Machine Learning

CSCI 4622 Fall 2019

Prof. Claire Monteleoni

Today: Lecture 5

- Perceptron, convergence (review)
- Decision Trees

With much credit to T. Jaakkola and S. Dasgupta

Perceptron

Examples, x in $X = R^d$

Labels, y in $Y = \{-1, 1\}$

Perceptron: If $y_t(v_t \cdot x_t) < 0$ Filtering rule

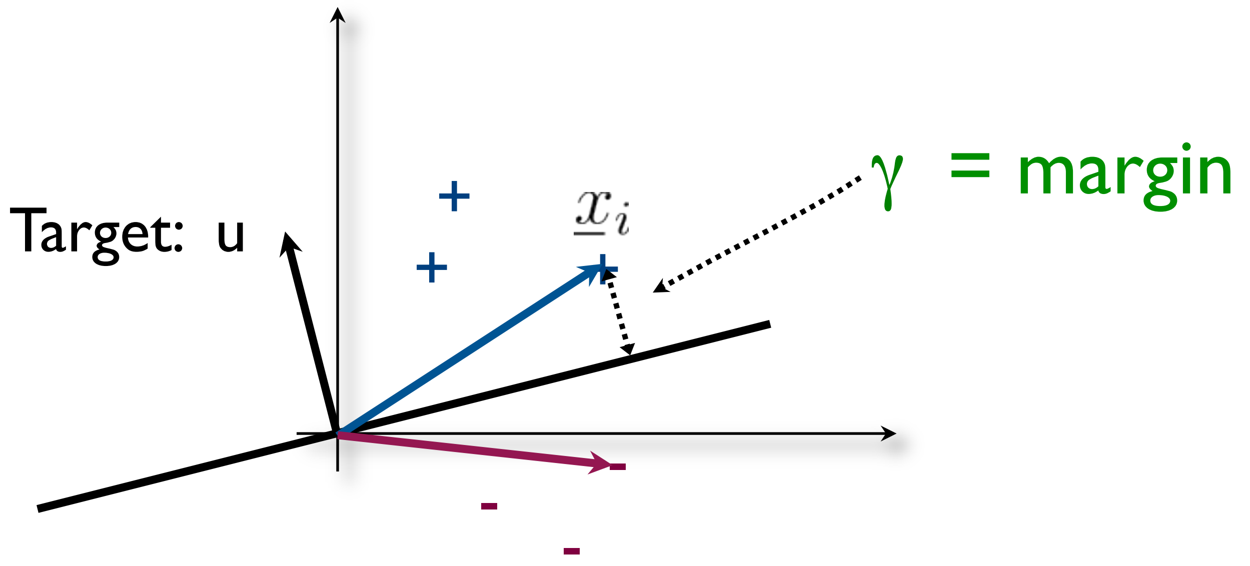
$v_{t+1} = v_t + y_t x_t$ Update step

Initialization: $v_1 = 0$ (vector of zeros).

NOTE: Additive updates. Algorithm credited to [Rosenblatt '58].

"Margin"

- One way to analyze Perceptron's convergence is to assume there exists a target classifier, u, with good properties
- One such property is margin, i.e., how close the separating boundary is to the points



Problem framework: margin assumption

$$x_t \in \mathbb{R}^d, \ \|x\| \leq R, \ y_t \in \{-1, +1\}$$

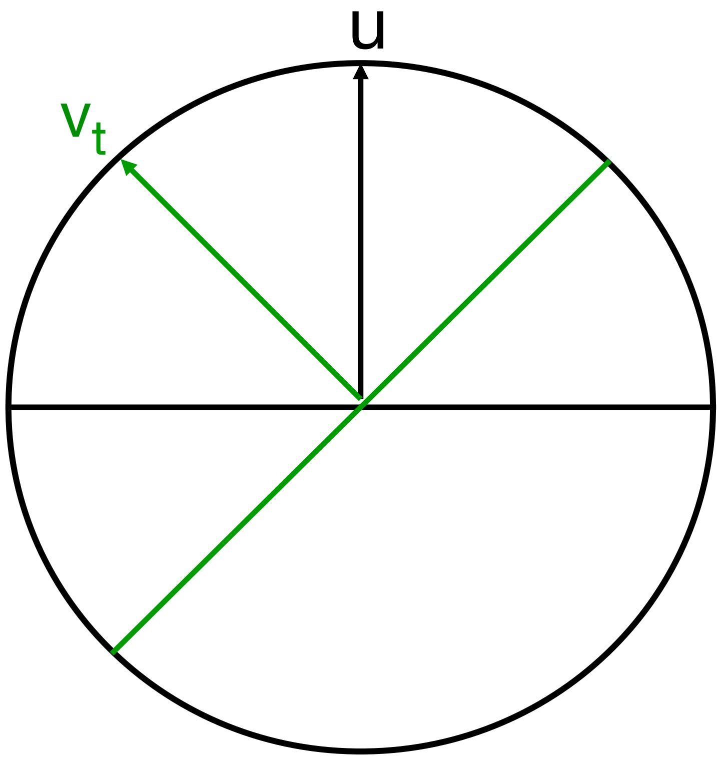
Margin assumption: there exists some perfect classifier, u, such that:

$$\boxed{u : y_t(u \cdot x_t) > \gamma \ \forall t, \ \ \|u\| = 1}$$

Assume u is through origin.

→ Always possible, by increasing dimension by 1.

Algorithm's current classifier: v_t



Perceptron analysis with margin

Margin assumption: no distribution assumption except separable (through origin), AND: $y_t(u \,.\, x_t) \geq \gamma$ for all t.

- $O(1/\gamma^2)$ mistakes to reach zero error [Novikoff '62].

Proof: Let $||u|| = 1$. Let (x, y) be a mistake, i.e. $y\,(v_t \,.\, x) < 0$, $||x|| \leq R$.

Lemma 1: $u \,.\, v_{t+1} \geq u \,.\, v_t + \gamma$.

Proof: $u \,.\, v_{t+1} = u \,.\, (v_t + y\,x) = u \,.\, v_t + y\,(u \,.\, x) \geq u \,.\, v_t + \gamma$

(by definition of margin, γ).

Lemma 2: $||v_{t+1}||^2 \leq ||v_t||^2 + R^2$

Proof:
$$\begin{aligned} ||v_{t+1}||^2 = ||v_t + y\,x||^2 &= ||v_t||^2 + 2y(v_t \,.\, x) + ||x||^2 \\ &\leq ||v_t||^2 + 2y(v_t \,.\, x) + R^2 \\ &< ||v_t||^2 + R^2 \end{aligned}$$

because v_t makes a mistake on (x, y), i.e. $2y(v_t \,.\, x) < 0$.

Perceptron analysis with margin

Proof continued:

Let $||u|| = 1$. Let (x, y) be a mistake, i.e. $y (v_t . x) < 0$, $||x|| \leq R$.

Lemma 1: $u . v_{t+1} \geq u . v_t + \gamma$.

Lemma 2: $||v_{t+1}||^2 \leq || v_t||^2 + R^2$.

Finally, after M mistakes:

a. $u . v_{M+1} \geq M \gamma$, by Lemma 1 (expanding the recurrence).

b. $||v_{M+1}||^2 \leq M R^2$, by Lemma 2. So $||v_{M+1}|| \leq M^{½} R$.

Since u is a unit vector, $u . v_t \leq || v_t||$ by Cauchy-Schwartz: $|u . v| \leq ||u|| \, ||v||$

So, $u . v_{M+1} \leq || v_{M+1}||$.

Using a. and b. for LHS and RHS respectively,

$M \gamma \leq u . v_{M+1} \leq || v_{M+1}|| \leq M^{½} R$

$M \gamma \leq M^{½} R$

$M^{½} \leq R / \gamma$, and $M \leq (R / \gamma)^2$ □

Perceptron: convergence

In summary, we showed that:

$$M \leq (R / \gamma)^2$$

Since R is a constant (upper bound on ||x||), this means:

$$M = O(1 / \gamma^2)$$

M is the number of mistakes. So after $O(1 / \gamma^2)$ mistakes, the Perceptron algorithm will not make any more mistakes!

Recall: this assumes a margin of at least γ for all data.

Fisher’ s IRIS data

Four features

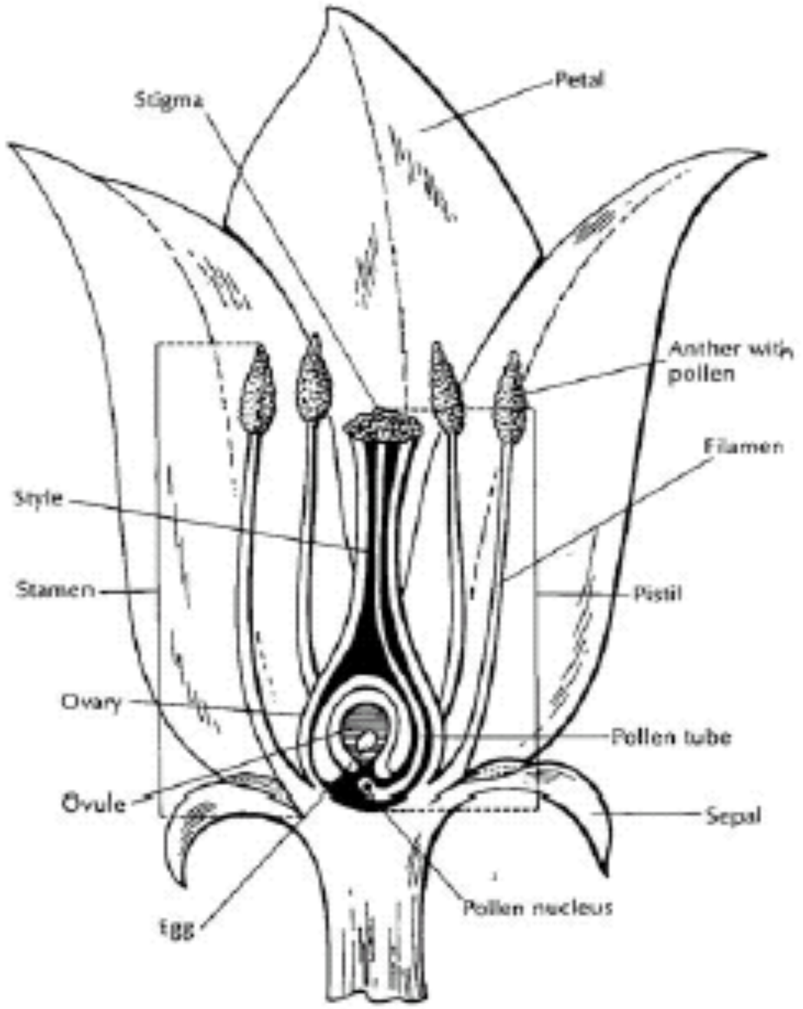
- sepal length
- sepal width
- petal length
- petal width

Three classes (species of iris)

- setosa
- versicolor
- virginica

50 instances of each

Parts of a Flower

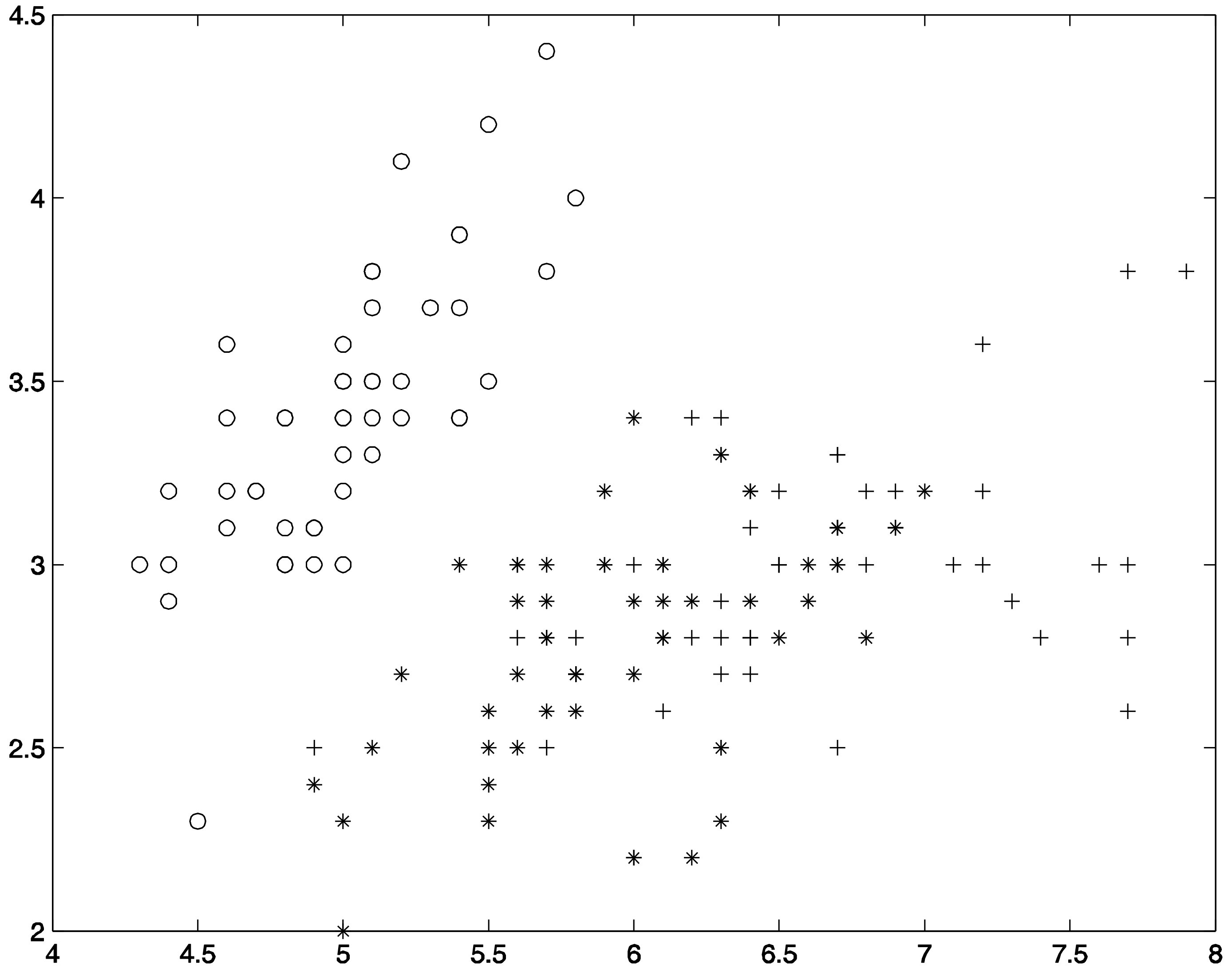


THE USE OF MULTIPLE MEASUREMENTS IN TAXONOMIC PROBLEMS

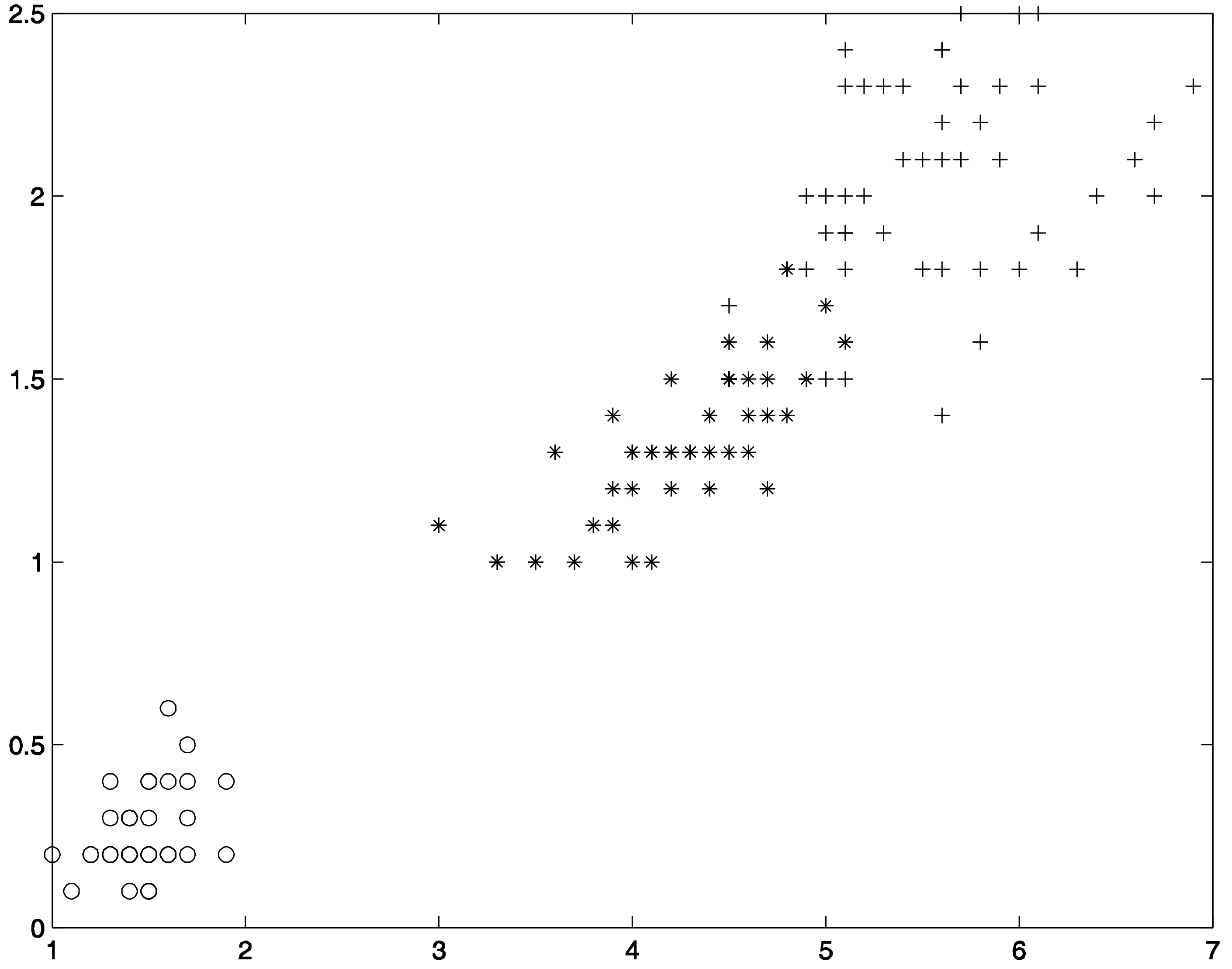
By R. A. FISHER, Sc.D., F.R.S.

I. Discriminant functions

When two or more populations have been measured in several characters, $x_1, \ldots, x_s$, special interest attaches to certain linear functions of the measurements by which the populations are best discriminated. At the author's suggestion use has already been made of this fact in craniometry (*a*) by Mr E. S. Martin, who has applied the principle to the sex differences in measurements of the mandible, and (*b*) by Miss Mildred Barnard, who showed how to obtain from a series of dated series the particular compound of cranial measurements showing most distinctly a progressive or secular trend. In the present paper the application of the same principle will be illustrated on a taxonomic problem; some questions connected with the precision of the processes employed will also be discussed.

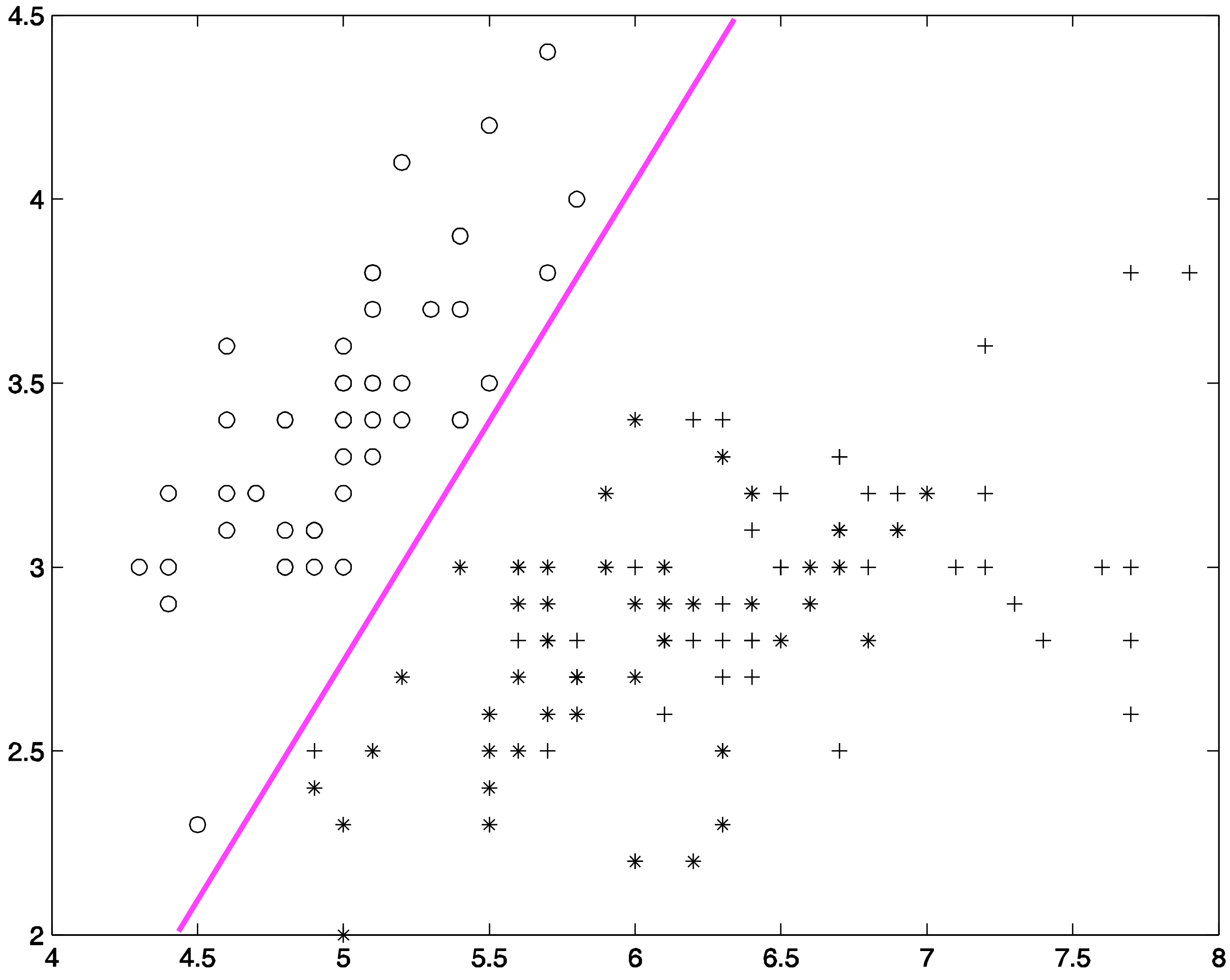


Features 1 and 2 (sepal width/length)



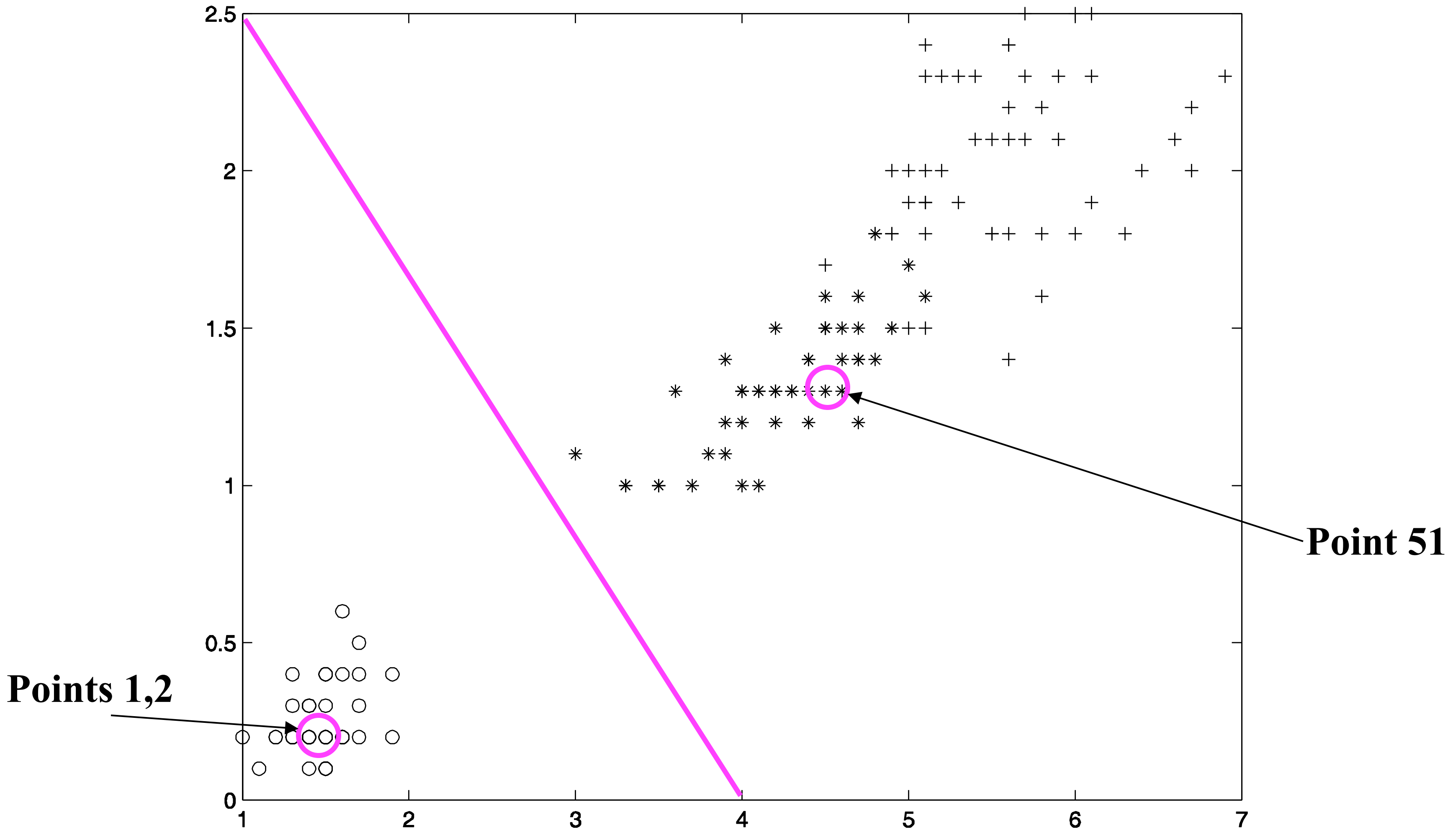
Features 3 and 4 (petal width/length)

Features 1 and 2; goal: separate setosa from other two



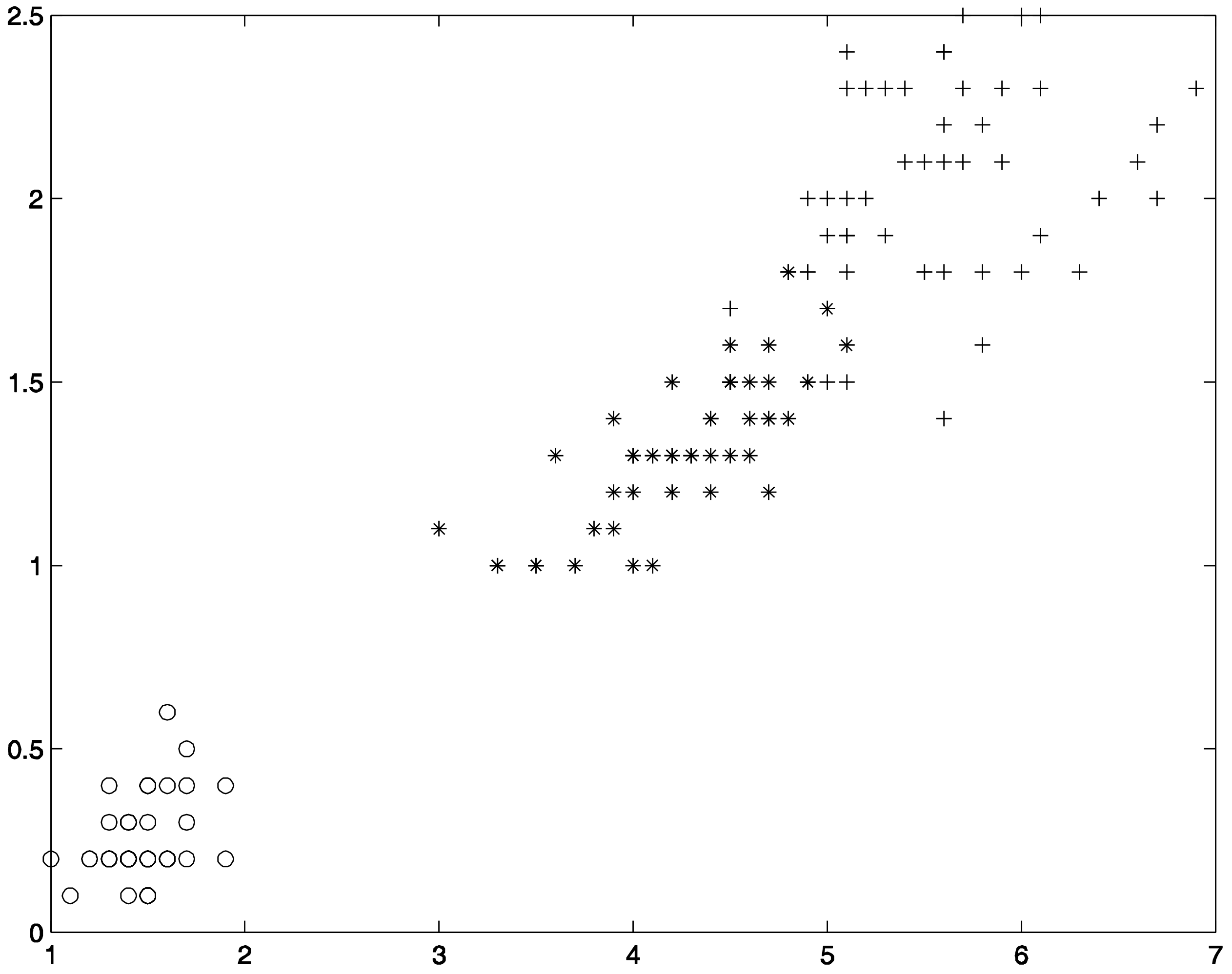
1500 updates (different permutation: 900)

Features 3 and 4; goal: separate setosa from other two



Iteration 1 [1,51] Iteration 2 [1,2] Iteration 3 [ ]

Features 3 and 4; goal: separate versicolor from other two

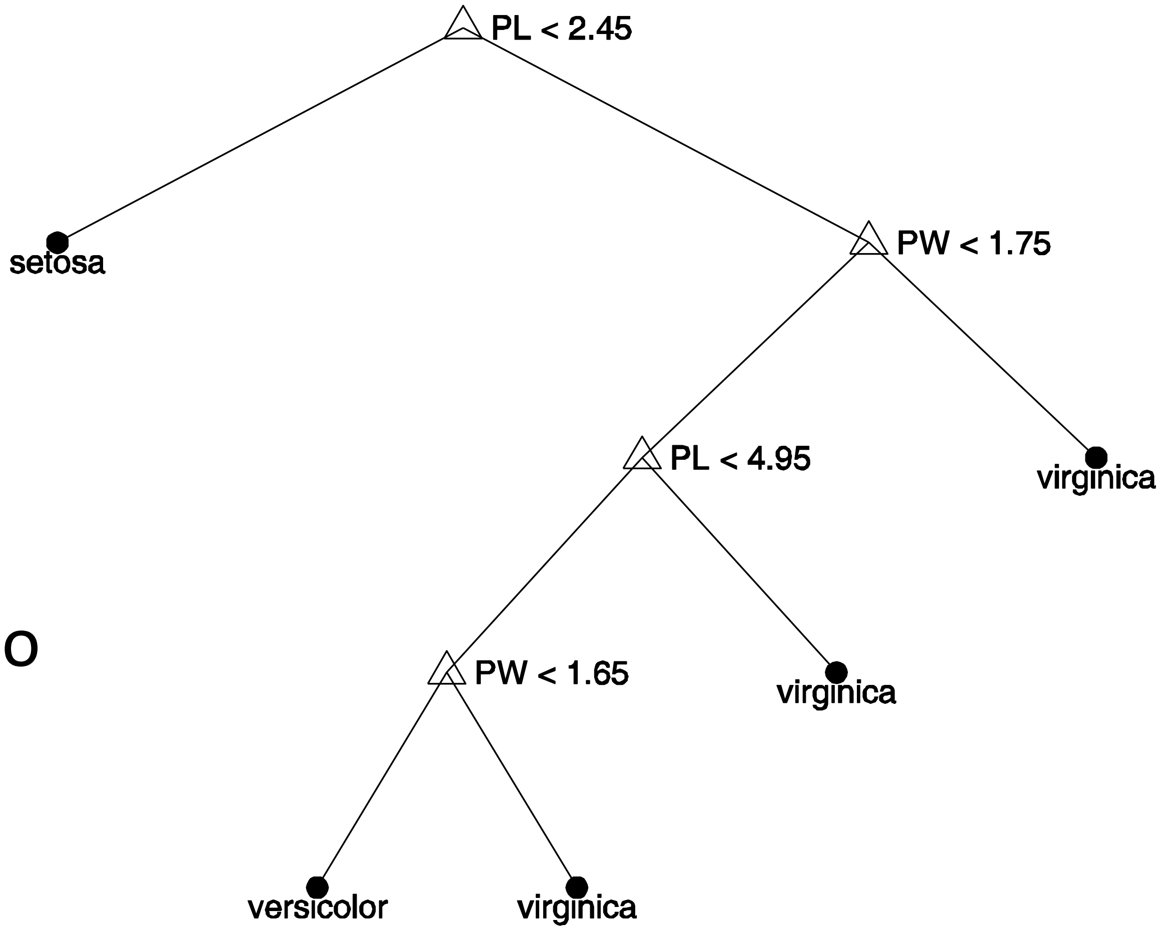


What if the data is not linearly separable?

Decision trees

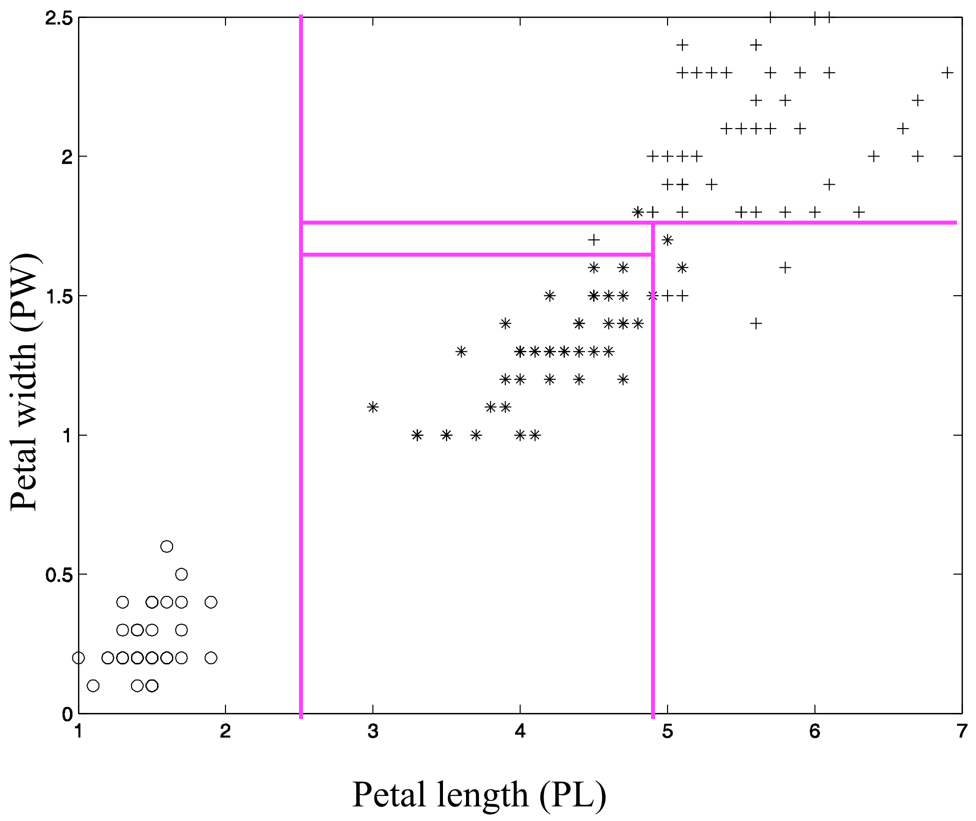
- Simple
- Multiclass
- Not perfect, but close
- Comprehensible to humans

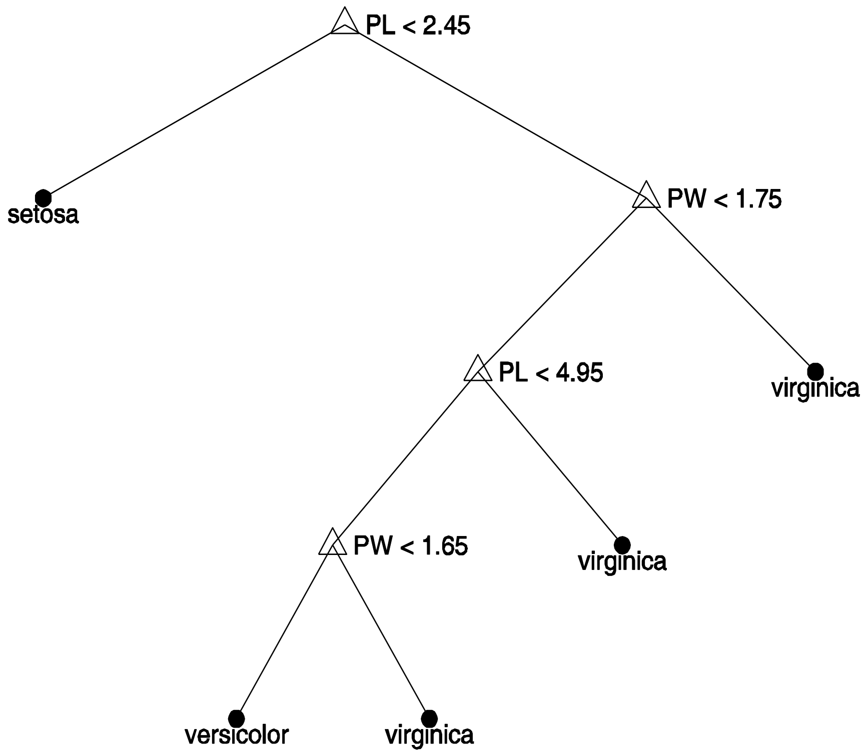
How to build them?



Example

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

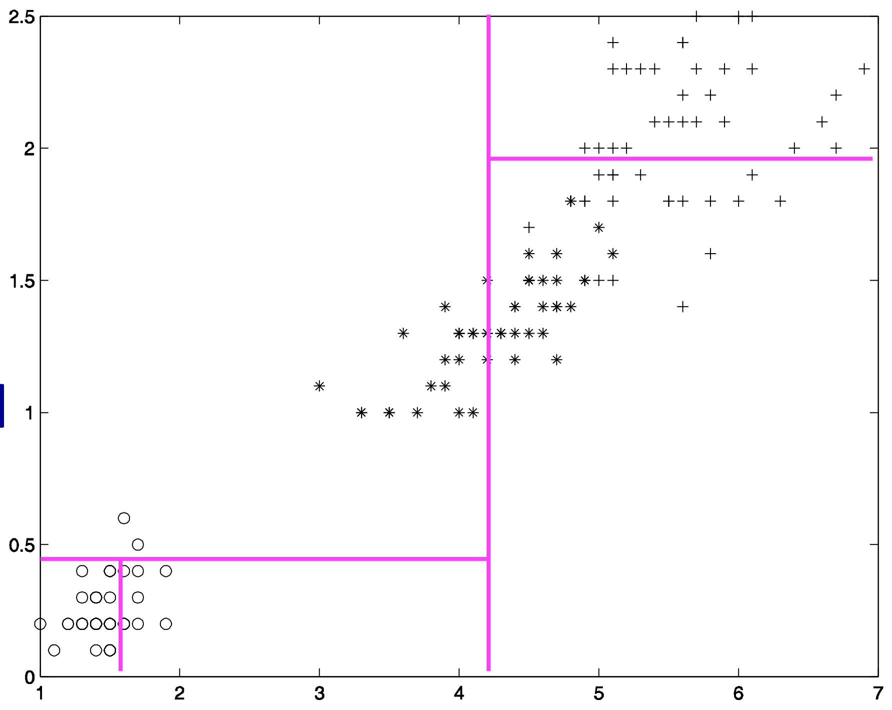




K-d trees

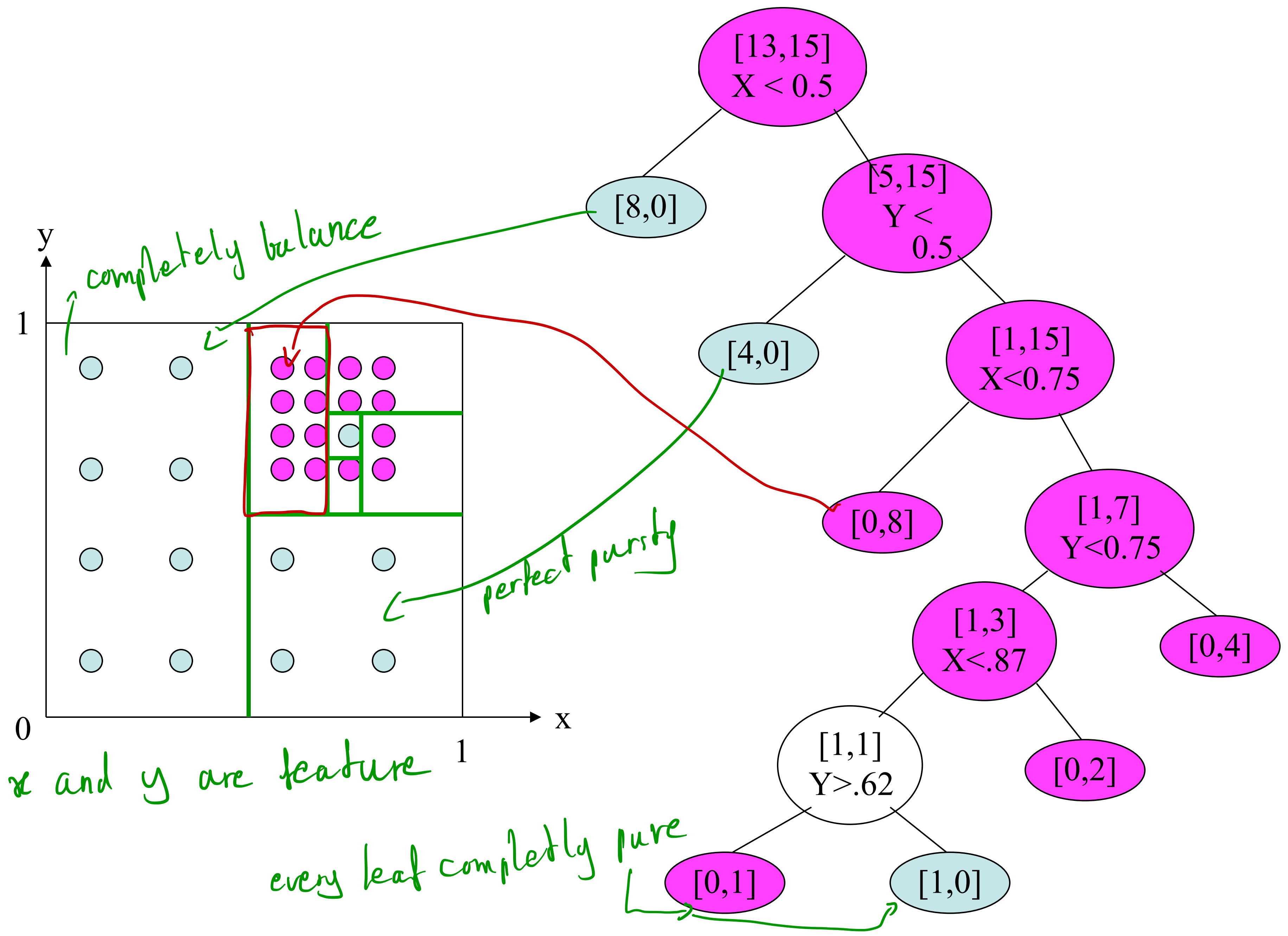
Rapidly partition the space into rectangular regions which contain few points

NOTE: Also useful for creating a nearest-neighbor data structure.

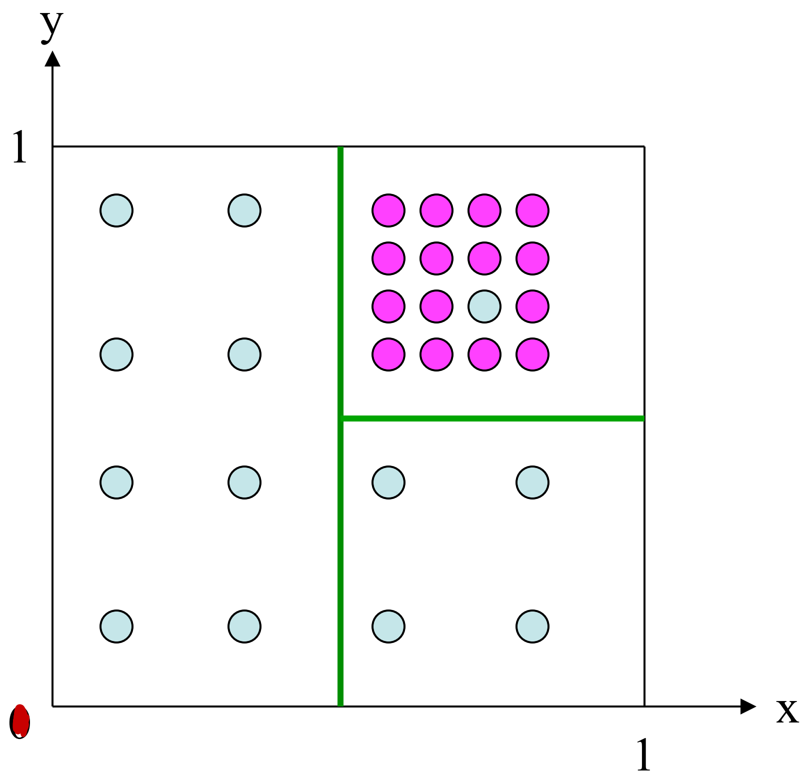


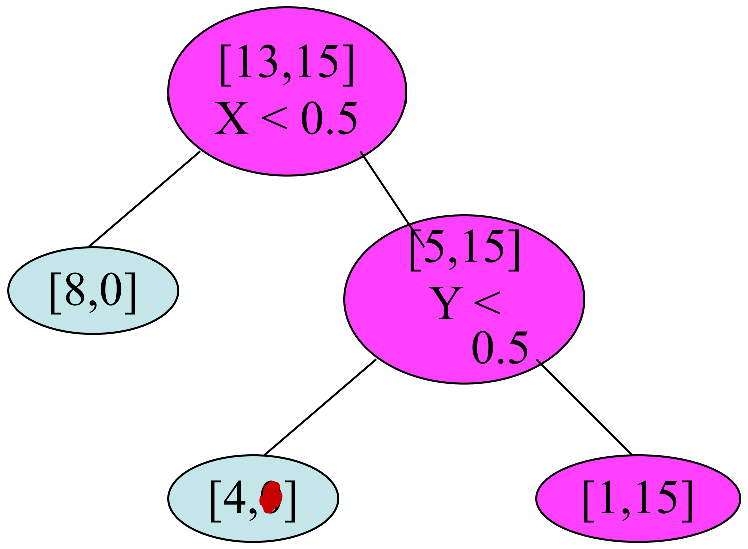
- Cycle through dimensions.
- Split at median.
- Recurse on each side (along next dimension).

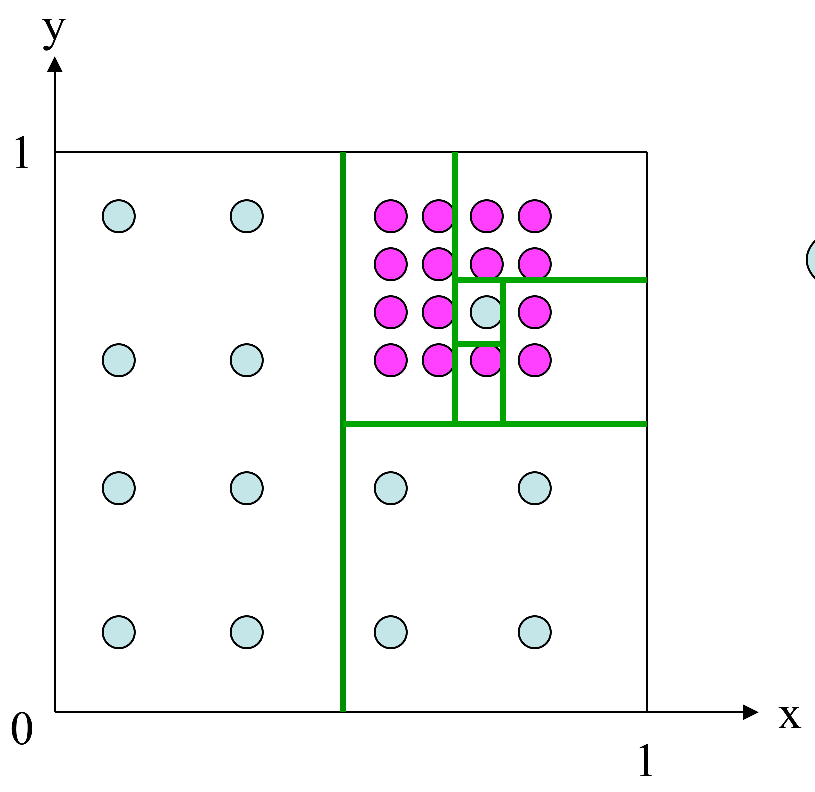
But for the current problem, we don't care how many points each region contains: we just want them to consist (almost) exclusively of points from one class…

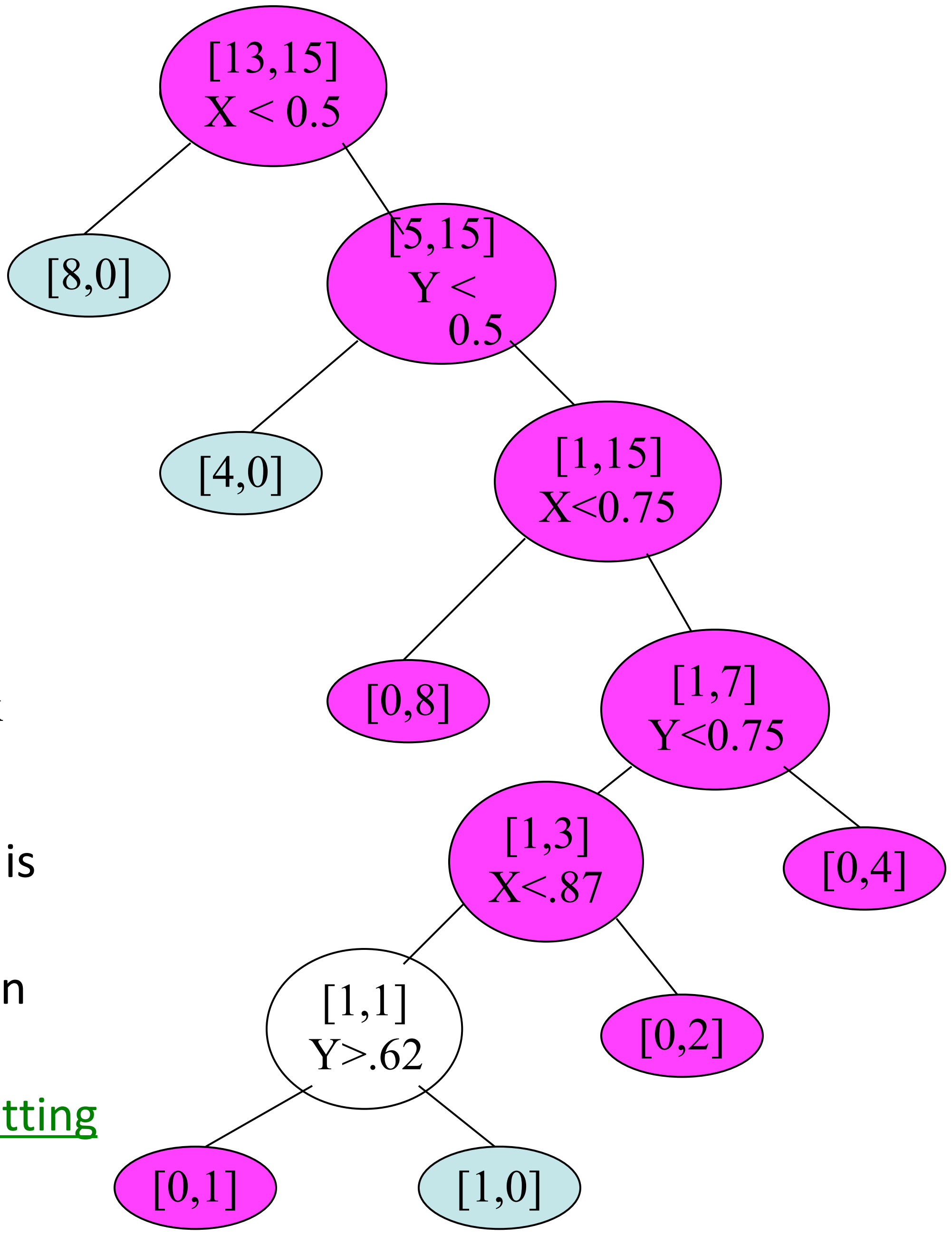

[13,15]
X < 0.5
[8,0]
[5,15]
Y < 0.5
[4,0]
[1,15]
X<0.75
[0,8]
[1,7]
Y<0.75
[1,3]
X<.87
[0,4]
[1,1]
Y>.62
[0,2]
[0,1]
[1,0]
y
1
0
x
1
completely balance
perfect purity
x and y are feature
every leaf completly pure

Retracing a few steps...









- This tree does slightly better – but 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 numbers, Boolean, categorical, …

Can perfectly fit any self-consistent training set
- I.e., data with no pairs of examples of the form (x, y), (x, y’)

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

Building a decision tree

Greedy algorithm: build the 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

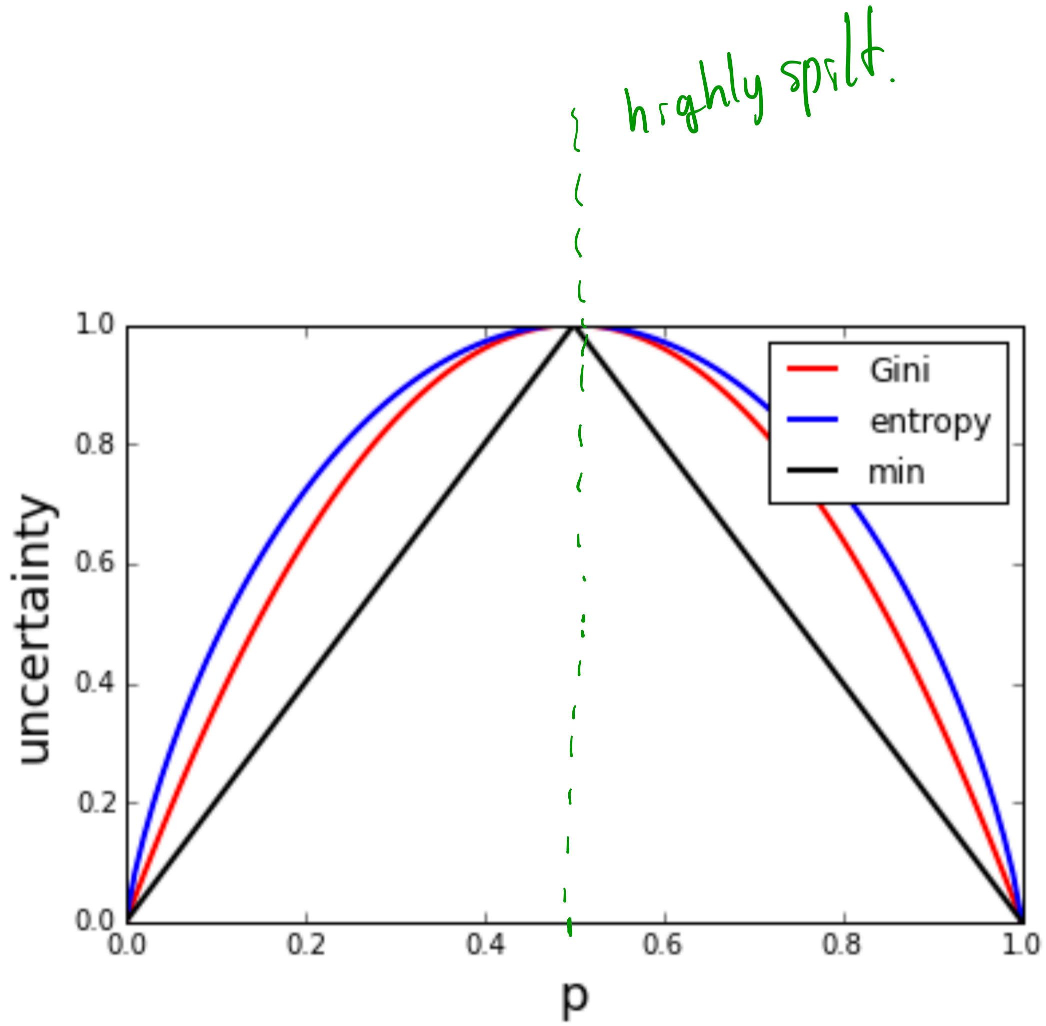
$$p \log \frac{1}{p} + (1-p) \log \frac{1}{1-p}$$

(ii) Gini index

$$2p(1-p)$$

(iii) Simplest of all

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



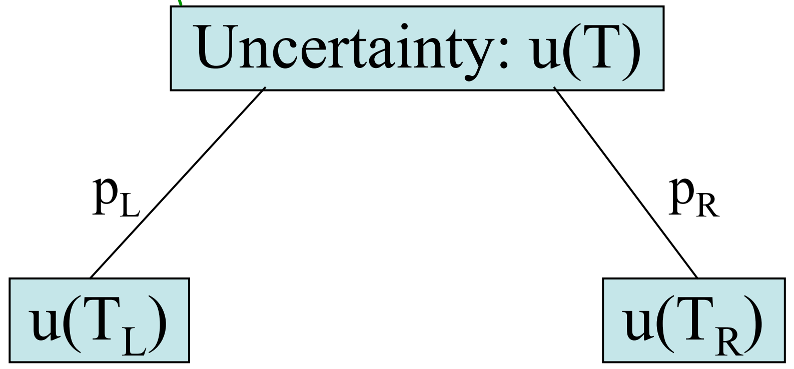
Uncertainty

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	$2p(1-p)$	$\sum_{i=1}^{k} p_i(1-p_i) = 1 - \lVert p \rVert^2$
Simplest	$\min\{p, 1-p\}$	$\min_i p_i$

The benefit of a split

we look many point to look at left tree and right tree



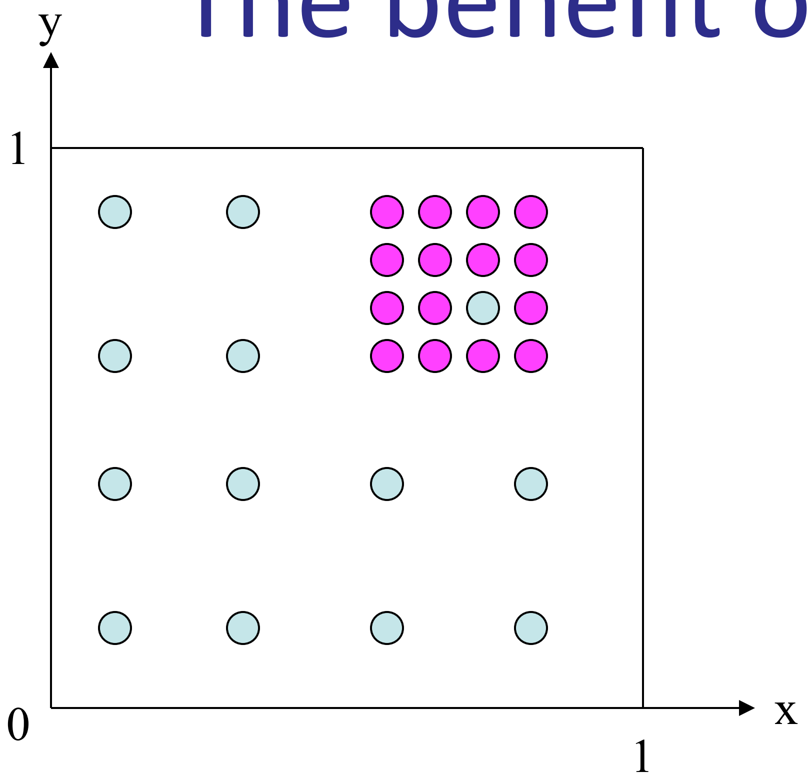
Of the points in T:

p_L fraction go to T_L

p_R fraction go to T_R

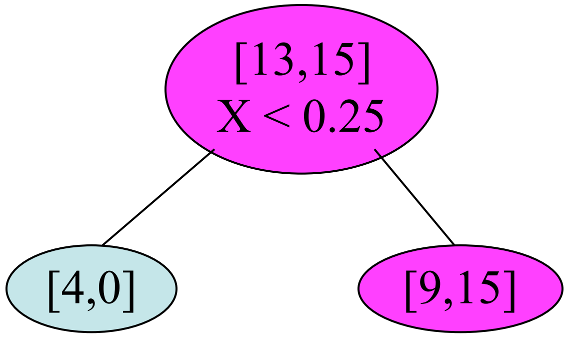
$$\text{Benefit of split} = \left(u(T) - \underbrace{\{p_L\, u(T_L) + p_R\, u(T_R)\}}_{\text{Expected uncertainty after split}}\right) \times \underbrace{|T|}_{\text{\# points in T}}$$

The benefit of a split: example



Initial uncertainty (Gini):

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



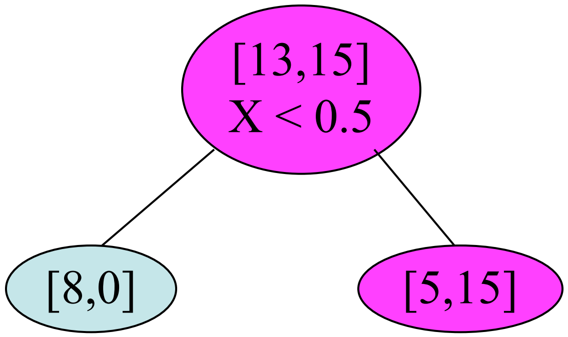
$p_L = 4/28$

$u_L = 0$

$p_R = 24/28$

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

Expected uncertainty: $p_L u_L + p_R u_R = 22.5/56$



Expected uncertainty: $p_L u_L + p_R u_R = 15/56$

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.

Bonus slides

What is overfitting?

Data comes from some true underlying distribution D on X × Y.
[X = input space, Y = label space]

All we ever see are samples from D: training set, test set, etc.

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

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

But we can also talk about its true error:

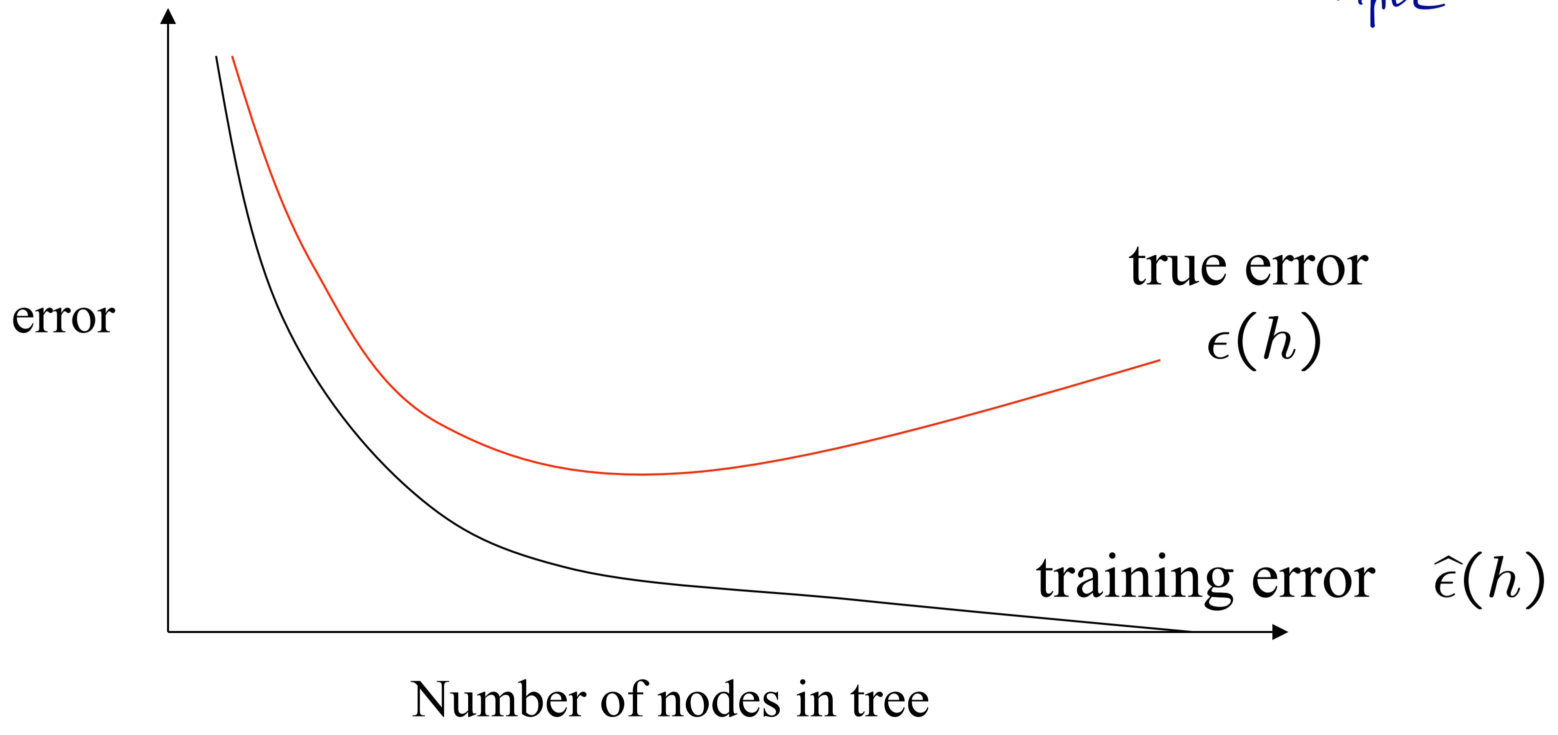
$$\epsilon(h) = \mathbf{P}_{(x,y)\sim D}(h(x) \neq y)$$

How are these two quantities related?

Overfitting: picture

Building a decision tree

increase k, classifier more simple



As we make our tree more and more complicated:

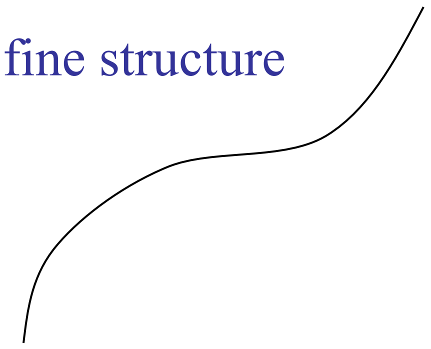
training error keeps going down

but true error stops improving and may even get worse!

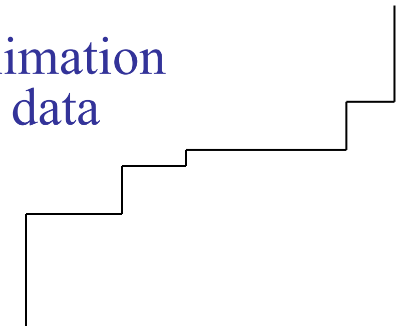
Overfitting: one perspective

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 <u>even</u>: but this is just <u>chance structure</u>. It would be bad to build this into a classifier.

Overfitting: another perspective

"Fit a line to a point": absurd
"Fit a plane to two points": likewise

Moral: It is not good to use a model which is so complex that there isn't enough data to reliably estimate its parameters.

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 → as approx as true error

```
repeat
    if there a node u in T such that removing the subtree
    rooted at u decreases the 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.]

Example: SPAM data set

4601 points, each corresponding to an email message

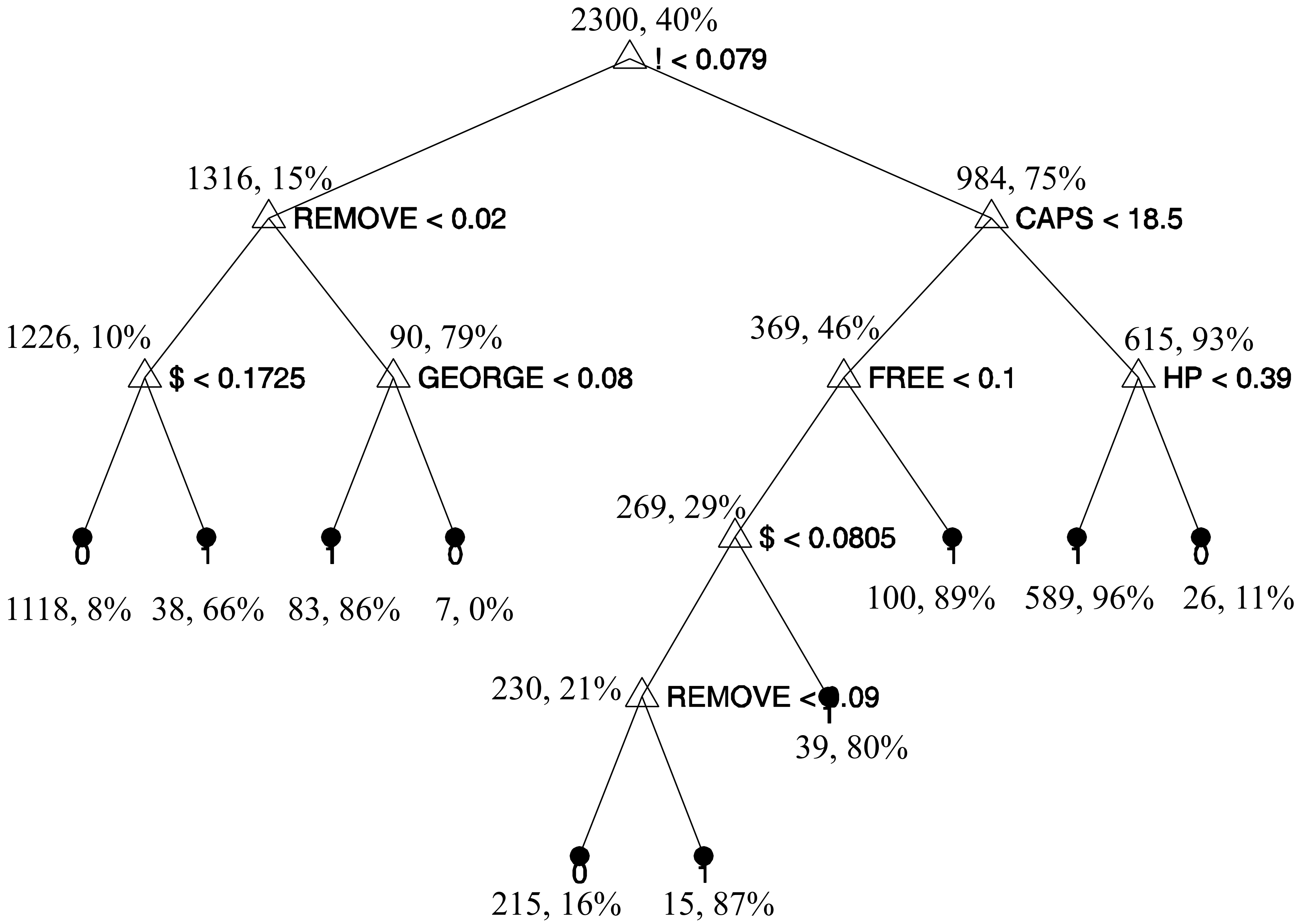
39.4% are SPAM

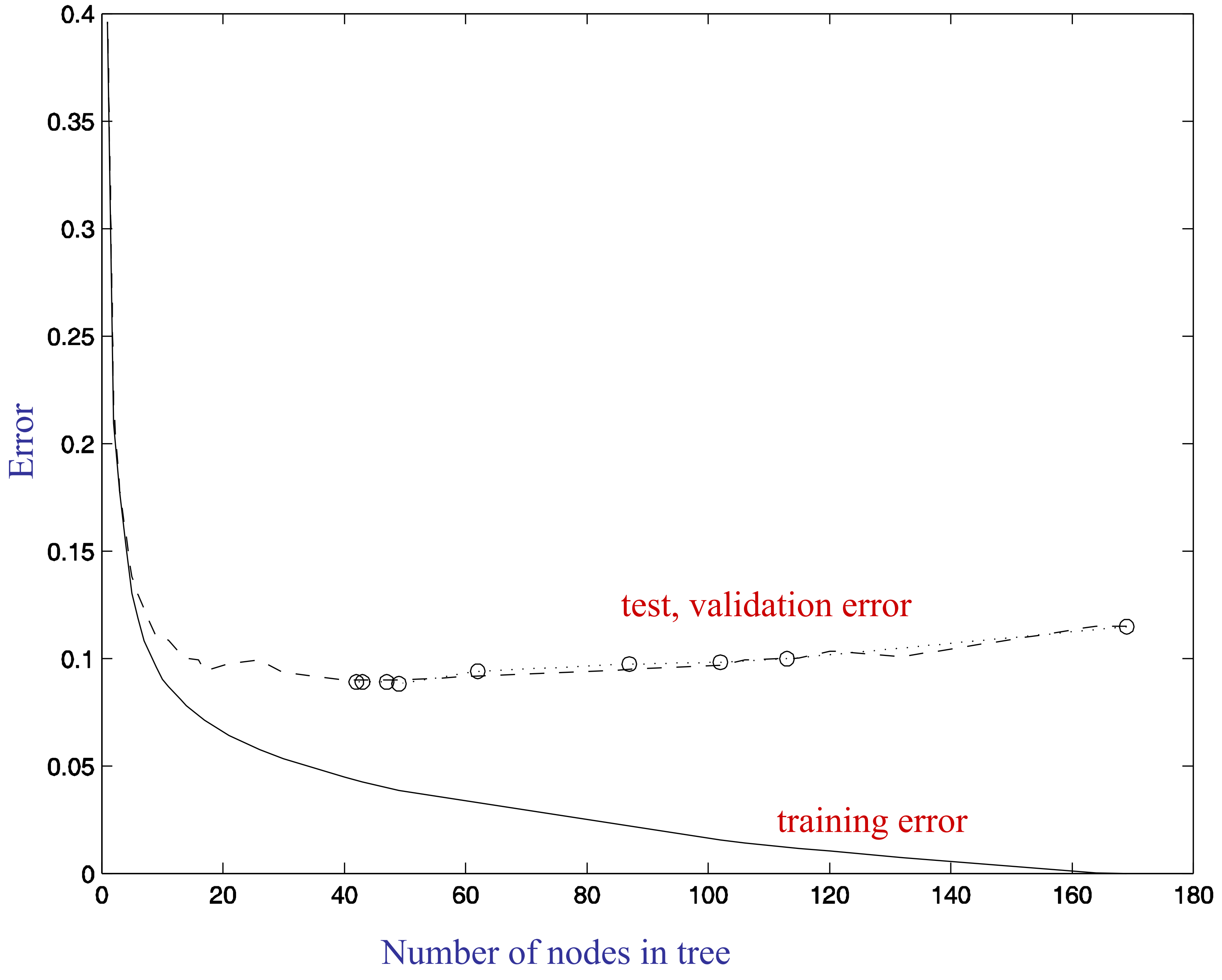
Each point has 57 features:

- 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
269, 29%
$ < 0.0805
0
1
1
0
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%



Error
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
test, validation error
training error
Number of nodes in tree

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 × Y:

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

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

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

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

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

How accurate is the validation set?

Fix any h. 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 $1/\sqrt{|S|}$

(i) In this scenario, S is used to assess the accuracy of a <u>single</u>, <u>prespecified</u> 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 <u>not</u> apply. In such situations, err(h,S) might be a very poor estimate of err(h).