\mathcal{X}_1, \ldots, \mathcal{X}_n$) /* eq. 9.8 */
      If e<MinEnt MinEnt ← e; bestf ← i
    Else /* $\boldsymbol{x}_i$ is numeric */
      For all possible splits
        Split $\mathcal{X}$ into $\mathcal{X}_1, \mathcal{X}_2$ on $\boldsymbol{x}_i$
        e←SplitEntropy($\mathcal{X}_1, \mathcal{X}_2$)
        If e<MinEnt MinEnt ← e; bestf ← i
  Return bestf

Regression Trees

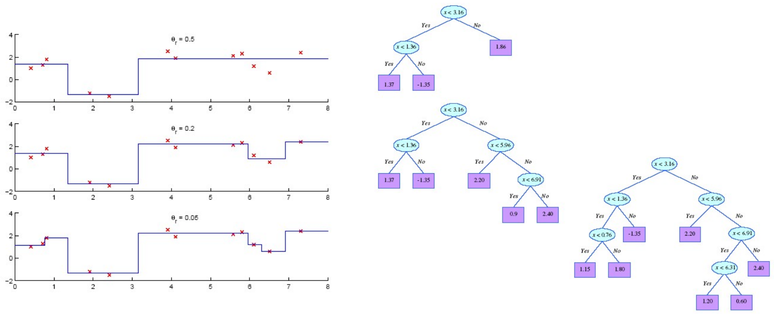
- Error at node m:

$$b_m(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \boldsymbol{x} \in \mathcal{X}_m : \boldsymbol{x} \text{ reaches node } m \\ 0 & \text{otherwise} \end{cases}$$
$$E_m = \frac{1}{N_m}\sum_t \left(r^t - g_m\right)^2 b_m(\boldsymbol{x}^t) \qquad g_m = \frac{\sum_t b_m(\boldsymbol{x}^t) r^t}{\sum_t b_m(\boldsymbol{x}^t)}$$

- After splitting:

$$b_{mj}(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \boldsymbol{x} \in \mathcal{X}_{mj} : \boldsymbol{x} \text{ reaches node } m \text{ and branch } j \\ 0 & \text{otherwise} \end{cases}$$
$$E'_m = \frac{1}{N_m}\sum_j \sum_t \left(r^t - g_{mj}\right)^2 b_{mj}(\boldsymbol{x}^t) \qquad g_{mj} = \frac{\sum_t b_{mj}(\boldsymbol{x}^t) r^t}{\sum_t b_{mj}(\boldsymbol{x}^t)}$$

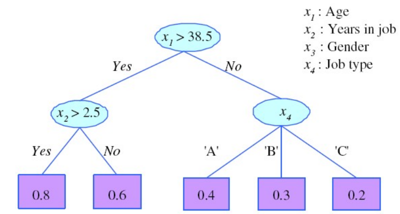
Model Selection in Trees



33

Rule Extraction from Trees

C4.5Rules
(Quinlan, 1993)



R1: IF (age>38.5) AND (years-in-job>2.5) THEN y =0.8
R2: IF (age>38.5) AND (years-in-job≤2.5) THEN y =0.6
R3: IF (age≤38.5) AND (job-type='A') THEN y =0.4
R4: IF (age≤38.5) AND (job-type='B') THEN y =0.3
R5: IF (age≤38.5) AND (job-type='C') THEN y =0.2

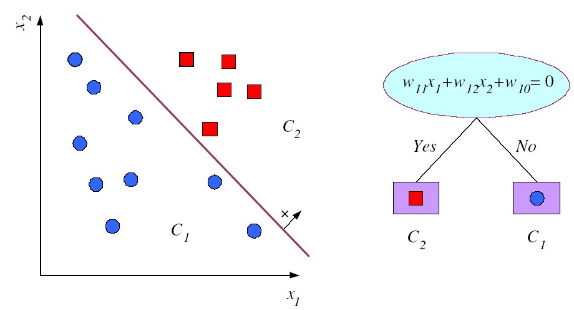
34

Learning Rules

- Rule induction is similar to tree induction but
  - tree induction is breadth-first,
  - rule induction is depth-first; one rule at a time
- Rule set contains rules; rules are conjunctions of terms
- Rule **covers** an example if all terms of the rule evaluate to true for the example
- **Sequential covering:** Generate rules one at a time until all positive examples are covered
- IREP (Fürnkrantz and Widmer, 1994), Ripper (Cohen, 1995)

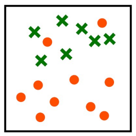
35

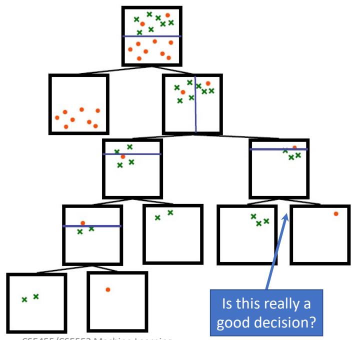
Multivariate Trees



36

Overfitting

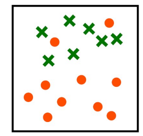


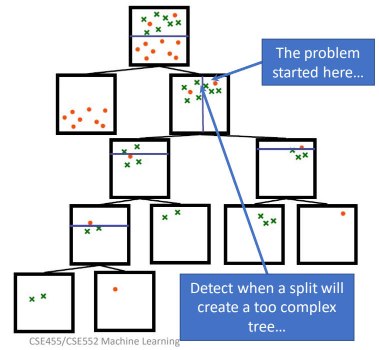


- noise makes the decision tree more complex than it should be
- the algorithm tries to classify all of the training set perfectly

37

Overfitting

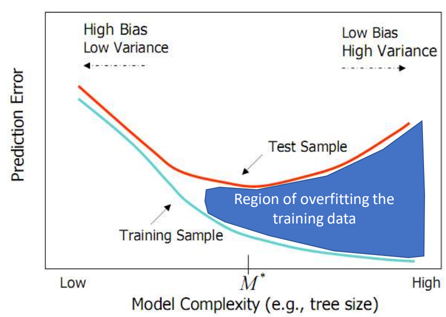




- noise makes the decision tree more complex than it should be
- the algorithm tries to classify all of the training set perfectly

38

Bias and Variance



39

Addressing Overfitting

- Grow tree based on training data
- This yields an unpruned tree
- Then prune nodes from the tree that are unhelpful.
- How do we know when this is the case?
  - Use additional data not used in training, i.e., test data
  - Use a statistical significance test to see if extra nodes are different from noise
  - Penalize the complexity of the tree

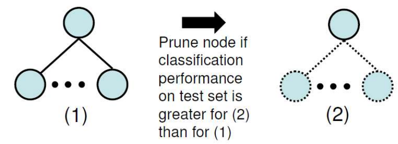
40

Pruning Trees

- Remove subtrees for better generalization (or to avoid overfitting) (or decrease variance)
  - Prepruning: Early stopping
  - Postpruning: Grow the whole tree then prune subtrees which overfit on the pruning set
- Prepruning is faster, postpruning is more accurate (requires a separate pruning set)

Pruning

- Construct standard decision tree, but keep a test data set on which the model is not trained
- Prune leaves recursively
- Splits are eliminated (or pruned) by evaluating performance on the test data
- A leaf is pruned if classification on the test data increases by removing the split



Statistical Significance Tests

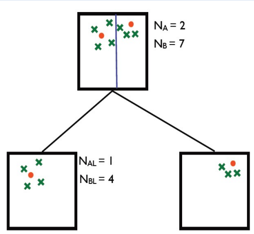
- For each split, ask of there is a significant increase in the information gain
- If we're splitting noise, then data are random
- What proportion of data go to left node?

$$p_L = \frac{N_{AL} + N_{BL}}{N_A + N_B}$$

- If data were random, how many would we *expect* to go to the left?

$$M_{AL} = N_A \times p_L = \frac{10}{9}$$
$$M_{BL} = N_B \times p_L = \frac{35}{9}$$

- Is there a statistically significant difference from what we observe and what we expect?



- # class A in root node is $N_A = 2$
- # class B in root node is $N_B = 7$
- # class A in left node is $N_{AL} = 1$
- # class B in left node is $N_{BL} = 4$

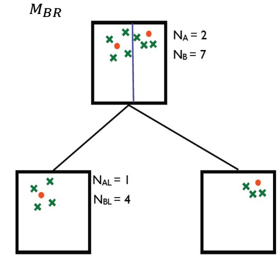
Statistical Significance Tests

- A measure of statistical significance

$$K = \frac{(M_{AL} - N_{AL})^2}{M_{AL}} + \frac{(M_{BL} - N_{BL})^2}{M_{BL}} + \frac{(M_{AR} - N_{BR})^2}{M_{BR}} + \frac{(M_{BR} - N_{BR})^2}{M_{BR}}$$

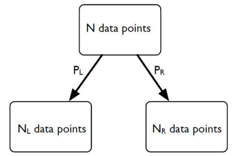
- K measures how much the split deviates from what we would expect from random data
- K small → *the information gain from the split is not significant*
- Here,

$$K = \frac{(10/9 - 1)^2}{10/9} + \frac{(35/9 - 4)^2}{35/9} + \ldots = 0.0321$$



χ^2 Criterion – General Case

- Small "Chi-square" values imply low statistical significance
- Nodes that have K smaller than threshold are pruned
- The threshold regulates the complexity of the model
  - Low thresholds allow larger trees and more overfitting
  - High thresholds keep trees small but may sacrifice performance



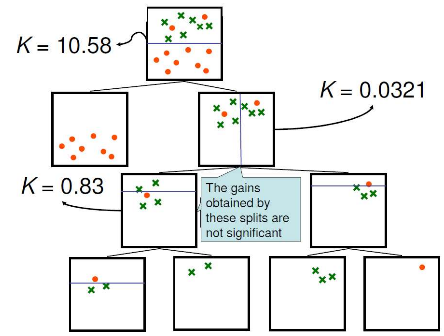
$$K = \sum_{\substack{\text{all classes } i \\ \text{all children } j}} \frac{(N_{ij} - N'_{ij})^2}{N'_{ij}}$$

N_{ij} = Number of points from class i in child j

N'_{ij} = Number of points from class i in child j *assuming random selection*

$N'_{ij} = N_i \times p_j$

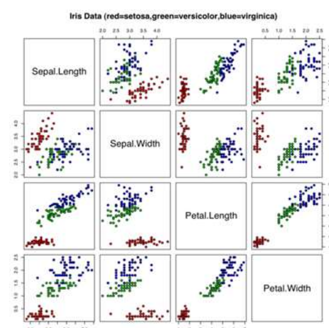
Example

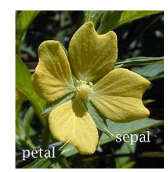


Better Example –Fisher's Iris data

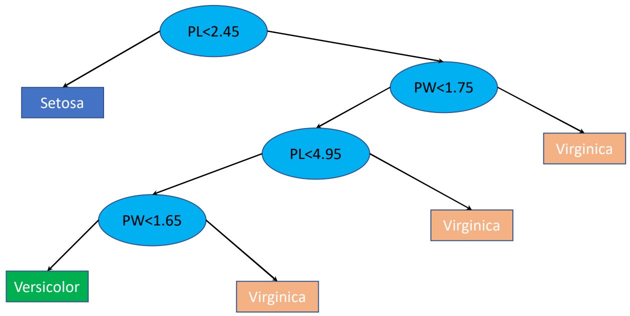
Class	Sepal Length (SL)	Sepal Width (SW)	Petal Length (PL)	Petal Width (PW)
Setosa	5.1	3.5	1.4	0.2
Setosa	4.9	3	1.4	0.2
Setosa	5.4	3.9	1.7	0.4
Versicolor	5.2	2.7	3.9	1.4
Versicolor	5	2	3.5	1
Versicolor	6	2.2	4	1
Virginica	6.4	2.8	5.6	2.1
Virginica	7.2	3	5.8	1.6

Features



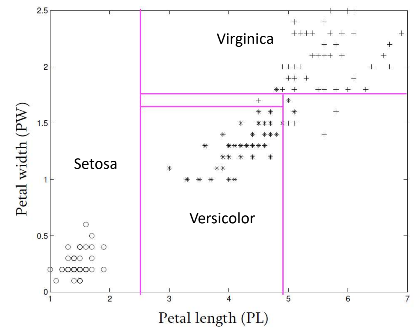


Unpruned Decision Tree



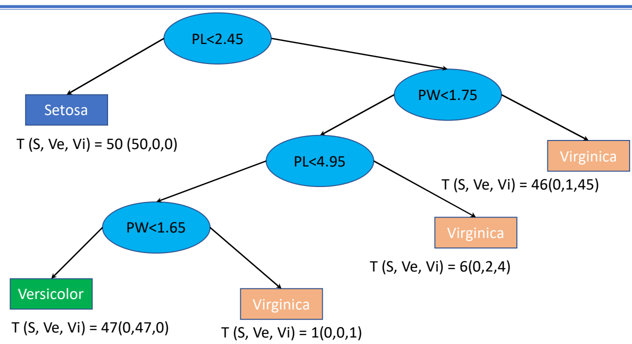
49

Scatter Plot Data w/ Decision Boundaries



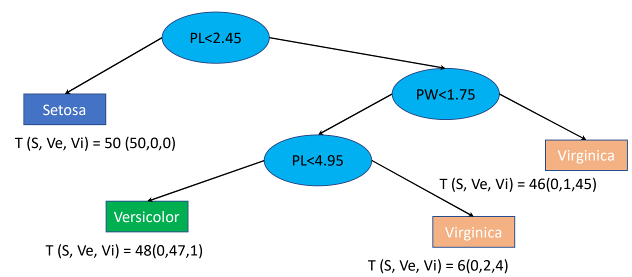
50

Tree Statistics



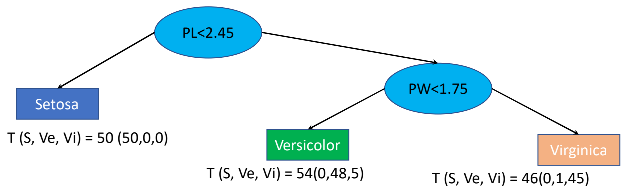
51

Pruning One Level



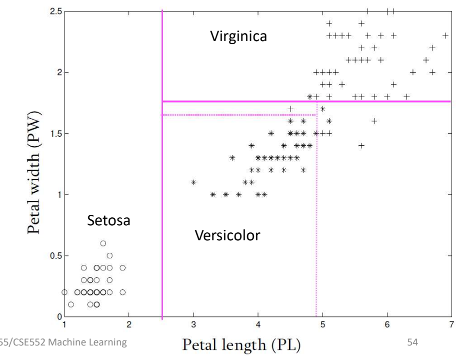
52

Pruning Two Levels



53

Tree w/ Pruned Decision Boundaries



54

Decision Tree Advantages

- Easy to interpret the decision rules
- Nonparametric so it is easy to incorporate a range of numeric or categorical data layers
- Universally applicable to both classification and regression problems
- Invariant to monotone transformation of input variables
- Robust against outliers in training data
- High resistance to irrelevant input variables
- Classification is fast once rules are developed
- Provide valuable insights for data structure

55

Decision Tree Disadvantages

- Poor accuracy - SVM often have 30% lower error rates
- Decision trees tend to overfit training data which can give poor results when applied to the full data set
- Instability (high variance) – If we change the data a little, the tree picture can change a lot
- Splitting perpendicular to feature space axes is not always efficient
- Not possible to predict beyond the minimum and maximum limits of the response variable in the training data

56

Thanks for listening!