Artificial Intelligence:
Representation and Problem Solving

15-381

April 12, 2007

Decision Trees 2

20 questions

- Consider this game of 20 questions on the web:

  20Q.net Inc.

Pick your poison

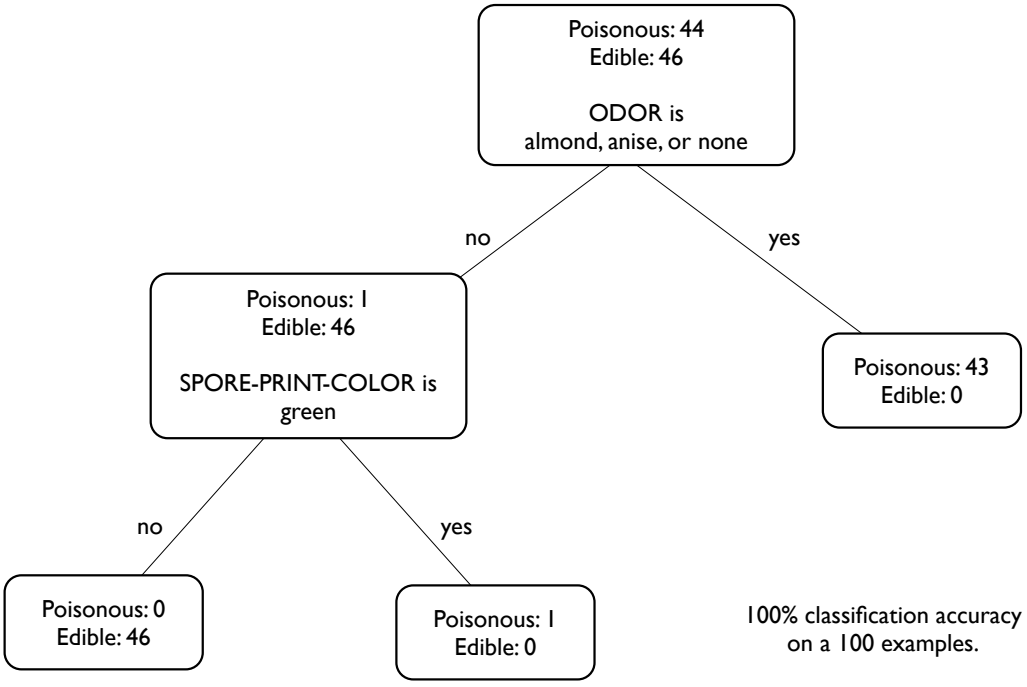
- How do you decide if a mushroom is edible?
- What's the best identification strategy?
- Let's try decision trees.

"Death Cap"

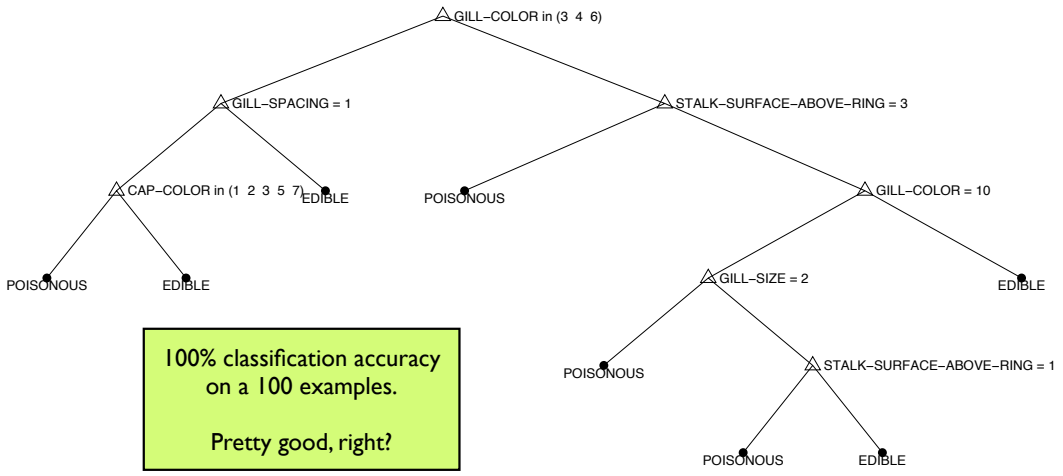
Some mushroom data (from the UCI machine learning repository)

	EDIBLE?	CAP-SHAPE	CAP-SURFACE	CAP-COLOR	ODOR	STALK-SHAPE	POPULATION	HABITAT	• • •
1	edible	flat	fibrous	red	none	tapering	several	woods	• • •
2	poisonous	convex	smooth	red	foul	tapering	several	paths	• • •
3	edible	flat	fibrous	brown	none	tapering	abundant	grasses	• • •
4	edible	convex	scaly	gray	none	tapering	several	woods	• • •
5	poisonous	convex	smooth	red	foul	tapering	several	woods	• • •
6	edible	convex	fibrous	gray	none	tapering	several	woods	• • •
7	poisonous	flat	scaly	brown	fishy	tapering	several	leaves	• • •
8	poisonous	flat	scaly	brown	spicy	tapering	several	leaves	• • •
9	poisonous	convex	fibrous	yellow	foul	enlarging	several	paths	• • •
10	poisonous	convex	fibrous	yellow	foul	enlarging	several	woods	• • •
11	poisonous	flat	smooth	brown	spicy	tapering	several	woods	• • •
12	edible	convex	smooth	yellow	anise	tapering	several	woods	• • •
13	poisonous	knobbed	scaly	red	foul	tapering	several	leaves	• • •
14	poisonous	flat	smooth	brown	foul	tapering	several	leaves	• • •
15	poisonous	flat	fibrous	gray	foul	enlarging	several	woods	• • •
16	edible	sunken	fibrous	brown	none	enlarging	solitary	urban	• • •
17	poisonous	flat	smooth	brown	foul	tapering	several	woods	• • •
18	poisonous	convex	smooth	white	foul	tapering	scattered	urban	• • •
19	poisonous	flat	scaly	yellow	foul	enlarging	solitary	paths	• • •
20	edible	convex	fibrous	gray	none	tapering	several	woods	• • •
	• • •	• • •	• • •	• • •	• • •	• • •	• • •	• • •	• • •

An easy problem: two attributes provide most of the information



Same problem with no odor or spore-print-color



100% classification accuracy on a 100 examples.

Pretty good, right?

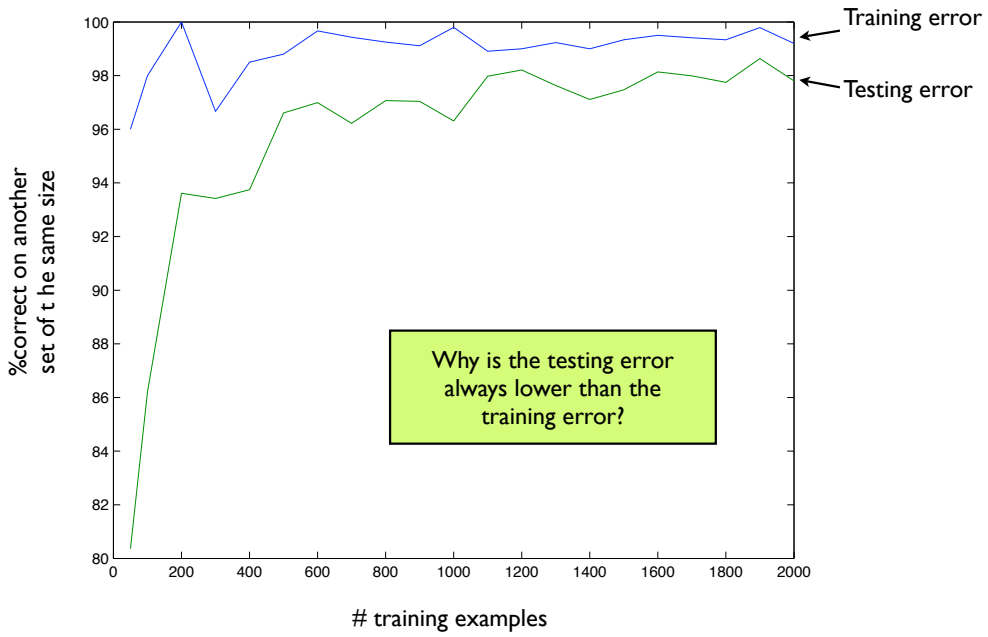
What if we go off hunting with this decision tree?

Performance on another set of 100 mushrooms:

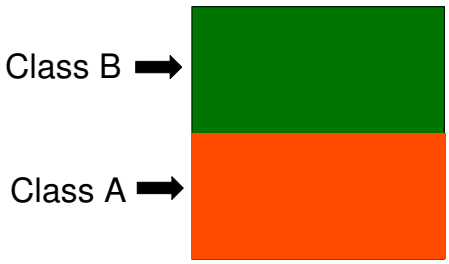
80%

Why?

Not enough examples?



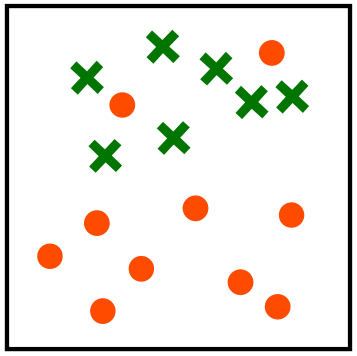
The Overfitting Problem: Example



- 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
- Generating a decision tree would be trivial, right?

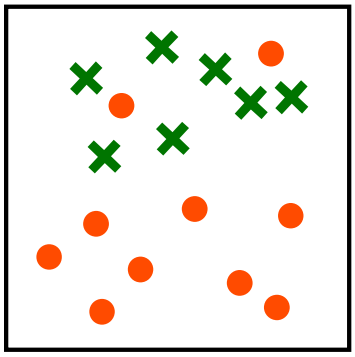
The following examples are from Prof Hebert.

The Overfitting Problem: Example

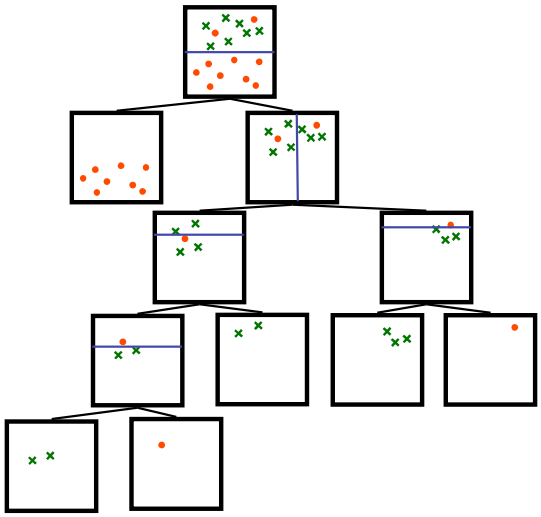


- But in the real world, our observations have variability.
- They can also be corrupted by noise.
- Thus, the observed pattern is more complex than it appears.

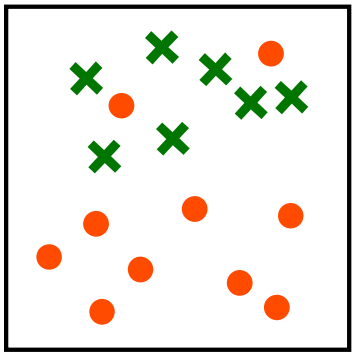
The Overfitting Problem: Example



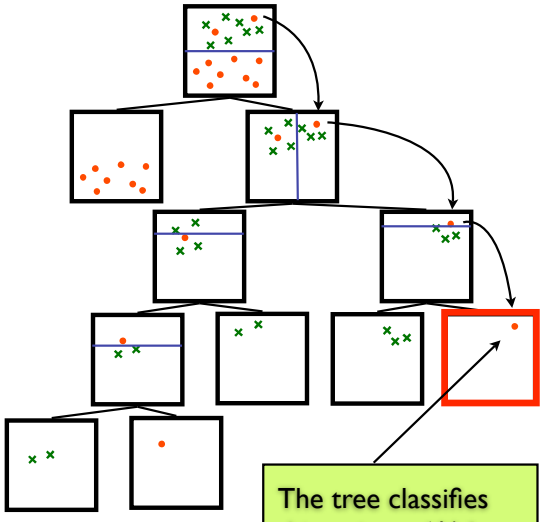
- noise makes the decision tree more complex than it should be
- The algorithm tries to classify all of the training set perfectly
- This is a fundamental problem in learning and is called *overfitting*



The Overfitting Problem: Example

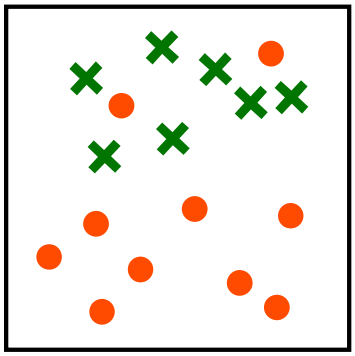


- noise makes the decision tree more complex than it should be
- The algorithm tries to classify all of the training set perfectly
- This is a fundamental problem in learning and is called *overfitting*

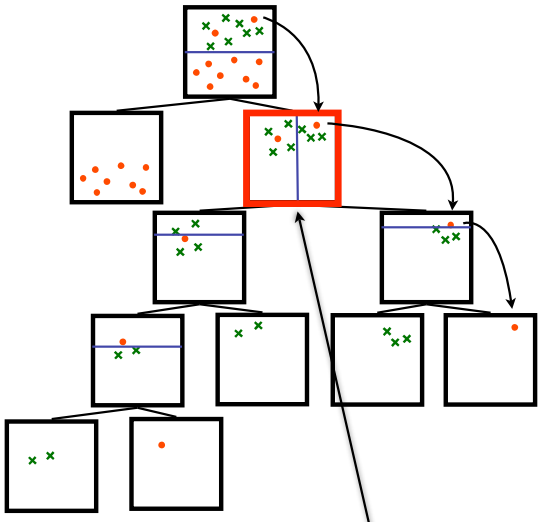


The tree classifies this point as 'A', but it won't generalize to new examples.

The Overfitting Problem: Example

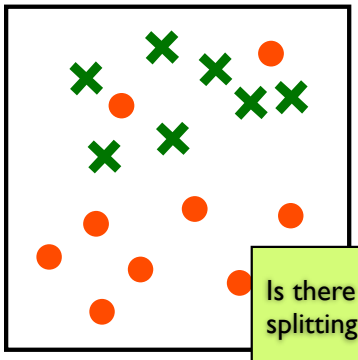


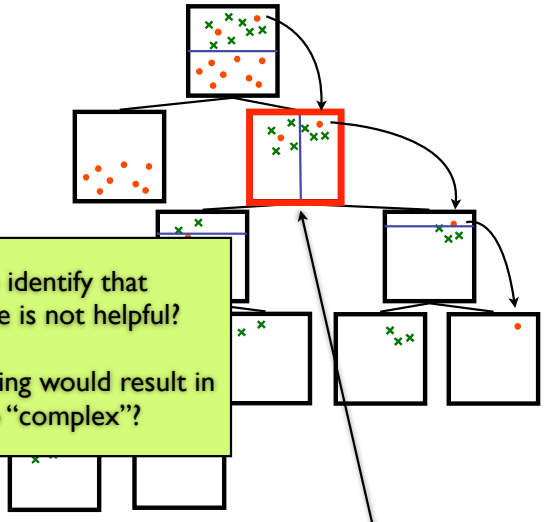
- noise makes the decision tree more complex than it should be
- The algorithm tries to classify all of the training set perfectly
- This is a fundamental problem in learning and is called *overfitting*



The problem started here. X_1 is irrelevant to the underlying structure.

The Overfitting Problem: Example





- noise makes the de
  more complex tha
- The algorithm tries to classify all of the training set perfectly
- This is a fundamental problem in learning and is called *overfitting*

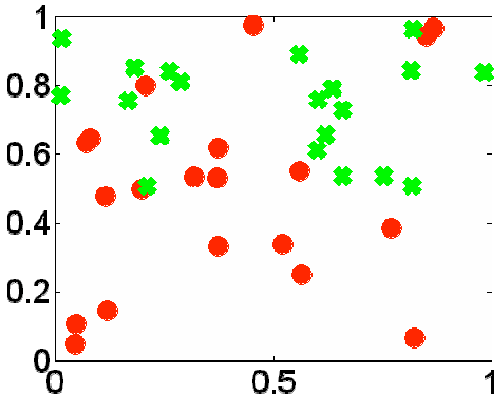
Is there a way to identify that splitting this node is not helpful?

Idea: When splitting would result in a tree that is too "complex"?

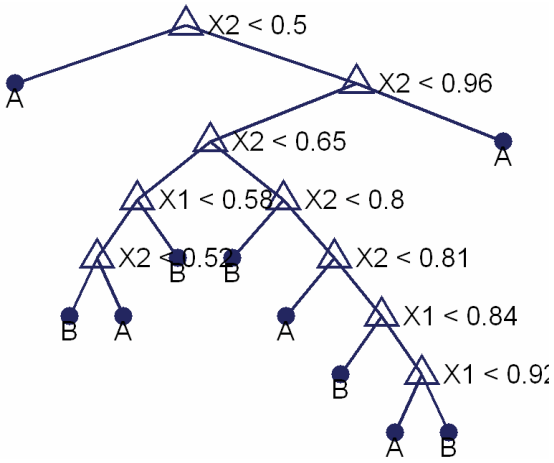
The problem started here. X_1 is irrelevant to the underlying structure.

Addressing overfitting

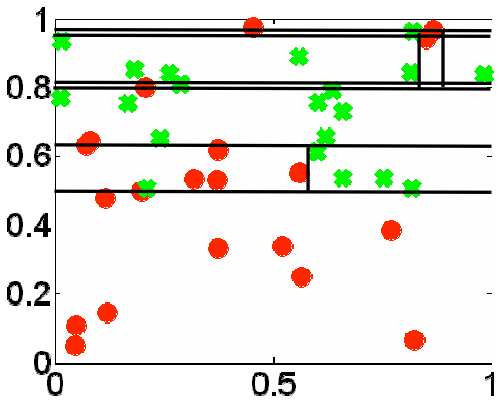
- Grow tree based on training data.
- This yields an *unpruned* tree.
- Then prune nodes from the tree that are unhelpful.
- How do we know when this is the case?
  - Use additional data not used in training, ie *test data*
  - Use a statistical significance test to see if extra nodes are different from noise
  - Penalize the complexity of the tree



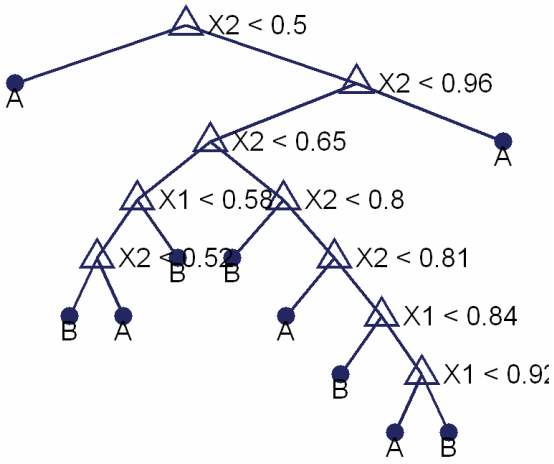
Training Data



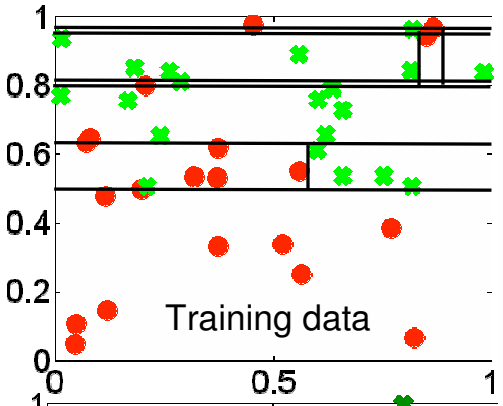
Unpruned decision tree from training data

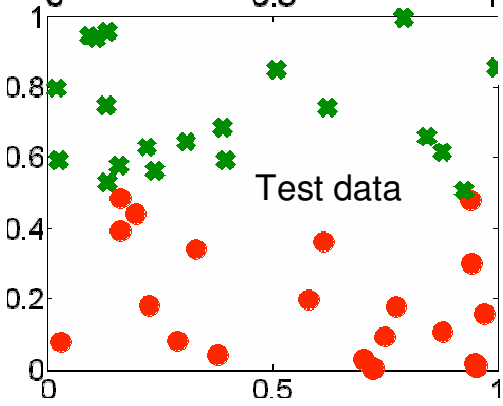


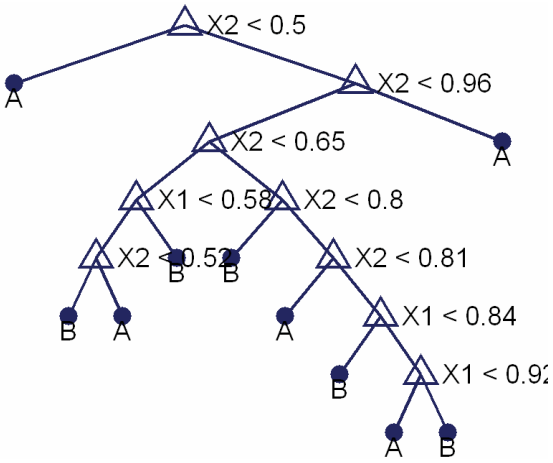
Training data with the partitions induced by the decision tree (Notice the tiny regions at the top necessary to correctly classify the 'A' outliers!)



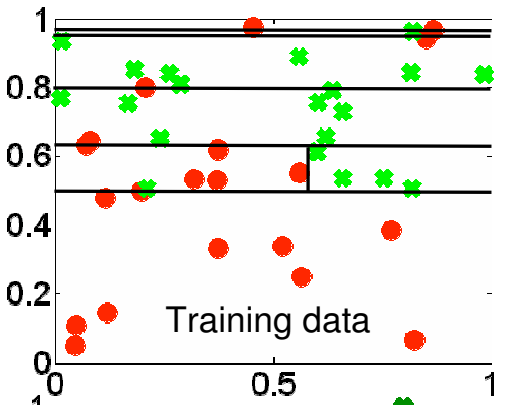
Unpruned decision tree from training data

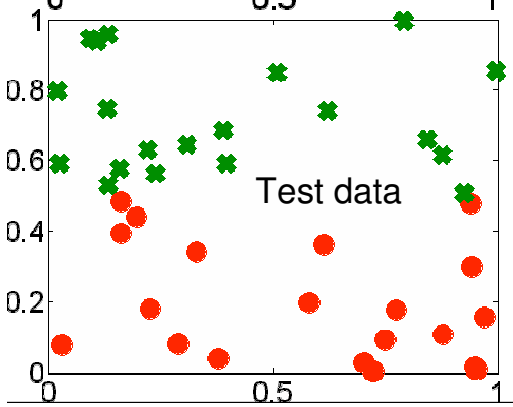


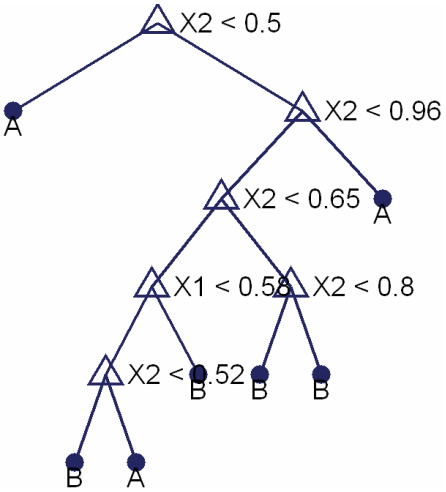




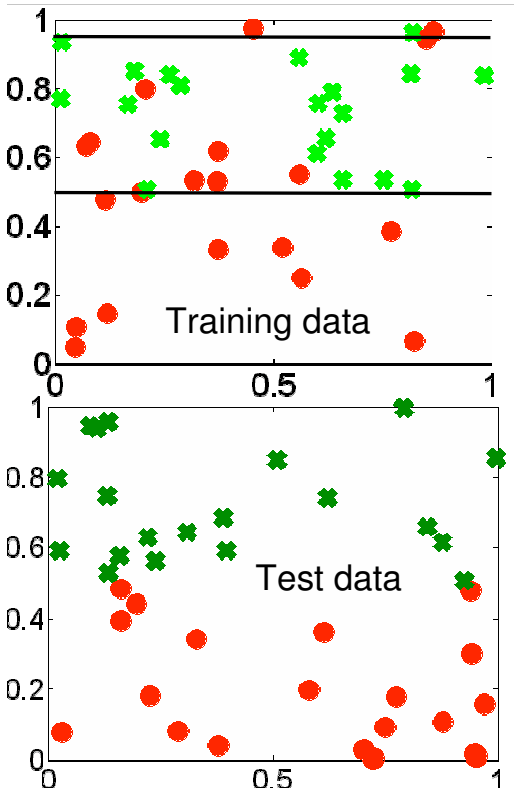
Unpruned decision tree from training data
Performance (% correctly classified)
Training: 100%
Test: 77.5%

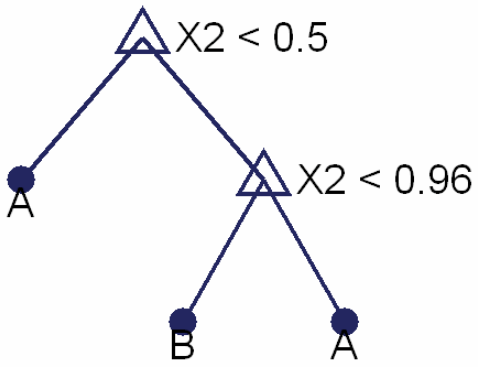






Pruned decision tree from training data
Performance (% correctly classified)
Training: 95%
Test: 80%



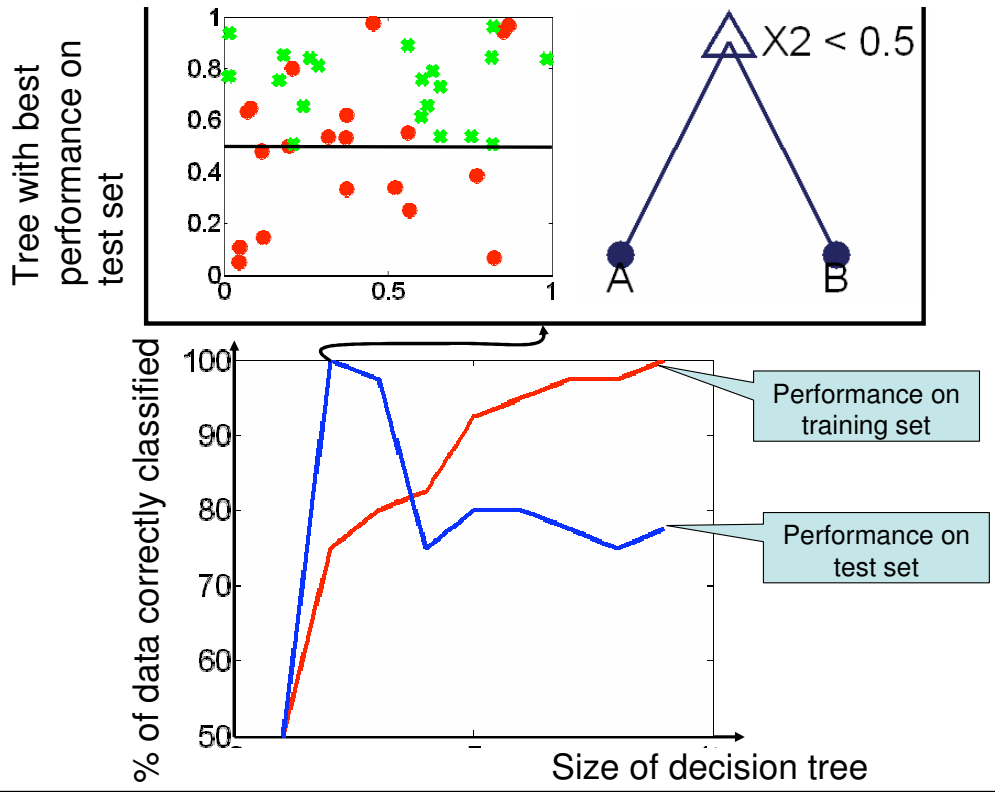


Pruned decision tree from training data

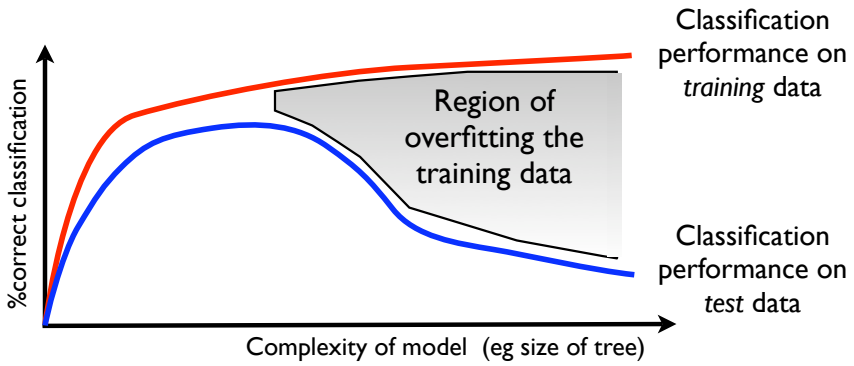
Performance (% correctly classified)

Training: 80%

Test: 97.5%



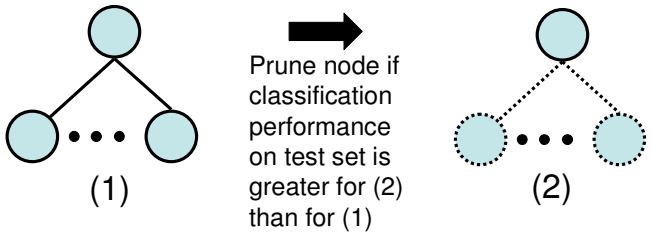
General principle



- As its complexity increases, the model is able to better classify the training data
- Performance on the test data initially increases, but then falls as the model *overfits*, or becomes specialized for classifying the noise training
- The complexity in decision trees is the number of free parameters, ie the number of nodes

Strategies for avoiding overfitting: Pruning

- Ovoiding overfitting is equivalent to achieving good generalization
- All strategies need some way to control the complexity of the model
- Pruning:
  - constructs a standard decision tree, *but* keep a test data set on which the model is *not* trained
  - prunes leaves recursively
  - splits are eliminated (or pruned) by evaluating performance on the test data
  - a leaf is pruned if classification on the test data increases by removing the split



Strategies for avoiding overfitting: Statistical significance tests

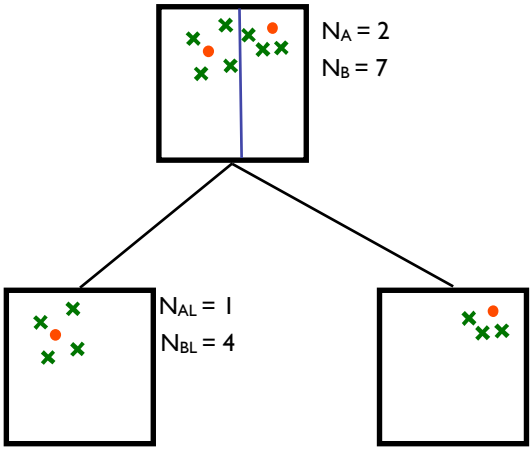
- For each split, ask of there is a *significant* increase in the info. gain
- If we're splitting noise, then data are *random*
- What proportion of data go to left node?

$$p_L = \frac{N_{AL} + N_{BL}}{N_A + N_B} = \frac{5}{9}$$

- If data were random, how many would we *expect* to go to the left?

$$N'_{AL} = N_A \times p_L = 10/9$$
$$N'_{BL} = N_B \times p_L = 35/9$$

- Is there a statistically significant different from what we observe and what we expect?



- # class A in root node is N_A = 2
- # class B in root node is N_B = 7
- # class A in left node is N_{AL} = 1
- # class B in left node is N_{BL} = 4

If not, don't split!

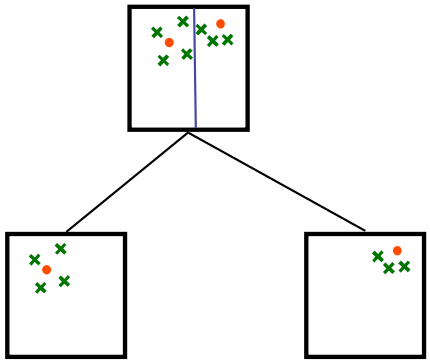
Detecting statistically significant splits

- A measure of statistical significance

$$K = \frac{(N'_{AL} - N_{AL})^2}{N'_{AL}} + \frac{(N'_{BL} - N_{BL})^2}{N'_{BL}} + \frac{(N'_{AR} - N_{BR})^2}{N'_{BR}} + \frac{(N'_{BR} - N_{BR})^2}{N'_{BR}}$$

- K measures how much the split deviates from what we would expect from random data
- K small ⇒

  the information gain from the split is not significant

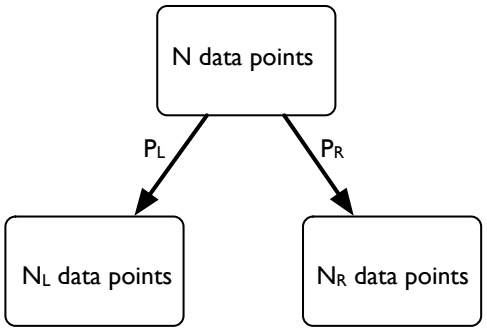


- Here,

$$K = \frac{(10/9 - 1)^2}{10/9} + \frac{(35/9 - 4)^2}{35/9} + \cdots = 0.0321$$

"χ^2 criterion": general case

- Small "Chi-square" values imply low statistical significance
- Nodes that have K smaller than threshold are *pruned*
- The threshold regulates the complexity of the model
  - Low thresholds allow larger trees and more overfitting
  - High thresholds keep trees small but may sacrifice performance



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

N_{ij} = Number of points from class i in child j

N'_{ij} = Number of points from class i in child j *assuming random selection*

$N'_{ij} = N_i \times p_j$

Illustration on our toy problem

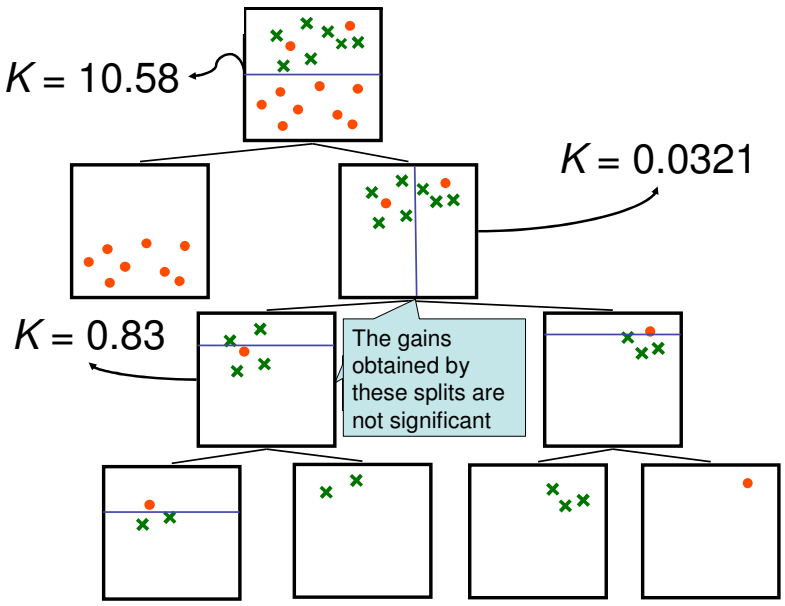
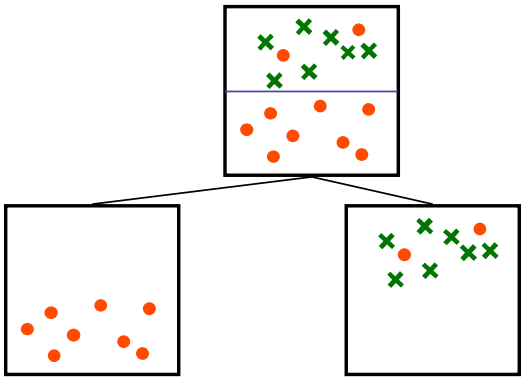


Illustration on our toy problem



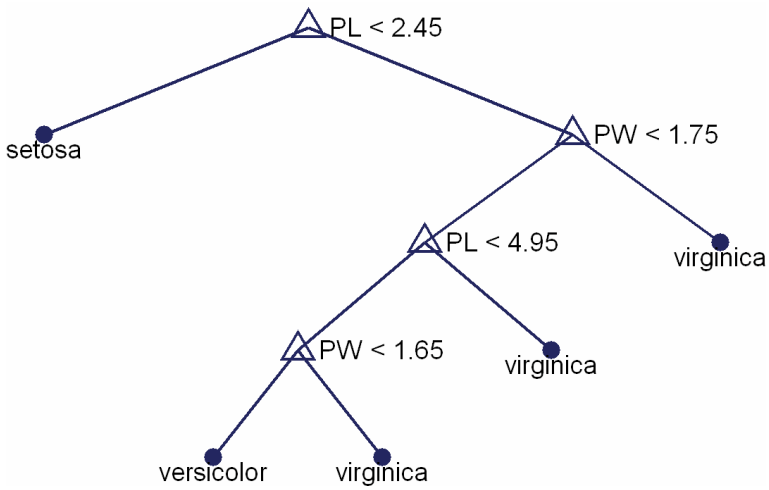
- With appropriate thresholding, we get the decision tree we expect, ie only one split.
- Note: this approach can be applied to both continuous and discrete attributes

A real example: Fisher's Iris data

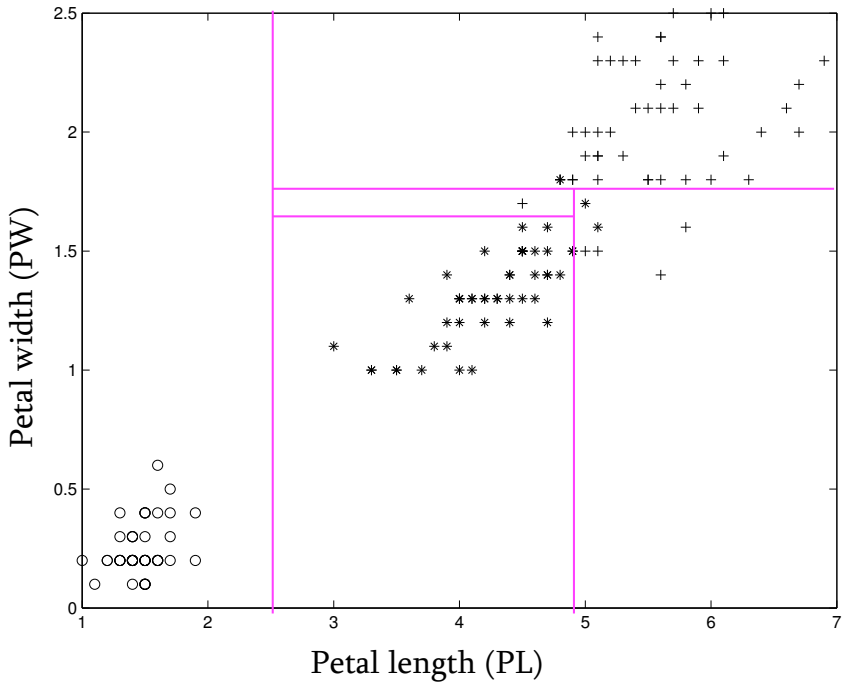
- three classes of irises
- four attributes

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

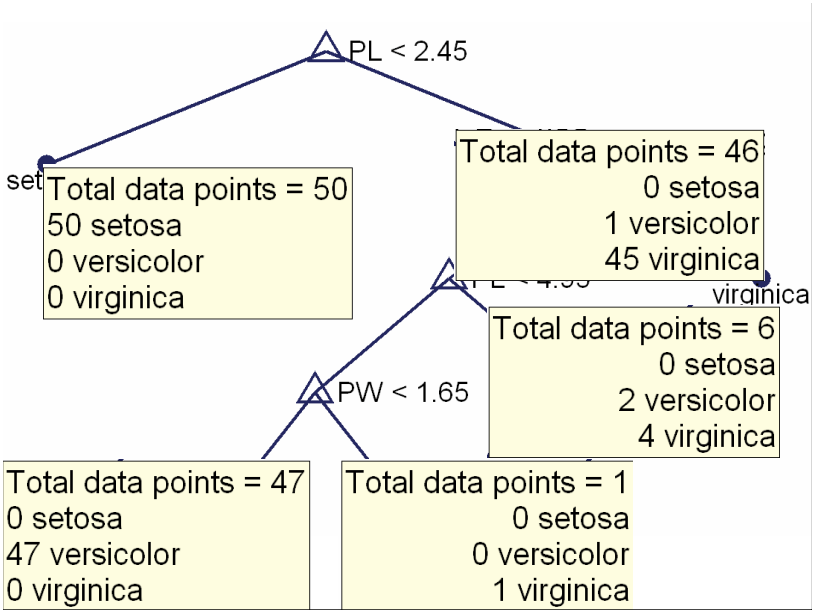
Full (unpruned) decision tree



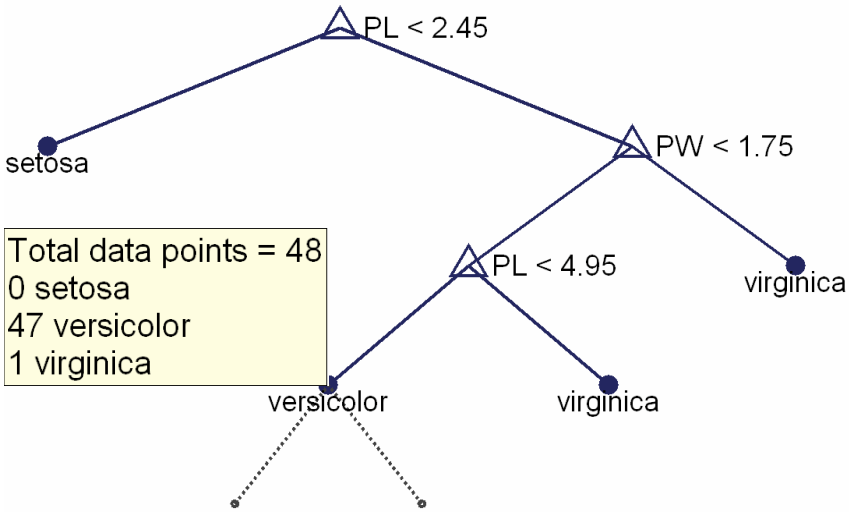
The scatter plot of the data with decision boundaries



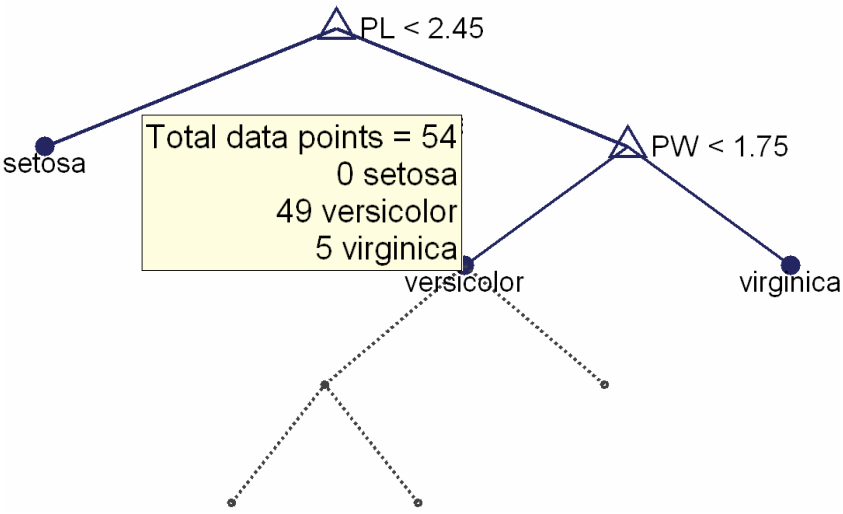
Tree statistics



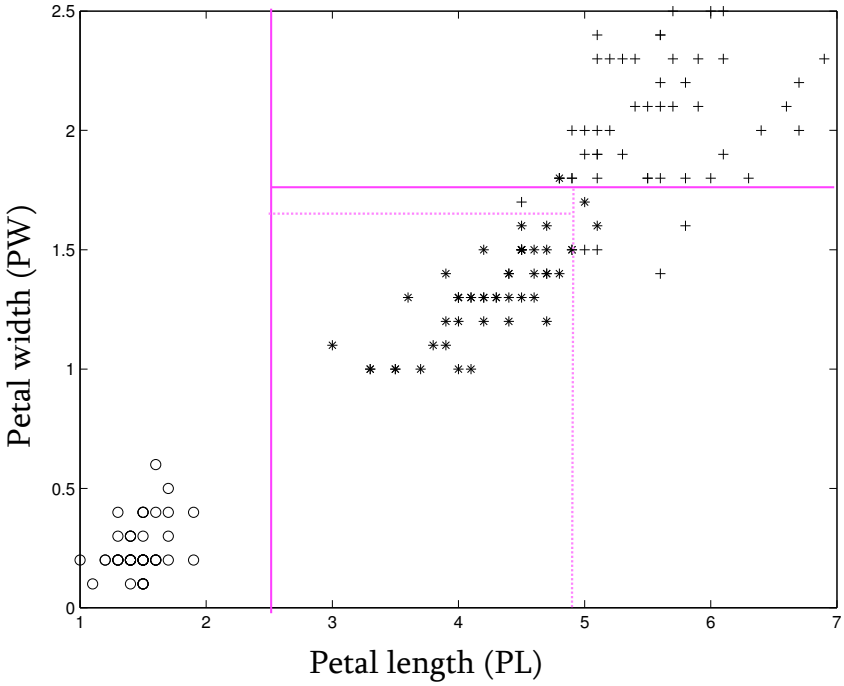
Pruning one level



Pruning two levels



The tree with pruned decision boundaries



Recap: What you should understand

- *Learning* is fitting models (estimating their parameters) from data
- The goal of learning is to achieve good predictions/classifications for novel data, ie good *generalization*
- There complexity of a model (related but not identical to the number of parameters) determines how well it can *fit* the data
- If there are insufficient data relative to the complexity, the model will exhibit poor generalization, ie it will *overfit the data*
- To avoid this learning algorithms divide examples into *training and testing data*
- Decision Trees:
  - a simple hierarchical approach to classification
  - goal is to achieve best classification with minimal # of decisions
  - works with binary, categorical, or continuous data
  - *information gain* is a useful splitting strategy (there are many others)
  - the decision tree is built *recursively*
  - it can be *pruned* to reduce the problem of overfitting