Data Mining

CS57300
Purdue University

Bruno Ribeiro

February 8, 2018

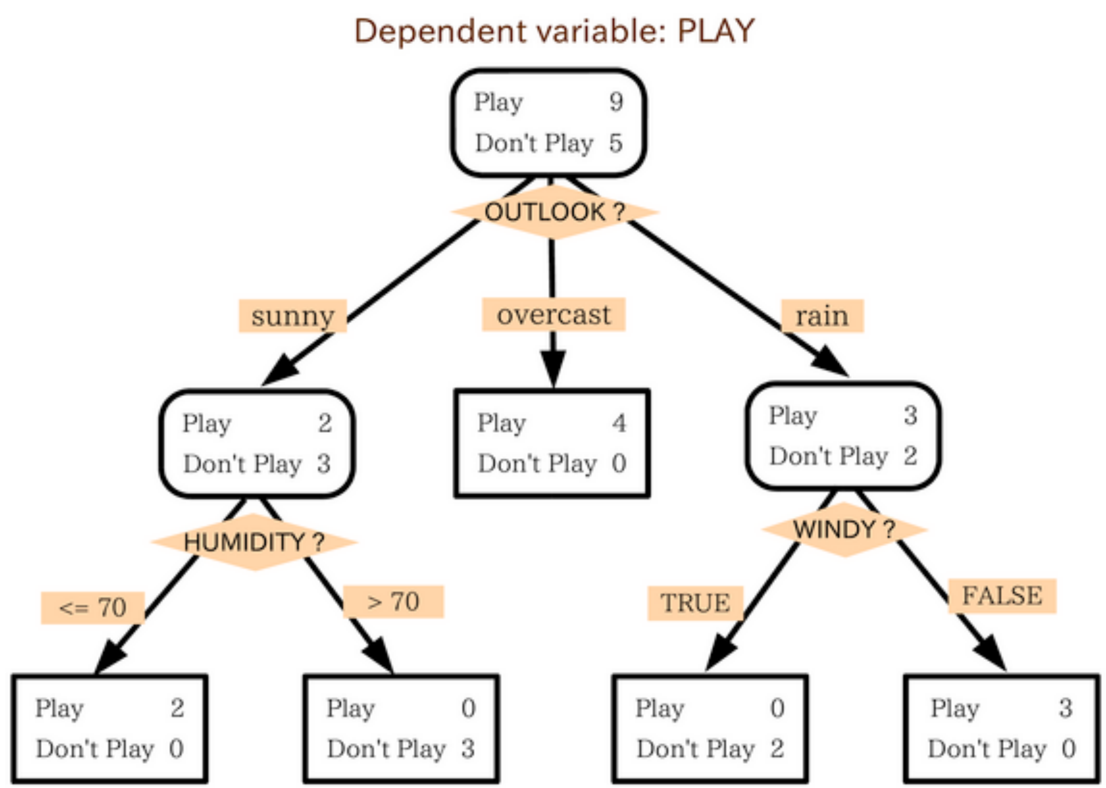
Decision trees

Why Trees?

- interpretable/intuitive, popular in medical applications because they mimic the way a doctor thinks
- model discrete outcomes nicely
- can be very powerful, can be as complex as you need them
- C4.5 and CART - from “top 10” entries on Kaggle - decision trees are very effective and popular

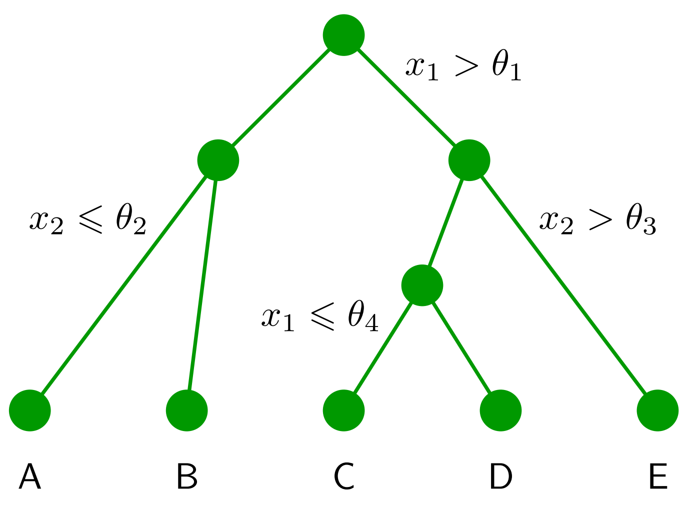
Sure, But Why Trees?

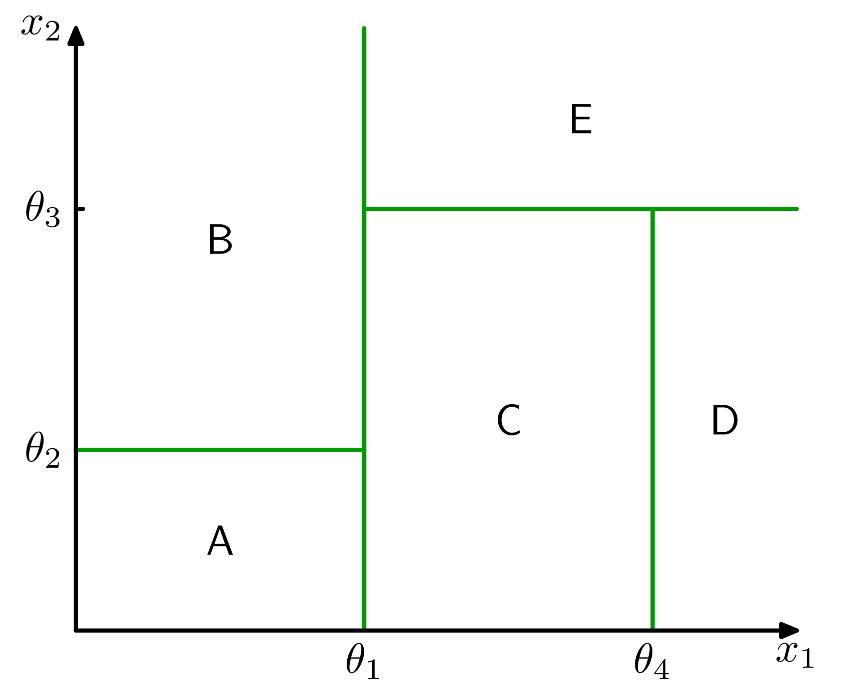
- Easy to understand knowledge representation
- Can handle mixed variables
- Recursive, divide and conquer learning method
- Efficient inference
- Example: Play outside =>



”Divide-and-conquer” Classification

- Consider input tuples $(\mathbf{x}_i, y_i)$ for i-th observation
- s





Tree learning

- Finding best tree is intractable
  Must consider all 2^m combinations, where m is number of features
- Often just greedily grow it by “splitting” attributes one by one.
  To determine which attribute to split, look at “node impurity.”
- Top-down recursive divide and conquer algorithm
  - Start with all examples at root
  - Select **best** attribute/feature
  - Recurse and repeat
- Other issues:
  - How to construct features
  - When to stop growing
  - Pruning irrelevant parts of the tree

Fraud	Age	Degree	StartYr	Series7
+	22	Y	2005	N
-	25	N	2003	Y
-	31	Y	1995	Y
-	27	Y	1999	Y
+	24	N	2006	N
-	29	N	2003	N

Score each attribute split for these instances:
Age, Degree, StartYr, Series7

Y **choose split on *Series7*** **N**

Fraud	Age	Degree	StartYr	Series7
-	25	N	2003	Y
-	31	Y	1995	Y
-	27	Y	1999	Y

Fraud	Age	Degree	StartYr	Series7
+	22	Y	2005	N
+	24	N	2006	N
-	29	N	2003	N

choose split on *Age>28*

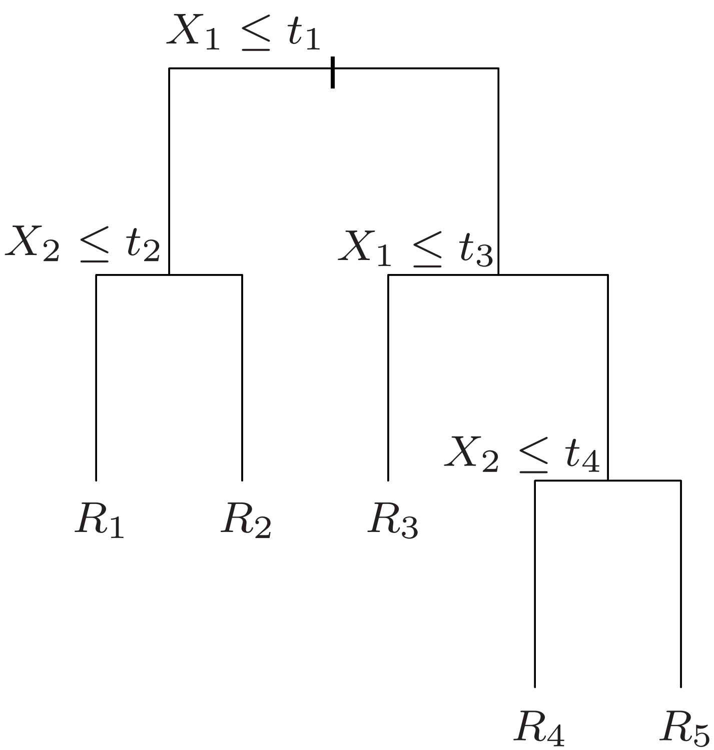
Score each attribute split for these instances:
Age, Degree, StartYr

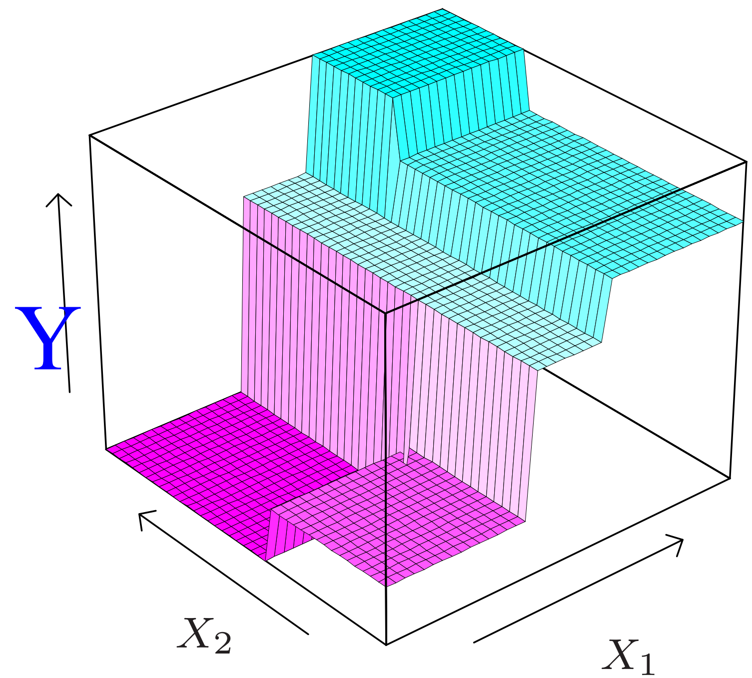
Y **N**

Fraud	Age	Degree	StartYr	Series7
-	29	N	2003	N

Fraud	Age	Degree	StartYr	Series7
+	22	Y	2005	N
+	24	N	2006	N

Overview (with two features and 1D target)





Features: X_1, X_2

Target: Y

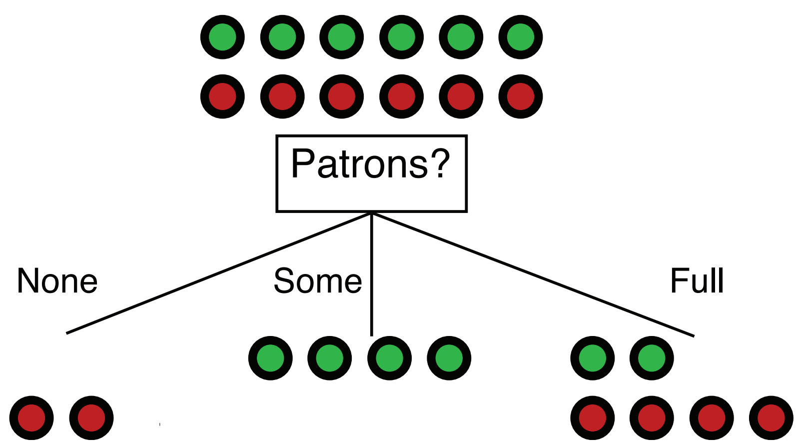
Tree models

- Most well-known systems
    - CART: Breiman, Friedman, Olshen and Stone
    - ID3, C4.5: Quinlan
- How do they differ?
    - **Split scoring function**
    - Stopping criterion
    - Pruning mechanism
    - Predictions in leaf nodes

Scoring functions: Local split value

Choosing an attribute/feature

- Idea: a good feature splits the examples into subsets that distinguish among the class labels as much as possible... ideally into pure sets of "all positive" or "all negative"

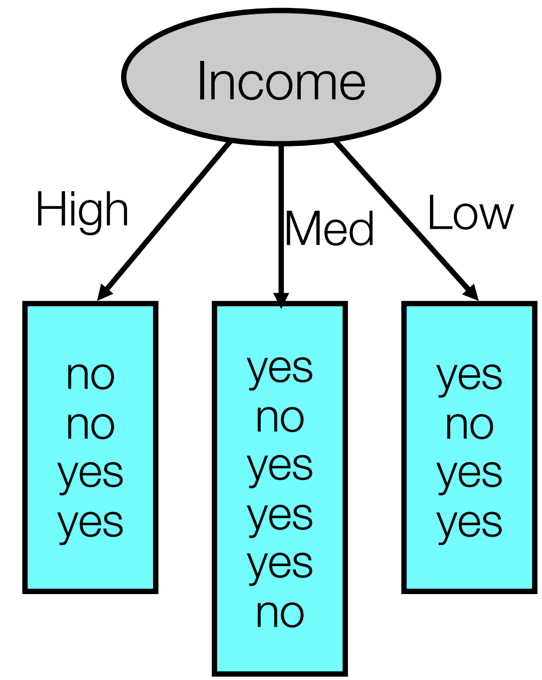


- Bias-variance tradeoff: Choosing most discriminating attribute first may not be best tree (*bias*), but it can make tree small (*low variance*).

Association between attribute and class label

Data

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no



Contingency table

Class label value

Attribute value

	Buy	No buy
High	2	2
Med	4	2
Low	3	1

Mathematically Defining “Good Split”

- We start with information theory
    - How uncertain of the answer will be if we split the tree this way?
- Say need to decide between k options
    - Uncertainty in the answer $Y \in \{1,\ldots,k\}$ when probability is $(p_1, \ldots, p_k)$ can be quantified via entropy:

$$H(p_1, \ldots, p_k) = \sum_i -p_i \log_2 p_i$$

- Convenient notation: $\mathrm{B}(p) = \mathrm{H}(p, 1-p)$, number of bits necessary to encode

Amount of Information in the Tree

- Suppose we have p positive and n negative examples at the root
  ⇒ $B(p/(p+n))$ bits needed to classify a new example
- Information is always conserved
  - If encoding the information in the leaves is lossless then tree has lossless encoding
  - The entropy of the leaves (amount of bits) + the tree information (bits) carried in the tree = total information in the data

- Let split Y_i have p_i positive and n_i negative examples
  ⇒ $B(p_i/(p_i+n_i))$ bits needed to classify a new example
  ⇒ expected number of bits per example over all branches is

$$\sum_i \frac{p_i+n_i}{p+n} B(p_i/(p_i+n_i))$$

⇒ choose the next attribute to split that minimizes the remaining information needed

- Which maximizes the information in the tree (as information is conserved)

Information gain

- Information Gain (Gain) is the amount of information that the tree structure encodes

H[X] is the entropy: expected number of bits to encode a randomly selected subset X

$\mathcal{A}$ is the set of subsets of the data with a given split

$\mathcal{S}$ is the entire data

$$\text{Gain}(S, \mathcal{A}) = H[S] - \sum_{A \subset \mathcal{A}} \frac{|A|}{|S|} H[A]$$

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

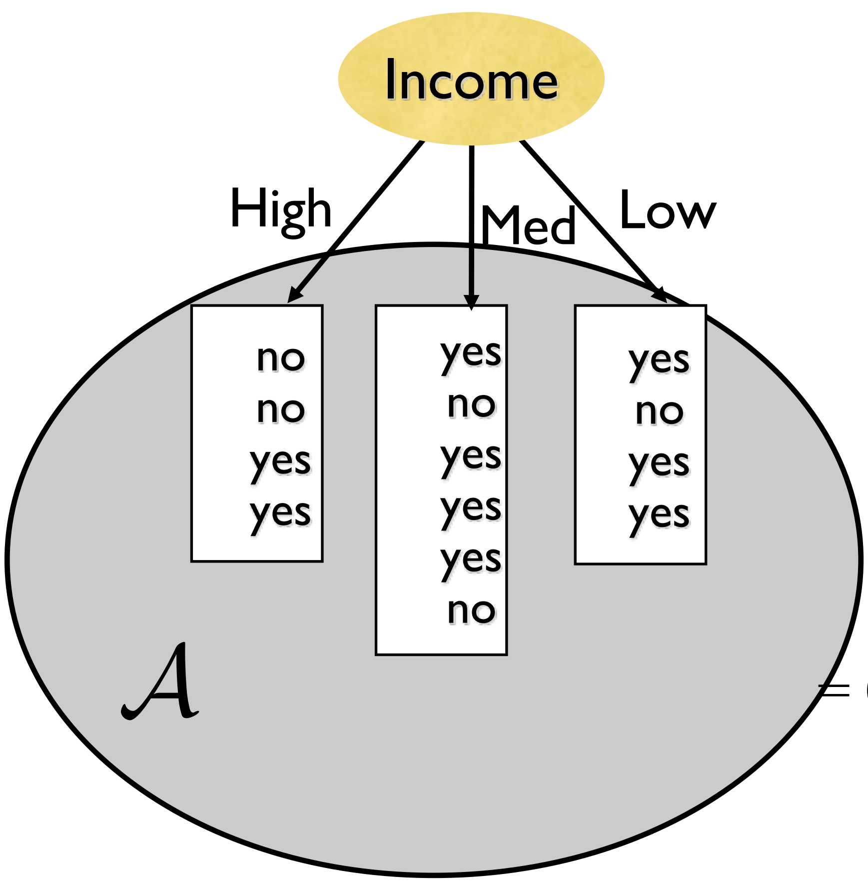
$H[buys\_computer]$

$= -9/14 \log 9/14 - 5/14 \log 5/14$

$= 0.9400$

Information gain

$$\text{Gain}(S, \mathcal{A}) = H[S] - \sum_{A \subset \mathcal{A}} \frac{|A|}{|S|} H[A]$$



Entropy(Income=high)
= -2/4 log 2/4 -2/4 log 2/4 = 1

Entropy(Income=med)
= -4/6 log 4/6 -2/6 log 2/6 = 0.9183

Entropy(Income=low)
= -3/4 log 3/4 -1/4 log 1/4 = 0.8113

Gain(D,Income)
= 0.9400 - (4/14 [1] + 6/14 [0.9183] + 4/14 [0.8113])
= 0.029

Gini gain

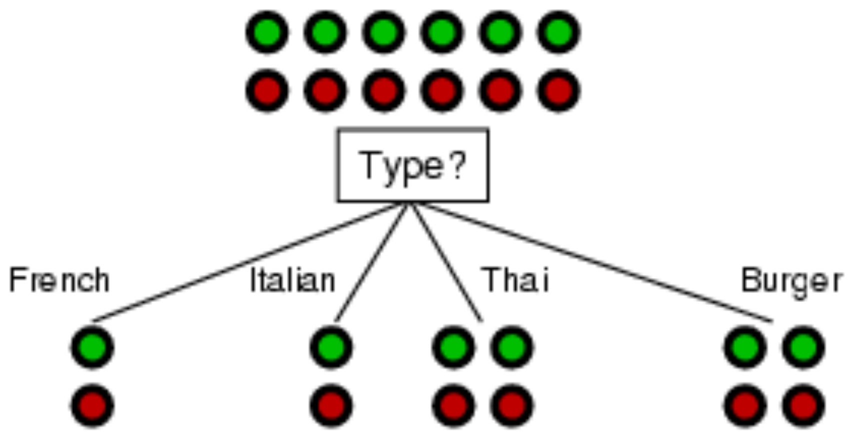
- Similar to information gain
- Uses gini index instead of entropy

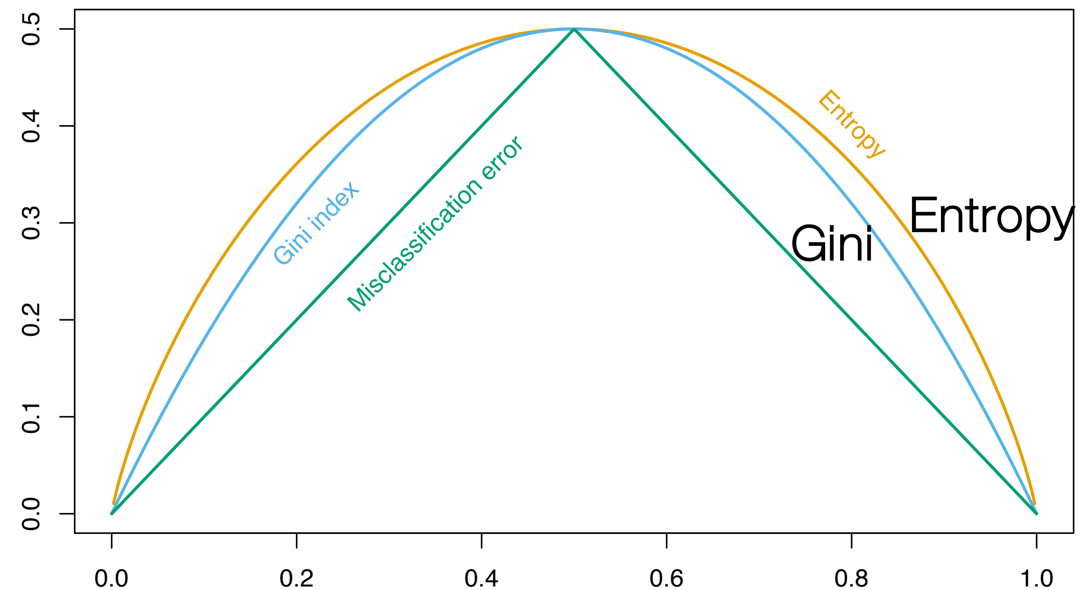
$$Gini(X) = 1 - \sum_{x} p(x)^2$$

- Measures decrease in gini index after split:

$$\mathrm{Gain}(S, \mathcal{A}) = \mathrm{Gigi}(S) - \sum_{A \subset \mathcal{A}} \frac{|A|}{|S|} \mathrm{Gini}(A)$$

Comparing information gain to gini gain



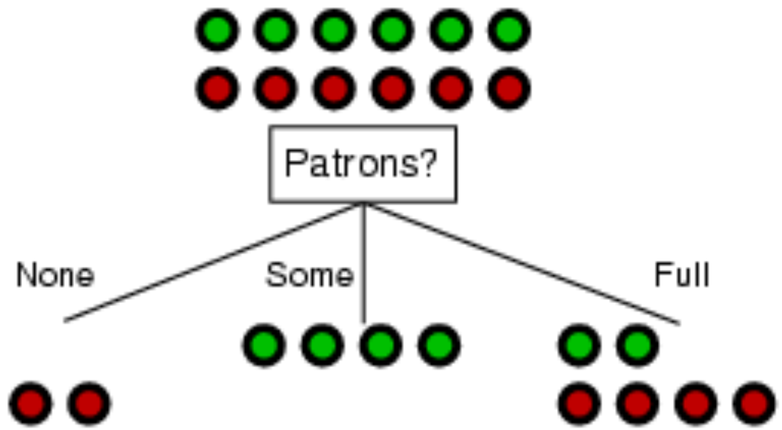


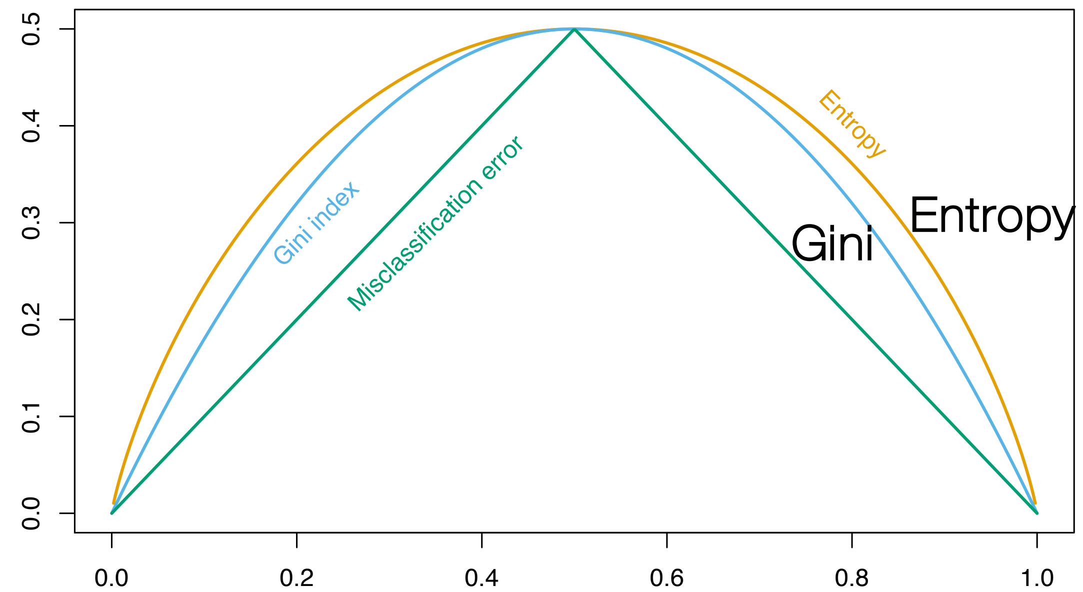
Fraction of target A into branch that outputs B

Information Gain $IG = 0$

Gini Gain $GG = 0$

Comparing information gain to gini gain



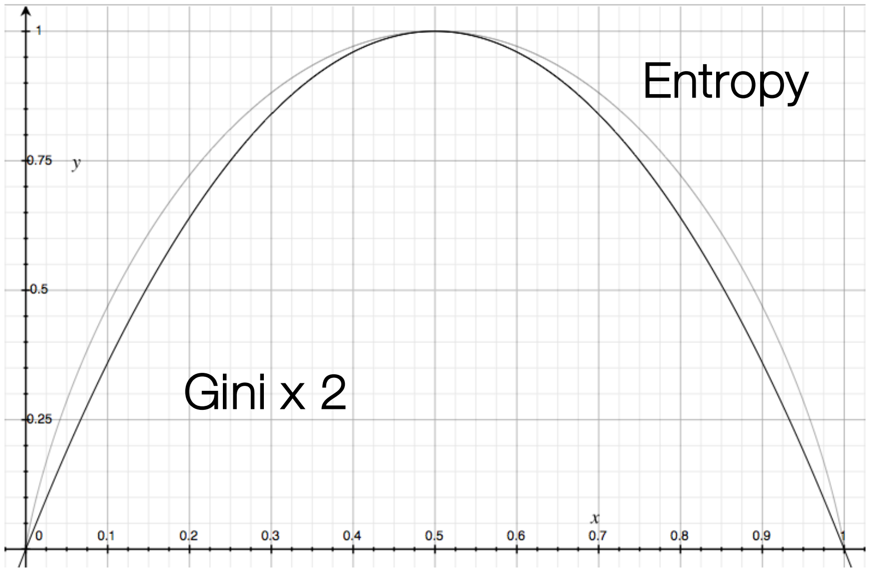


Fraction of target A into branch that outputs B

$$IG = 1.0 - \left[\frac{2}{12}\,0\right] - \left[\frac{4}{12}\,0\right] - \left[\frac{6}{12}\,0.919\right] = 0.541$$

$$GG = 0.5 - \left[\frac{2}{12}\,0\right] - \left[\frac{4}{12}\,0\right] - \left[\frac{6}{12}\,0.444\right] = 0.278$$

How does score function affect feature selection?



$$66\% \text{ split} : Entropy = 0.919 \qquad 85\% \text{ split} : Entropy = 0.610$$

$$Gini \times 2 = 0.889 \qquad Gini \times 2 = 0.510$$

Gini score can produce larger gain

Chi-Square score

- Widely used to test independence between two categorical attributes (e.g., feature and class label)
- Hypothesis → H_0 : Attributes are independent
- Consider a contingency table with **k** entries (k = rows x columns)
- Considers counts in a contingency table and calculates the normalized squared deviation of observed (predicted) values from expected (actual) values given H_0

$$\mathcal{X}^2 = \sum_{i=1}^{k} \frac{(o_i - e_i)^2}{e_i}$$

- If counts are large (large number of examples), sampling distribution can be approximated by a chi-square distribution

Contingency tables

Income

	Buy	No buy	
High	2	2	4
Med	4	2	6
Low	3	1	4
	9	5	14

Calculating expected values for a cell

$$\mathcal{X}^2 = \sum_{i=1}^{k} \frac{(o_i - e_i)^2}{e_i}$$

		Class	
		+	–
Attribute	0	a	b
	1	c	d
			N

$$o_{(0,+)} = a$$

$$\begin{aligned} e_{(0,+)} &= p(A=0, C=+) \cdot N \\ &= p(A=0)p(C=+|A=0) \cdot N \\ &= p(A=0)p(C=+) \cdot N \qquad \text{(assuming independence)} \\ &= \left[\frac{a+b}{N}\right] \cdot \left[\frac{a+c}{N}\right] \cdot N \end{aligned}$$

Example calculation

Observed

	Buy	No buy
High	2	2
Med	4	2
Low	3	1

Expected

	Buy	No buy
High	2.57	1.43
Med	3.86	2.14
Low	2.57	1.43

$$\chi^2 = \sum_{i=1}^{k} \frac{(o_i - e_i)^2}{e_i} = \left(\frac{(2-2.57)^2}{2.57}\right) + \left(\frac{(4-3.86)^2}{3.86}\right) + \left(\frac{(3-2.57)^2}{2.57}\right)$$

$$+ \left(\frac{(2-1.43)^2}{1.43}\right) + \left(\frac{(2-2.14)^2}{2.14}\right) + \left(\frac{(1-1.43)^2}{1.43}\right)$$

$$= 0.57$$

Tree learning

- Top-down recursive divide and conquer algorithm
  - Start with all examples at root
  - Select **best** attribute/feature
  - Partition examples by selected attribute
  - Recurse and repeat
- Other issues:
  - How to construct features
  - ***When to stop growing***
  - ***Pruning irrelevant parts of the tree***

Controlling Variance

- One major problem with trees is their high variance.
- Often a small change in the data can result in a very different series of splits, making interpretation somewhat precarious.
- The major reason for this instability is the hierarchical nature of the process:
  - the effect of an error in the top split is propagated down to all of the splits below it.

Overfitting

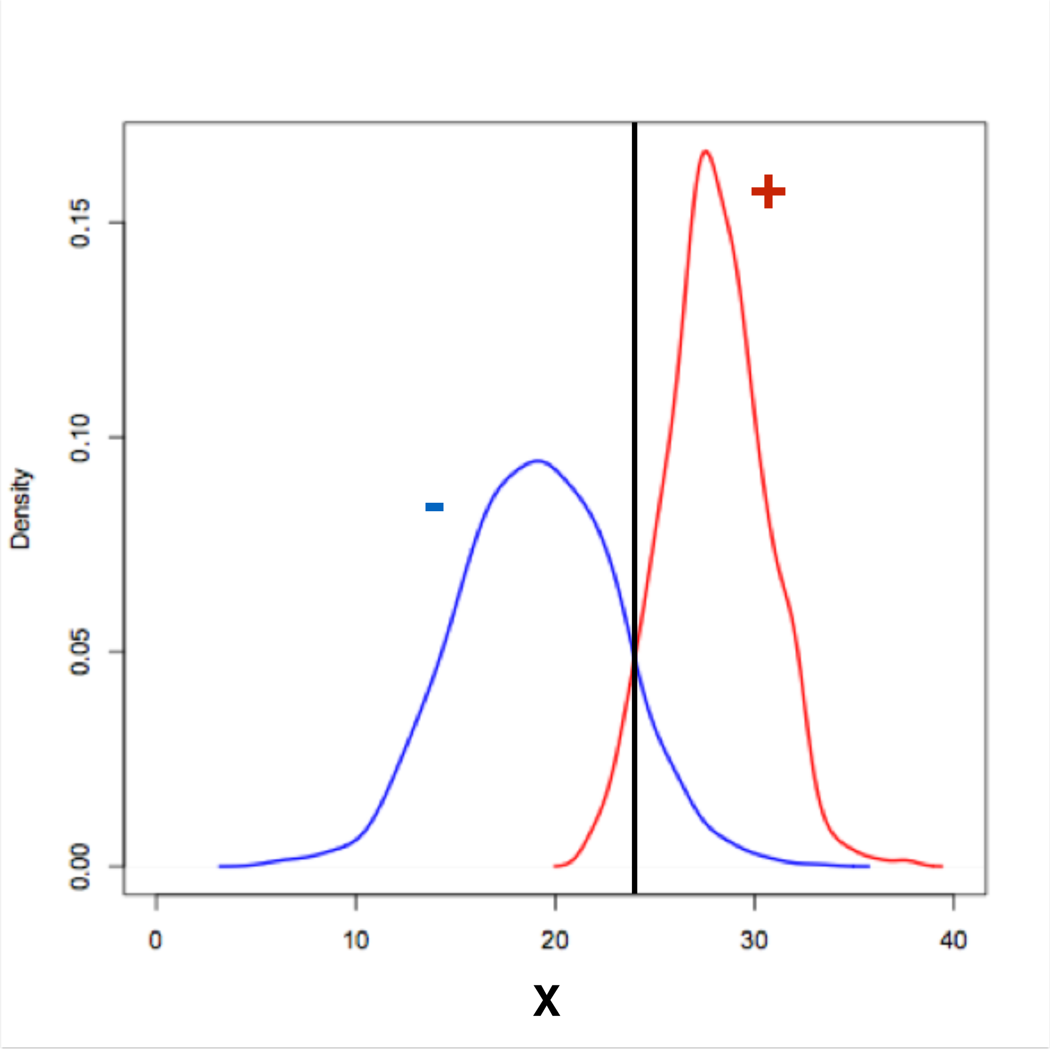
- Consider a distribution D of data representing a population and a sample D_S drawn from D, which is used as training data
- Given a model space M, a score function S, and a learning algorithm that returns a model m ∈ M, the algorithm **overfits** the training data D_S if:
  $\exists\ m' \in M$ such that $S(m,D_S) > S(m',D_S)$ but
  $S(m,D) < S(m',D)$
  - In other words, there is another model (m') that is better on the entire distribution and if we had learned from the full data we would have selected it instead

Example learning problem

Task: Devise a rule to classify items based on the attribute X

Knowledge representation:
If-then rules

Example rule:
If x > 25 then +
Else -



What is the model space?

All possible thresholds

What score function?

Prediction error rate

Approaches to avoid overfitting

- Regularization (Priors)
- Hold out evaluation set, used to adjust structure of learned model
  - e.g., pruning in decision trees
- Statistical tests during learning to only include structure with significant associations
  - e.g., pre-pruning in decision trees
- Penalty term in classifier scoring function
  - i.e., change scorre function to prefer simpler models

How to avoid overfitting in decision trees

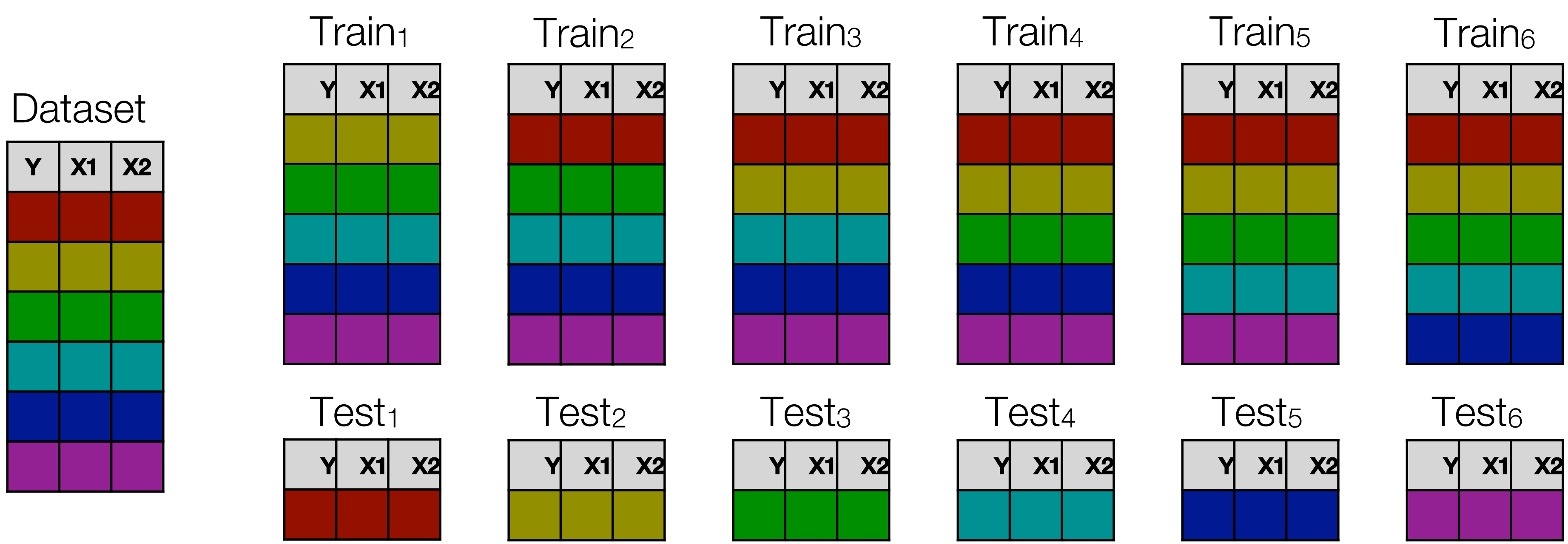
- Postpruning
  - Use a separate set of examples to evaluate the utility of pruning nodes from the tree (after tree is fully grown)
- Prepruning
  - Apply a statistical test to decide whether to expand a node
  - Use an explicit measure of complexity to penalize large trees (e.g., Minimum Description Length)

Algorithm comparison

- CART
  - Evaluation criterion: **Gini index**
  - Search algorithm: Simple to complex, hill-climbing search
  - Stopping criterion: When leaves are pure
  - Pruning mechanism: **Cross-validation to select Gini threshold**
- C4.5
  - Evaluation criterion: **Information gain**
  - Search algorithm: Simple to complex, hill-climbing search
  - Stopping criterion: When leaves are pure
  - Pruning mechanism: **Reduce error pruning**

CART: Finding Good Gini Threshold Background: K-fold cross validation

- Randomly **partition** training data into k folds
- For i=1 to k
    - Learn model on D - ith fold; evaluate model on ith fold
- Average results from all k trials

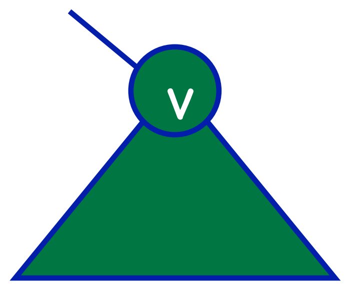


Choosing a Gini threshold with cross validation

- For i in 1.. k
  - For t in threshold set (e.g, [0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8])
    - Learn decision tree on Train_i with Gini gain threshold t (i.e. stop growing when max Gini gain is less than t)
    - Evaluate learned tree on Test_i (e.g., with accuracy)
  - Set $t_{max,i}$ to be the t with best performance on Test_i
- Set t_{max} to the average of $t_{max,i}$ over the k trials
- Relearn the tree on all the data using t_{max} as Gini gain threshold

C4.5: reduced error pruning

- Use **pruning set** to estimate accuracy in sub-trees and for individual nodes
- Let T be a sub-tree rooted at node v



- Define:

$$\text{Gain from prunning at } v = \#\text{misclassification in } T - \#\text{misclassification at } v$$

- Repeat: Prune at node with largest gain until until only negative gain nodes remain
- “Bottom-up restriction”: T can only be pruned if it does not contain a sub-tree with lower error than T

Source: www.ailab.si/blaz/predavanja/uisp/slides/uisp05-PostPruning.ppt

Pre-pruning methods

- Stop growing tree at some point during top-down construction when there is no longer sufficient data to make reliable decisions
- Approach:
  - Choose threshold on feature score
  - Stop splitting if the best feature score is below threshold

Determine chi-square threshold analytically

- Stop growing when chi-square feature score is not **statistically significant**

- Chi-square has known sampling distribution, can look up significance threshold

  - Degrees of freedom= (#rows-1)(#cols-1)

  - 2X2 table:
    3.84 is 95% critical value

$$\chi^2 = \sum_{i=1}^{k} \frac{(o_i - e_i)^2}{e_i}$$