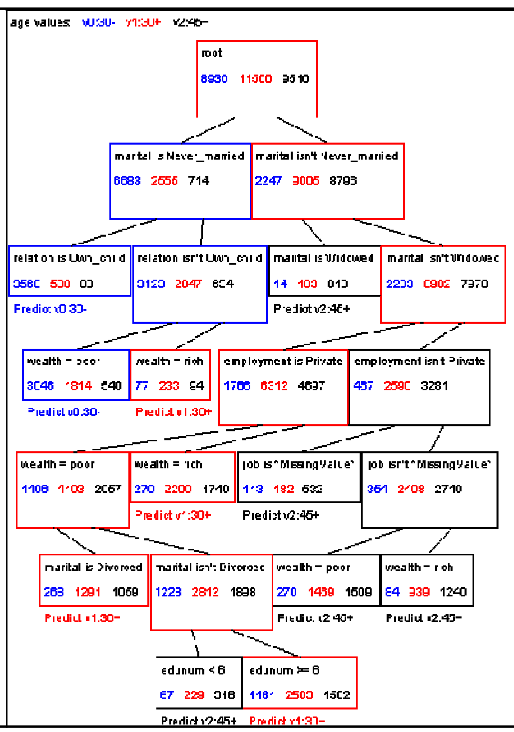
Decision Trees (Cont.)

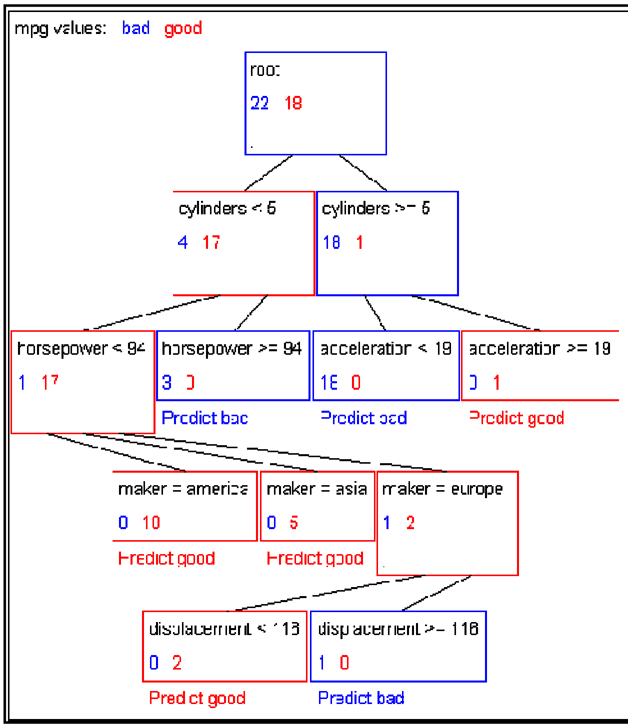
R&N Chapter 18.2,18.3



Side example with discrete (categorical) attributes: Predicting age (3 values: less than 30, 30-45, more than 45 yrs old) from census data.

Attributes (split in that order):

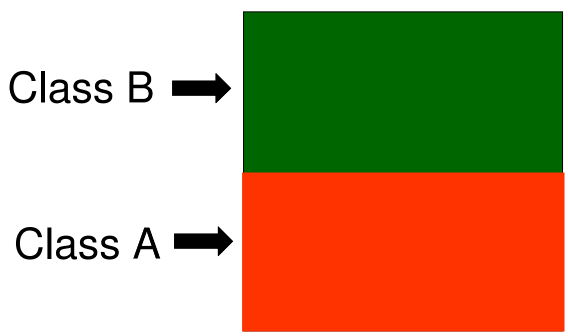
- Married
- Have a child
- Widowed
- Wealth (rich/poor)
- Employment type (private or not private), etc.



Side example with both discrete and continuous attributes: Predicting MPG ('GOOD' or 'BAD') from attributes:

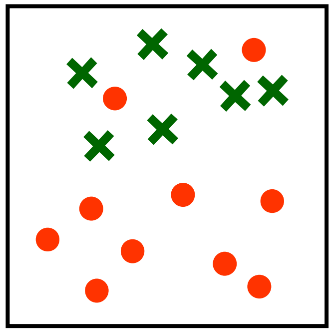
- Cylinders
- Horsepower
- Acceleration
- Maker (discrete)
- Displacement

The Overfitting Problem: Example



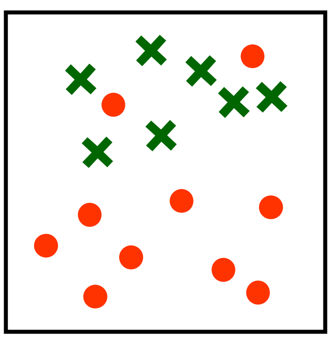
- Suppose that, in an ideal world, class B is everything such that $X_2 >= 0.5$ and class A is everything with $X_2 < 0.5$
- Note that attribute X_1 is irrelevant
- Seems like generating a decision tree would be trivial

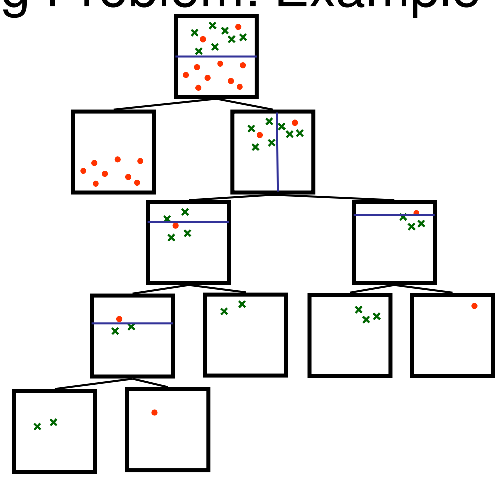
The Overfitting Problem: Example



- However, we collect training examples from the perfect world through some imperfect observation device
- As a result, the training data is corrupted by *noise*.

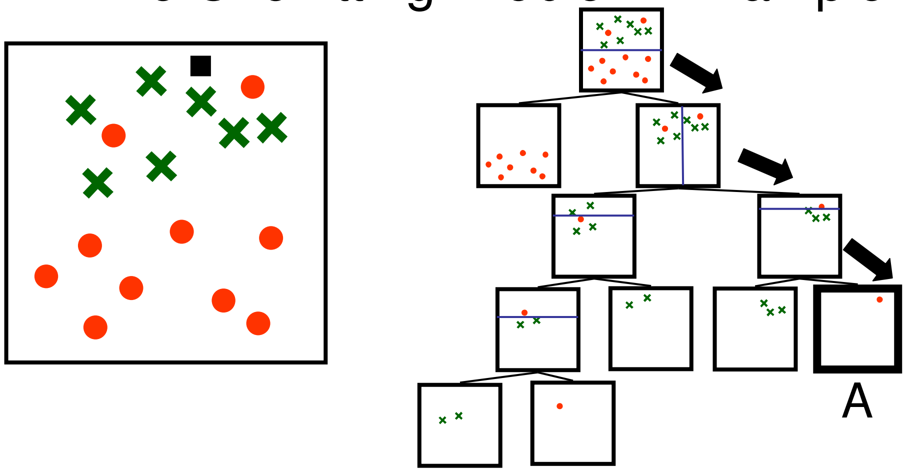
The Overfitting Problem: Example





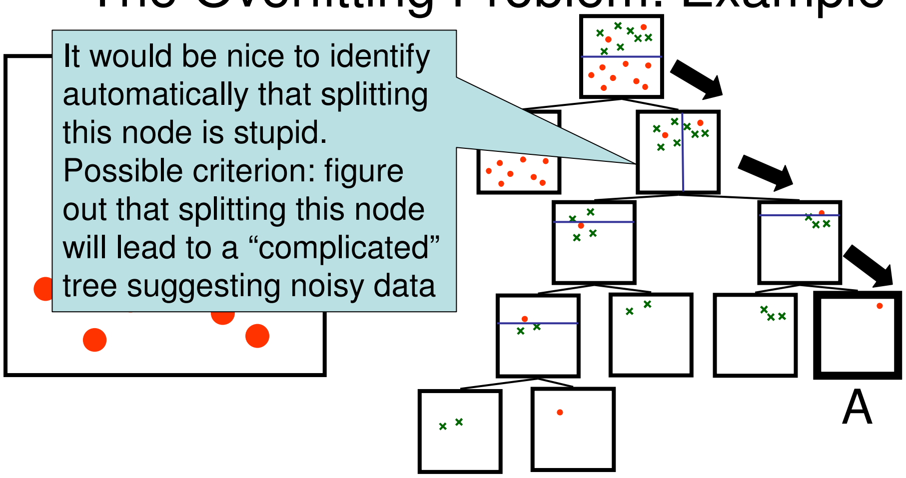
- Because of the noise, the resulting decision tree is far more complicated than it should be
- This is because the learning algorithm tries to classify *all of the training set perfectly* → This is a fundamental problem in learning: *overfitting*

The Overfitting Problem: Example



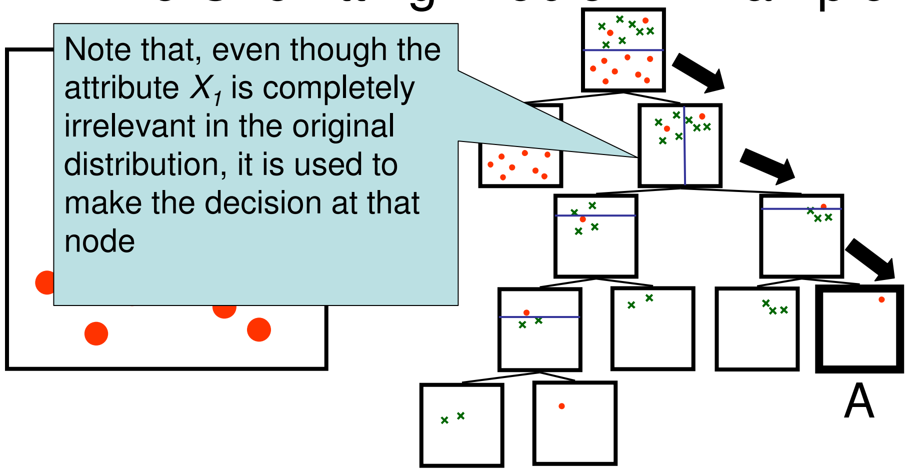
- The effect of overfitting is that the tree is guaranteed to classify the training data perfectly, but it may do a terrible job at classifying new test data.
- Example: (0.6,0.9) is classified as ‘A’

The Overfitting Problem: Example



- The effect of overfitting is that the tree is guaranteed to classify the training data perfectly, but it may do a terrible job at classifying new test data.
- Example: (0.6,0.9) is classified as ‘A’

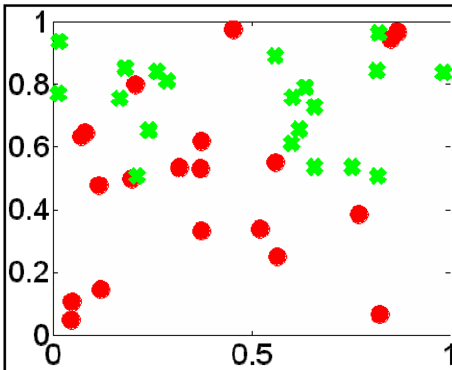
The Overfitting Problem: Example



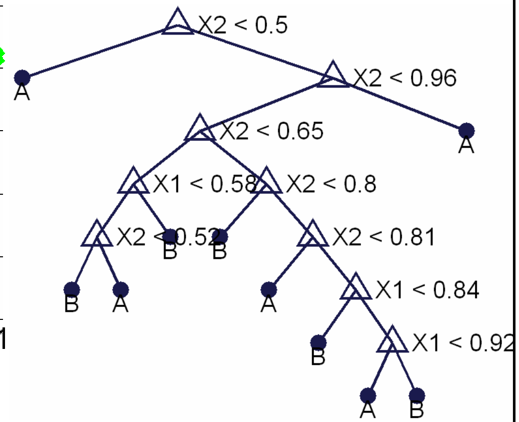
- The effect of overfitting is that the tree is guaranteed to classify the training data perfectly, but it may do a terrible job at classifying new test data.
- Example: (0.6,0.9) is classified as ‘A’

Possible Overfitting Solutions

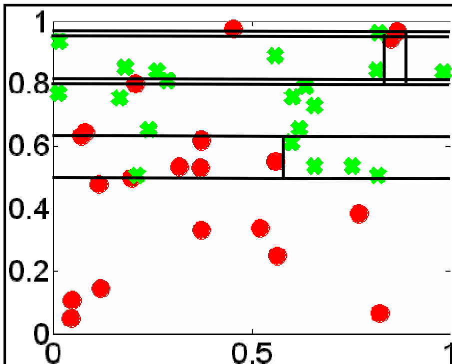
- Grow tree based on training data (*unpruned* tree)
- Prune the tree by removing useless nodes based on:
  - Additional test data (not used for training)
  - Statistical significance tests



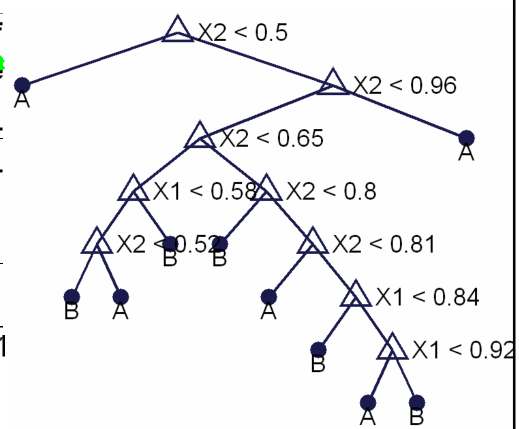
Training Data



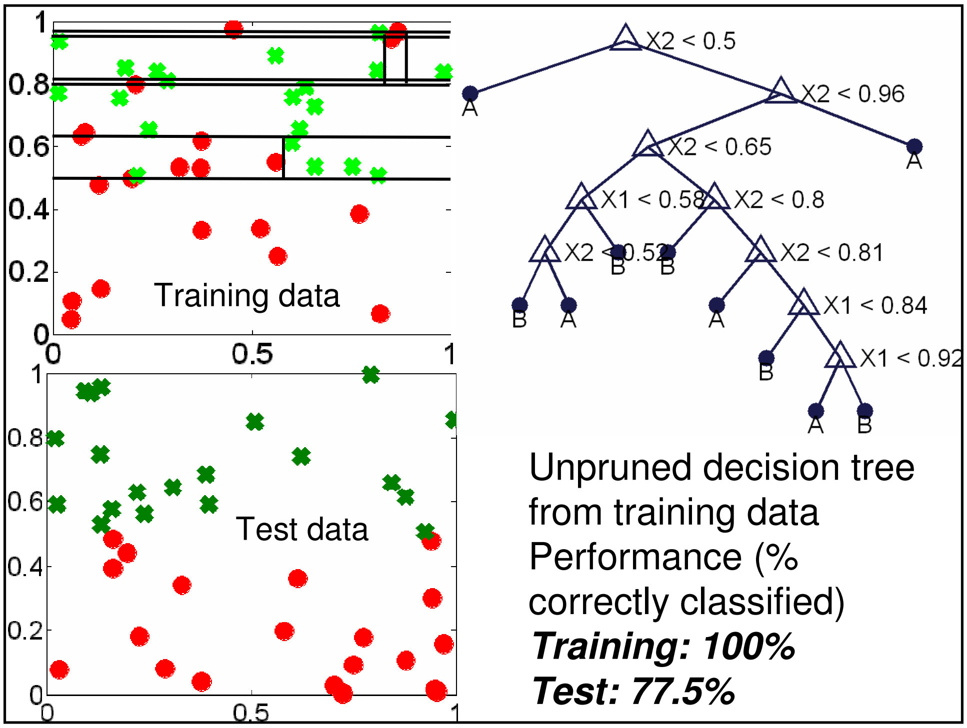
Unpruned decision tree from training data



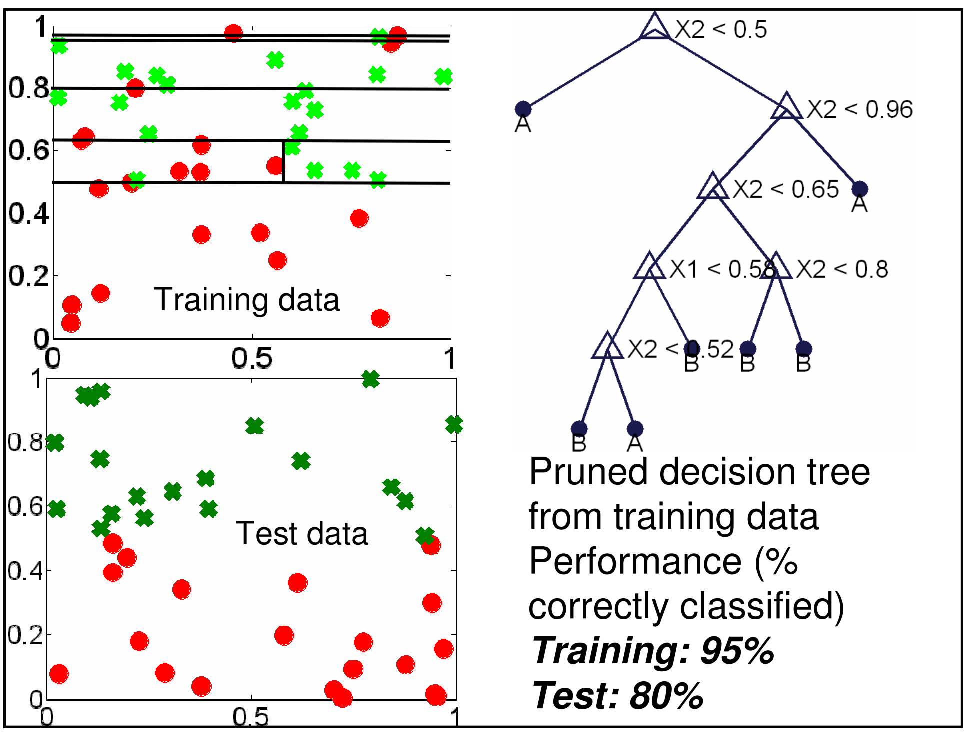
Training data with the partitions induced by the decision tree (Notice the tiny regions at the top necessary to correctly classify the 'A' outliers!)



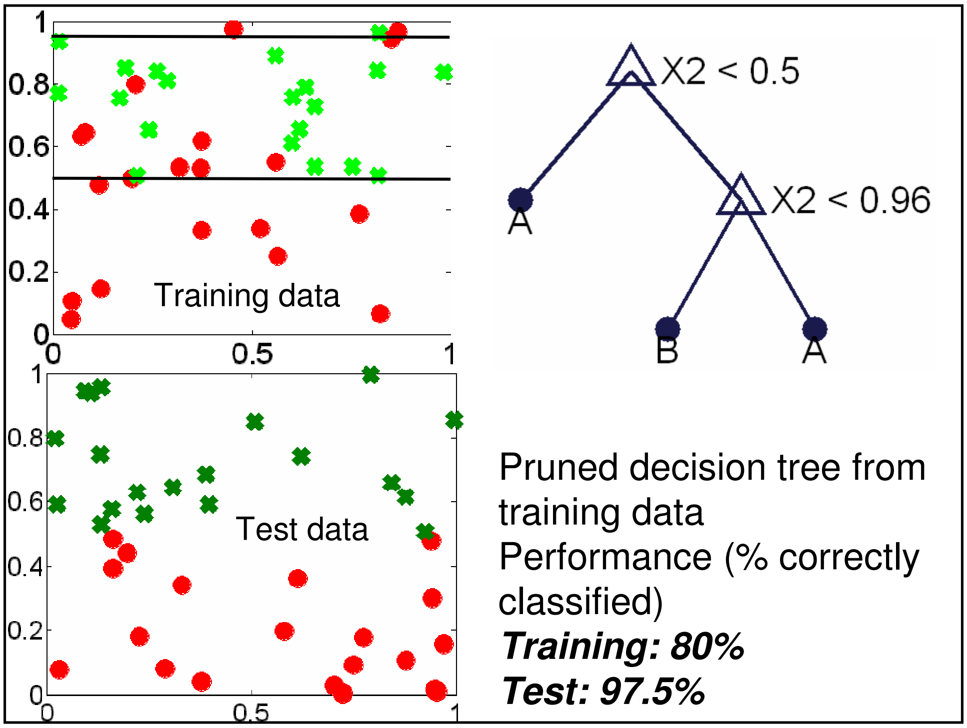
Unpruned decision tree from training data



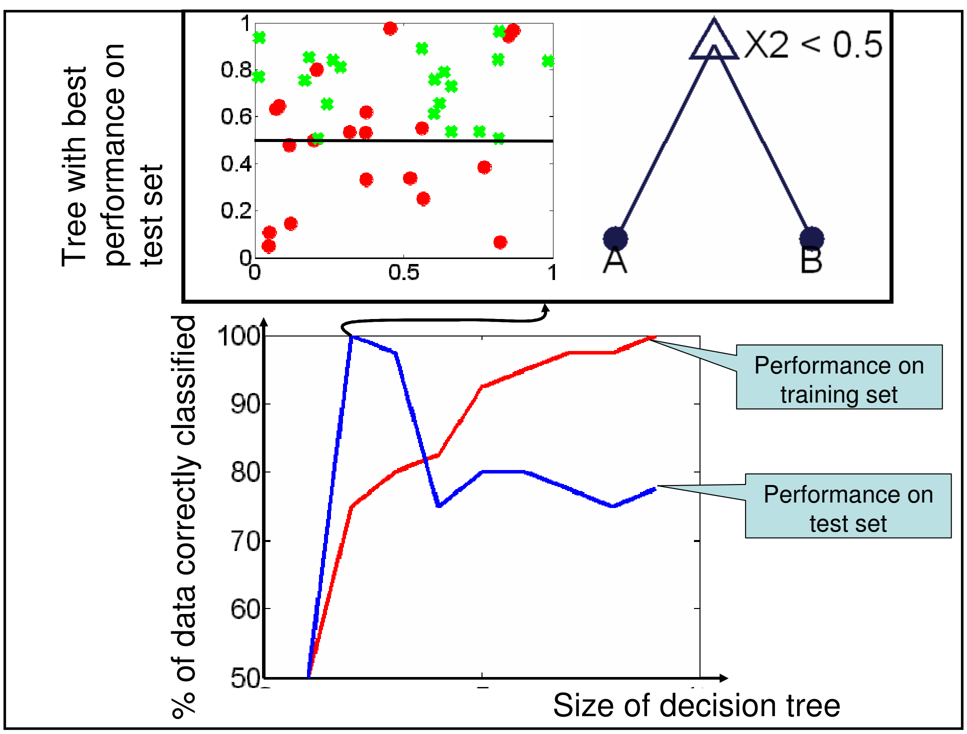
Unpruned decision tree from training data
Performance (% correctly classified)
Training: 100%
Test: 77.5%



Pruned decision tree from training data
Performance (% correctly classified)
Training: 95%
Test: 80%

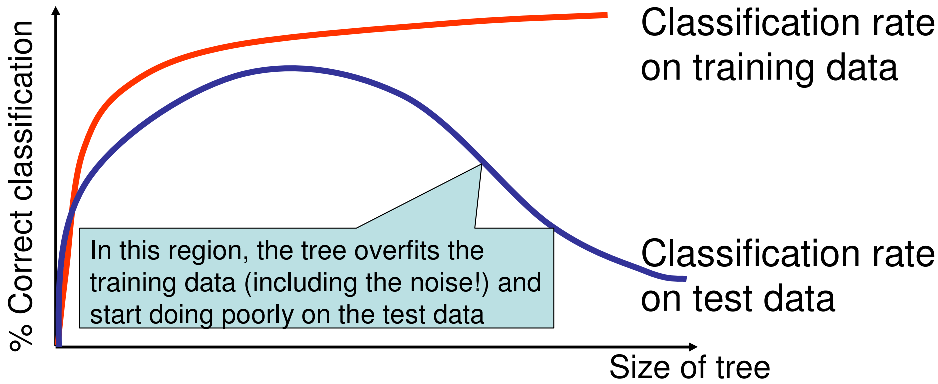


Training data
Test data
X2 < 0.5
X2 < 0.96
A
B
A
Pruned decision tree from training data
Performance (% correctly classified)
Training: 80%
Test: 97.5%



Tree with best performance on test set
X2 < 0.5
A
B
% of data correctly classified
Size of decision tree
Performance on training set
Performance on test set

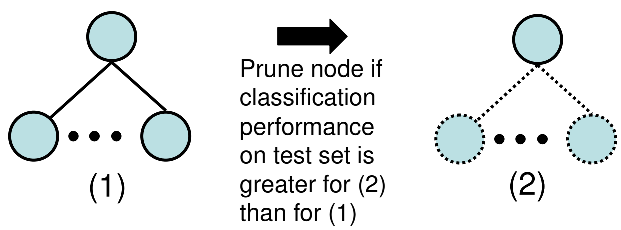
Using Test Data



- General principle: As the complexity of the classifier increases (depth of the decision tree), the performance on the training data increases and the performance on the test data decreases when the classifier overfits the training data.

Decision Tree Pruning

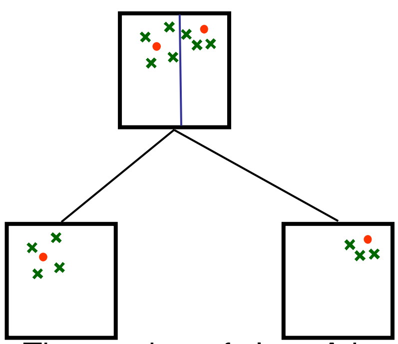
- Construct the entire tree as before
- Starting at the leaves, recursively eliminate splits:
  - Evaluate performance of the tree on test data (also called validation data, or hold out data set)
  - Prune the tree if the classification performance increases by removing the split



Possible Overfitting Solutions

- Grow tree based on training data (*unpruned* tree)
- Prune the tree by removing useless nodes based on:
  – Additional test data (not used for training)
  – Statistical significance tests

A Criterion to Detect Useless Splits



• The number of class A in the root node is $N_A = 2$

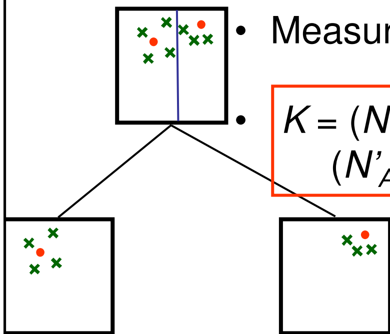
• The number of class B in the root node is $N_B = 7$

• The number of class A in the left node is $N_{AL} = 1$

• The number of class B in the left node is $N_{BL} = 4$

- The problem is that we split whenever the IG increases, but we never check if the change in entropy is *statistically significant*
- *Reasoning:*
- The proportion of the data going to the left node is $p_L = (N_{AL} + N_{BL})/(N_A+N_B) = 5/9$
- Suppose now that the data is *completely randomly* distributed (i.e., it does not make sense to split):
- The expected number of class A in the left node would be $N'_{AL} = N_A \times p_L = 10/9$
- The expected number of class B in the left node would be $N'_{BL} = N_B \times p_L = 35/9$
- *Question:*
- Are N_A and N_B sufficiently different from N'_A and N'_B. *If not, it means that the split is not statistically significant and we should not split the root* → The resulting children are not significantly different from what we would get by splitting a random distribution at the root node.

A Criterion to Detect Useless Splits



- Measure of statistically significance:
- $K = (N'_{AL} - N_{AL})^2/N'_{AL} + (N'_{BL} - N_{BL})^2/N'_{BL} + (N'_{AR} - N_{AR})^2/N'_{AR} + (N'_{BR} - N_{BR})^2/N'_{BR}$

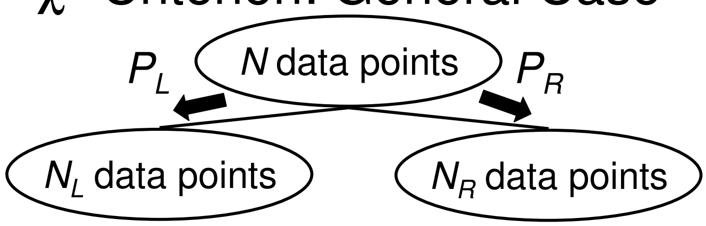
K measures how much the split deviates from what we would get if the data where random

K small → The increase in IG of the split is not significant

In this example: $K =$

$(10/9 – 1)^2/(10/9) + (35/9 – 4)^2/(35/9) + \ldots = 0.0321$

χ^2 Criterion: General Case



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

- N_{ij} = Number of points from class i in child j
- N'_{ij} = Number of points from class i in child j *assuming a random selection*
- $N'_{ij} = N_i \times P_j$

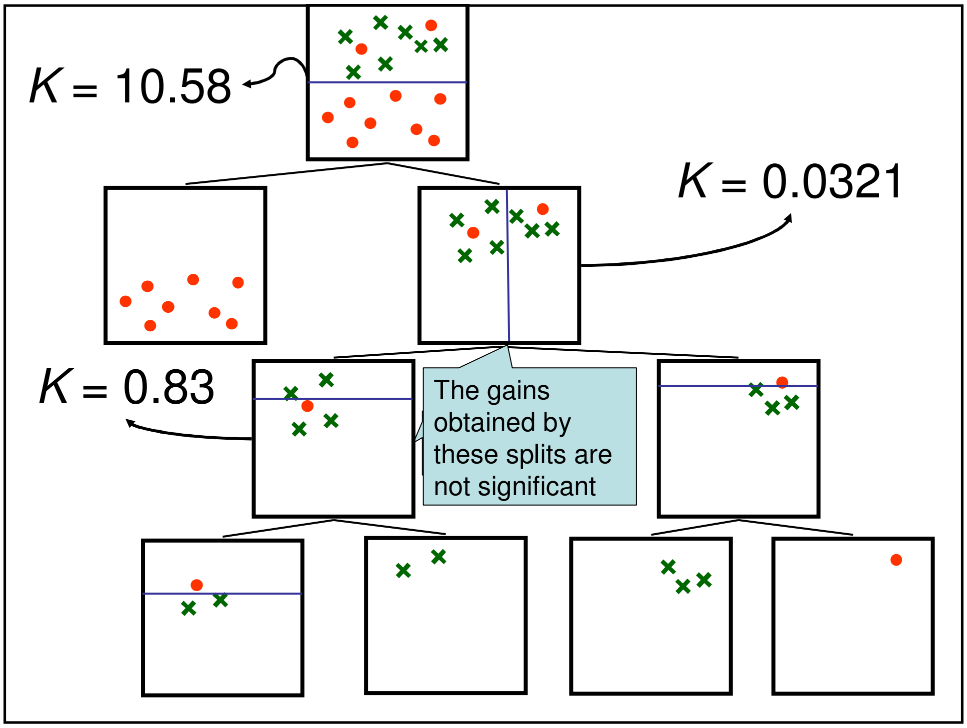
Small (Chi-square) values indicate low statistical significance → Remove the splits that are lower than a threshold $K < t$.
Lower t → bigger trees (more overfitting).
Larger t → smaller trees (less overfitting, but worse classification error).

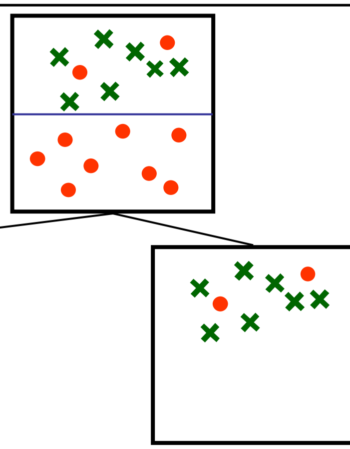
Difference between the distribution of class i from the proposed split and the distribution from randomly drawing data points in the same proportions as the proposed split

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

Decision Tree Pruning

- Construct the entire tree as before
- Starting at the leaves, recursively eliminate splits:
  - At a leaf $\mathcal{N}$:
    - Compute the K value for $\mathcal{N}$ and its parent $\mathcal{P}$.
    - If the K values is lower than the threshold t:
      - Eliminate all of the children of $\mathcal{P}$
      - $\mathcal{P}$ becomes a leaf
  - Repeat until no more splits can be eliminated





- By thresholding K we end up with the decision tree that we would expect (i.e., one that does not overfit the data)
- Note: The approach is presented with continuous attributes in this example but it works just as well with discrete attributes

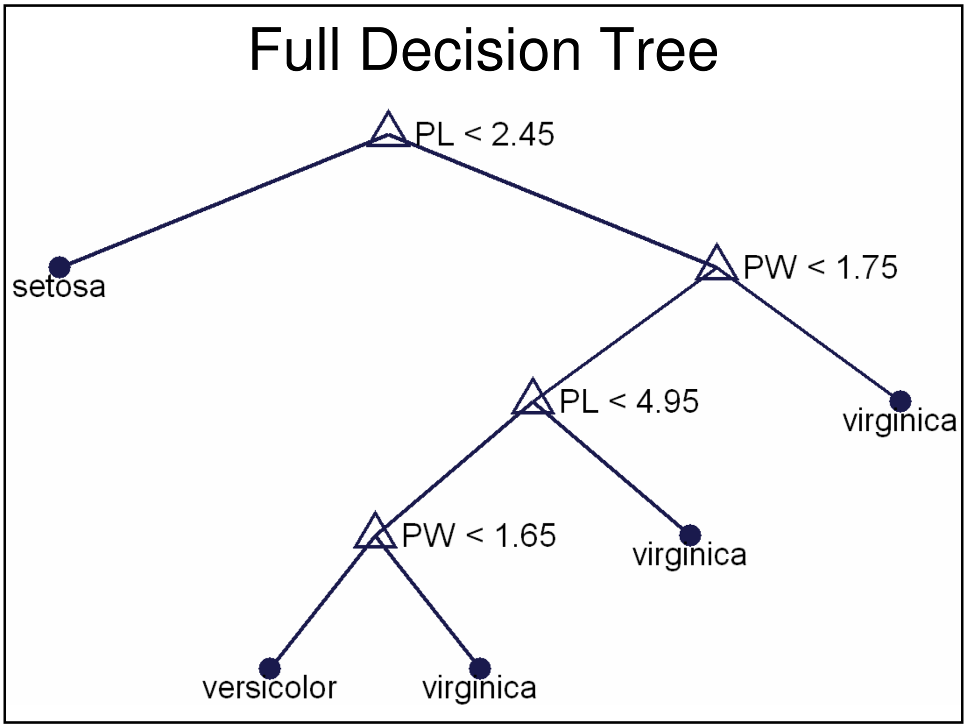
χ^2 Pruning

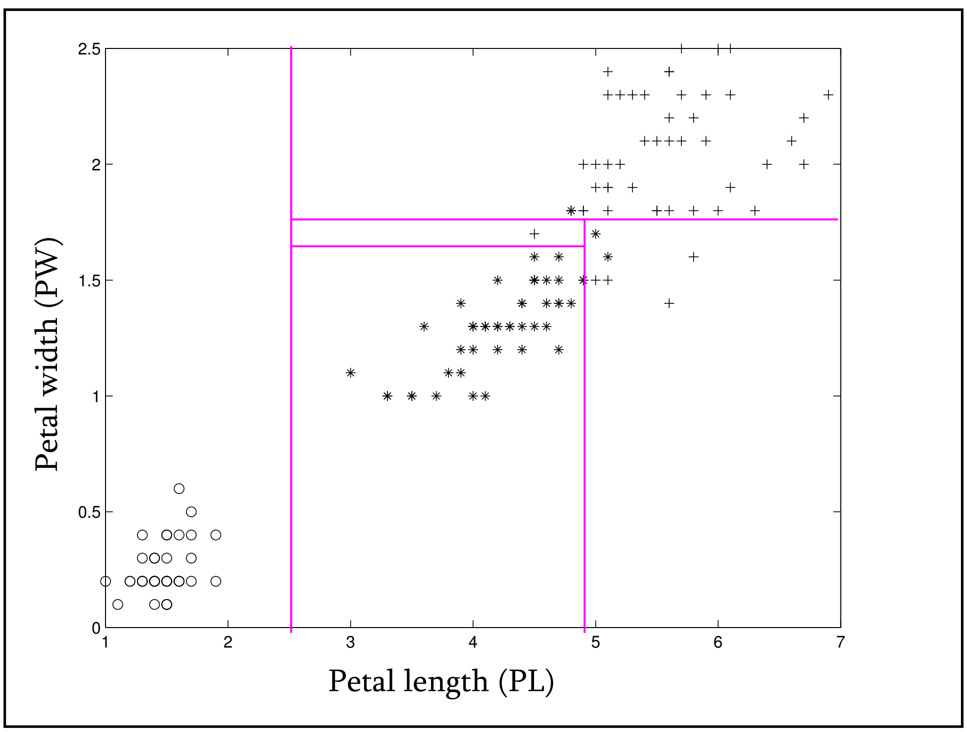
- The test on K is a version of a standard statistical test, the χ^2 ('chi-square') test.
- The value of t is retrieved from statistical tables. For example, $K > t$ means that, with confidence 95%, the information gain due to the split is significant.
- If $K < t$, with high confidence, the information gain will be 0 over very large training samples
  - Reduces *overfitting*
  - Eliminates *irrelevant attributes*

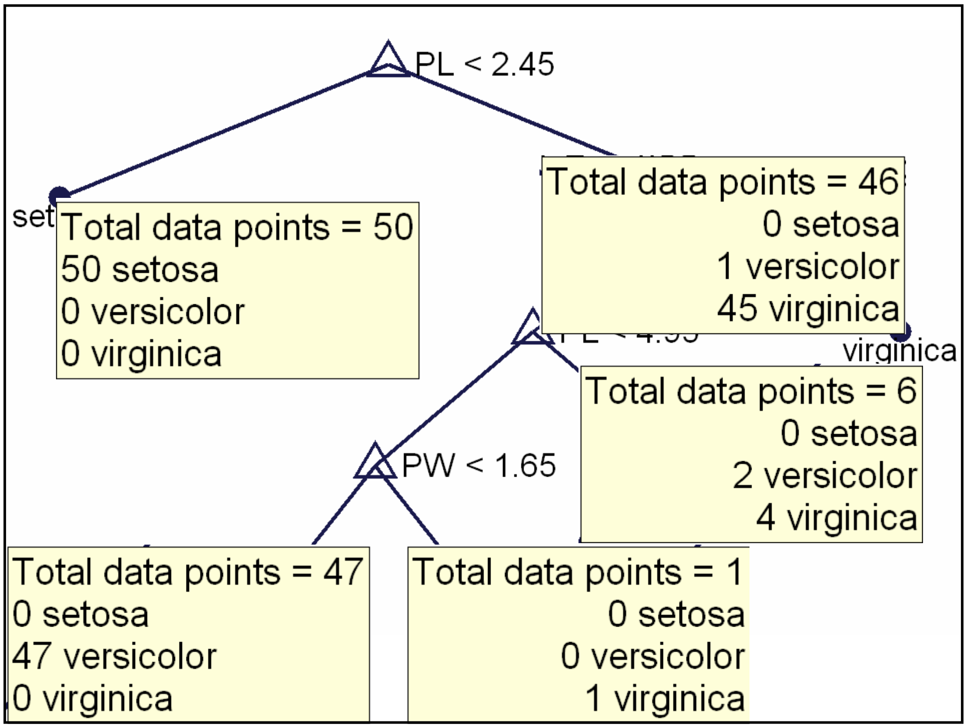
Example

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

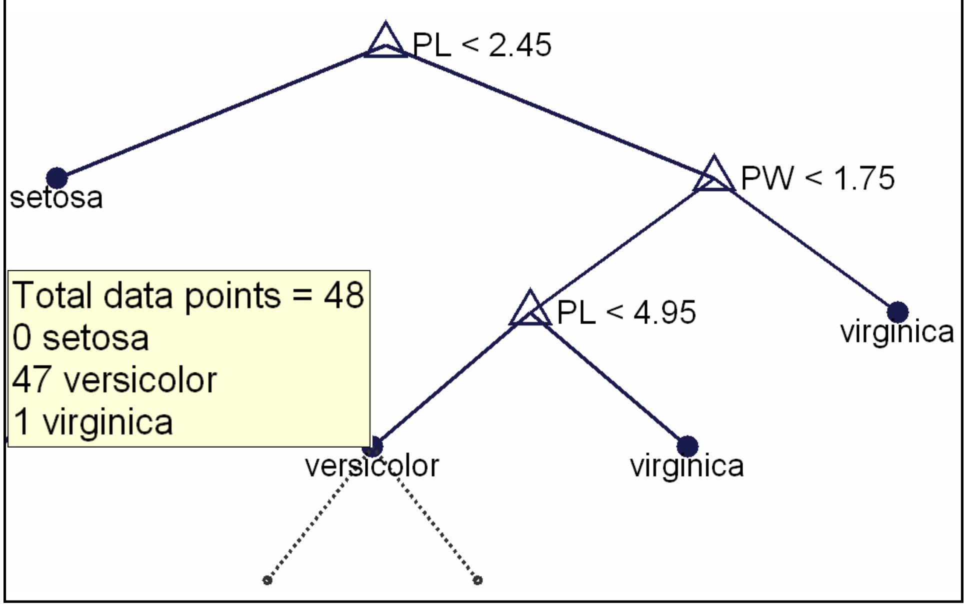
● ● ● ● ● 50 examples from each class ● ● ● ● ●


Full Decision Tree
PL < 2.45
setosa
PW < 1.75
PL < 4.95
virginica
PW < 1.65
virginica
versicolor
virginica

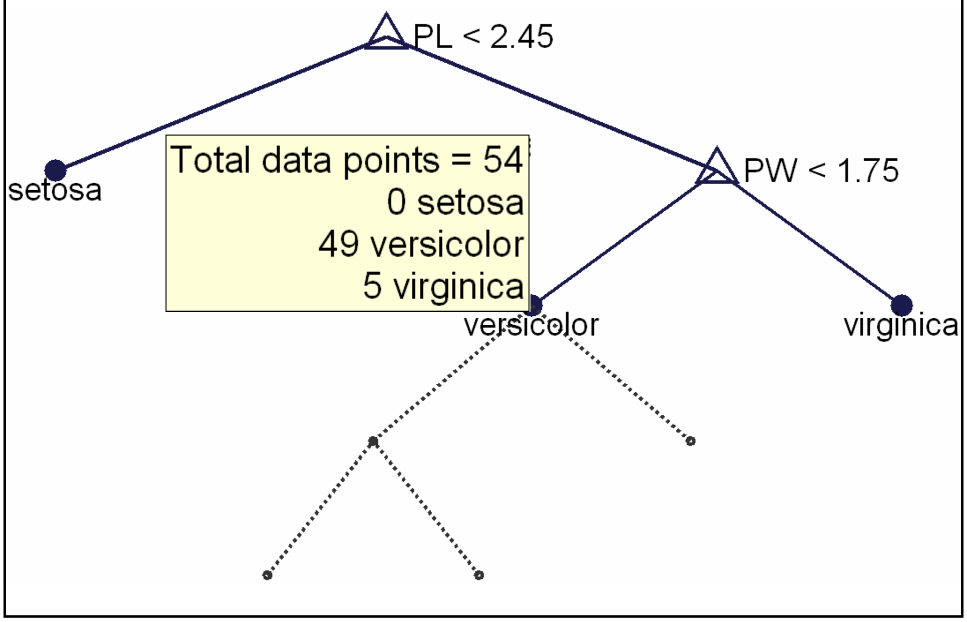

Petal width (PW)
Petal length (PL)

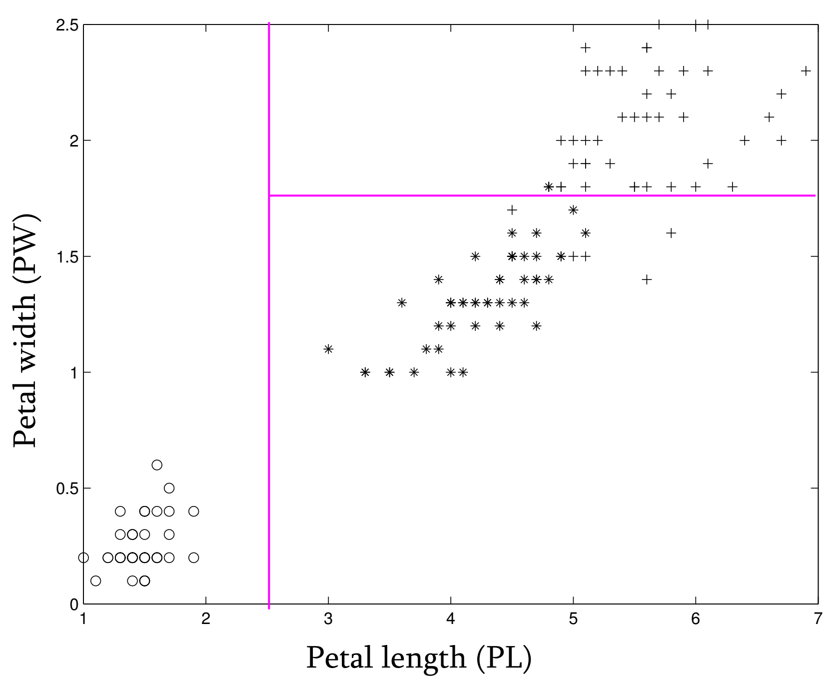


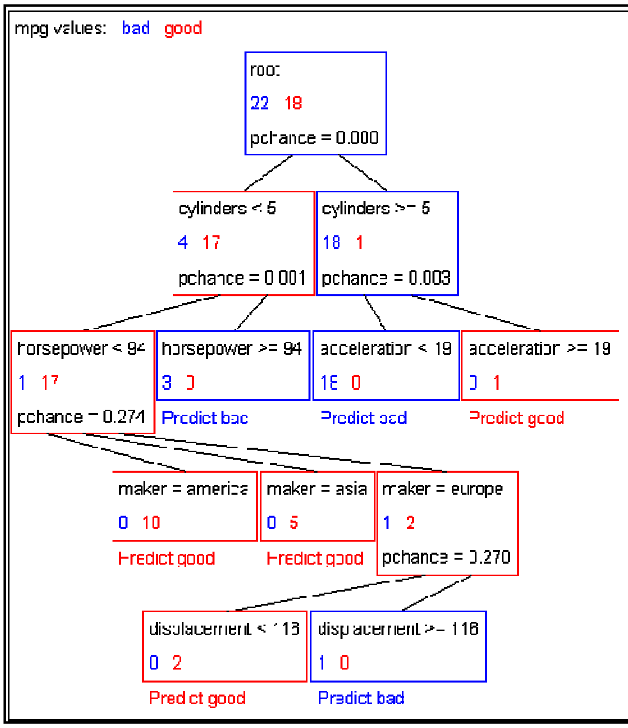
Pruning One Level



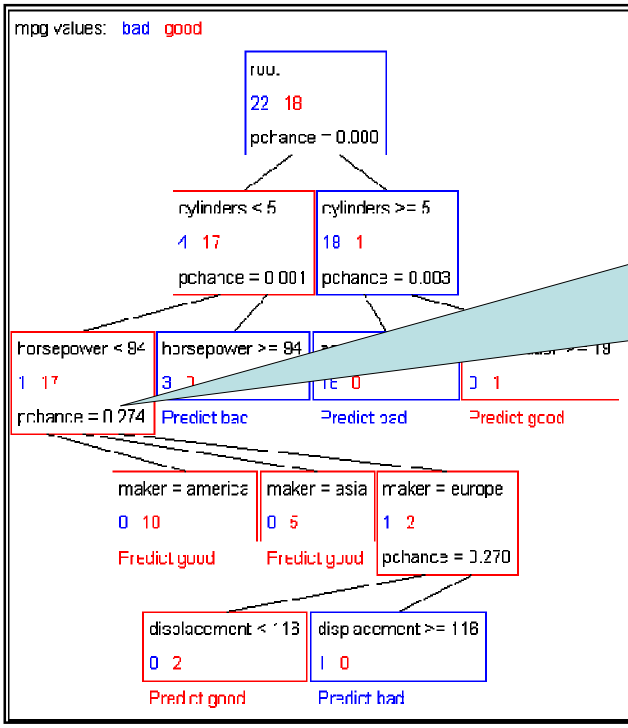
Pruning Two Levels







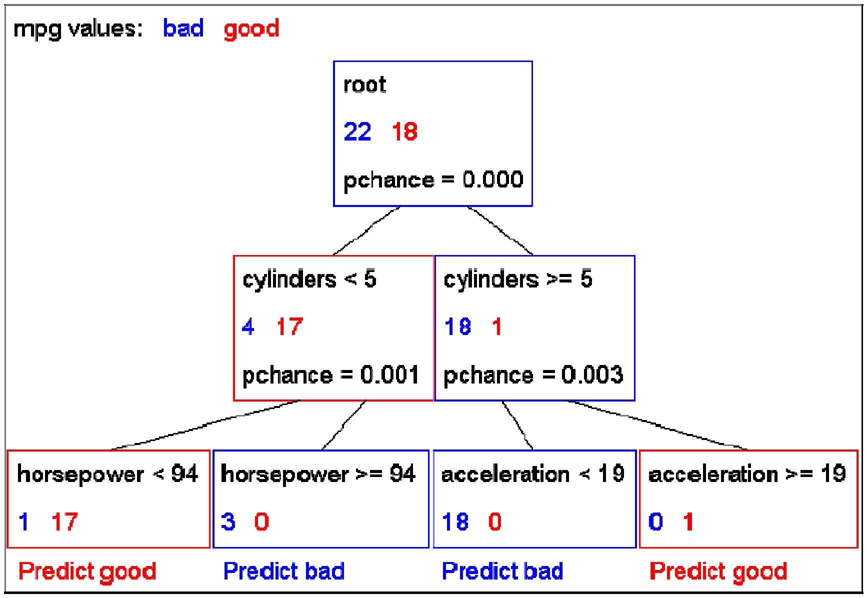
Unpruned



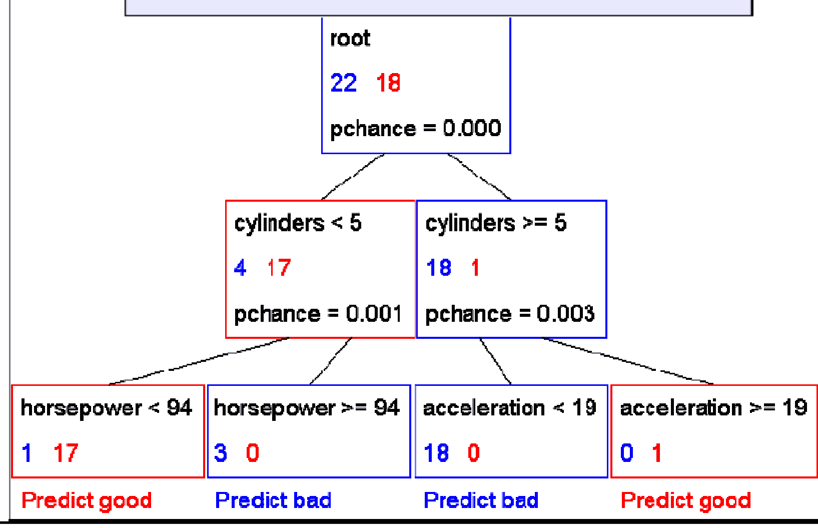
Unpruned

27% probability that this is a “chance” node according to χ^2 test.
Node should be pruned.

Pruned



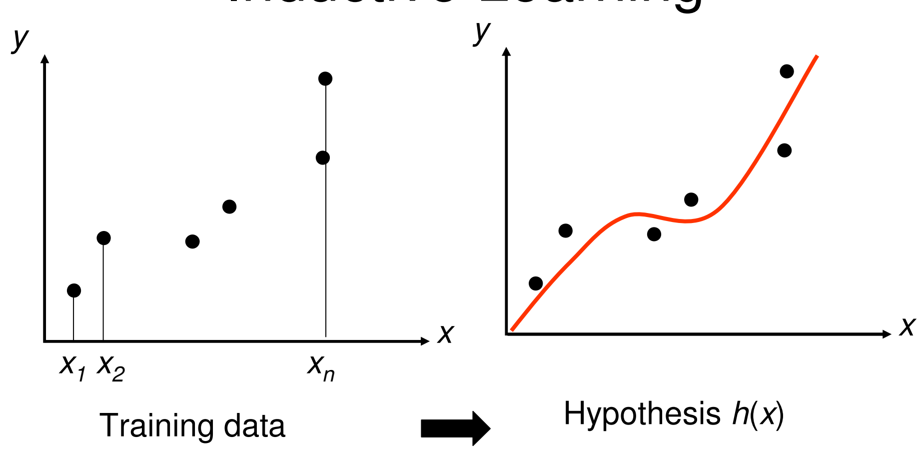
	Num Errors	Set Size	Percent Wrong
Training Set	1	40	2.50
Test Set	53	352	15.06



Note: Inductive Learning

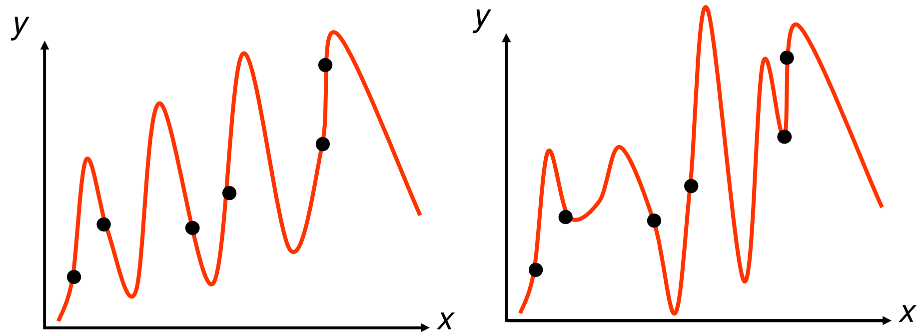
- The decision tree approach is one example of an inductive learning technique:
- Suppose that data x is related to output y by a unknown function $y = f(x)$
- Suppose that we have observed training examples $\{(x_1,y_1),..,(x_n,y_n)\}$
- *Inductive learning problem*: Recover a function h (the "hypothesis") such that $h(x) \approx f(x)$
- $y = h(x)$ predicts y from the input data x
- *The challenge*: The hypothesis space (the space of all hypothesis h of a given form; for example the space of all of the possible decision trees for a set of M attributes) is huge + many different hypotheses may agree with the training data.

Inductive Learning



- What property should h have?
- It should agree with the training data…

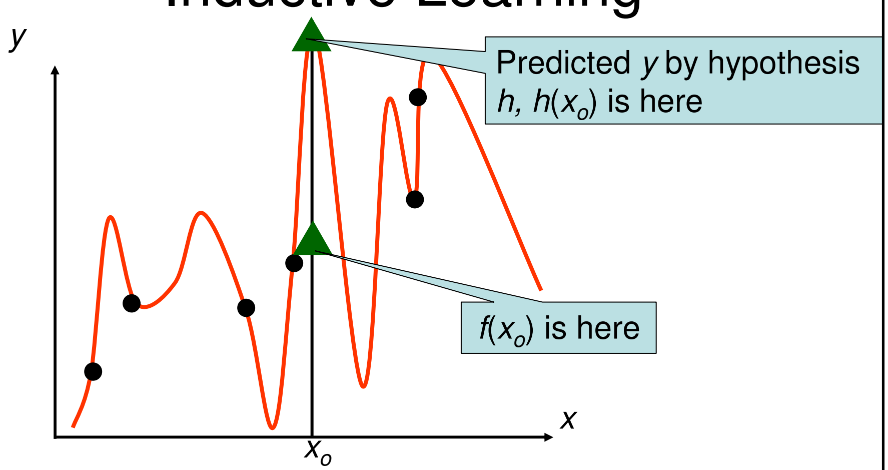
Inductive Learning



Two stupid hypotheses that fit the training data perfectly

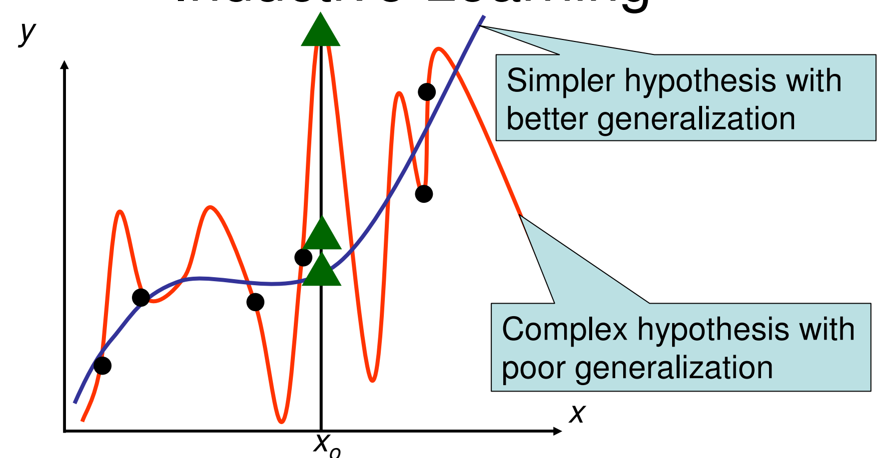
- What property should *h* have?
- It should agree with the training data…
- But that can lead to arbitrarily complex hypotheses and there are many of them; which one should we choose?…

Inductive Learning



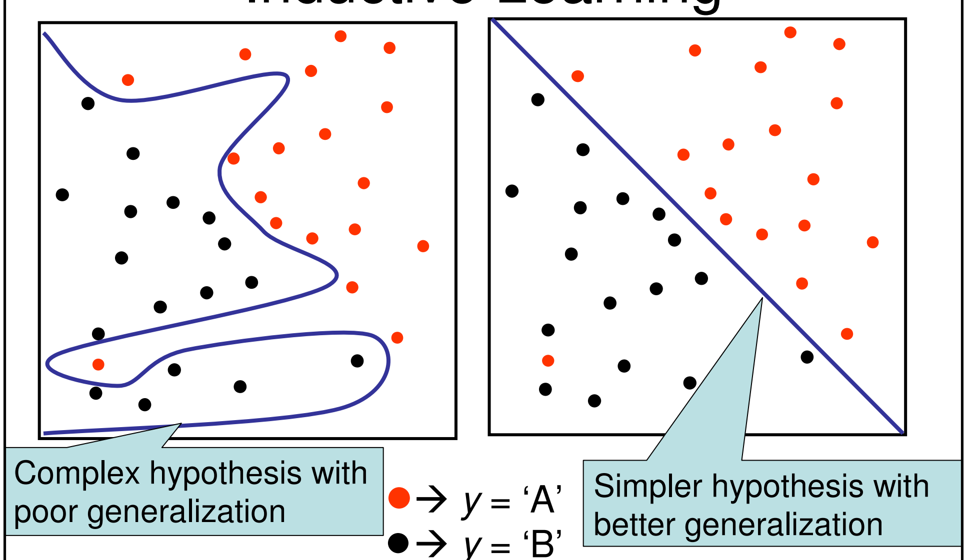
- What property should *h* have?
- It should agree with the training data…
- But that can lead to arbitrarily complex hypotheses…
- Which leads to completely wrong prediction on new test data…
- The model does not *generalize* beyond the training data…it overfits the training data

Inductive Learning



- Simplicity principle (*Occam's razor*): "entities are not to be multiplied beyond necessity"
- The simpler hypothesis is preferred
- Compromise between:
  - Error on data under hypothesis *h*
  - Complexity of hypothesis *h*

Inductive Learning



- Different illustration, same concept….

Inductive Learning

- Decision tree is one example of inductive learning
- To be covered next:
  - Instance-based learning and clustering
  - Neural networks
- In all cases, minimize:

  Error on data + complexity of model

Decision Trees

- Information Gain (IG) criterion for choosing splitting criteria at each level of the tree.
- Versions with continuous attributes and with discrete (categorical) attributes
- Basic tree learning algorithm leads to overfitting of the training data
- Pruning with:
  - Additional test data (not used for training)
  - Statistical significance tests
- Example of inductive learning