Expert systems

- Research in symbolic artificial intelligence in 1970's and 1980's to develop decision support systems
- *Expert systems* used domain knowledge extracted from a human expert
- MYCIN diagnosed and planned treatment for blood infections, using rules:

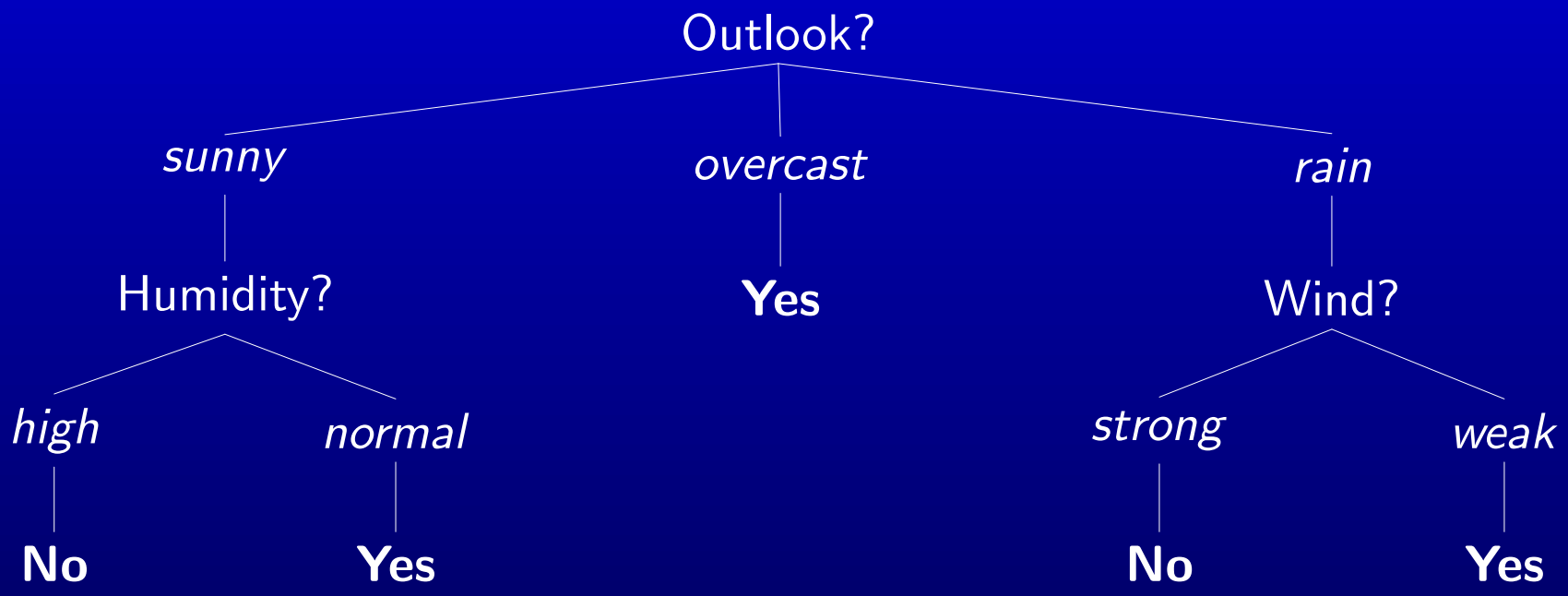
```
RULE037

IF the organism
        1) stains grampos
        2) has coccus shape
        3) grows in chains
THEN
  There is suggestive evidence (.7) that the
  identity of the organism is streptococcus.
```

Expert systems

- Sometimes domain information is organized into a tree structure (dichotomous key)
- *Productions* of an expert system can often be organized into a *decision tree*
- Decision theory — maximize expected utility

Decision trees

- Is today a good day to play tennis?



Decision trees

- Decision trees are well suited to represent problems where instances are vectors of discrete-valued features and the target function has predefined discrete values
- Value of taget function is a ‘logical’ combination of feature values (no weights, logs, sums, etc.)
- NLP classification problems very often look like this
- If necessary, continuous (interval) variables can be converted to discrete (ordinal or nominal) variables (*discretization*)

Decision trees

- Decision trees can be constructed manually by a *knowledge engineer*
- It's more fun to *induce* a decision tree from a collection of labeled instances
- Early work on Divide and conquer algorithms: Hoveland, Hunt (1950's and 60's)
- Friedman, Breiman $\rightarrow$ CART (1984)
- Algorithms ID3 and C4.5 (and others) developed by Ross Quinlan (1978–now)

Algorithm ID3

- Take a set of labeled instances T and a set of features F
- If T is empty, assign the most frequent class
- If all T are in the same class C, assign C
- If F is empty, assign the most frequent class in T
- Otherwise, choose the feature f_i which best classifies T
- For each possible value v_j of f_i, construct a subtree based on the instances in T with $f_i = v_j$ and features $F - f_i$

Algorithm ID3

- What makes a feature “good” for classification?
- Remember entropy:

$$H(X) = -\sum P(x_i) \log_2 P(x_i)$$

- The best feature would divides T into subsets S with only one class, with $H(S_v) = 0$
- More generally, we can measure the discriminative power of a feature by its information gain:

$$\begin{aligned} G(T, f) &= H(T) - H(T|f) \\ &= H(T) - \sum_{v \in f} \frac{|S_v|}{|T|} H(S_v) \end{aligned}$$

Algorithm ID3

Day	*Outlook*	*Temp*	*Humid*	*Wind*	*Play?*
d1	sunny	hot	high	weak	no
d2	sunny	hot	high	strong	no
d3	overcast	hot	high	weak	yes
d4	rain	mild	high	weak	yes
d5	rain	cool	normal	weak	yes
d6	rain	cool	normal	strong	no
d7	overcast	cool	normal	strong	yes
d8	sunny	mild	high	weak	no
d9	sunny	cool	normal	weak	yes
d10	rain	mild	normal	weak	yes
d11	sunny	mild	normal	strong	yes
d12	overcast	mild	high	strong	yes
d13	overcast	hot	normal	weak	yes
d14	rain	mild	high	strong	no

Algorithm ID3

- We compute the entropy of the training data:

$$\begin{aligned} H(T) &= -\frac{9}{14}\log_2\frac{9}{14} - \frac{5}{14}\log_2\frac{5}{14} \\ &= 0.94 \end{aligned}$$

- Now for each feature f_i, we compute $H(T|f_i)$:

$$\begin{aligned} H(T|\text{humid}) &= P(\text{humid=high})\,H(T|\text{humid=high}) + \\ &\qquad P(\text{humid=norm})\,H(T|\text{humid=norm}) \\ &= \frac{7}{14}(-\frac{3}{7}\log_2\frac{3}{7} - \frac{4}{7}\log_2\frac{4}{7}) + \frac{7}{14}(-\frac{6}{7}\log_2\frac{6}{7} - \frac{1}{7}\log_2\frac{1}{7}) \\ &= 0.79 \end{aligned}$$

- Branch on *outlook*, the feature with the lowest conditional entropy (or highest information gain)

Algorithm ID3

- Proceed recursive, with subset of the training data:

Day	*Outlook*	*Temp*	*Humid*	*Wind*	*Play?*
d1	sunny	hot	high	weak	no
d2	sunny	hot	high	strong	no
d8	sunny	mild	high	weak	no
d9	sunny	cool	normal	weak	yes
d11	sunny	mild	normal	strong	yes

- And:

Day	*Outlook*	*Temp*	*Humid*	*Wind*	*Play?*
d3	overcast	hot	high	weak	yes
d7	overcast	cool	normal	strong	yes
d12	overcast	mild	high	strong	yes
d13	overcast	hot	normal	weak	yes

Algorithm ID3

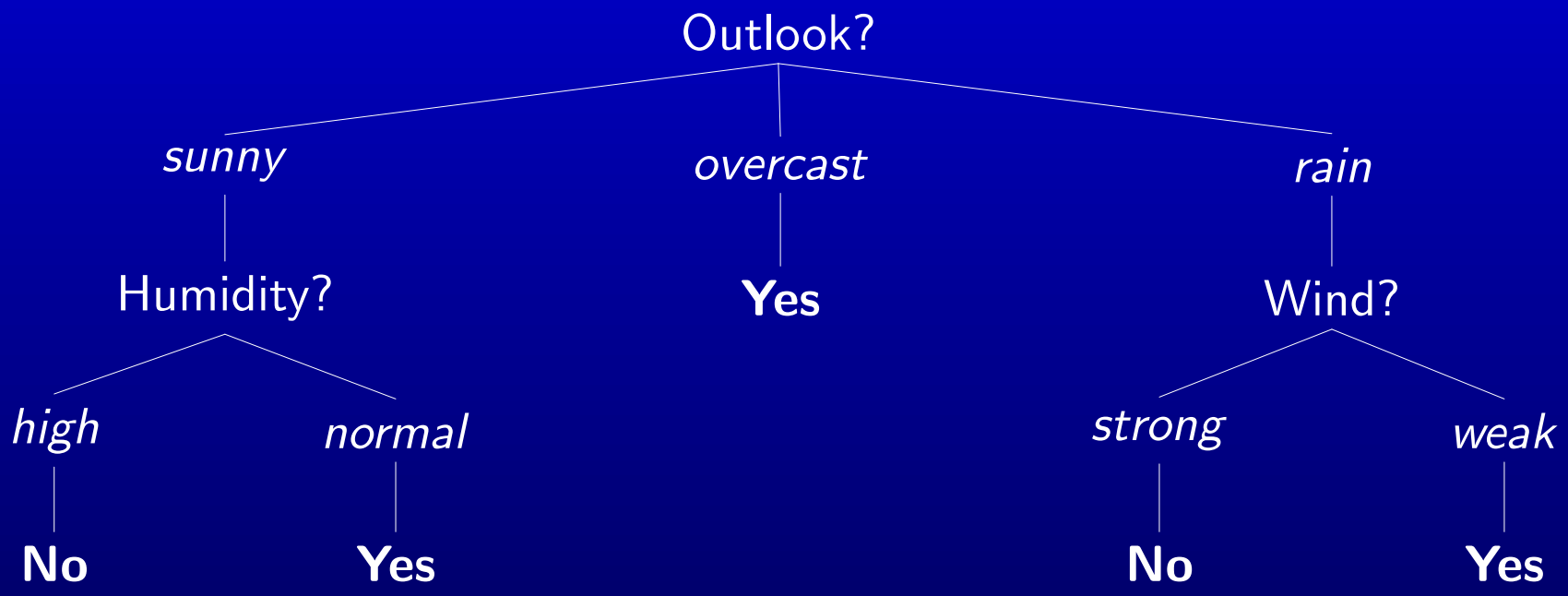
- And:

Day	*Outlook*	*Temp*	*Humid*	*Wind*	*Play?*
d4	rain	mild	high	weak	yes
d5	rain	cool	normal	weak	yes
d6	rain	cool	normal	strong	no
d10	rain	mild	normal	weak	yes
d14	rain	mild	high	strong	no

- Continue until subsets are unequivocal or we are out of features

Algorithm ID3

- Result:



Algorithm ID3

- ID3 searches a *complete hypothesis space* (i.e., every discrete-valued function can be represented by some decision tree)
- ID3 considers only one hypothesis, and never backtracks
- Search proceeds from shorter trees (with higher infogain features) over longer trees (with lower infogain features)
- This inductive bias is one interpretation of Occam's Razor:

  Pluralitas non est ponenda sine neccesitate.

- But what about *pluralitas cum neccesitate*?

Overfitting

- Even given its bias, ID3 often suffers from overfitting (aka overtraining) :

  Given a hypothesis space H, a hypothesis $h \in H$ is said to *overfit* the training data is there exists some alternative hypothesis $h' \in H$, such that h has a smaller error than h' over the training examples, but h' has a smaller error than h over the entire distribution of instances.

- Overfitting is particular worrisome for models which are not well specified, or for sparse, noisy, non deterministic data (sound familiar?)

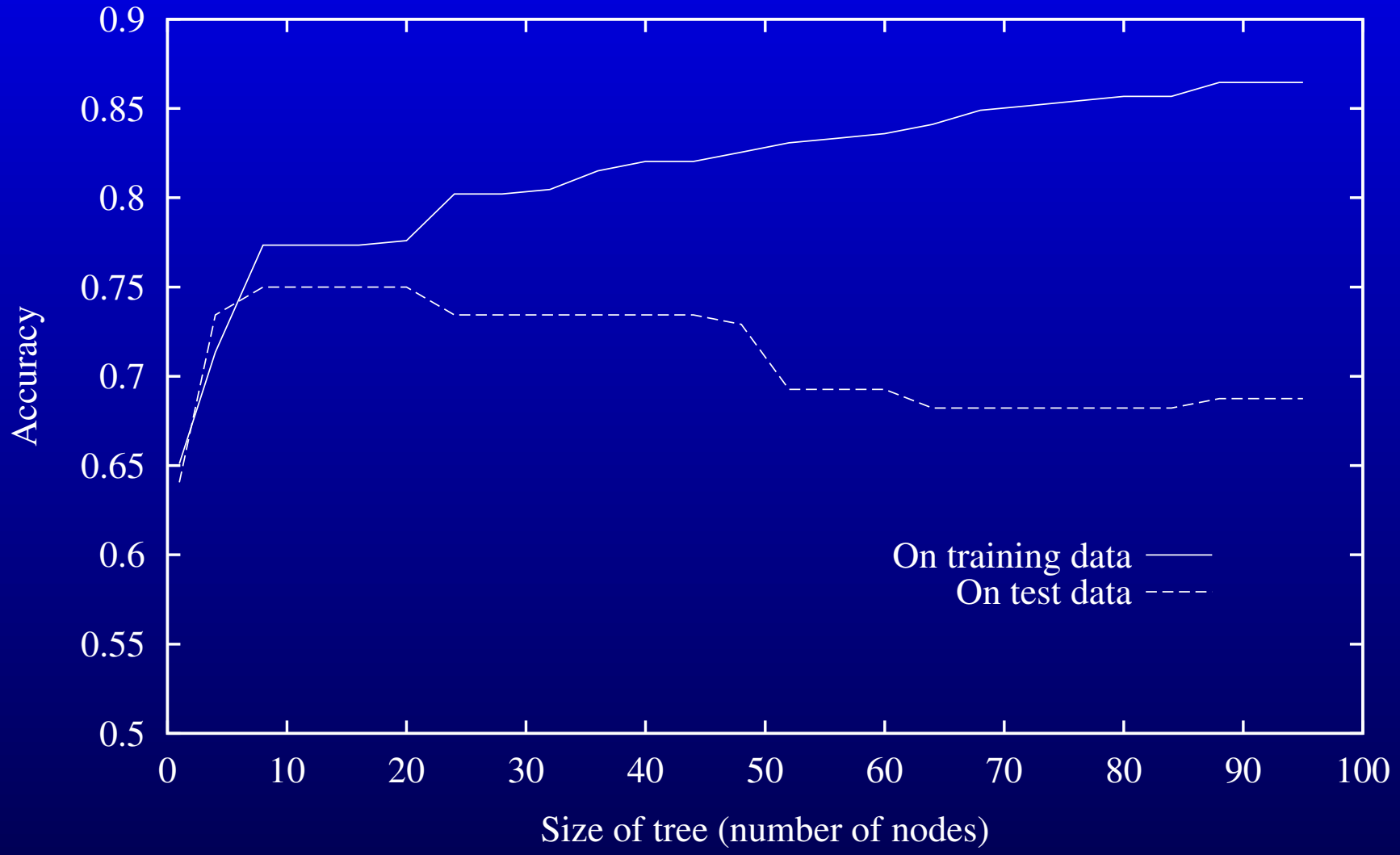
Overfitting

- An example: ten features, which take the value 0 or 1 with equal probability, and two classes $P(\mathit{yes}) = p = 0.75$ and $P(\mathit{no}) = 1 - p = 0.25$
- ID3 produced a tree with 119 nodes and a 35% error rate on test cases
- Always assigning the most frequent class gives an expected error rate of $1 - p = 0.25$
- Assigning *yes* with probability p yields an error rate of:

$$p(1-p) + (1-p)p = 2p(1-p) = 0.375$$

- A stopped clock is right twice a day!

Overfitting



Overfitting

- One way to control overfitting is to constrain the size of the tree
- Cross validation can be used to find the peak of the learning curve
- We make the stopping criterion more liberal (e.g., stop branching when 95% of the instances are in one class)
- We also could build a complete tree, and the prune it, replacing a subtree with a leaf or one of its branches (C4.5)

Reduced error pruning

- As we build the tree, we minimize the error w.r.t. the training data
- To avoid overfitting, we want to minimize expected error
- If e out of n training instances covered by a node are misclassified, the training error is $\frac{e}{n}$
- To get at the expected error rate, we could pretend that this is a sample statistic, and try to estimate its sampling distribution
- The upper limit of the confidence interval for the binomial distribution $U_{CF}(e,n)$ gives a pessimistic estimate of the expected error rate
- Make a post-order traversal of tree, replacing nodes with a child if the expected error rate of the child is no more than that of the node

Gain ratio

- Another weakness in ID3 comes from its use of information gain, which strongly prefers features with more values (e.g., *day*)
- We can normalize information gain by the entropy of the feature (what Quinlan calls *split info*):

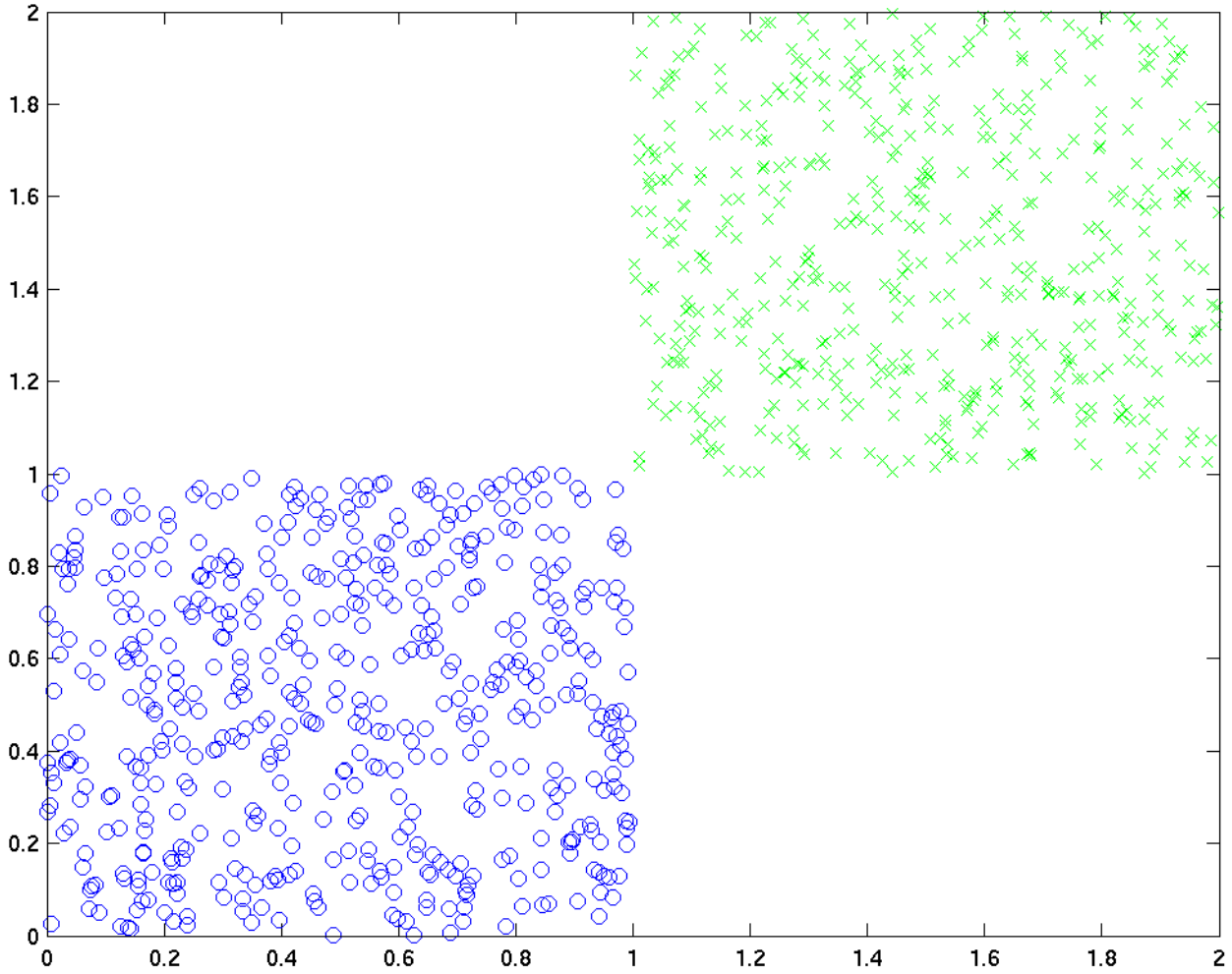
$$H(f) = -\sum_{v} \in P(v) \log_2 P(v)$$

- The *gain ratio* is the normalized information gain:

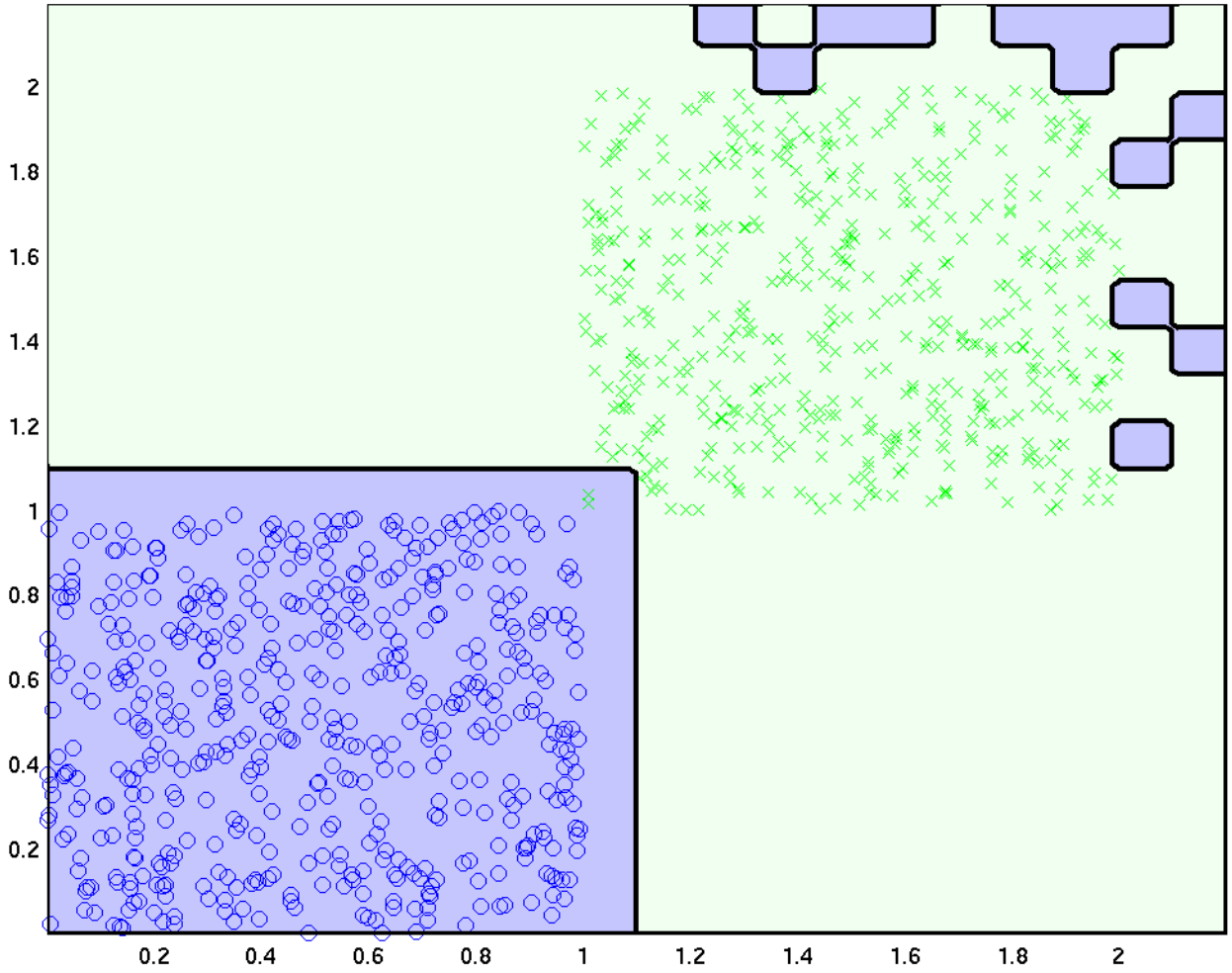
$$GR(T, f) = \frac{G(T, f)}{H(f)}$$

- Experimentally, gain ratio seems to be better than information gain as a measure of the usefulness of a feature to a classification problem

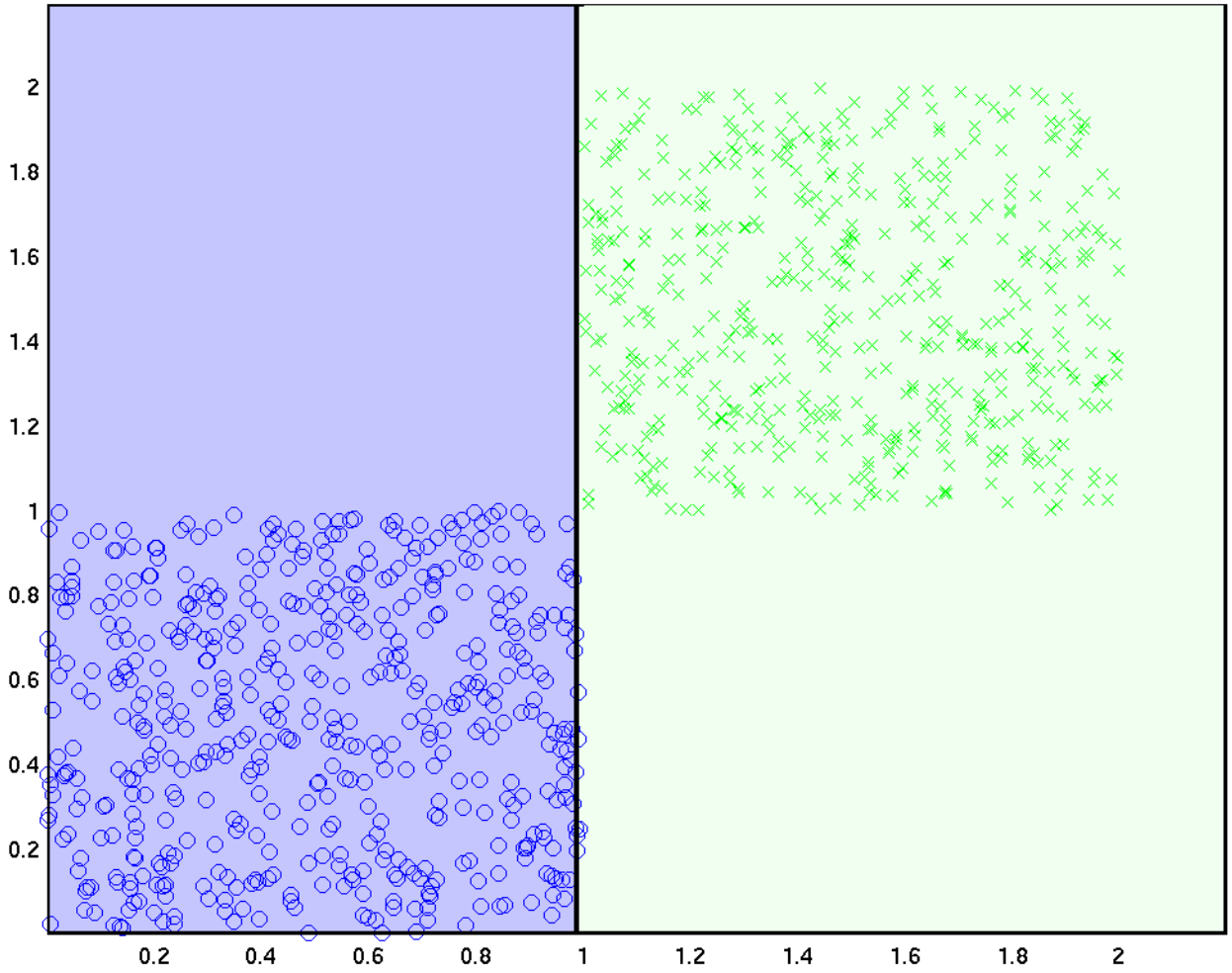
Decision boundaries



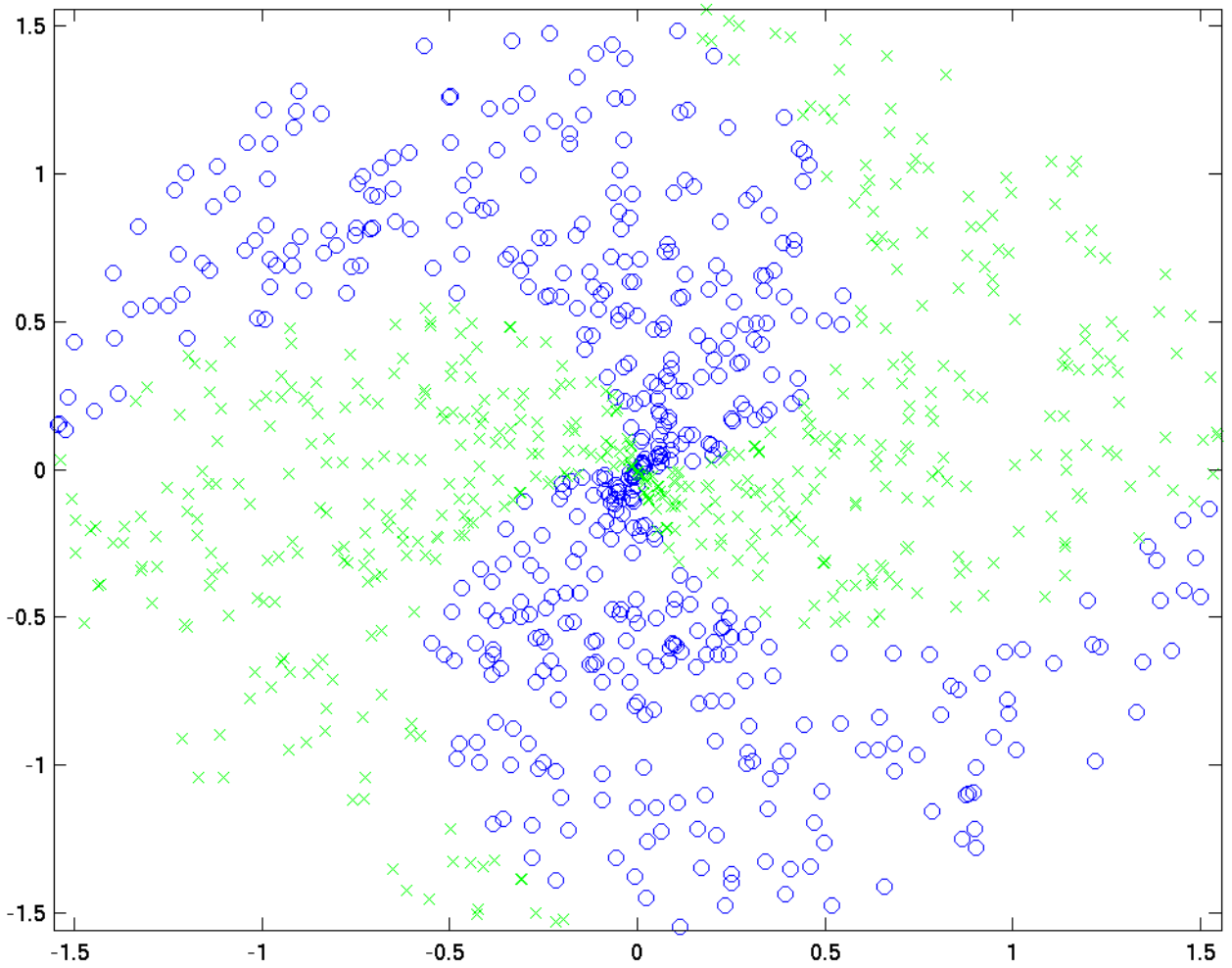
Decision boundaries ID3



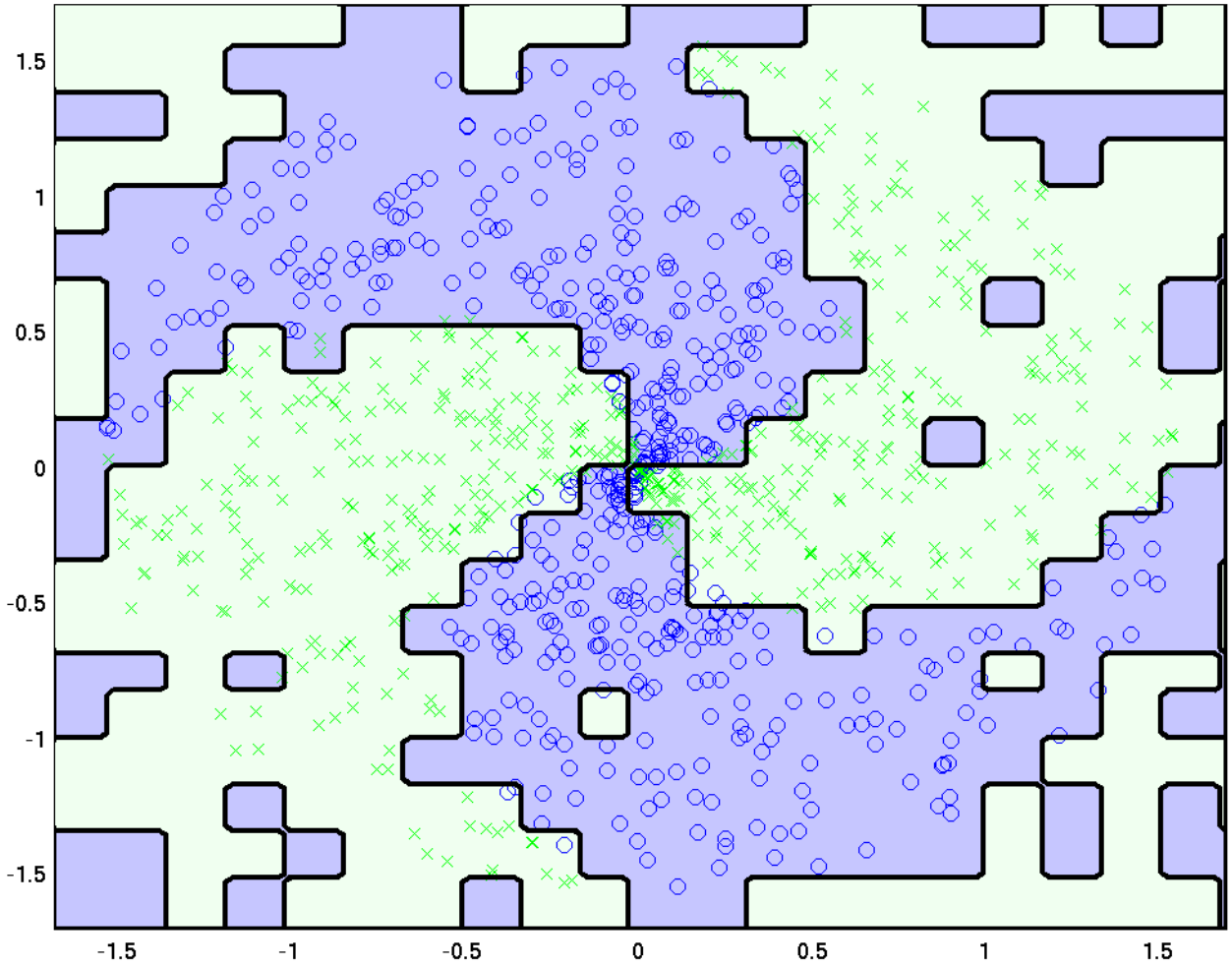
Decision boundaries C4.5



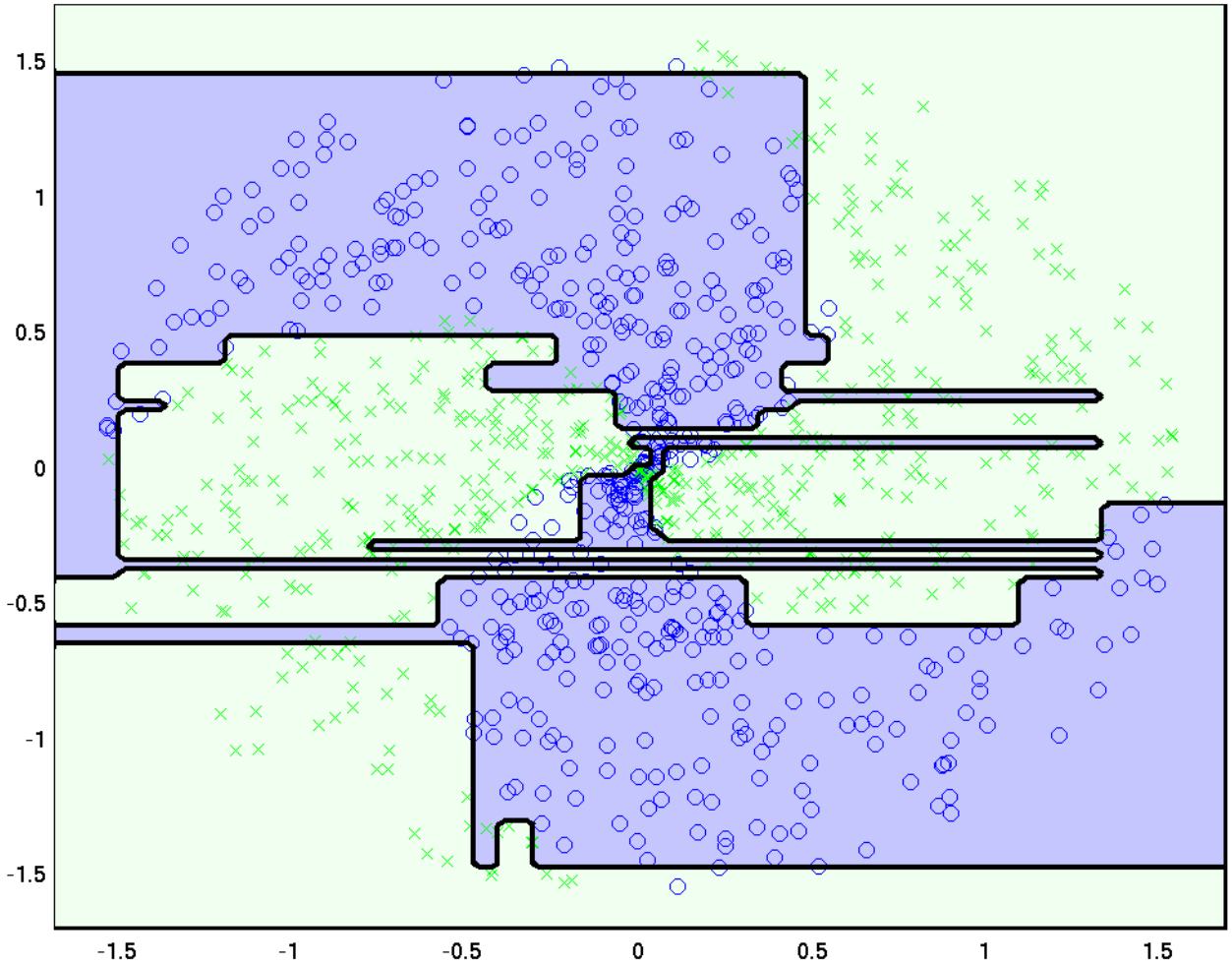
Decision boundaries



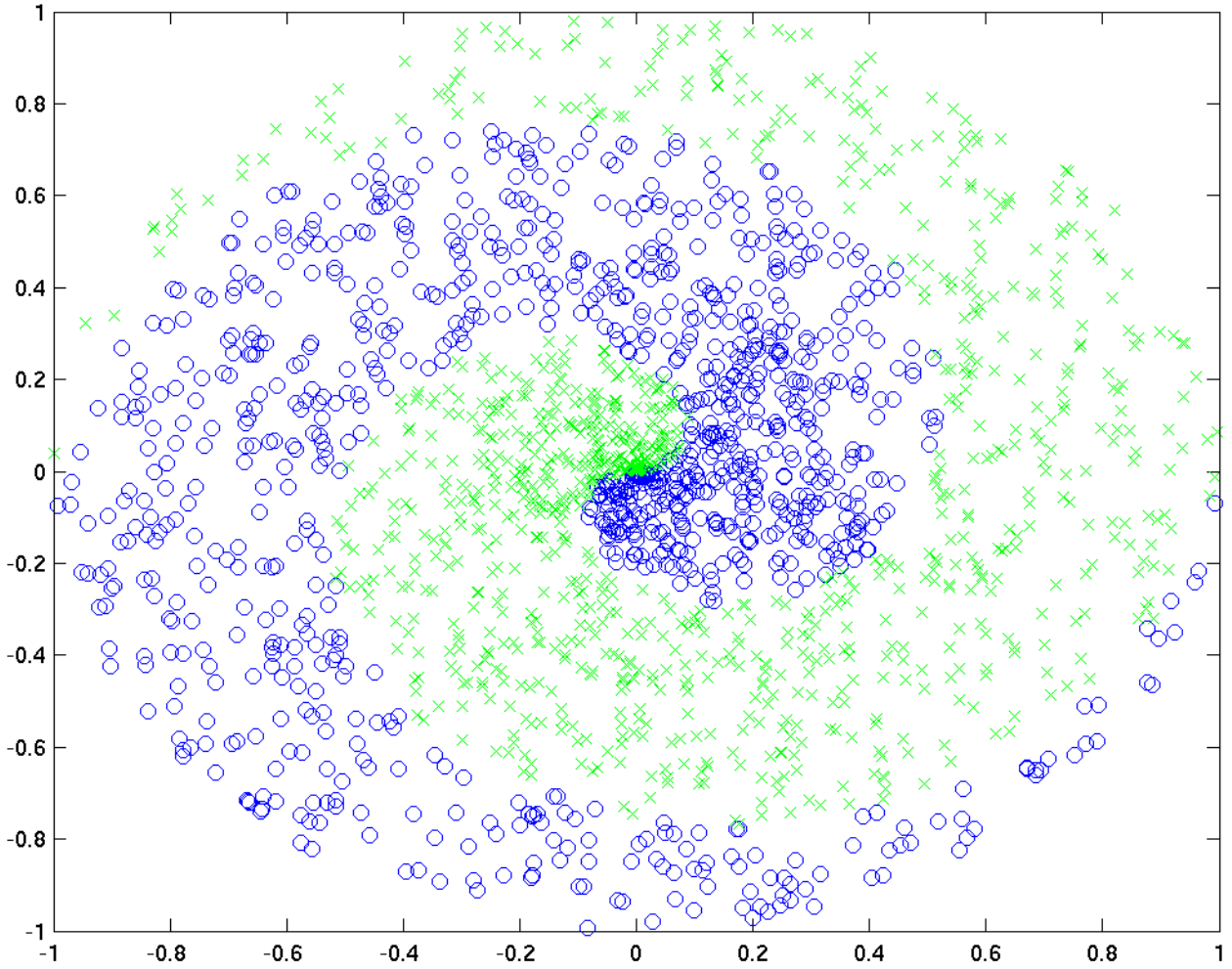
Decision boundaries ID3



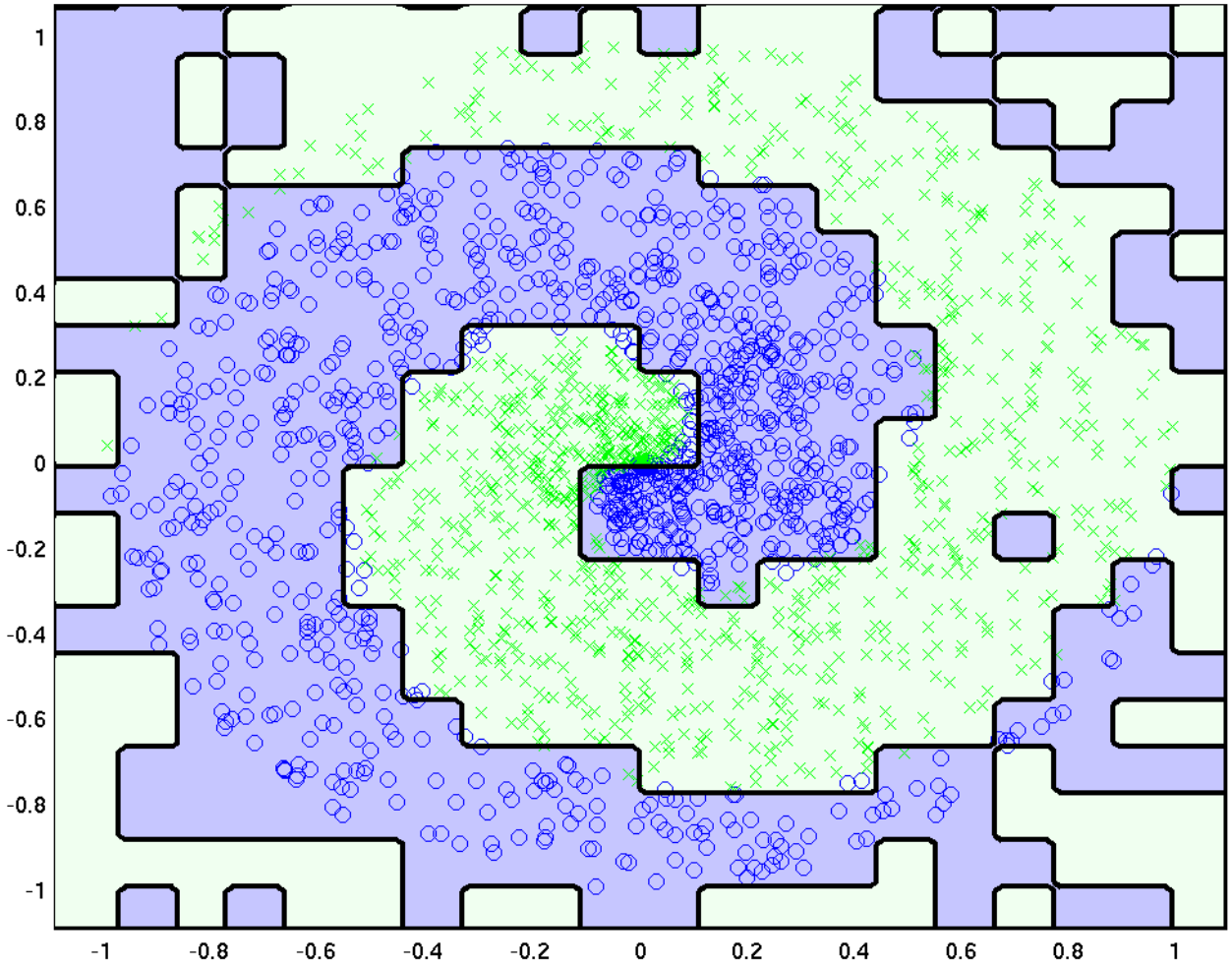
Decision boundaries C4.5



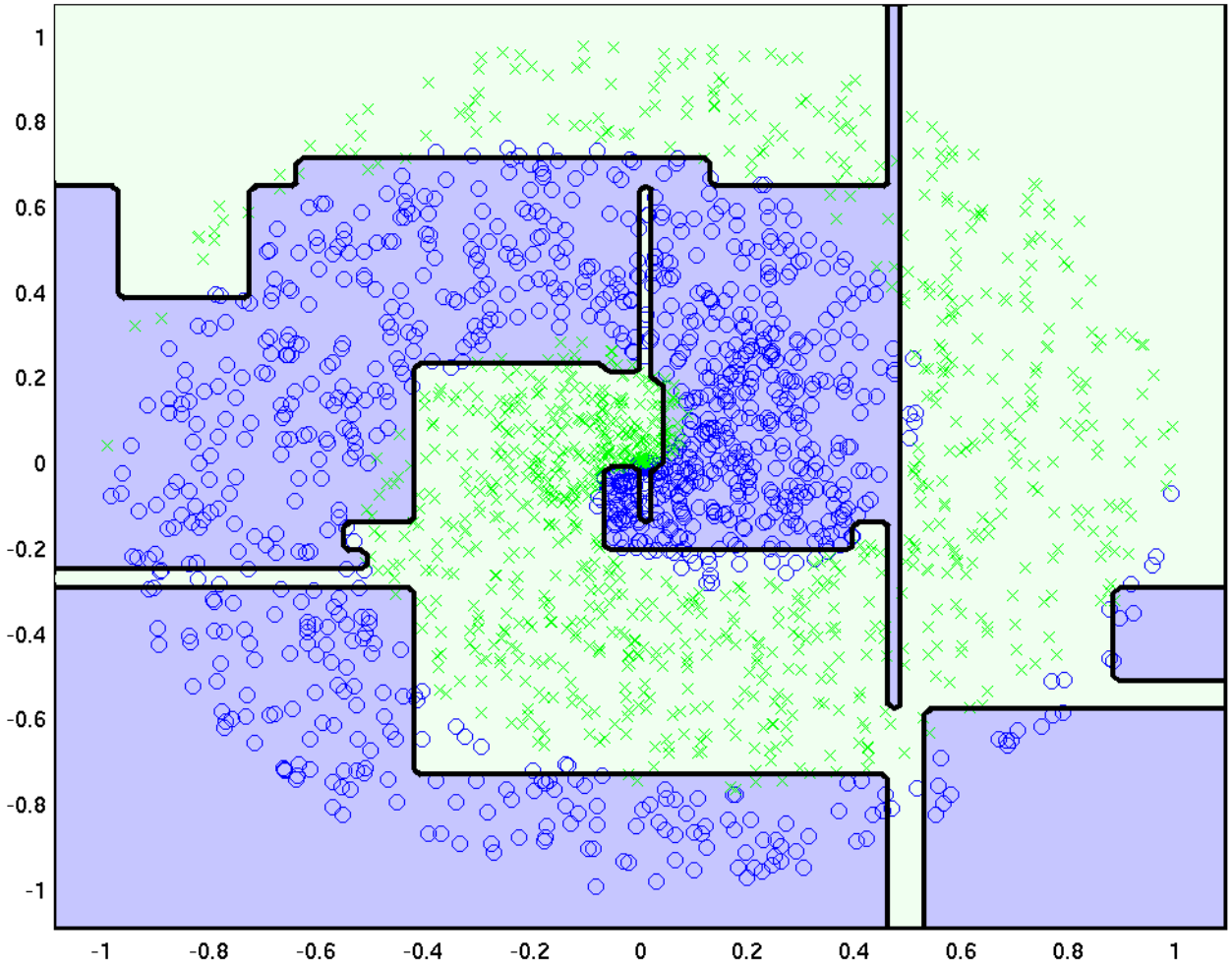
Decision boundaries



Decision boundaries ID3



Decision boundaries C4.5



Real world decision trees

- Decision trees are conceptually simple but, with heuristic add-ons, can be very effective
- Ross Quinlan released a C implementation of C4.5 in the early 1990's
- Both training and application of decision trees is cheap
- This combination has made C4.5 the default machine learning method