ECE7252
Statistical Learning for Signal Processing

Lectures 22-23: Decision Trees

Chin-Hui Lee
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA 30332, USA
chl@ece.gatech.edu

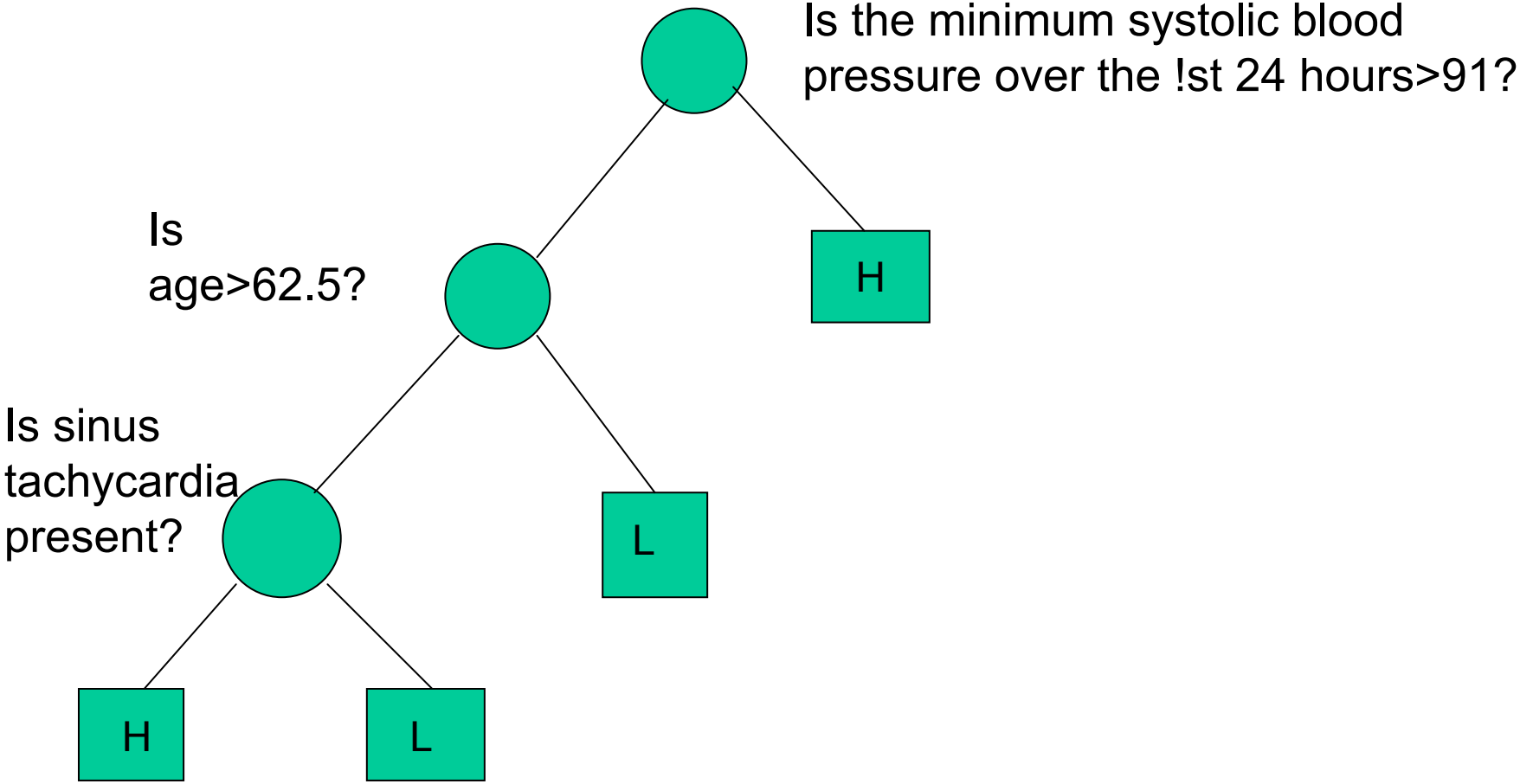
Variety of Approaches Used

- CART developed by Breiman, Friedman, Olsen and Stone: “Classification and Regression Trees”
- C4.5 : A Machine Learning Approach by Quinlan
- Engineering approach by Sethi and Sarvarayudu

Example

- University of California - a study into patients *after* admission for a heart attack
- 19 variables collected during the first 24 hours for 215 patients (for those who survived the 24 hours)
- Question: Can the high risk (will not survive 30 days) patients be identified?

Answer


Is the minimum systolic blood pressure over the !st 24 hours>91?
Is age>62.5?
Is sinus tachycardia present?
H
L
H
L

Plan for Construction of a Tree

- Selection of the Splits
- Decisions when to decide that a node is a terminal node (i.e. not to split it any further)
- Assigning a class to each terminal node
- Features of CART
  - Binary Splits
  - Splits based only on one variable

Combinatorics

- The branch of mathematics studying the enumeration, combination, and permutation of sets of elements and the mathematical relations which characterize these properties

- Examples of combinatorial problems:
  - Scheduling classes (or tournament play)
  - Determining cheapest route for a multi-city tour
  - Tracing a figure without picking up your pencil or repeating a line
  - Determining the odds of a full house in poker
  - Designing reliable networks
  - Creating efficient computer programs

An Example

- Example– A survey of 150 college students reveals that 83 own automobiles, 97 own bikes, 28 own motorcycles, 53 own a car and a bike, 14 own a car and a motorcycles, 7 own a bike and a motorcycle, and 2 own all three.

Question–

- How many students own only a bike?
- How many students do not own any of the three?

Pigeonhole Principle

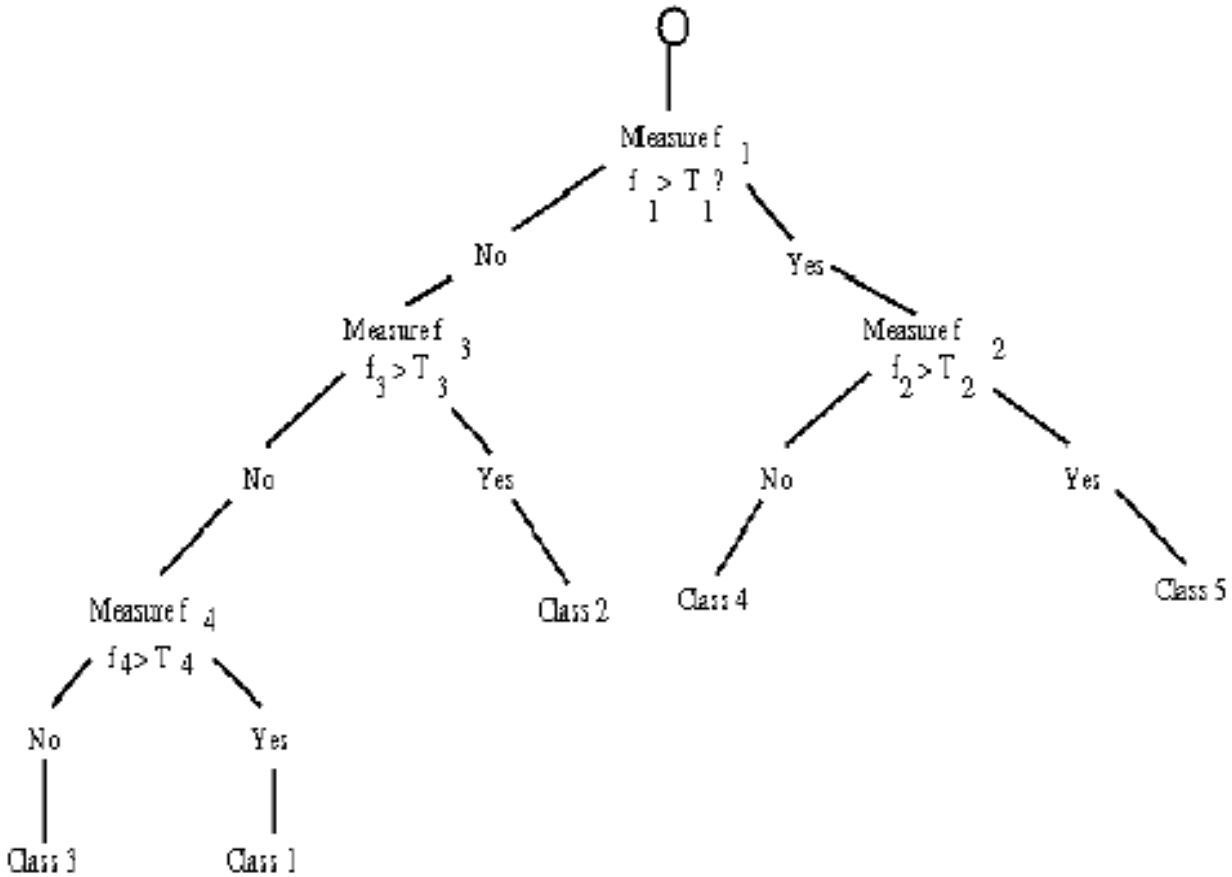
- If more than *k* items are placed into *k* bins, then at least one bin contains more than one item
- Example– How many times must a single die be rolled in order to guarantee getting the same value twice?

Classification Decision Trees

- ***Nonmetric:*** Data is list of binary attributes
  - Apple is *red* and *round*. Banana is *long* and *yellow*
  - Difficult to apply *nearest neighbor* and other techniques

- ***Decision Tree***: based on game of twenty questions
  - Apply a series of tests to the input pattern
  - Each test asks a question: e.g. “is the pattern yellow?”
  - The answer is “yes” or “no”
  - The answers give the classification -- e.g. the pattern is *yellow*, *not-round*, and *long* – so it is a banana.

A Decision Tree Example



CART: Design Decision Tree

- CART is a general framework for designing decision trees
- Basic Issues:

  (1). Should we restrict ourselves to binary questions?

  (2) Which attributes should be tested at each node?

  (3) When should a node be declared a leaf?

  (4) How can we prune a large tree?

  (5) How do we assign a class label to a leaf node?

Decision Tree Notation

- Set of classified data: $\{(x_a, \omega_a) : a \in \Lambda\}$.
- Set of tests: $\{T_j : j \in \Phi\}$.
  - Each test has response "T" or "F", e.g. $T_j(x_a) \in \{T, F\}$.
- Tree nodes: $\{\mu\}$. *Root node:* μ_0 at top of tree
- Each node either has two *child nodes*, or is a *leaf node*
- Each node $\{\mu\}$ has a test T_μ. Its child node μ_1 (on left) is for data x_a for which $T_\mu(x_a) = T$, μ_2 (right) is for $T_\mu(x_a) = F$.

Decision Tree Notation (Cont.)

- Define the data that gets to a node μ recursively

$$\Lambda_{\mu 1} = \{x_a \in \Lambda_\mu \ s.t. T_\mu(x_a) = T\}.$$
$$\Lambda_{\mu 2} = \{x_a \in \Lambda_\mu \ s.t. T_\mu(x_a) = F\}.$$

- The root node contains all data $\Lambda_{\mu_0} = \Lambda$

- Define Impurity (entropy)

$$I(\mu) = -\sum_j P_\mu(\omega_j) \log_2 P_\mu(\omega_j).$$
$$P_\mu(\omega_j) = \sum_{a \in \Lambda_\mu} \delta_{\omega_a, \omega_j} / |\Lambda_\mu|$$

Learning Decision Tree

- Iterative Design Principle: For all leaf nodes, calculate the ***maximal decrease in impurity*** (by searching over all tests). Expand the leaf node with maximal decrease and add its child nodes to the tree
- Decrease in impurity:

$$\Delta I(\mu) = I(\mu) - I(\mu_1, \mu_2).$$

$$I(\mu_1, \mu_2) = \frac{|\Lambda_{\mu 1}|}{|\Lambda_\mu} I(\mu_1) + \frac{|\Lambda_{\mu 2}|}{|\Lambda_\mu} I(\mu_2)$$

- The lower the impurity, the easier to classify

Learning Decision Tree (Cont.)

- Greedy Strategy:
  - Start at root node μ_0. select the test $T^*_{\mu_0}$ that has maximum $\Delta I(\mu_0)$,
  - For each child node, μ_a select the test that has maximal $\Delta I(\mu_a)$.
  - Repeat until each node is a pure leaf nod $I(\mu) = 0$
  - Classify leaf nodes by majority vote
- Note: learning algorithm is $O(|\Phi||\Lambda|\{\log|\Lambda|\}^2)$.
- Run time is $O(\log|\Lambda|)$. (Test Rules, and data points)

Learning Decision Tree (Cont.)

- Algorithm is greedy. There may be a more efficient tree (fewer tests) if you learned the tree by searching over the sequence of tests.

$$I(\mu) = \sum_{i,j,i \neq j} P_\mu(\omega_i) P_\mu(\omega_j)$$
$$= 1 - \sum_j P_\mu^2(\omega).$$

- The choice of number of children was arbitrary. There might be a more efficient tree (fewer tests) if the tests were multi-valued
- Alternative impurity measure. Gini index

When to Stop

- Generalization versus memorization: Key issue in learning
  - Don't want a decision tree that simply memorizes the training data. The tree should generalize, give good classification, to data that it has not been trained on
- The decision tree gives a rule α for classifying data. It has empirical risk:

$$(1/|\Lambda|)\sum_{a\in\Lambda} L(\omega_a, \alpha(x_a))$$

- But will it have small risk on the entire dataset?

When to Stop (Cont.)

- The decision tree learning algorithm can have if it proceeds until all nodes are pure. But $R(\alpha) = 0$ usually ***does not mean that:*** $R_{emp}(\alpha) = 0,$

where $R(\alpha) = \sum_j \sum x P(\omega_j, x) L(\omega_j, \alpha(x)).$

- *The closer you fit your model to the data, the more likely that you overfit*
- Often better to stop splitting the data when the impurity reaches $I(\mu) \leq \beta.$
- A positive threshold. i.e. set a *node to be a leaf* if $R_{emp}(\alpha) = 0$

When to Stop (Cont.)

- Alternatively, grow large tree and then remove sibling pairs (children of the same node) if this raises the impurity by an amount below threshold

- Both methods have the disadvantage of making the tree unbalanced (I.e. not all leaf nodes have the same distance to the root node)

Cross Validation

- General Principle for testing whether you are generalizing or memorizing
- Combine with ***validation*** or ***cross-validation***.
  - Validation – learn the decision tree on part of the dataset and evaluate performance on the other part
  - Cross-validation – split the dataset up into subsets. Learn on each subset and evaluate on the other subsets, e.g. learn decision trees with different impurity threshold beta
- Select the tree, and hence the beta, which has best validation

Summary

- Decision Trees: formulate the idea of "twenty questions"
- CART: procedure for learning decision trees
- Learning algorithm is greedy $O(|\Lambda||\Phi|\{\log|\Lambda|\}^2)$.
- Search is $O(\log|\Lambda|)$.
- Generalization or memorization: empirical risk and Bayes risk.
- Validation and cross-validation

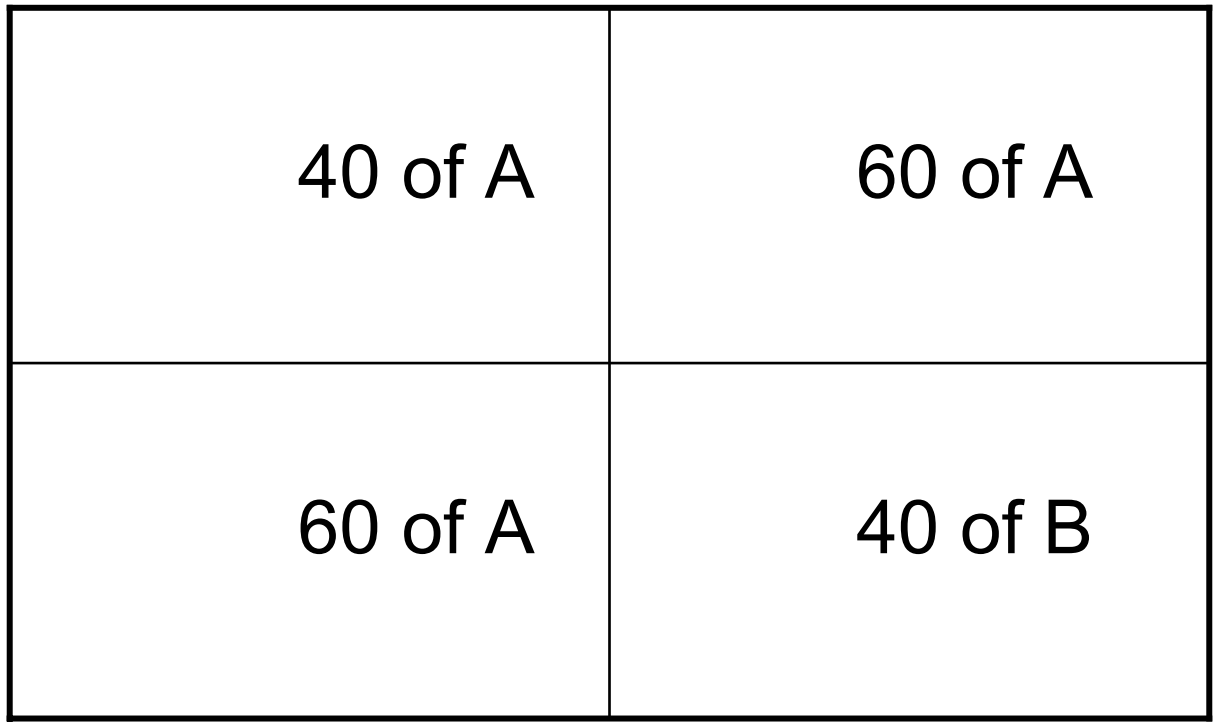
Impurity of a Node

- Need a measure of impurity of a node to help decide on how to split a node, or which node to split
- The measure should be at a maximum when a node is equally divided amongst all classes
- The impurity should be zero if the node is all one class

Measures of Impurity

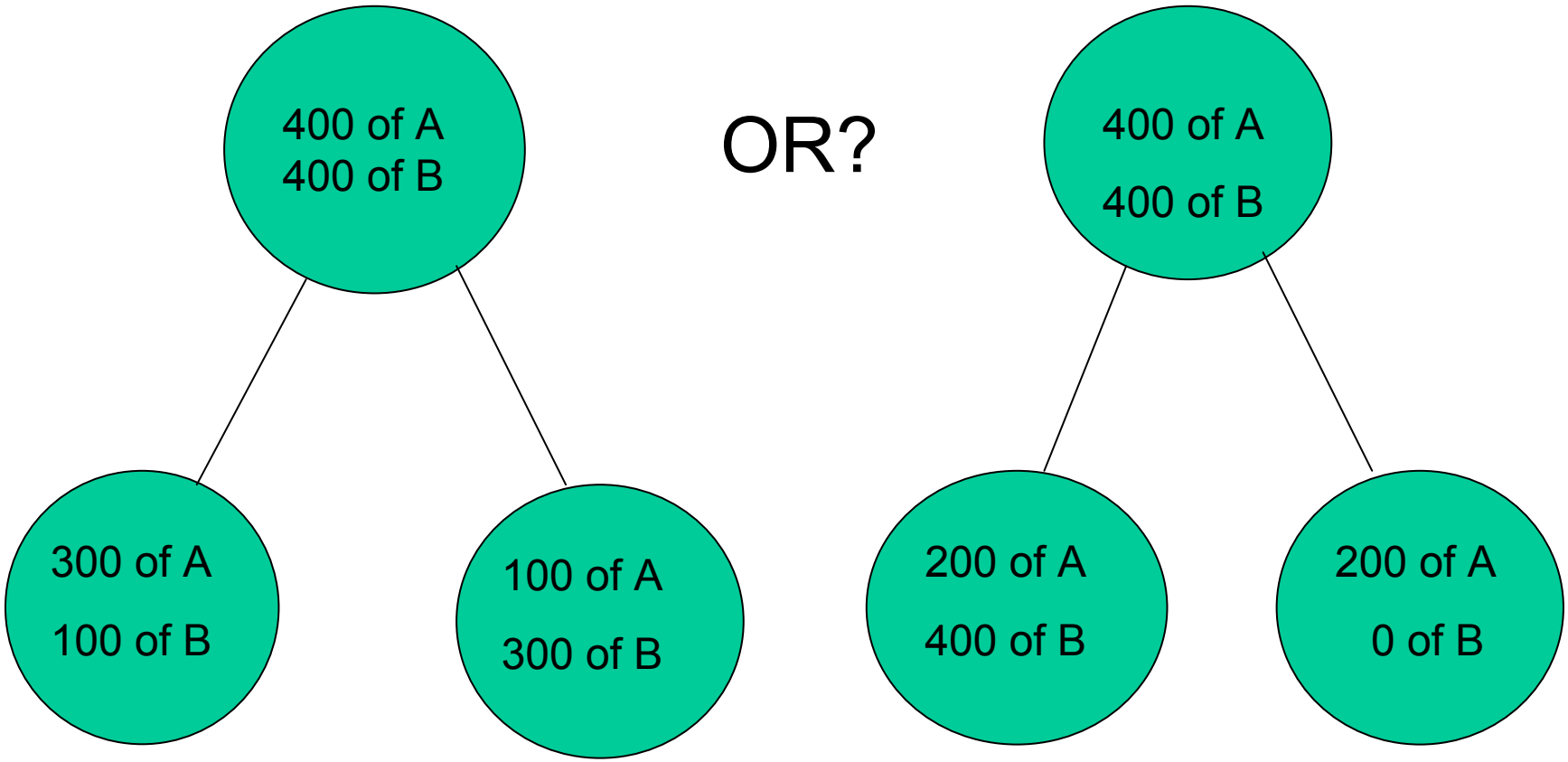
- Misclassification Rate,
- Information, or Entropy
- Gini Index
- In practice the first is not used for the following reasons:
- Situations can occur where no split improves the misclassification rate
- The misclassification rate can be equal when one option is clearly better for the next step

Problems with Misclassification Rate I

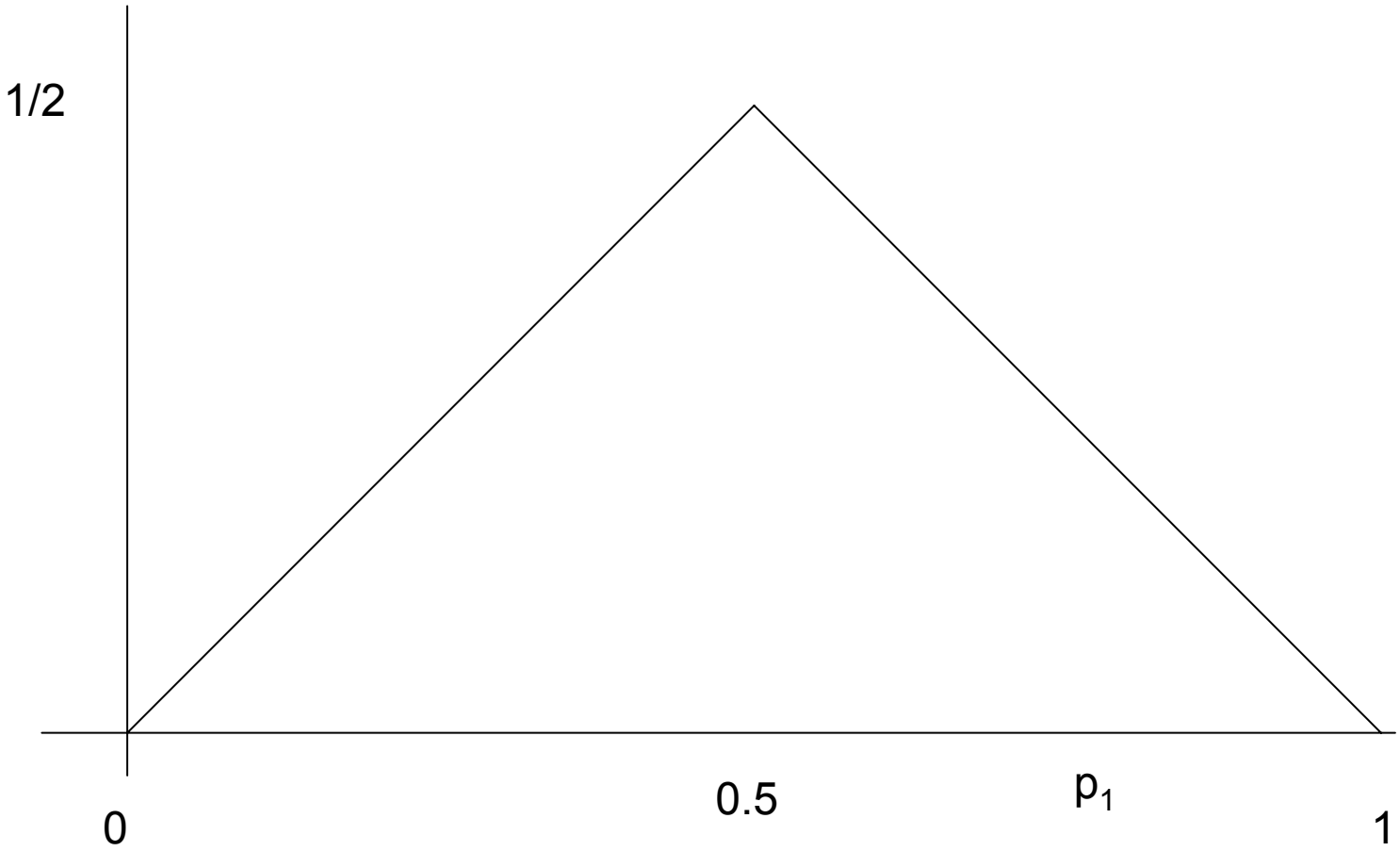


Neither improves misclassification, but together give perfect classification!

Problems with Misclassification Rate II



Misclassification Rate for Two Classes



Information

- If a node has a proportion of p_j of each of the classes then the information or entropy is:

$$i(p) = -\sum_j p_j \log p_j$$

where 0log0 = 0

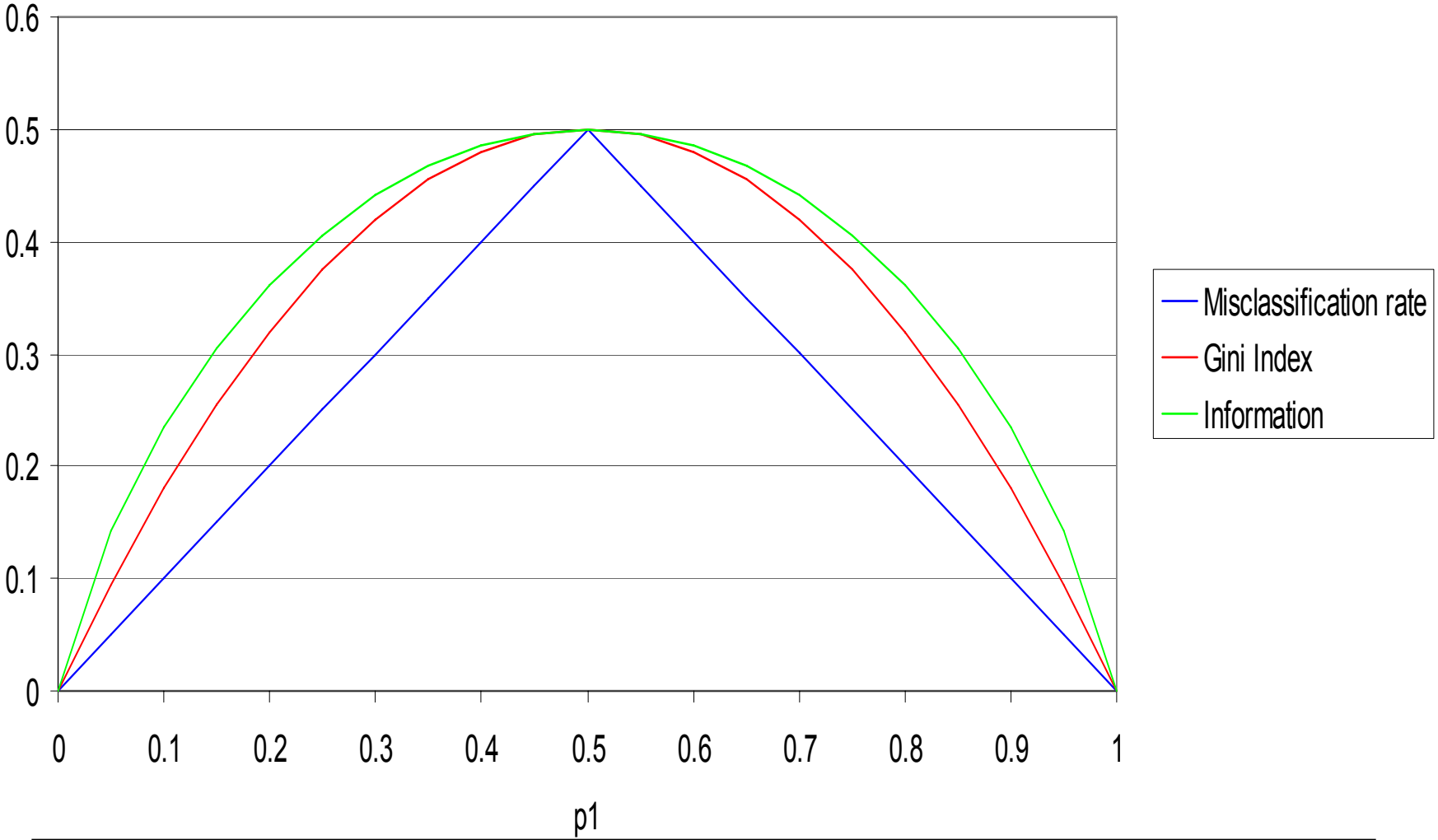
Note: $p=(p_1,p_2,\ldots. p_n)$

Gini Index

- This is the most widely used measure of impurity (at least by CART)
- Gini index is:

$$i(p) = \sum_{i \neq j} p_i p_j = 1 - \sum_j p_j^2$$

Scaled Impurity functions



Tree Impurity

- We define the *impurity of a tree* to be the sum over all terminal nodes of the impurity of a node multiplied by the proportion of cases that reach that node of the tree
- Example i) Impurity of a tree with one single node, with both A and B having 400 cases, using the Gini Index:
- Proportions of the two cases= 0.5
- Therefore Gini Index= $1-(0.5)^2- (0.5)^2 = 0.5$

Tree Impurity Calculations

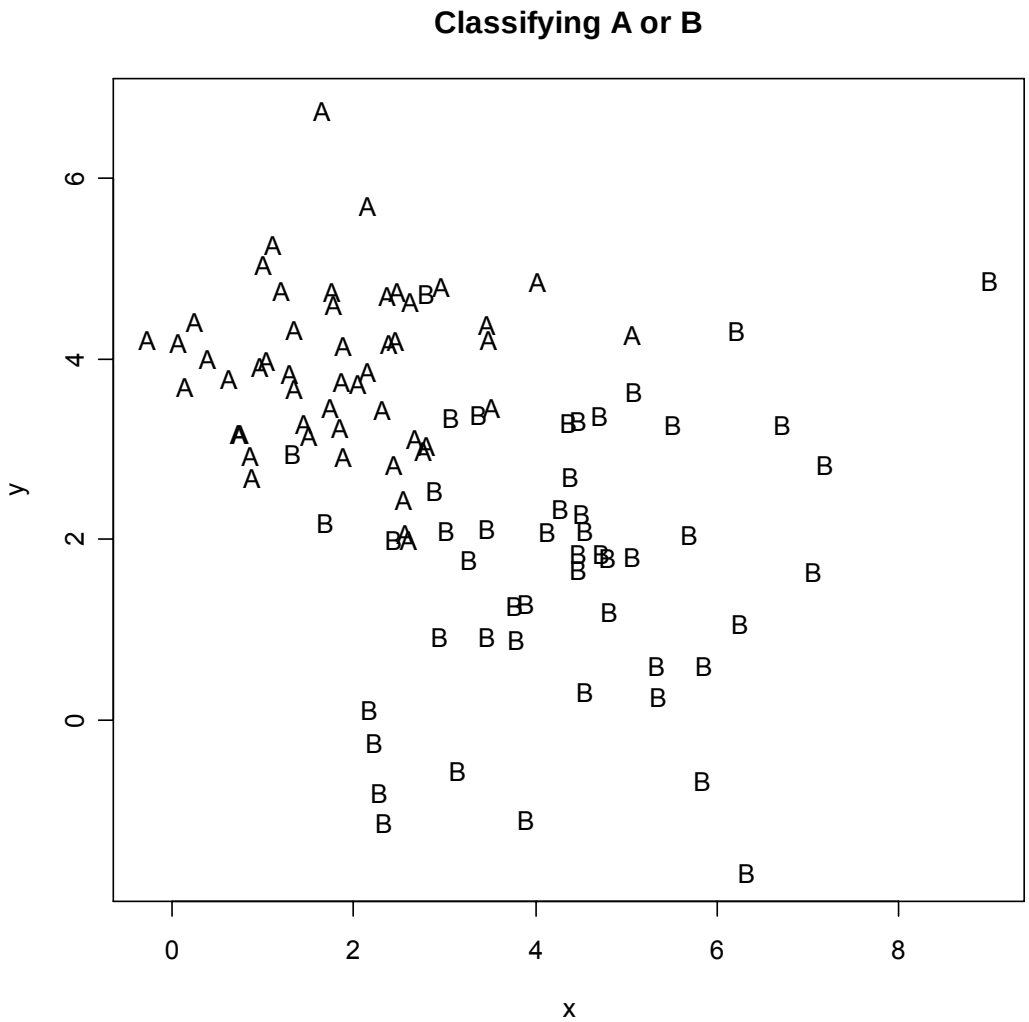
Numbers of Cases		Proportion of Cases				Gini Index
A	B	A	B			
		p_A	p_B	p^2_A	p^2_B	$1- p^2_A- p^2_B$
400	400	0.5	0.5	0.25	0.25	0.5

Number of Cases		Proportion of Cases				Gini Index	Contrib. To Tree
A	*B*	*A*	*B*				
		p_A	p_B	p^2_A	p^2_B	$1 - p^2_A - p^2_B$	
300	100	0.75	0.25	0.5625	0.0625	0.375	0.1875
100	300	0.25	0.75	0.0625	0.5625	0.375	0.1875
						Total	0.375
200	400	0.33	0.67	0.1111	0.4444	0.4444	0.3333
200	0	1	0	1	0	0	0
						Total	0.3333

Selection of Splits

- We select the split that most decreases the Gini Index. This is done over all possible places for a split and all possible variables to split.
- We keep splitting until the terminal nodes have very few cases or are all pure – this is an unsatisfactory answer to when to stop growing the tree, but it was realized that the best approach is to grow a larger tree than required and then to *prune* it!

Example



Possible Splits

- There are two possible variables to split on and each of those can split for a range of values of c i.e.:

$$x<c \text{ or } x\geq c$$

And:

$$y<c \text{ or } y\geq c$$

Example: Possible Splits

			Split=	2.81		
x	y	Class	A	B	A	B
2.61	2.02	A	1	0	0	0
2.57	2.10	A	1	0	0	0
2.85	2.46	B	0	0	0	1
2.45	2.85	A	1	0	0	0
2.76	3.00	A	1	0	0	0
2.82	3.07	A	0	0	1	0
2.68	3.13	B	0	1	0	0

Etc.

Example: Possible Splits (Cont.)

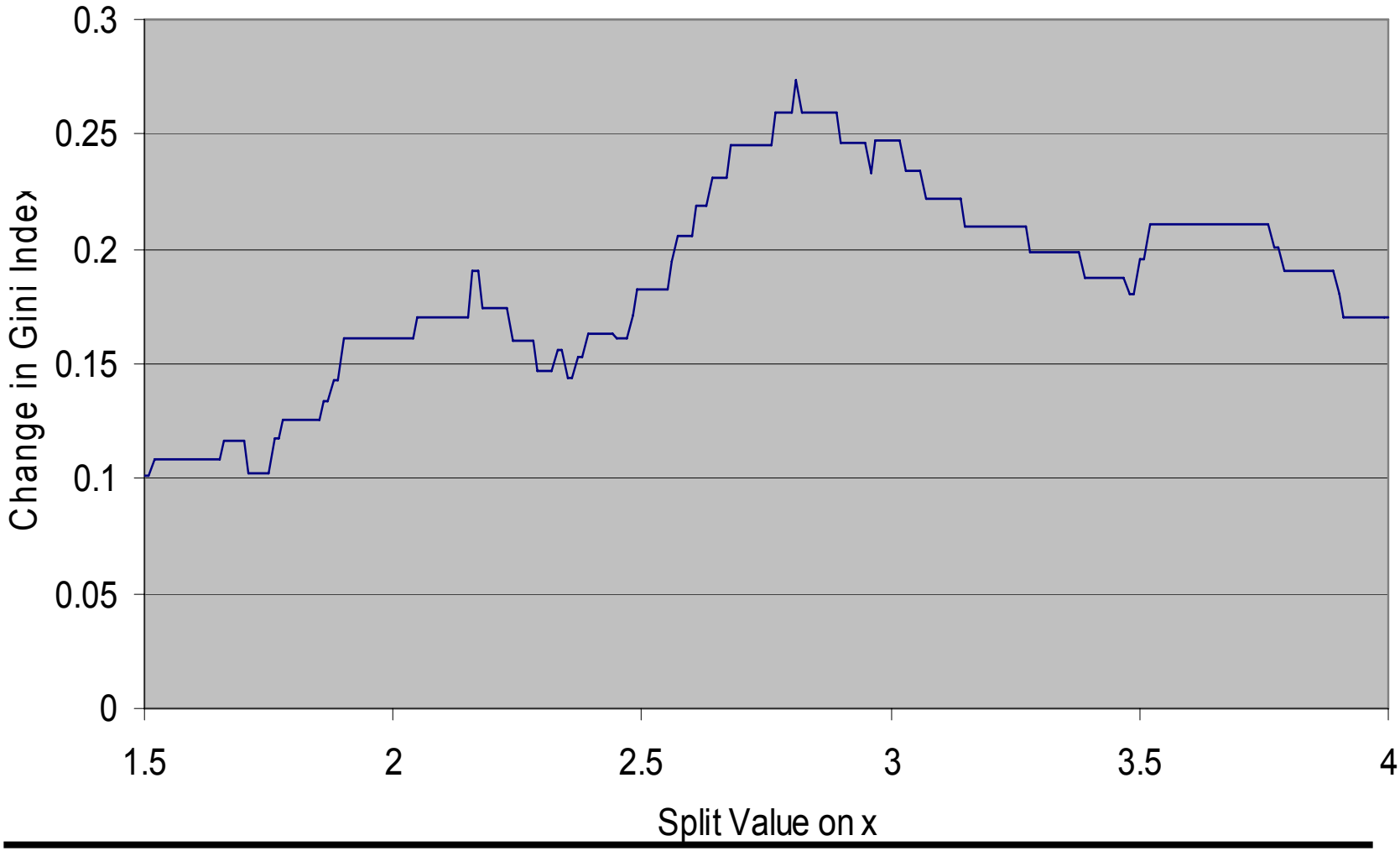
Top	50	50	100	0.5	0.5	0.25	0.25	0.5	
Split Left	44	7	51	0.86	0.14	0.74	0.02	0.24	0.12
Split Right	6	43	49	0.12	0.88	0.01	0.77	0.21	0.11
								Sum=	0.23
								Change in Gini Index	0.27

Example: Possible Splits (Cont.)

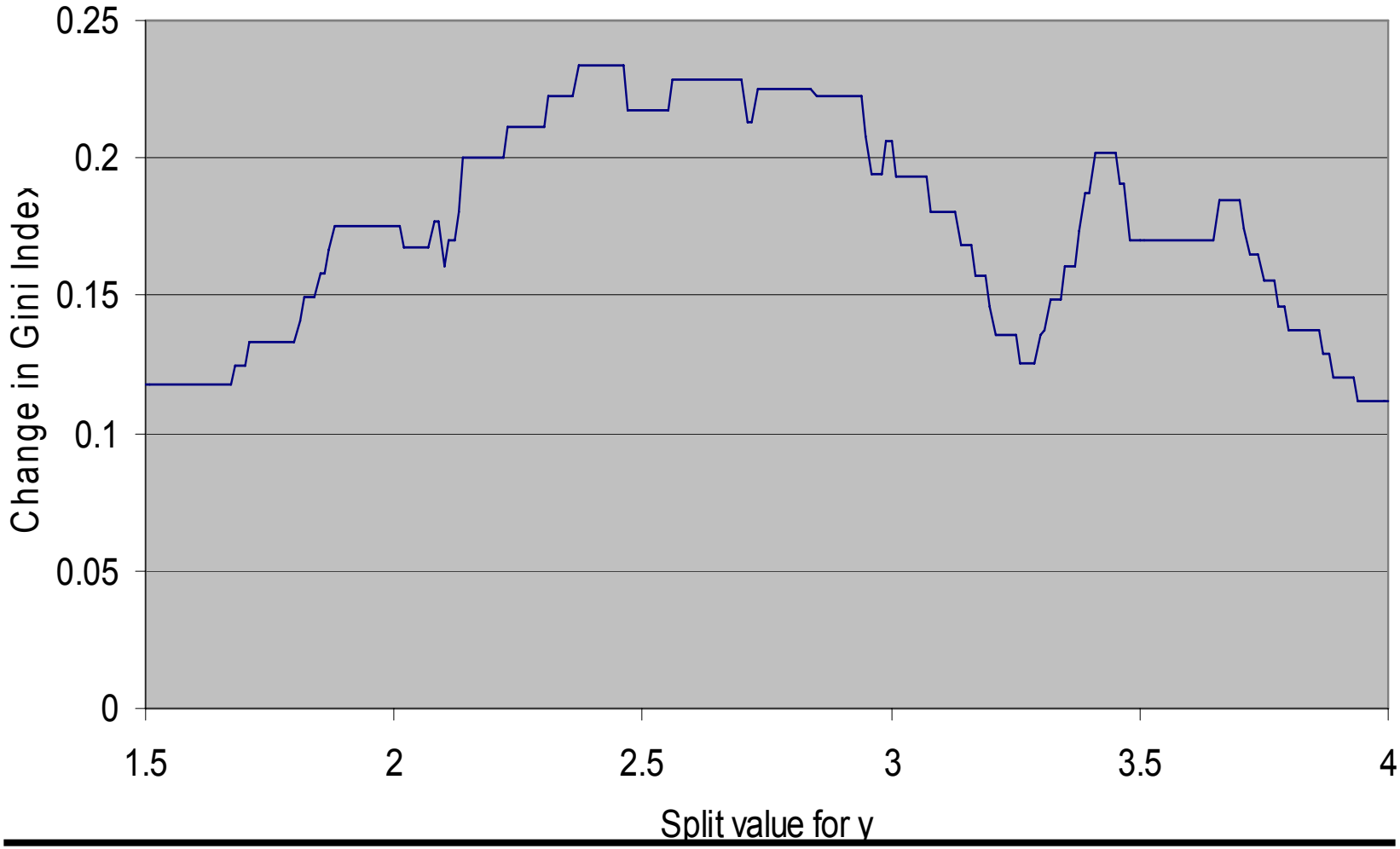
Then use Data table to find the best value for a split:

Split	Change
	0.27
1.5	0.10
1.6	0.11
1.7	0.12
1.8	0.13
1.9	0.16
2	0.16
2.1	0.17
2.2	0.17
2.3	0.15

Improvement in Gini Index

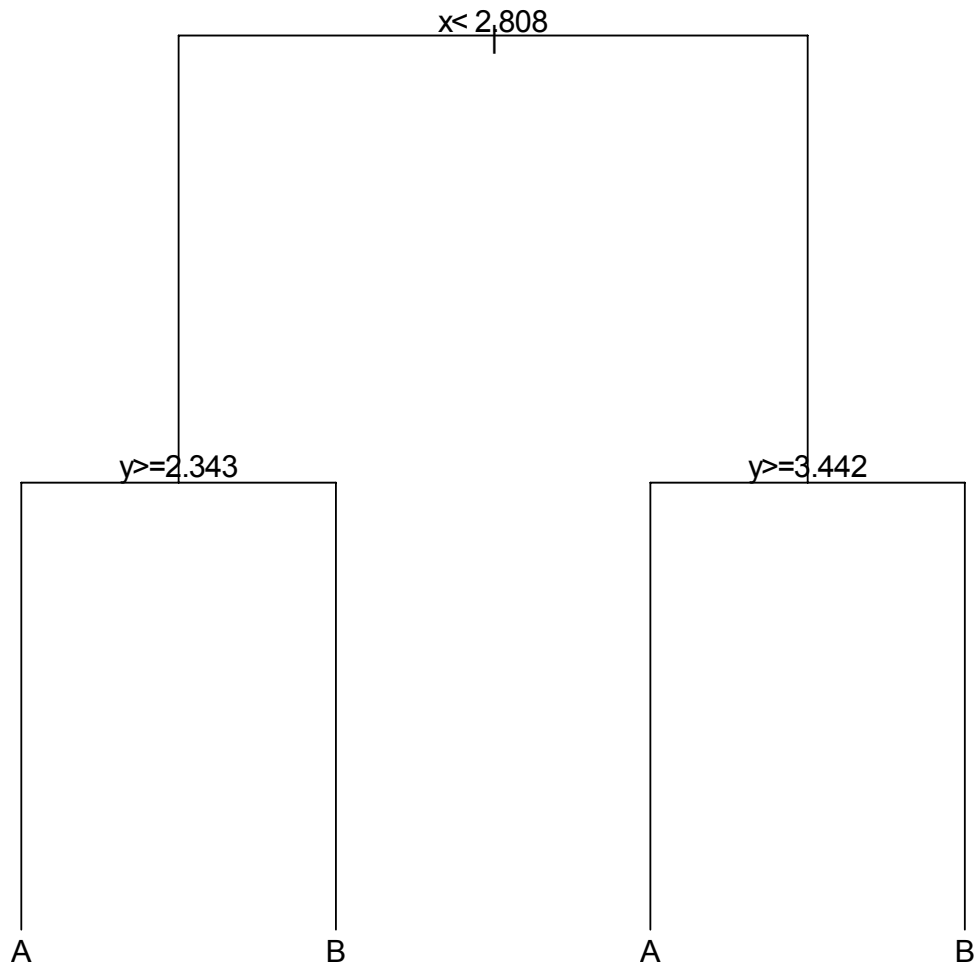


Improvement in Gini Index



The Next Step

You'd now need to develop a series of spreadsheets to work out the next best split



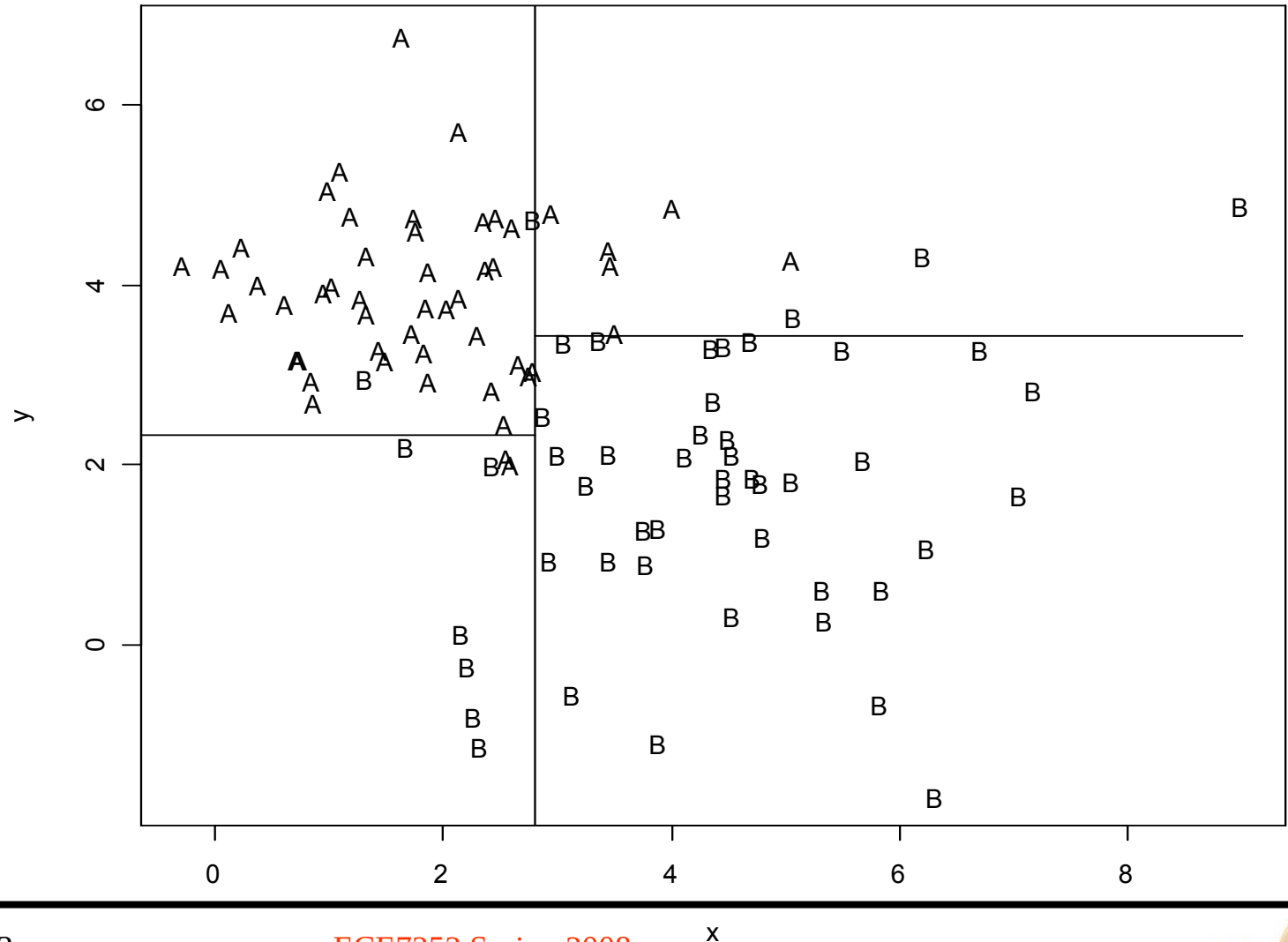
Developing Trees Using R

- Need to load the package “rpart” which contains the set of functions for CART
- The function looks like:

```
NNB.tree<-rpart(Type~., NNB[ , 1:2], cp = 1e-3)
```

- This takes the data in Type (which contains the classes for the data, i.e. A or B), and builds a model on all the variables indicated by “~.” . The data is in NNB[, 1,2] and cp is complexity parameter (more to come about this).

Plot showing how Tree works



Regression Trees

- Trees can be used to model functions though each end point will result in the same predicted value, a constant for that end point. Thus regression trees are like classification trees except that the end pint will be a predicted function value rather than a predicted classification

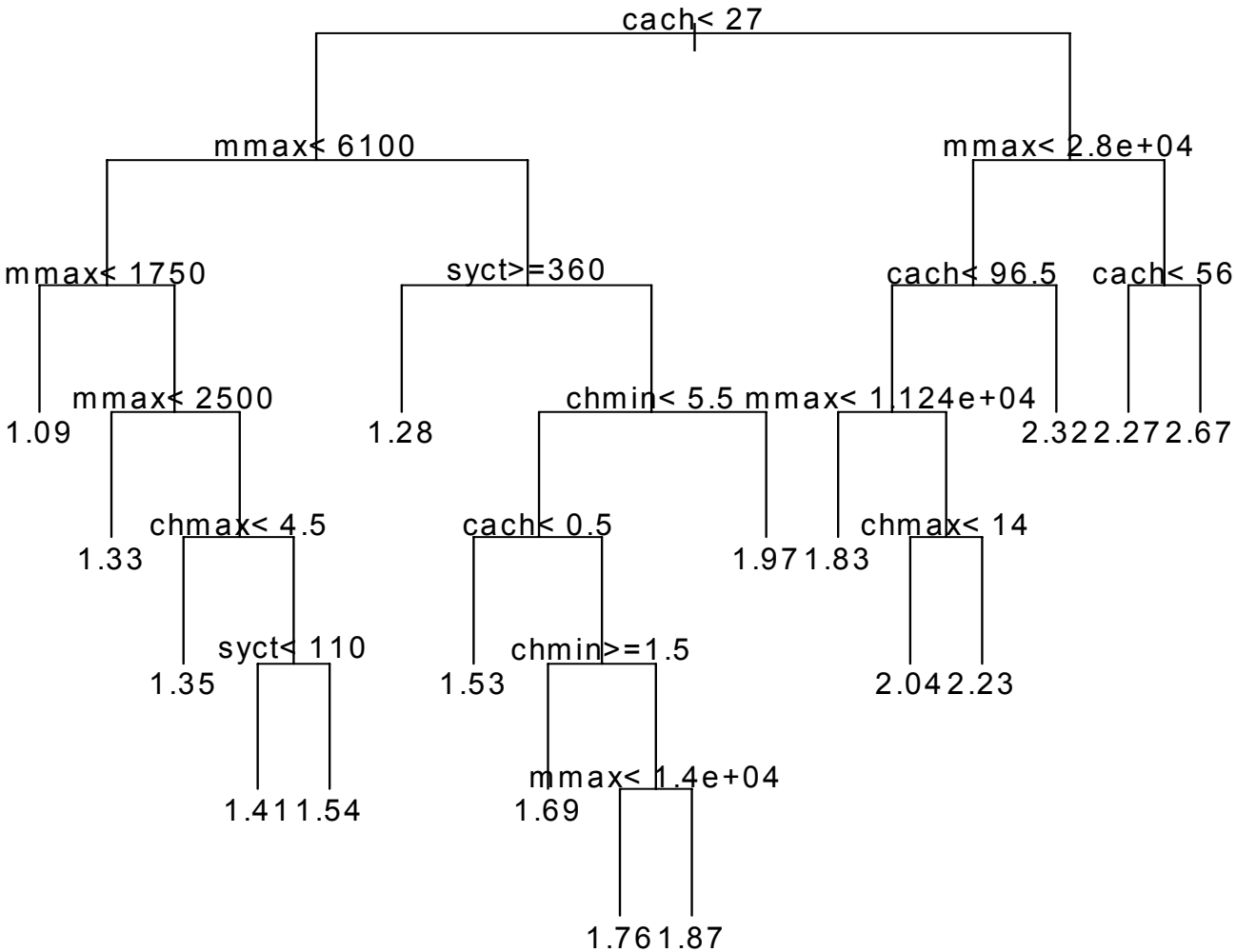
Measures Used in Fitting Regression Tree

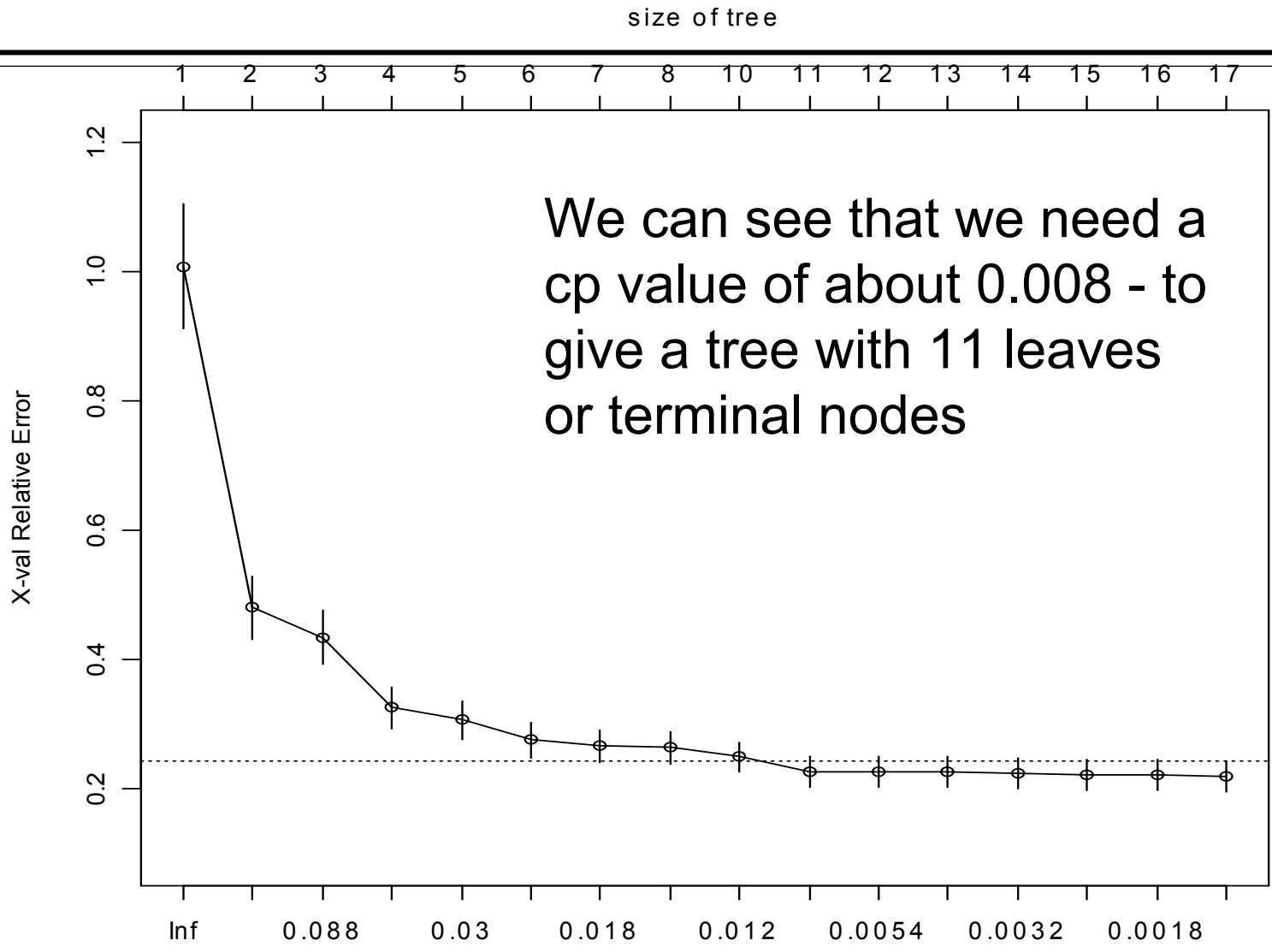
- Instead of using the Gini Index the impurity criterion is the sum of squares, so splits which cause the biggest reduction in the sum of squares will be selected
- In pruning the tree the measure used is the mean square error on the predictions made by the tree

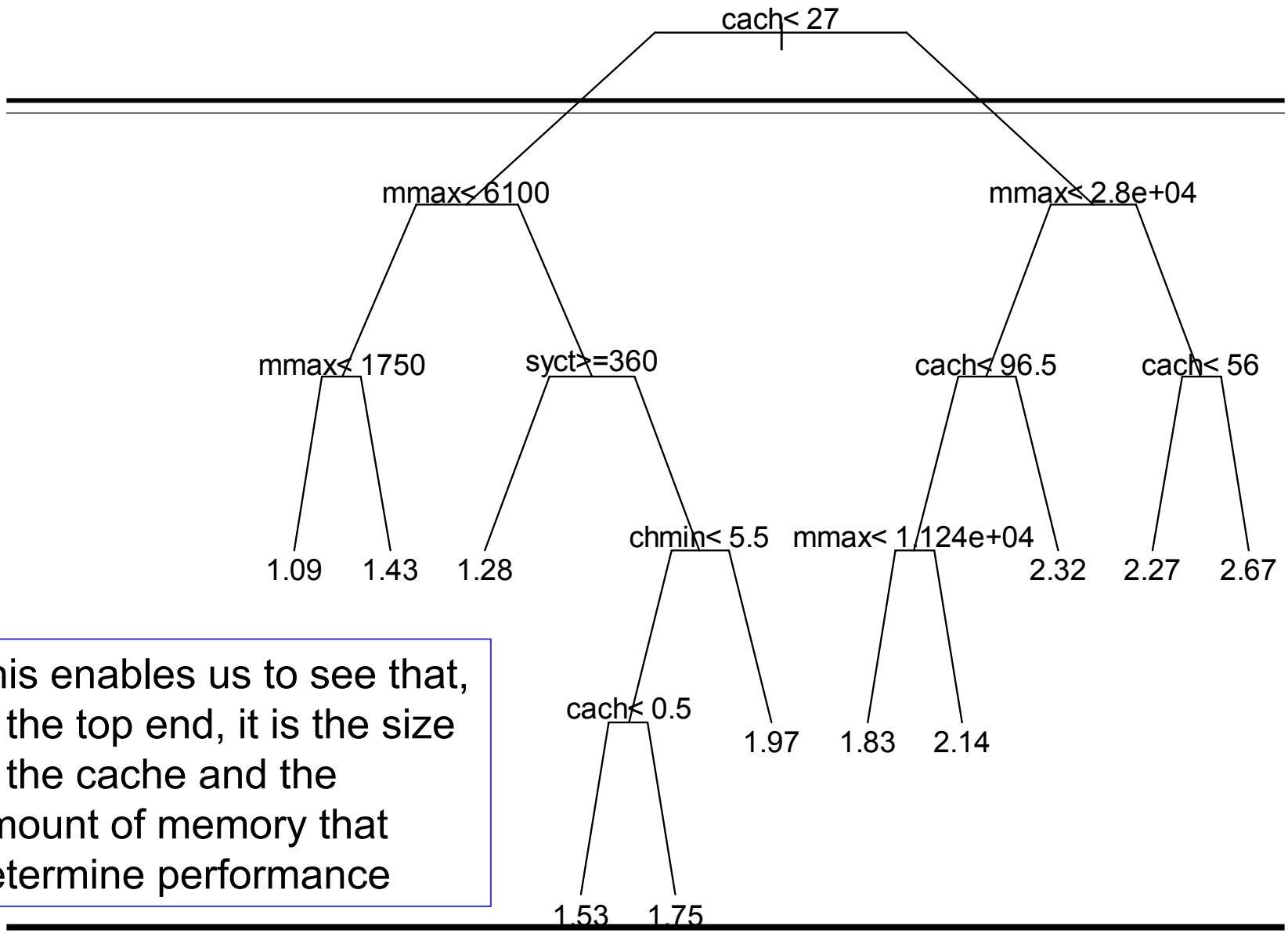
Regression Example

- In an effort to understand how computer performance is related to a number of variables which describe the features of a PC the following data was collected: the size of the cache, the cycle time of the computer, the memory size and the number of channels (both the last two were not measured but minimum and maximum values obtained).

The Tree Generated




size of tree
1 2 3 4 5 6 7 8 10 11 12 13 14 15 16 17
X-val Relative Error
0.2 0.4 0.6 0.8 1.0 1.2
Inf 0.088 0.03 0.018 0.012 0.0054 0.0032 0.0018
cp
We can see that we need a cp value of about 0.008 - to give a tree with 11 leaves or terminal nodes



This enables us to see that, at the top end, it is the size of the cache and the amount of memory that determine performance

Advantages of CART

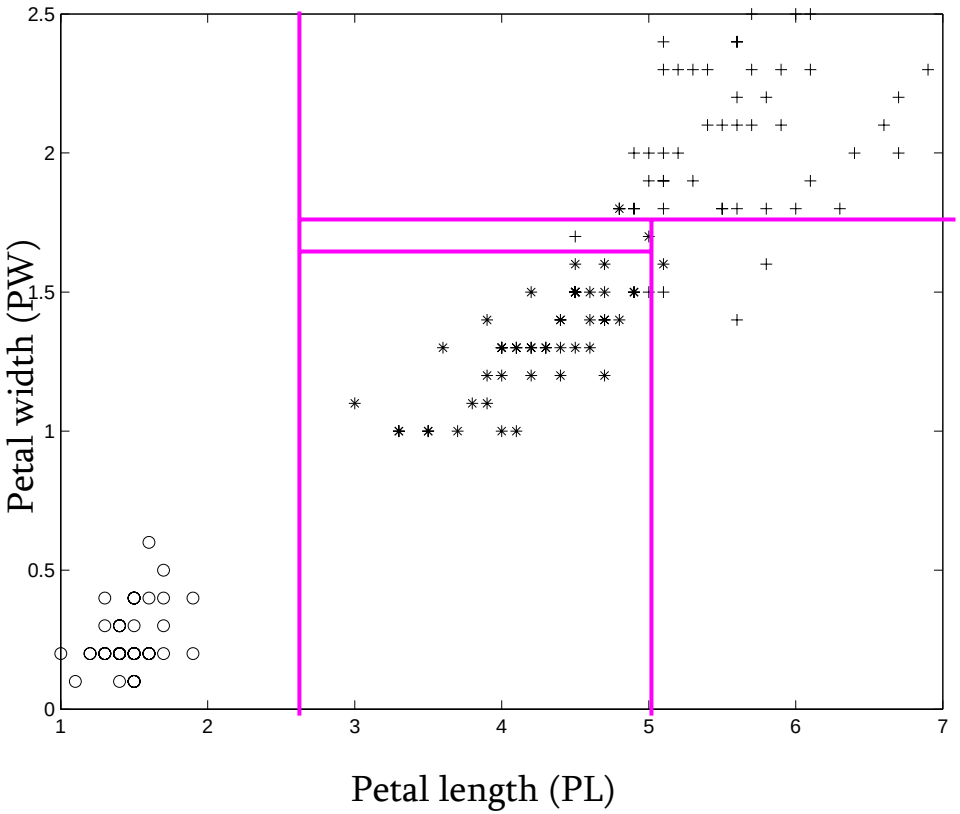
- Can cope with any data structure or type
- Classification has a simple form
- Uses conditional information effectively
- Invariant under transformations of the variables
- Is robust with respect to outliers
- Gives an estimate of the misclassification rate

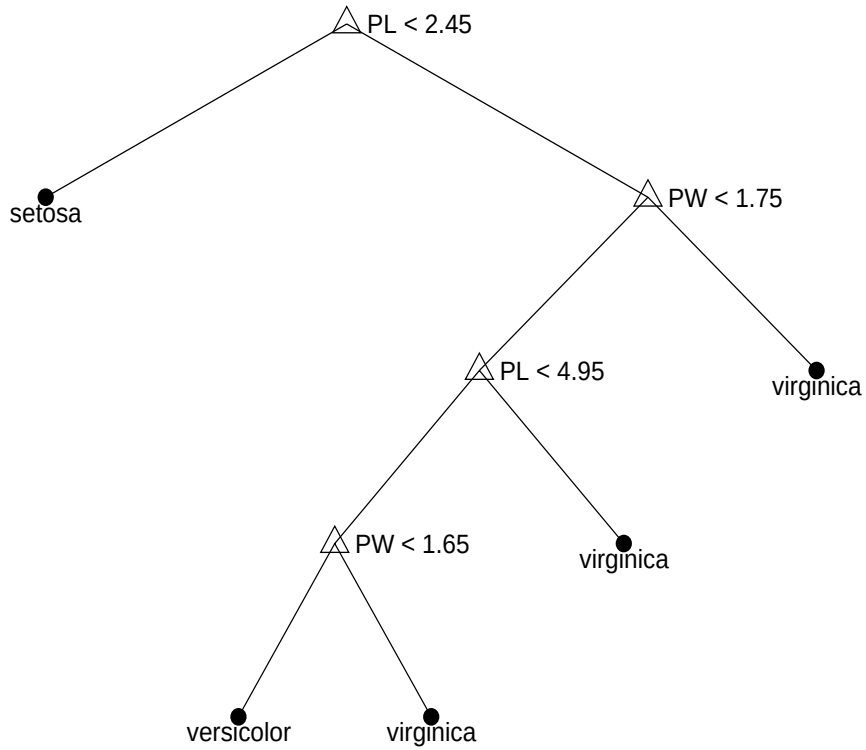
Disadvantages of CART

- CART does not use combinations of variables
- Tree can be deceptive – if variable not included it could be as it was “masked” by another
- Tree structures may be unstable – a change in the sample may give different trees
- Tree is optimal at each split – it may not be globally optimal.

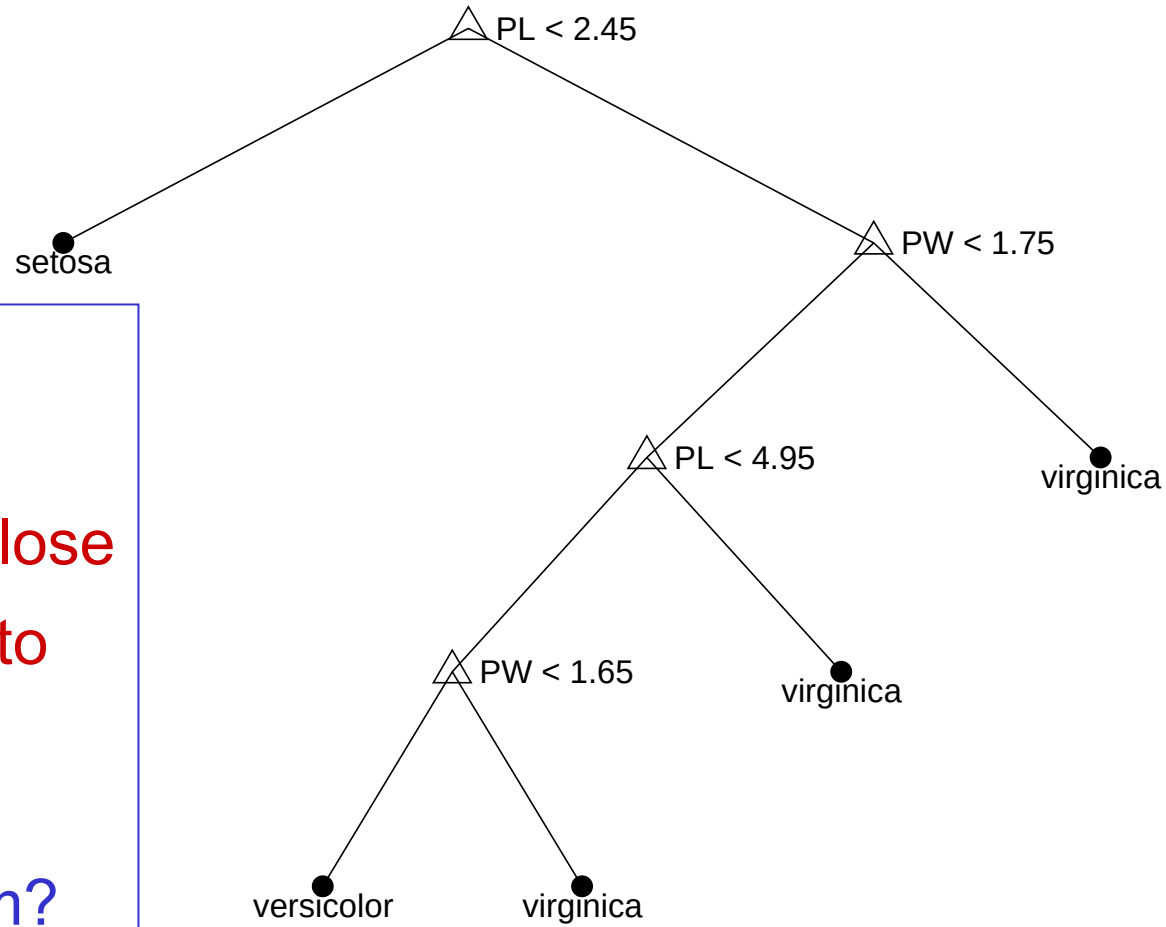
An Example

Iris data: 3 classes (setosa, viriginica, versicolor)





Decision Trees



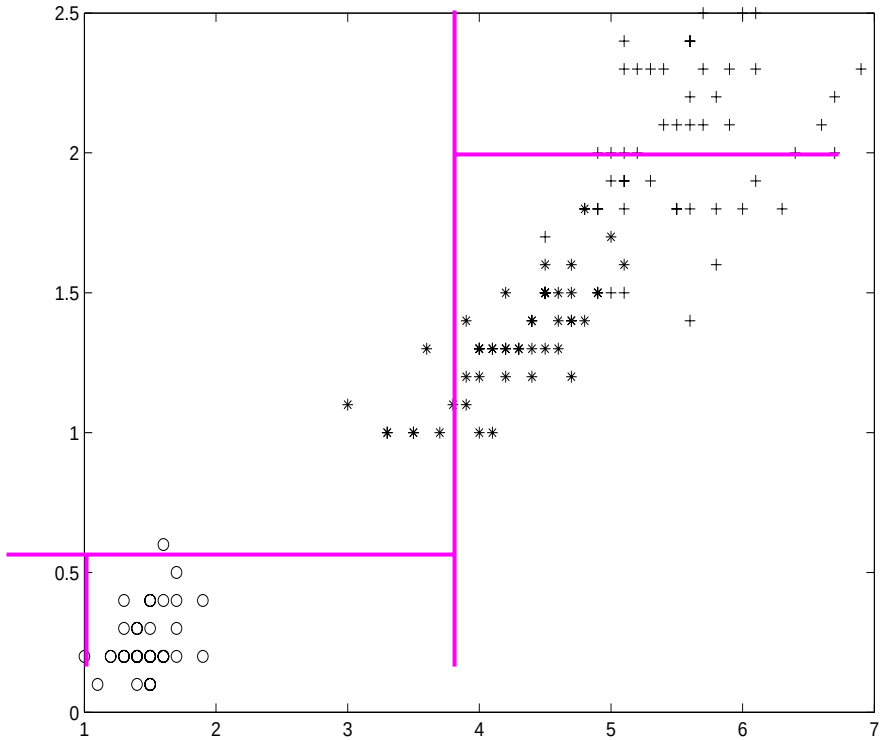
- Simple
- Multiclass
- Not perfect, but close
- Comprehensible to humans

- How to build them?

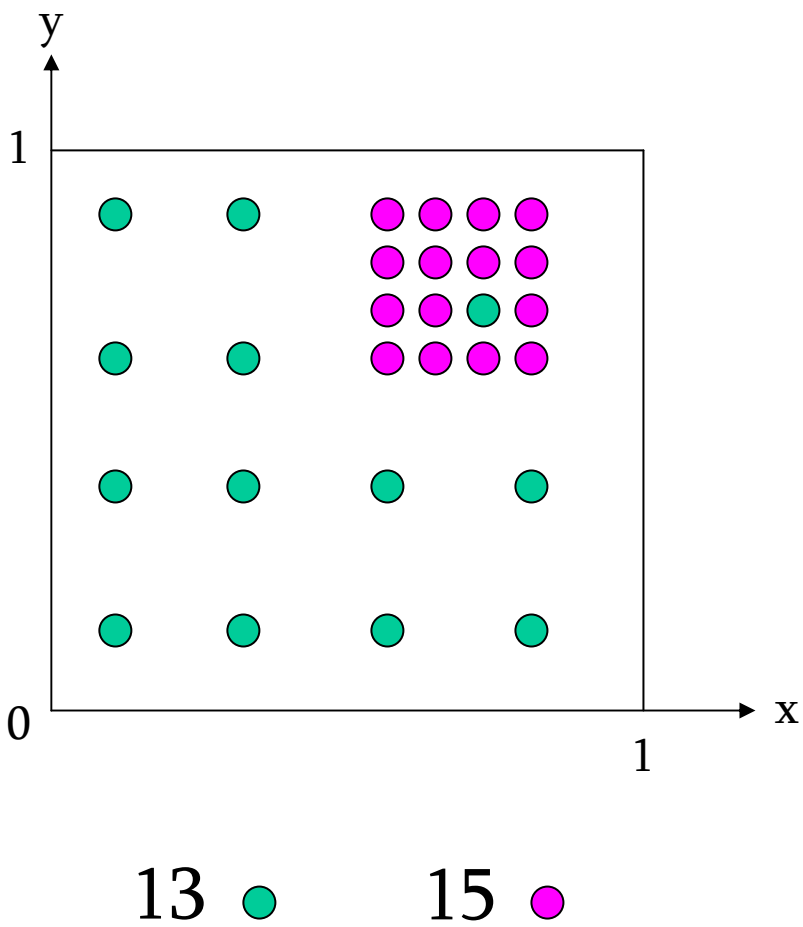
K-d Trees

- Rapidly partition the space into rectangular regions which contain few points

- But we don't care how many points regions contain: we just want them to consist (almost) exclusively of points from one class…



Example: Building a Decision Tree

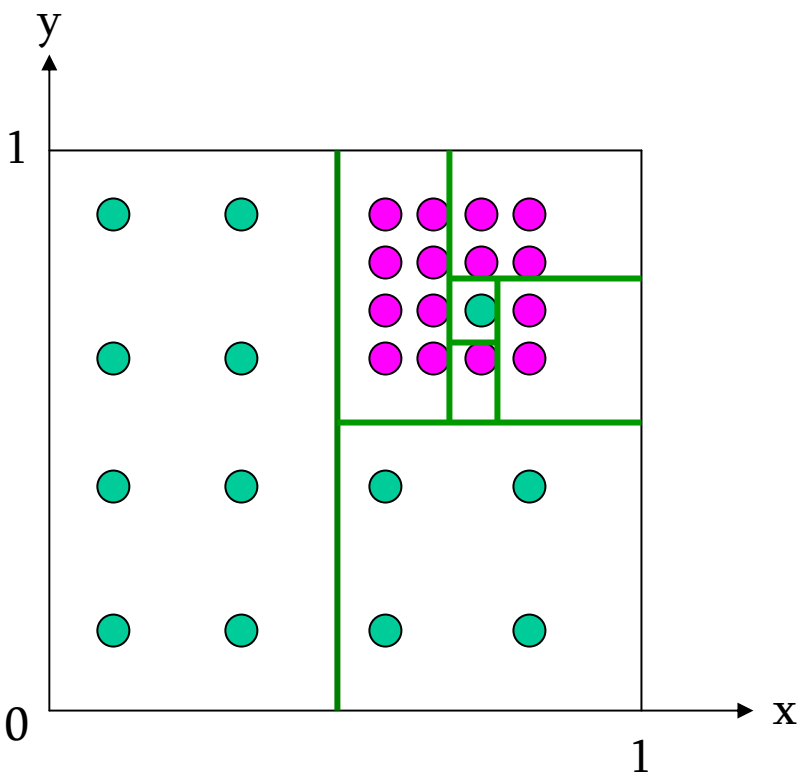


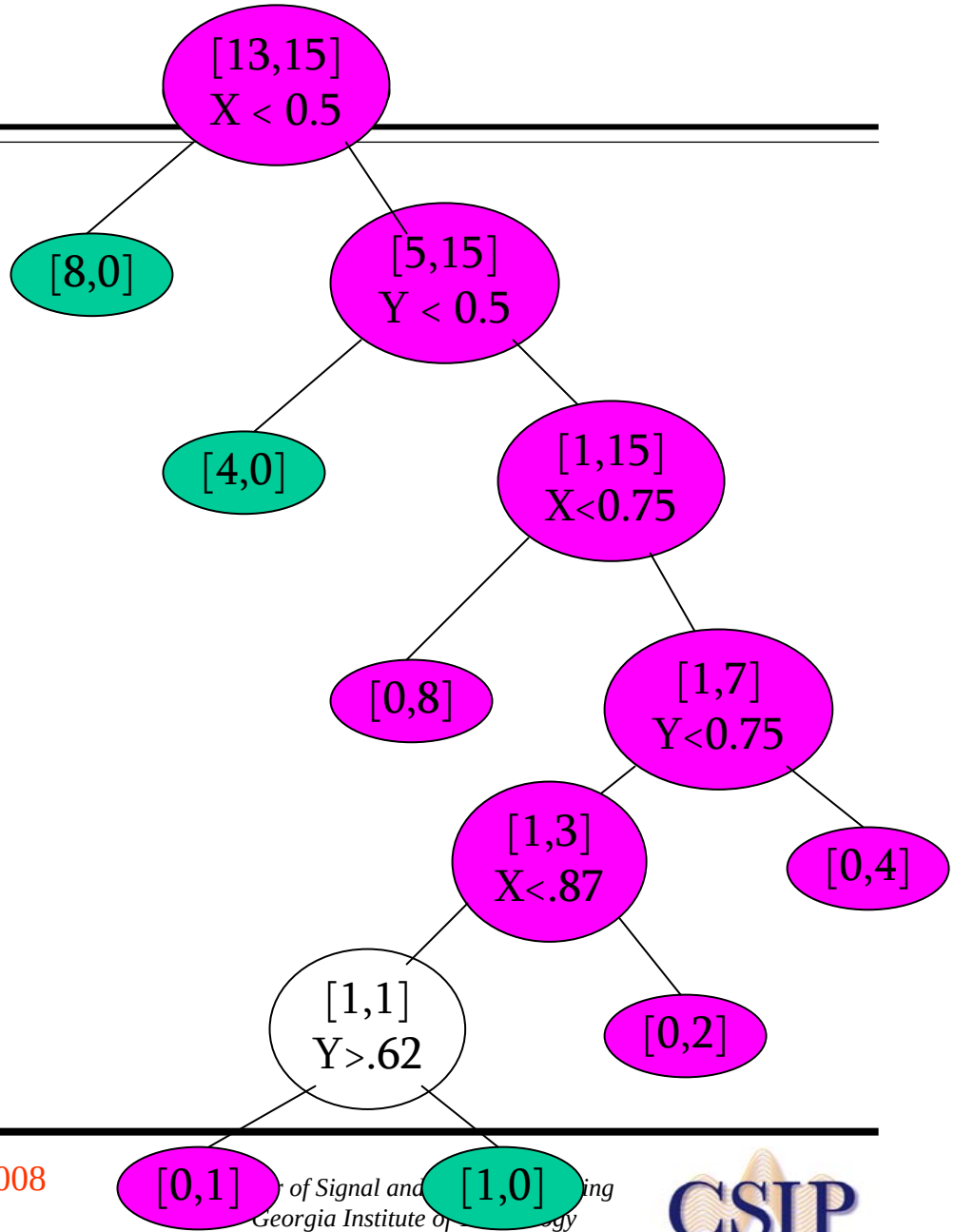
- Root of tree contains

  13 ● 15 ●

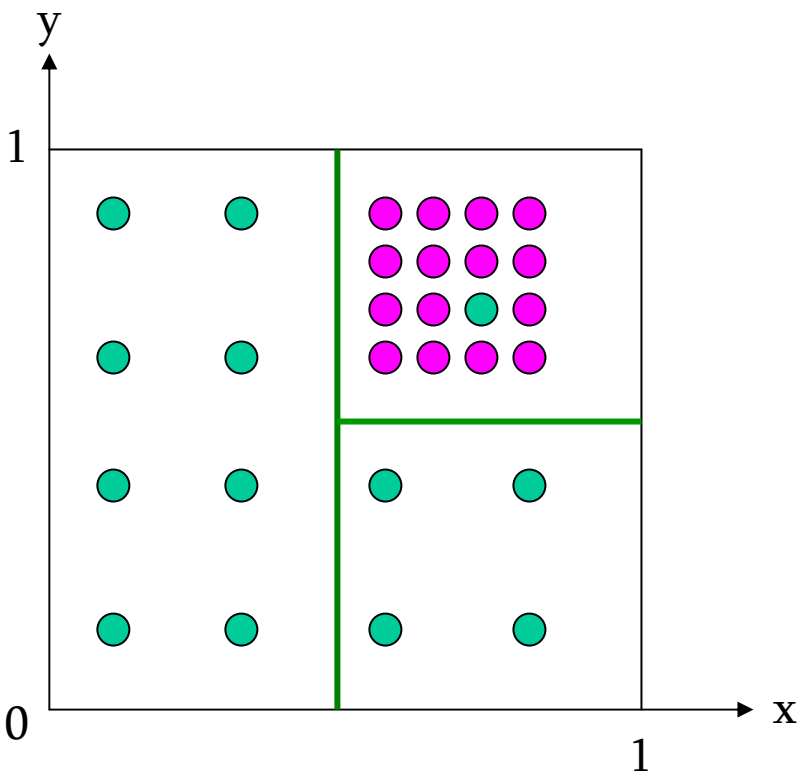
- If we had to guess, we'd pick ●. But there's too much uncertainty
- We want to improve our odds, so split

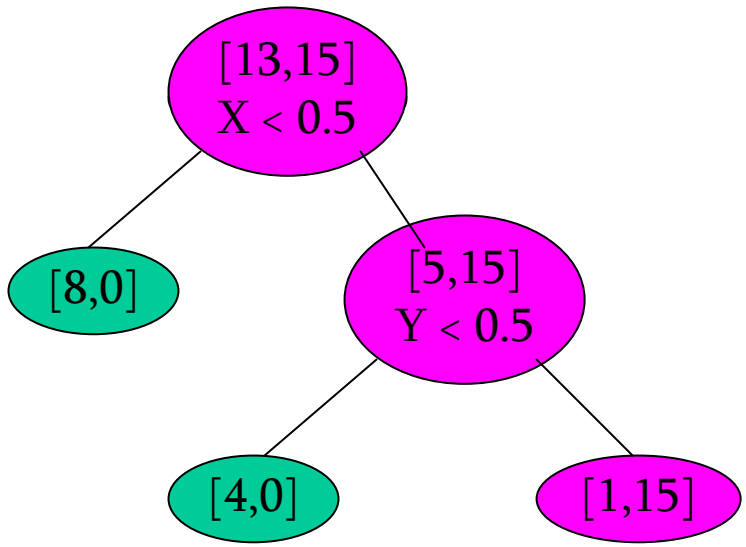
Split Results



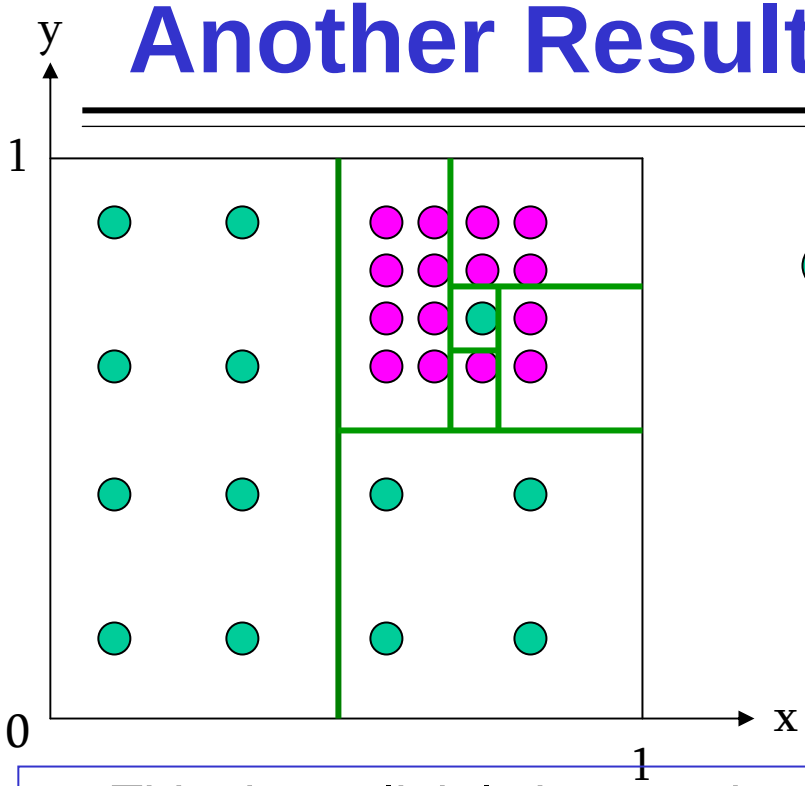


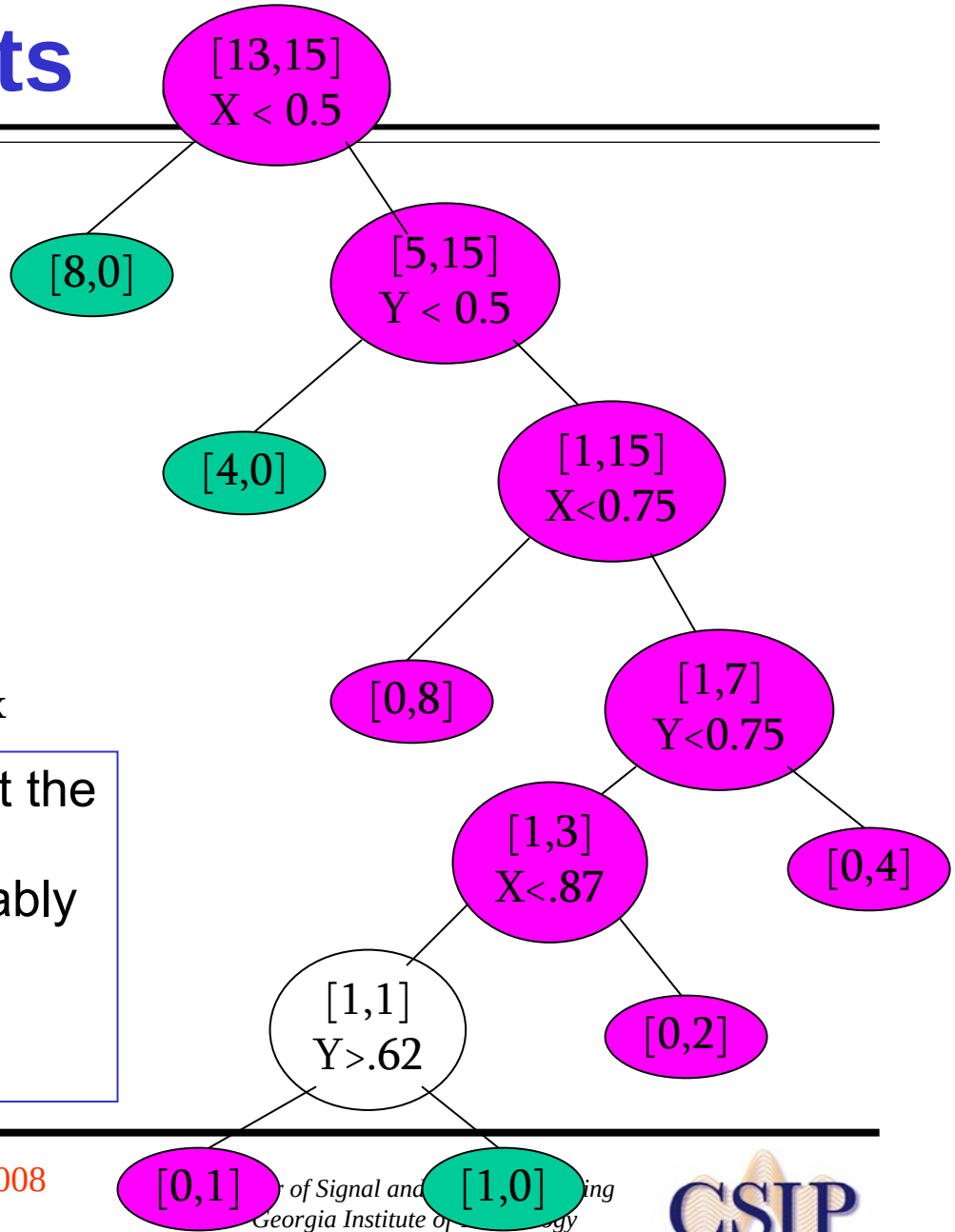
Retracing a Few Steps…





Another Results





- This does slightly better – but the tree is much more complex
- And that one point was probably an outlier anyway
- We have probably ended up overfitting the data

Decision Tree Issues

- Very expressive family of classifiers:
  - Any type of data can be accommodated: real, Boolean, categorical, …
  - Can perfectly fit any self-consistent training set

- But this also means that there is serious danger of overfitting.

Building a Decision Tree

- Greedy algorithm: build tree top-down
- At each stage:
  - Look at all current leaves and all possible splits
  - Choose the split that most decreases the uncertainty

- We need a measure of uncertainty…

Uncertainty

e.g.: + p fraction of the points

- 1-p fraction

How uncertain is this?

(i) Entropy

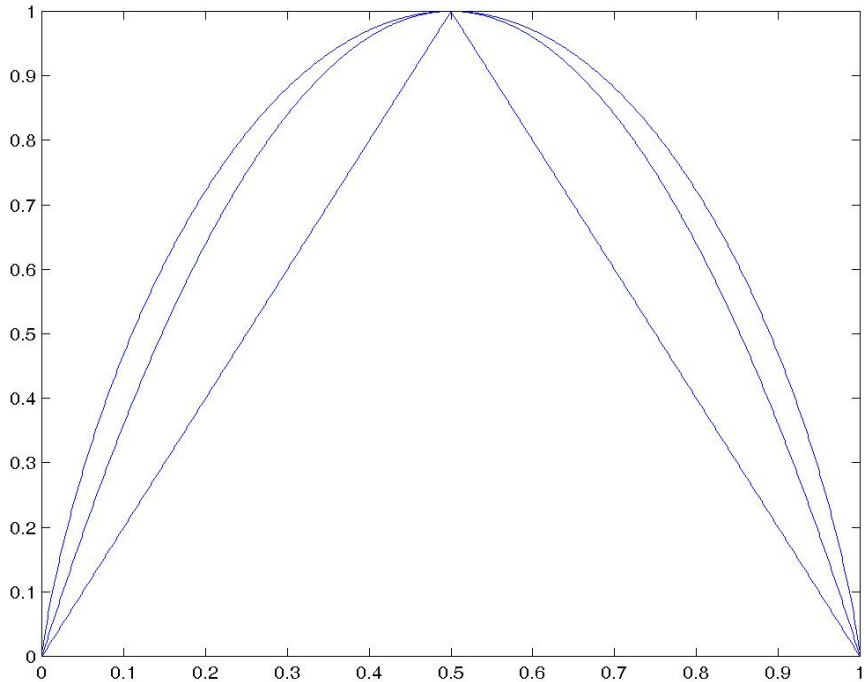
$$p \log \frac{1}{p} + (1-p) \log \frac{1}{1-p}$$

(ii) Gini index

$$2p(1-p)$$

(iii) Simplest of all

$$\min\{p, 1-p\}$$

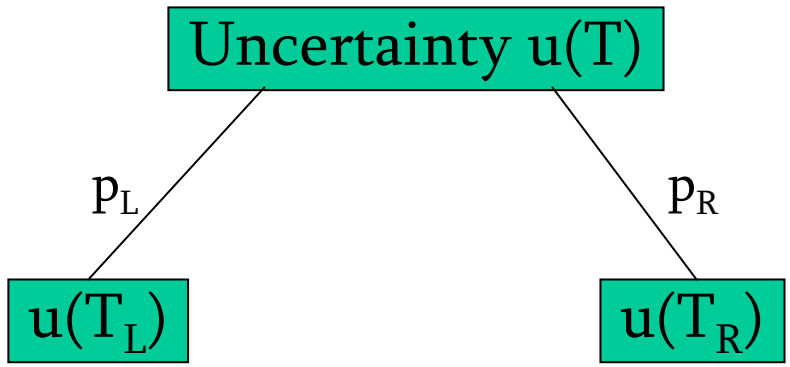


Uncertainty (Cont.)

- Generalize to k classes: $p_1, p_2, \ldots, p_k$ fraction of points

	$k = 2$	General k
Entropy	$p \log \frac{1}{p} + (1-p) \log \frac{1}{1-p}$	$\sum_{i=1}^{k} p_i \log \frac{1}{p_i}$
Gini index	$2p(1-p)$	$\sum_{i,j} p_i p_j = 1 - \|p\|^2$
Simplest	$\min\{p, 1-p\}$	$1 - \max_i p_i = 1 - \|p\|_\infty$

The Benefit of a Split



Of the points in T:

p_L fraction go to T_L

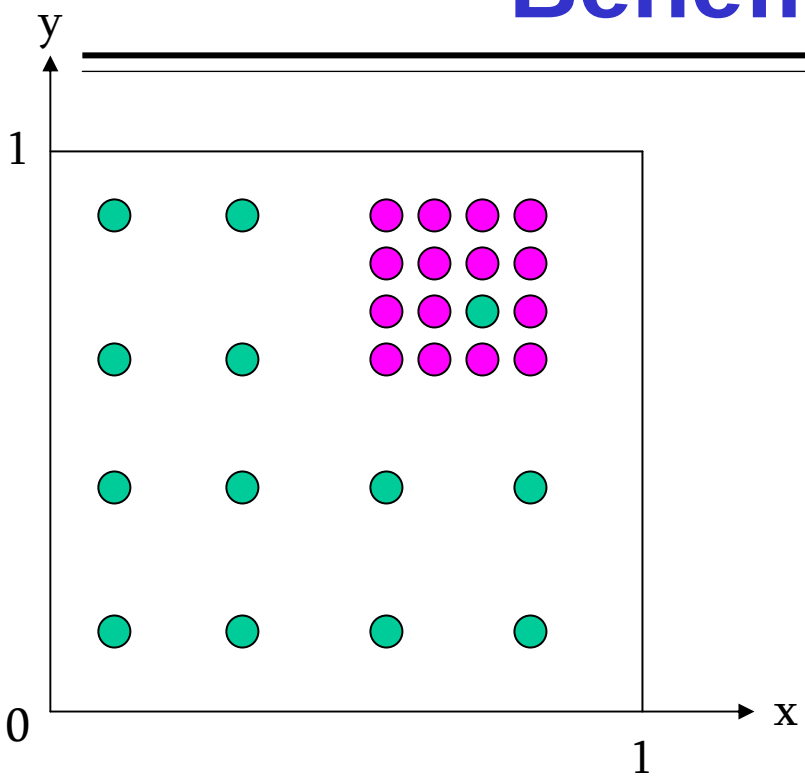
p_R fraction go to T_R

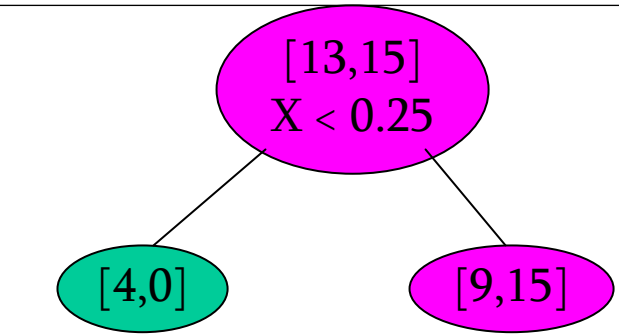
$$\text{Benefit of split} = (u(T) - \{p_L\, u(T_L) + p_R\, u(T_R)\}) \notin |T|$$

Expected uncertainty after split

points in T

Benefit: Example





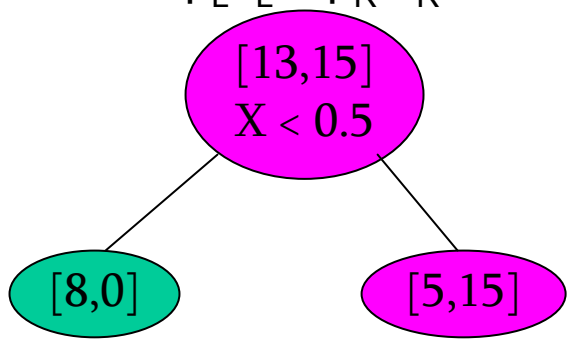
$p_L = 4/28$

$u_L = 0$

$p_R = 24/28$

$u_R = 2 \cdot 9/24 \cdot 15/24$

Gain = $u - p_L u_L - p_R u_R = u - 22.5/56$



Gain = $u - p_L u_L - p_R u_R = u - 15/56$

Initial uncertainty (Gini):

[13,15] $u = 2 \cdot \frac{13}{28} \cdot \frac{15}{28}$

Greedy Decision Tree Building

- Start with all points in a single node

Repeat

Pick the split with greatest benefit

Until ???

When to stop?

(i) When each leaf is pure?

(ii) When the tree is already pretty big?

(iii) When each leaf has uncertainty < some threshold?

- Common strategy: keep going until leaves are pure (recall: this didn't work too well for us earlier…)
- Then, shorten the tree by pruning, to correct the overfitting problem.

What is Overfitting?

- Data comes from some true underlying distribution D on X and Y. [X = input space, Y = label space]
- All we ever see are samples from D: training/test set, etc.

- When we choose a classifier h: X ! Y, we can talk about its error on the training set $(x_1, y_1), \ldots, (x_m, y_m)$:

$$\hat{\epsilon}(h) = \frac{1}{m} \sum_{i=1}^{m} 1(h(x_i) \neq y_i)$$

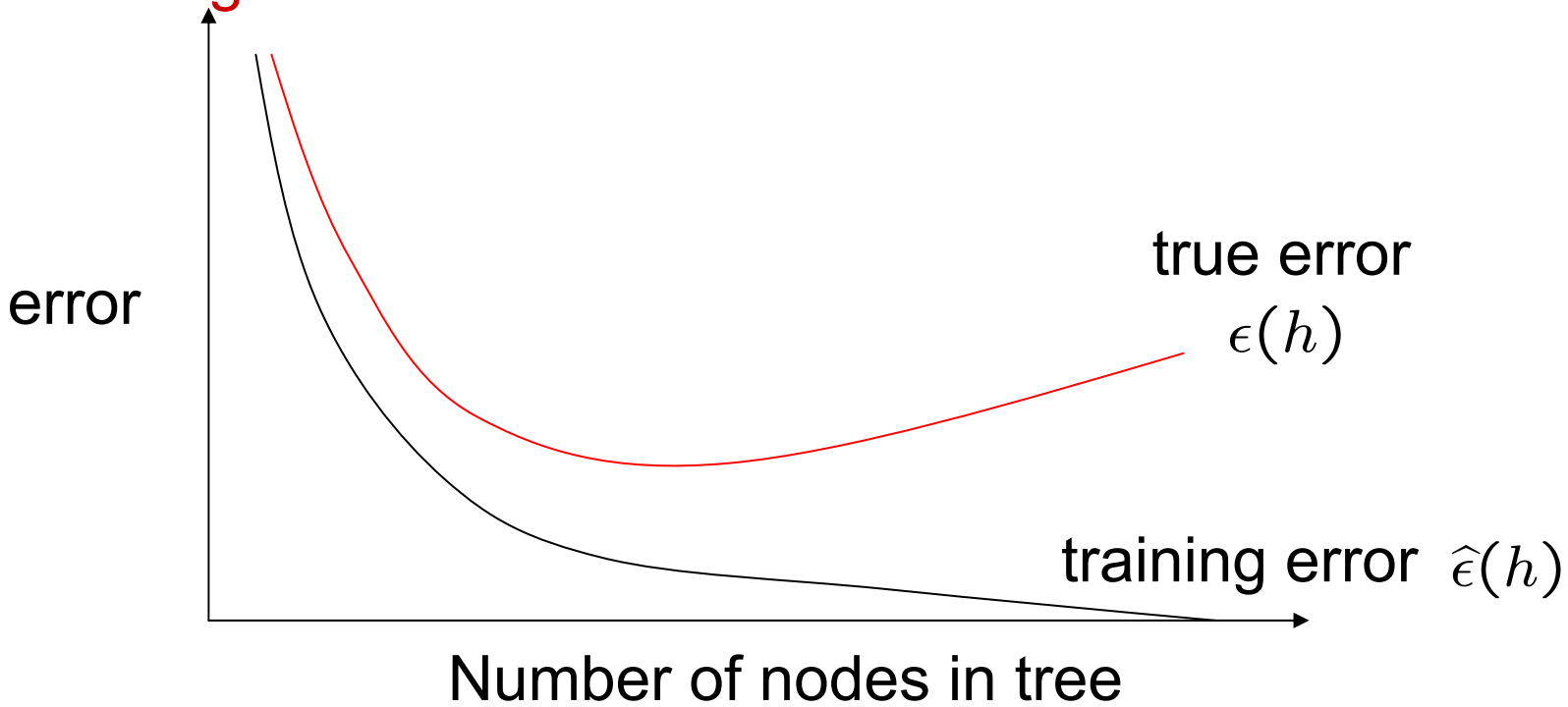
- But we can also talk about its true error:

$$\epsilon(h) = \mathbf{P}_{(x,y)\sim D}(h(x) \neq y)$$

- How are these two quantities related?

Overfitting: Illustration

Building a decision tree

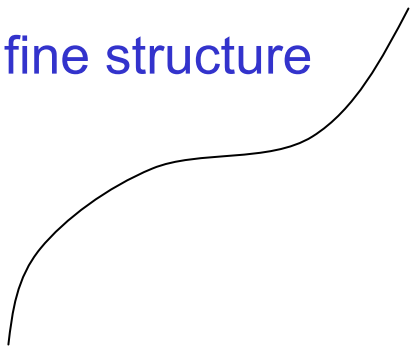


- As we make our tree more and more complicated:
  - training error keeps going down
  - but true error stops improving and may even get worse!

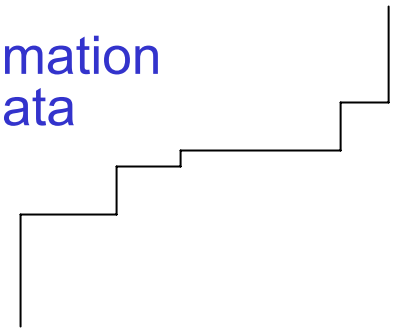
Overfitting: One Perspective

1. The true underlying distribution D is the one whose structure we would like to capture
2. The training data reflects the structure of D, so it helps us.
3. But it also has chance structure of its own – we must try to avoid modeling this.

Reality's fine structure



Coarse approximation by training data



- For instance: D = uniform distribution over {1,2,3,…,100}
- Pick three training points: eg. 6, 12, 98.
- They all happen to be even: but this is just chance structure. It would be bad to build this into a classifier.

Decision Tree Pruning

[1] Split the training data S_{full} into two parts
A smaller training set *S*
A validation set *V* (a model of reality, a surrogate test set)

[2] Build a full decision tree *T* using *S*

[3] Then prune using *V*
repeat
if there a node *u* in *T* such that removing the subtree rooted at *u* decreases error on *V*:
T = *T* – {subtree rooted at *u*}

[Of course, *V* has chance structure too, but its chance structure is unlikely to coincide with that of *S*]

SPAM Data Set

4601 points, each corresponding to an email message

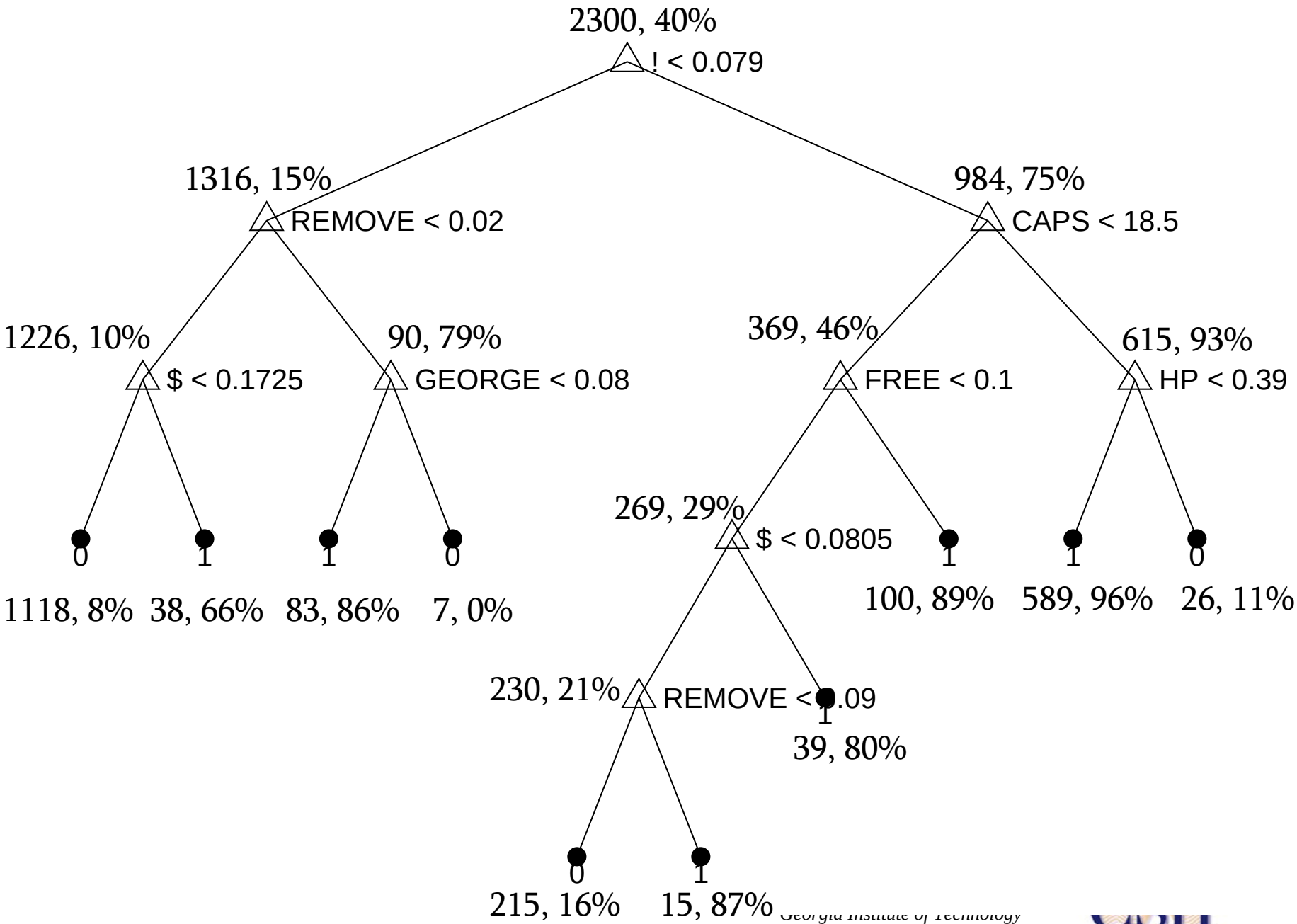
39.4% are SPAM

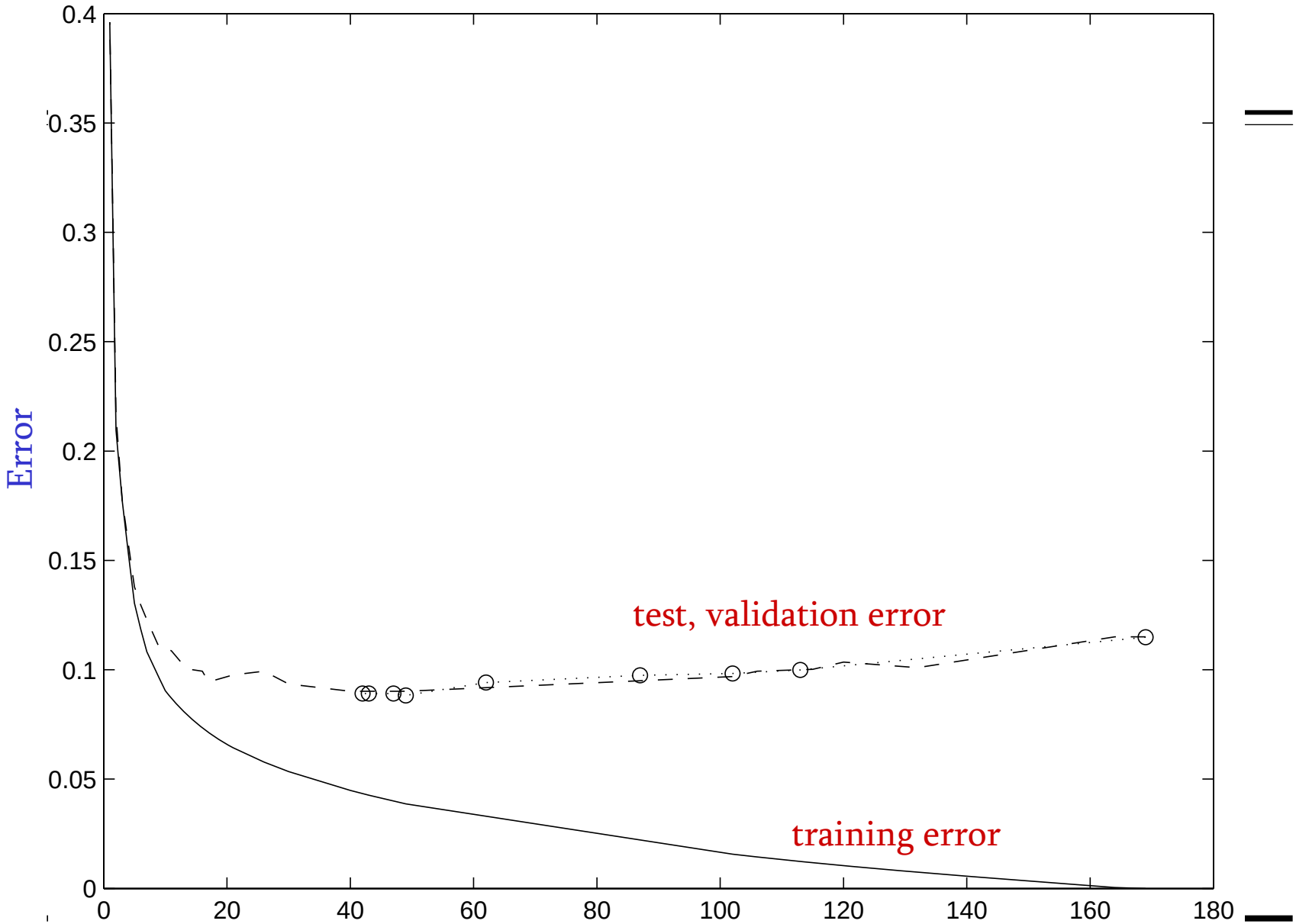
Each point has 57 features:

- 48 check for specific words, eg. FREE
- 6 check for specific characters, eg. !
- 3 others, eg. longest run of capitals

Randomly divide into three parts:

50%	training data
25%	validation
25%	testing


2300, 40%
! < 0.079
1316, 15%
REMOVE < 0.02
984, 75%
CAPS < 18.5
1226, 10%
$ < 0.1725
90, 79%
GEORGE < 0.08
369, 46%
FREE < 0.1
615, 93%
HP < 0.39
0
1
1
0
269, 29%
$ < 0.0805
1
1
0
1118, 8%
38, 66%
83, 86%
7, 0%
100, 89%
589, 96%
26, 11%
230, 21%
REMOVE < 0.09
1
39, 80%
0
1
215, 16%
15, 87%


0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
0
20
40
60
80
100
120
140
160
180
Error
Number of nodes in tree
test, validation error
training error

How Accurate is the Validation Set?

How accurate are error estimates based on the validation set?

For any classifier *h* and underlying distribution *D* on *X* and *Y*:

"true error" $\mathrm{err}(h) = \mathbf{P}_{(x,y)\sim D}(h(x) \neq y)$

"error on set *S*" $\mathrm{err}(h, S) = \frac{1}{|S|} \sum_{(x,y)\in S} \mathbf{1}(h(x) \neq y)$

Suppose *S* is chosen i.i.d. (independent, identically distributed) from *D*. Then (over the random choices of *S*),

$$\mathbf{E}[\mathrm{err}(h, S)] = \mathrm{err}(h)$$

And the standard deviation of err(*h*,*S*) is about $1/\sqrt{|S|}$

How Accurate is the Validation Set? (Cont.)

Fix any *h*. Suppose *S* is chosen i.i.d. (independent, identically distributed) from D. Then (over the random choices of S),

$$\mathbf{E}[\text{err}(h, S)] = \text{err}(h)$$

And the standard deviation of err(*h*,*S*) is about $1/\sqrt{|S|}$

(i) In this scenario, *S* is used to assess the accuracy of a single, prespecified classifier *h*
Can the same S be used to check many classifiers simultaneously?
Answer: VC theory

(ii) In particular, if *h* was created using *S* as a training set, then the above scenario does not apply. In such situations, err(*h*,*S*) might be a very poor estimate of err(*h*).

Summary

- Today's Class
  - Decision Trees (Chapter 9)
- Next Classes
  - Unsupervised Learning
- Project Presentation: April 21 and 23
- Reading Assignments
  - HTF, Chapter 9