Introduction to Machine Learning: CS 436/580L

Decision Trees

Instructor: Arti Ramesh

Binghamton University

Administrivia

- Project teams!!! Hope you are all working actively on that! Send me an email if you are not yet part of a team. Due **Feb 5th**.
- Project Ideas: Check if problem has interesting data!
- TA Office Space: N-21.

Administrivia: Homework 1

- Homework 1 will be out today. Due Wednesday **Feb 8th, midnight.**
- **Programming part of homework should compile on remote.cs.binghamton.edu**
- Check whether remote has the packages that you need before starting to implement
- Using existing decision-tree package is unacceptable!
- **Please, do not cheat!!! Unsure of what is considered cheating, stop by my office/send me an email!**
- Submit a single zip file: FirstName_LastName_hw1. Files that don't follow the naming scheme **WILL LOSE POINTS**.

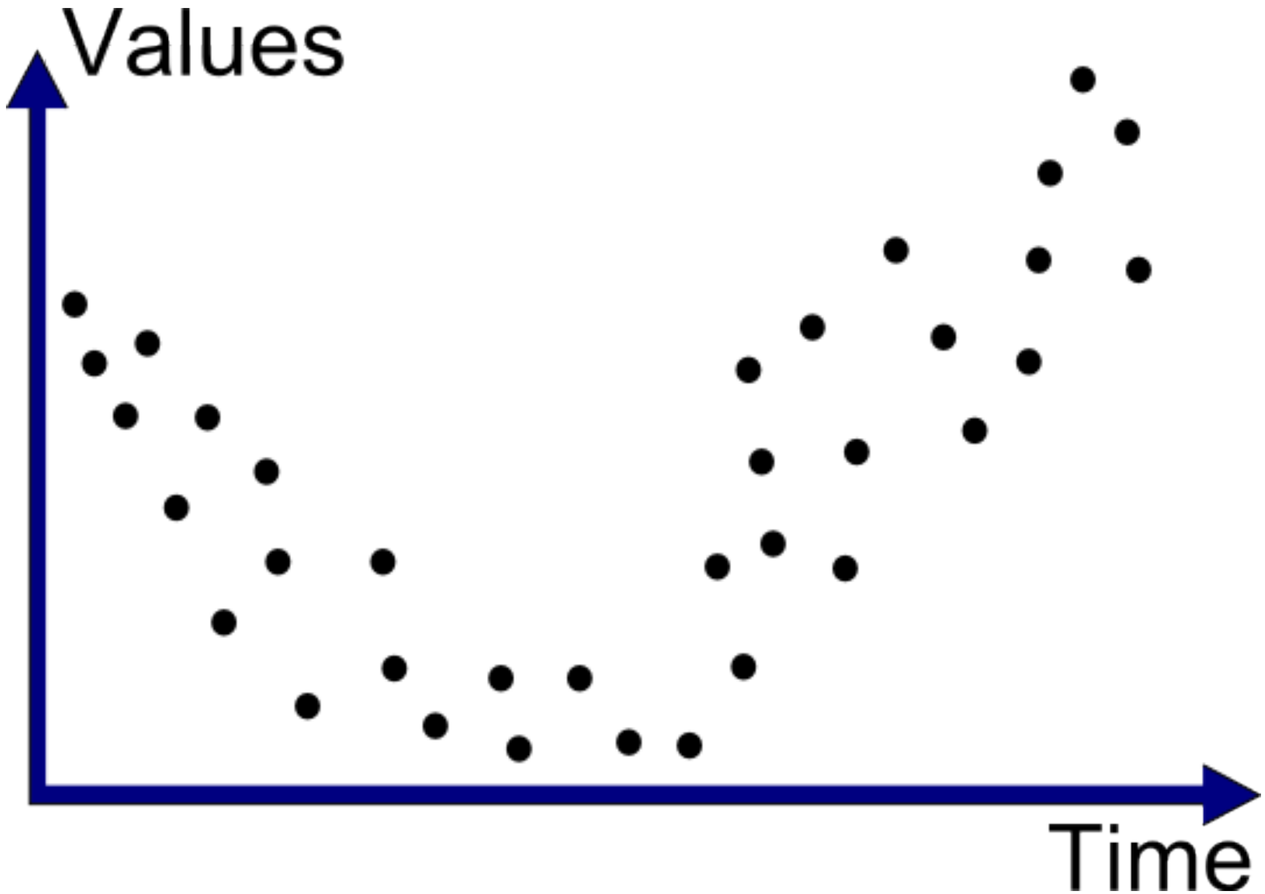
Recap

- Supervised learning
  - Given: Training data with desired output
  - Assumption: There exists a function *f* which transforms input “x” into output *f(x)*.
  - To do: find an approximation to *f*
- Classification: Output, i.e., *f(x)* is discrete
- What makes learning hard?
- Issues.

Recap

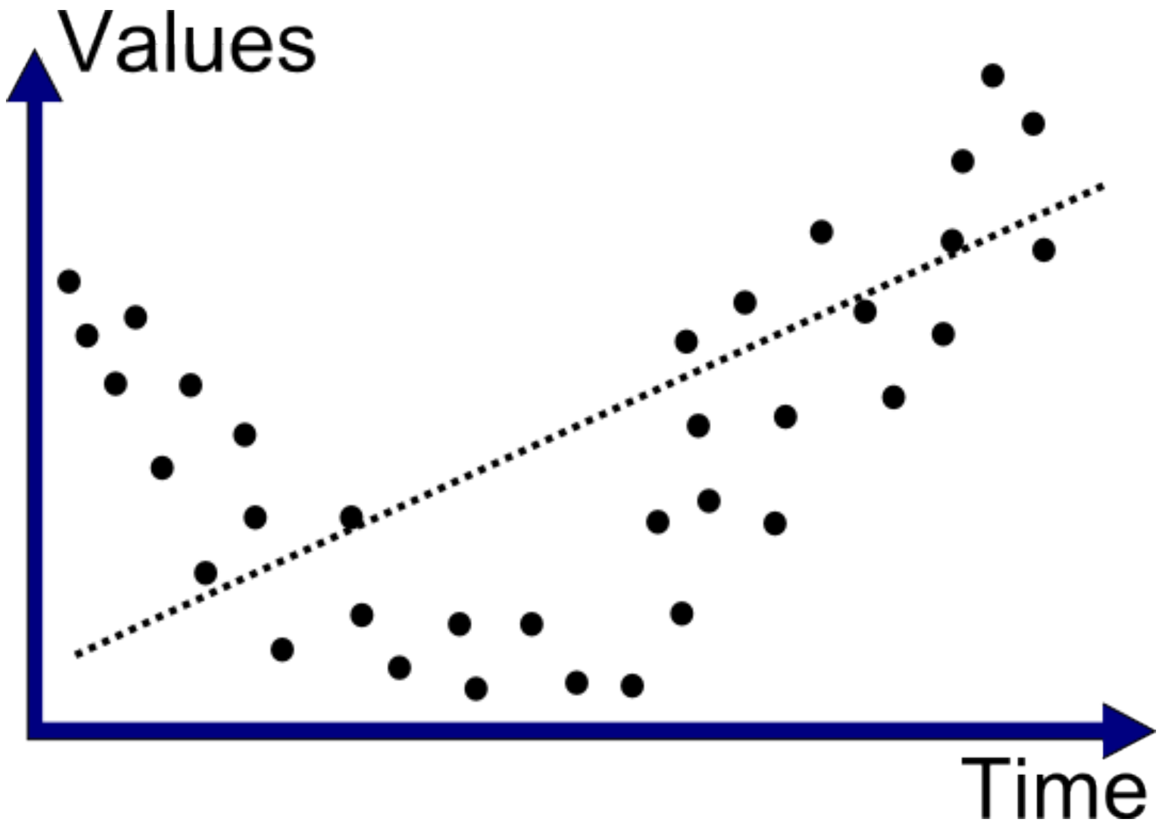
- What are hypothesis spaces?
- What is a validation set? How is it different from the test set/training set?
- What is “peeking” in machine learning?
- What are the three main components of machine learning?
- What is overfitting?

Forecasting Model

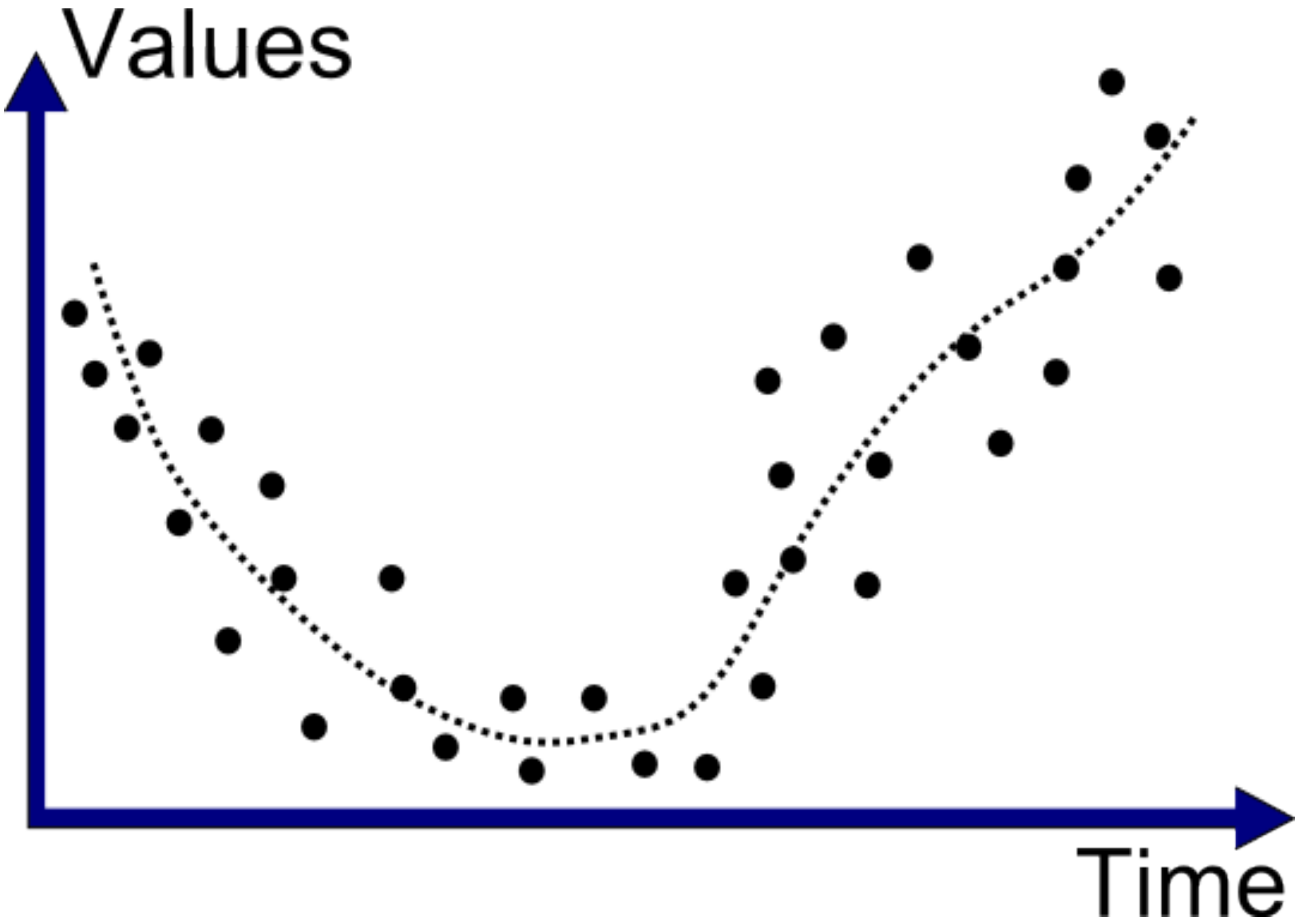


Linear model (error > 50%)

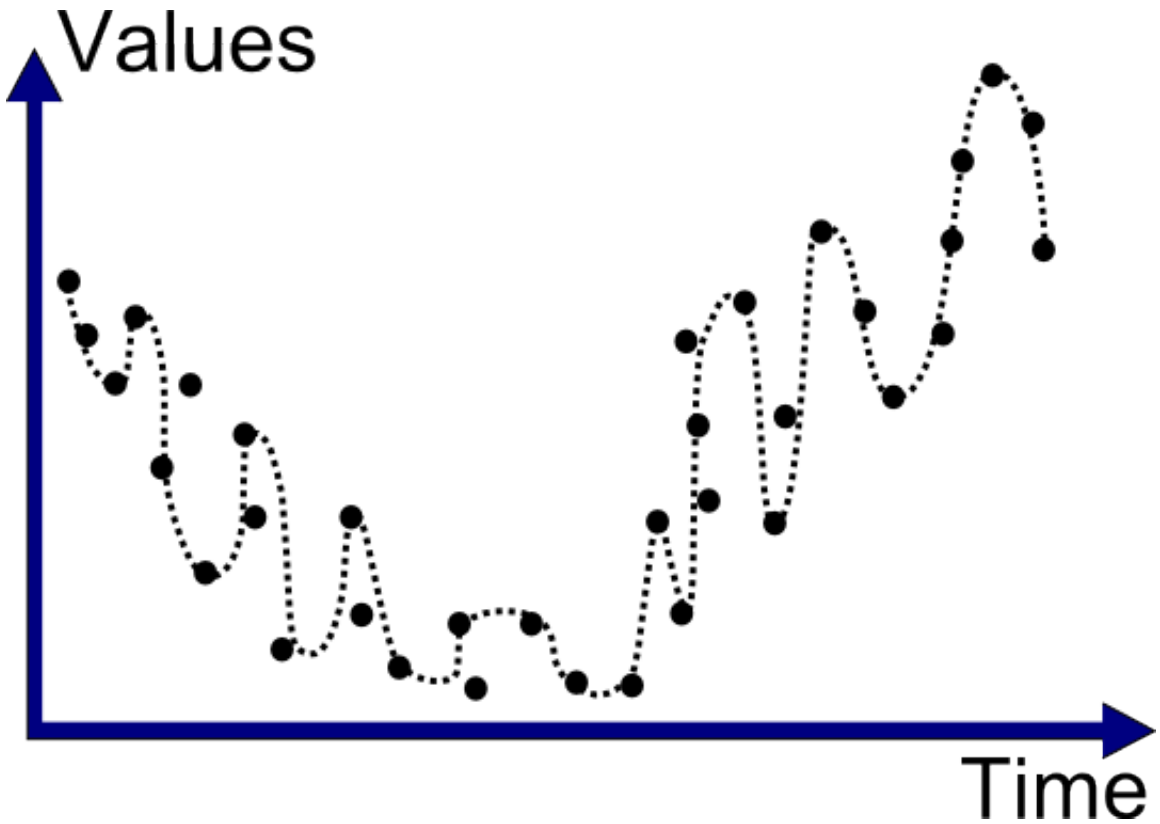
(numbers are made-up, just an illustration)



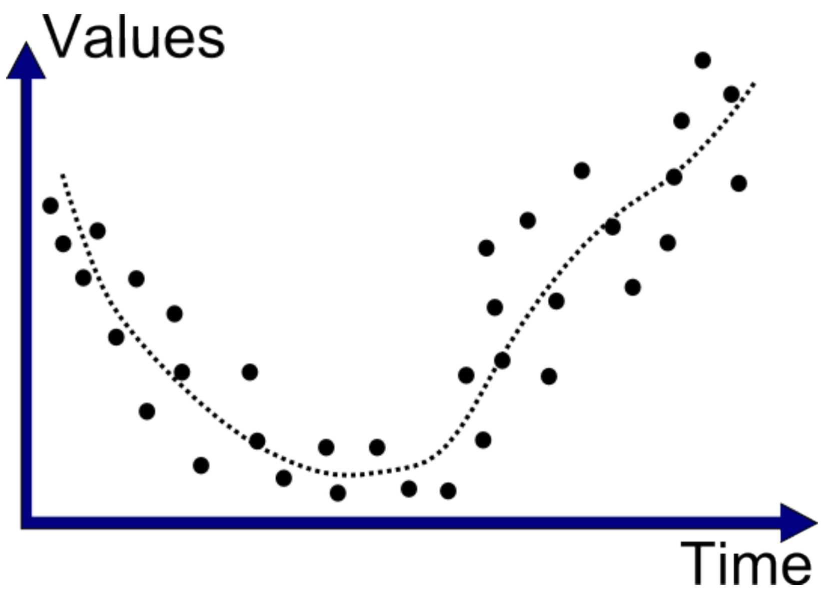
More complexity (error ≈ 10%)



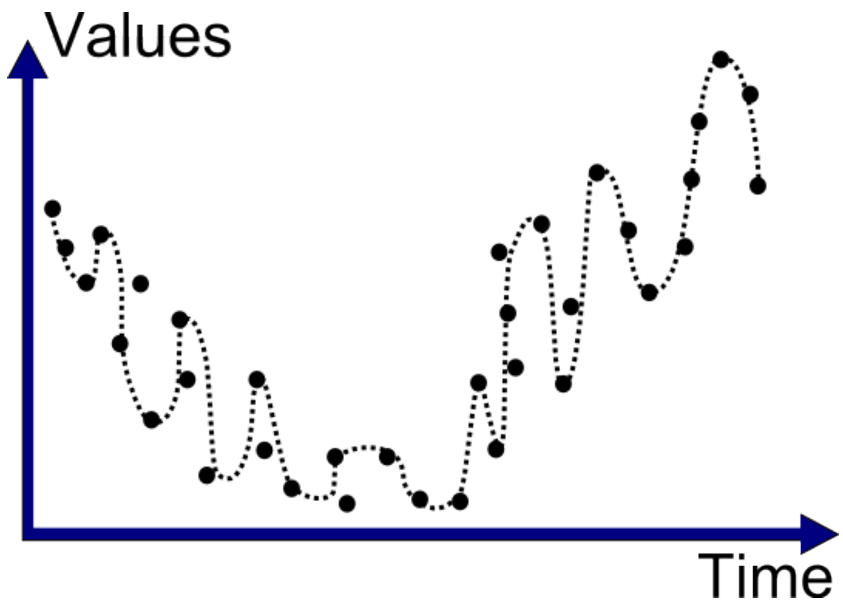
Too much complexity (error < 1%)



Comparing models

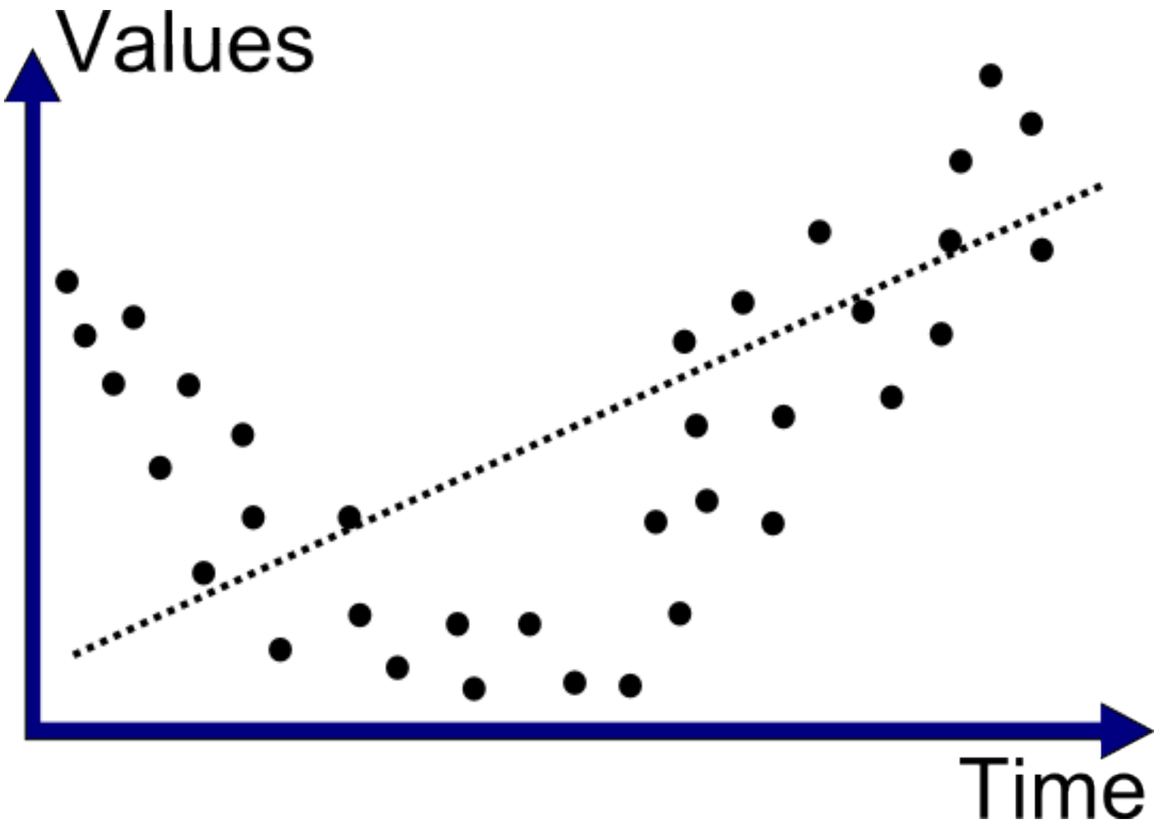


Model “fits”



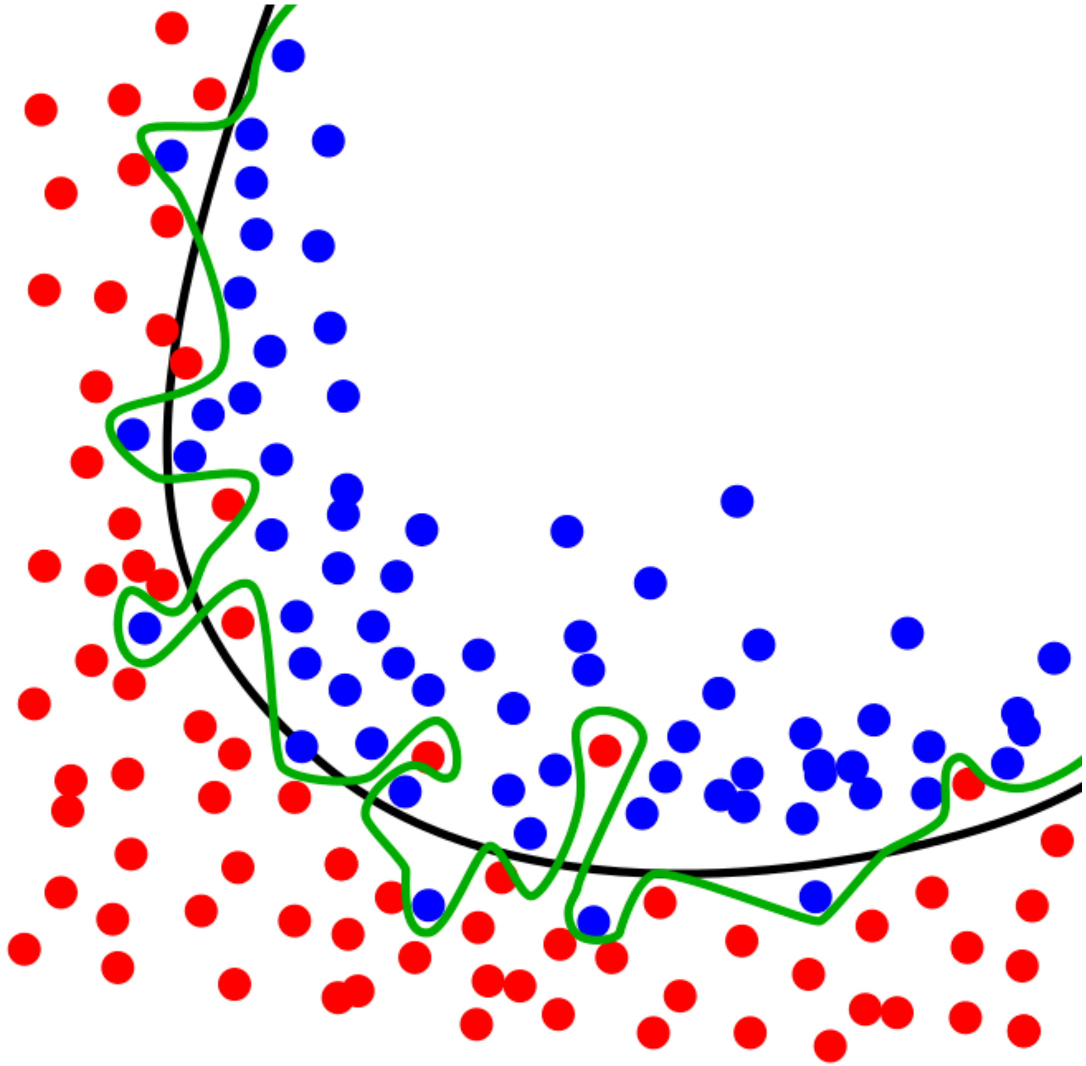
Model “overfits”

Underfitting



Model “underfits”

Overfitting in Classification



Occam’s Razor

- Occam’s Razor: principle attributed to the 14th-century English logician and Franciscan friar William of Ockham.
- Occam’s Razor “All other things being equal, the simplest solution is the best.”
- When multiple competing theories are equal in other respects, the principle recommends selecting the theory that introduces the fewest assumptions and postulates the fewest entities.
- Rule of thumb: **Prefer the simplest hypothesis <u>that fits the data</u>**

Ockham's Razor

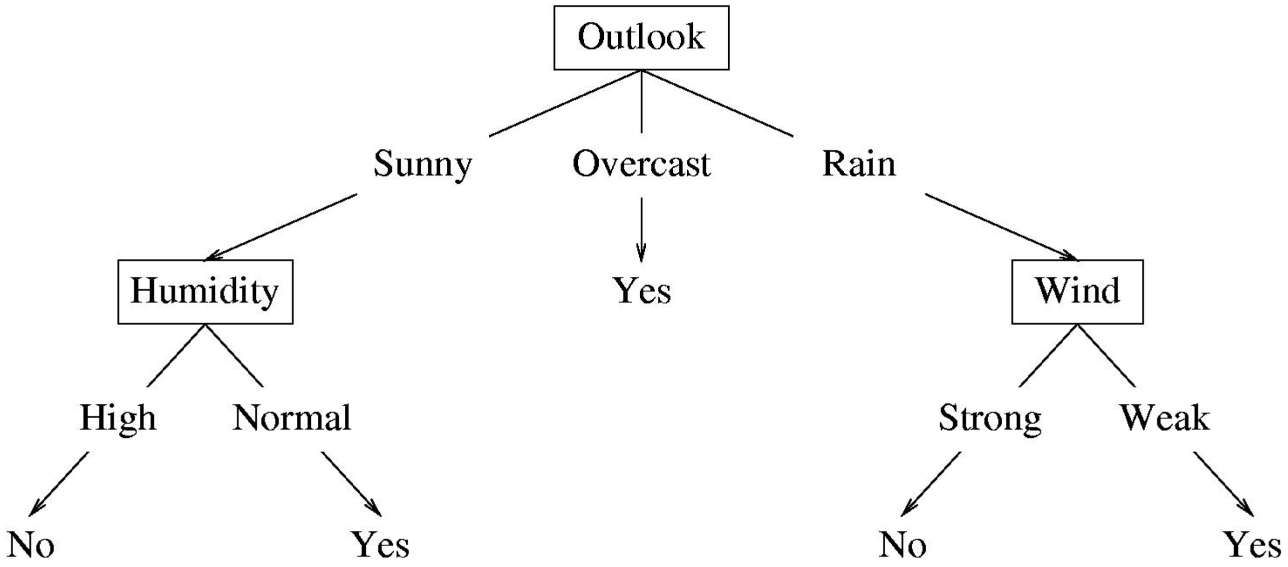
- A simpler hypothesis is less likely to be correct "by chance" and is therefore more likely to generalize well
- If we have 2 hypotheses with equally small training error, how can we pick the right one?
  - If we pick the wrong one, with enough data, we will eventually find out.

Ockham's Razor

- The amount of data we need (to be sure we pick the right hypothesis) depends on the complexity of the hypothesis class.
- If we want to avoid the possibility of overfitting, we should restrict the complexity of the hypothesis class, or use a larger training set.
- An overly simple hypothesis class may underfit.

Decision Tree Hypothesis Space

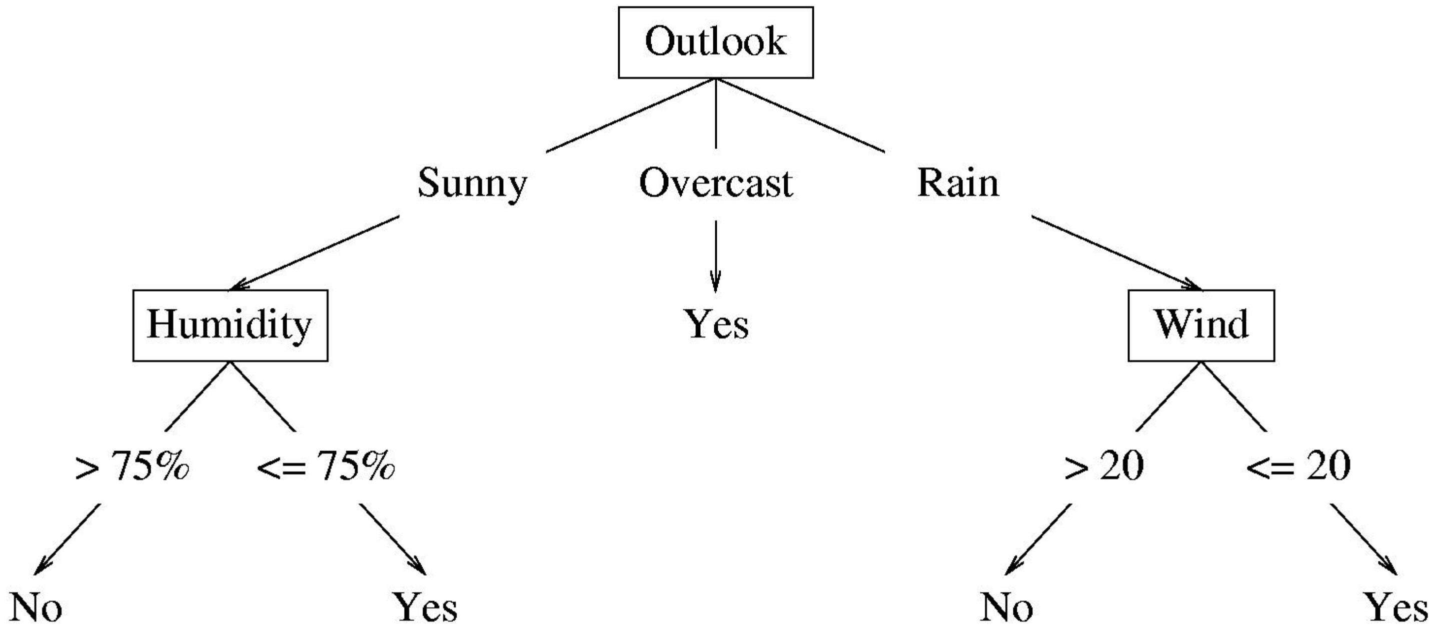
- **Internal nodes** test the value of particular features x_j and branch according to the results of the test.
- **Leaf nodes** specify the class $h(\mathbf{x})$.



Suppose the features are **Outlook** (x_1), **Temperature** (x_2), **Humidity** (x_3), and **Wind** (x_4). Then the feature vector $\mathbf{x} = (Sunny, Hot, High, Strong)$ will be classified as **No**. The **Temperature** feature is irrelevant.

Decision Tree Hypothesis Space

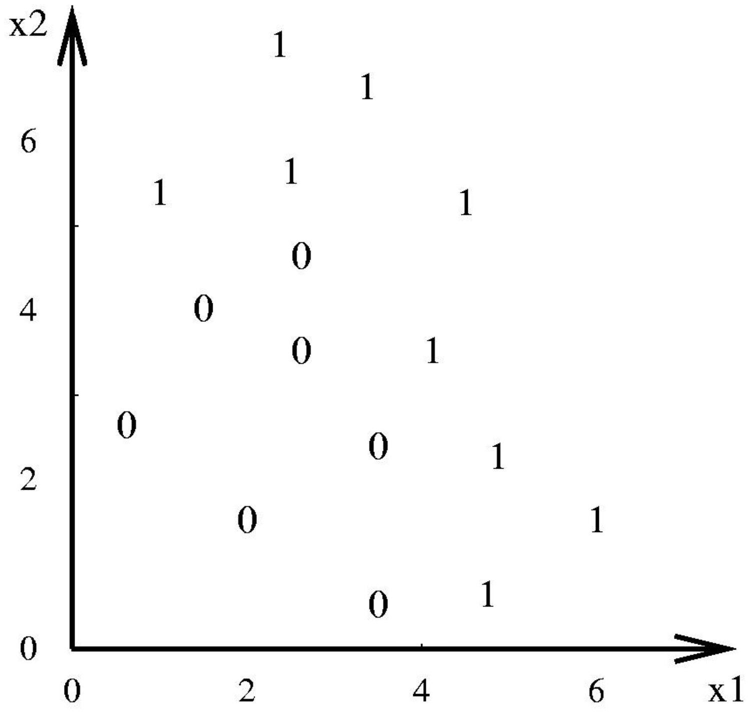
If the features are continuous, internal nodes may test the value of a feature against a threshold.

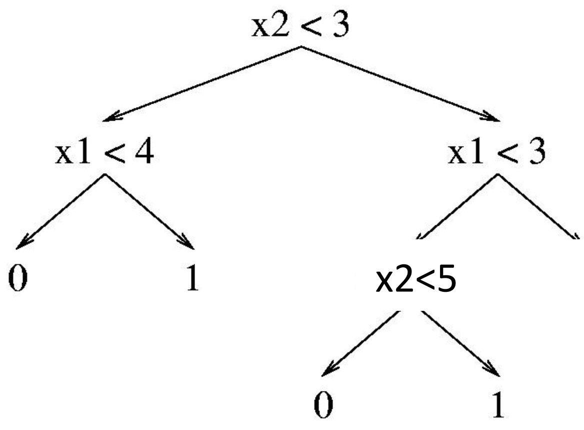


- **Notes:**
  - **A discrete feature can appear only once (or not appear at all) along the unique path from the root to a leaf.**
  - **Question: Can I test on Humidity with a threshold of 95?**
    - **YES (it is a different discrete feature).**

Decision Tree Decision Boundaries

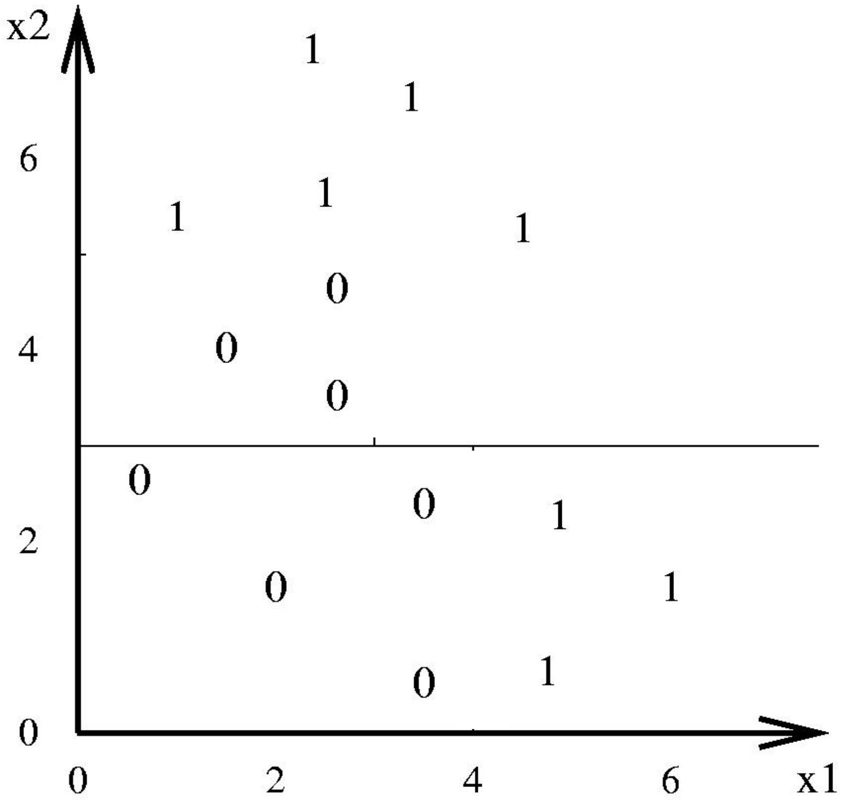
Decision trees divide the feature space into axis-parallel rectangles, and label each rectangle with one of the K classes.

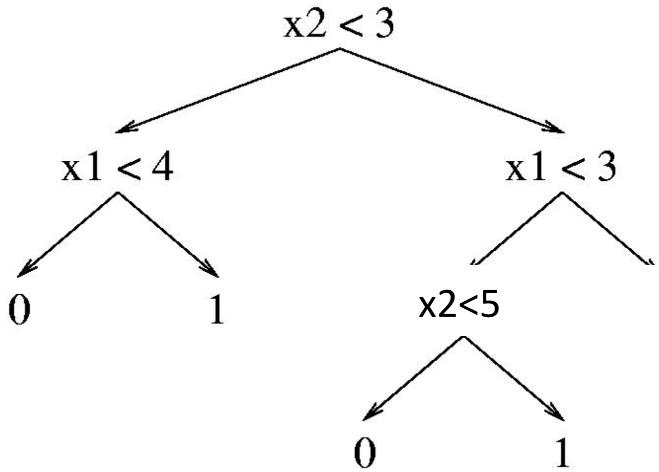




Decision Tree Decision Boundaries

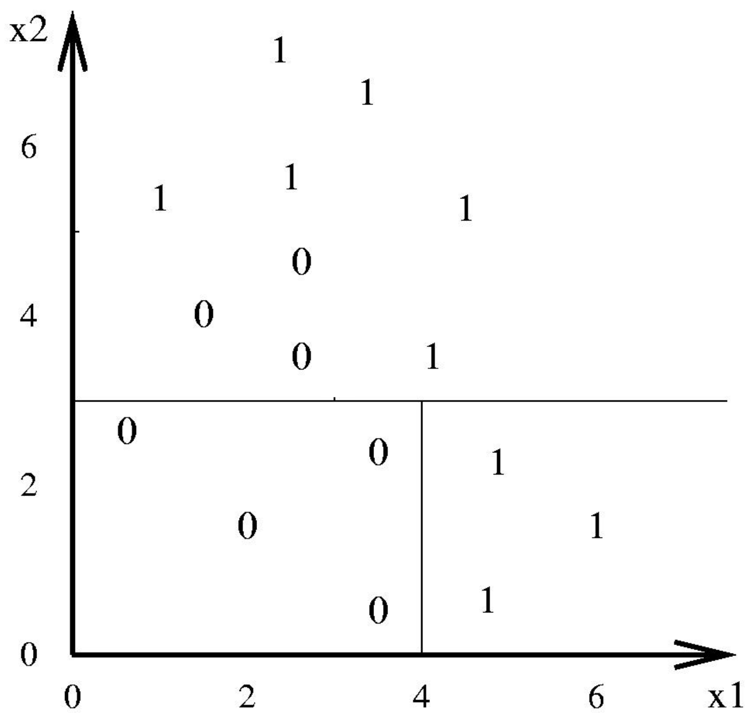
Decision trees divide the feature space into axis-parallel rectangles, and label each rectangle with one of the K classes.

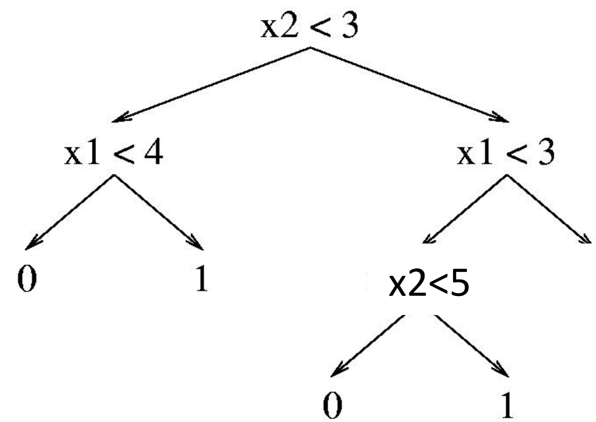




Decision Tree Decision Boundaries

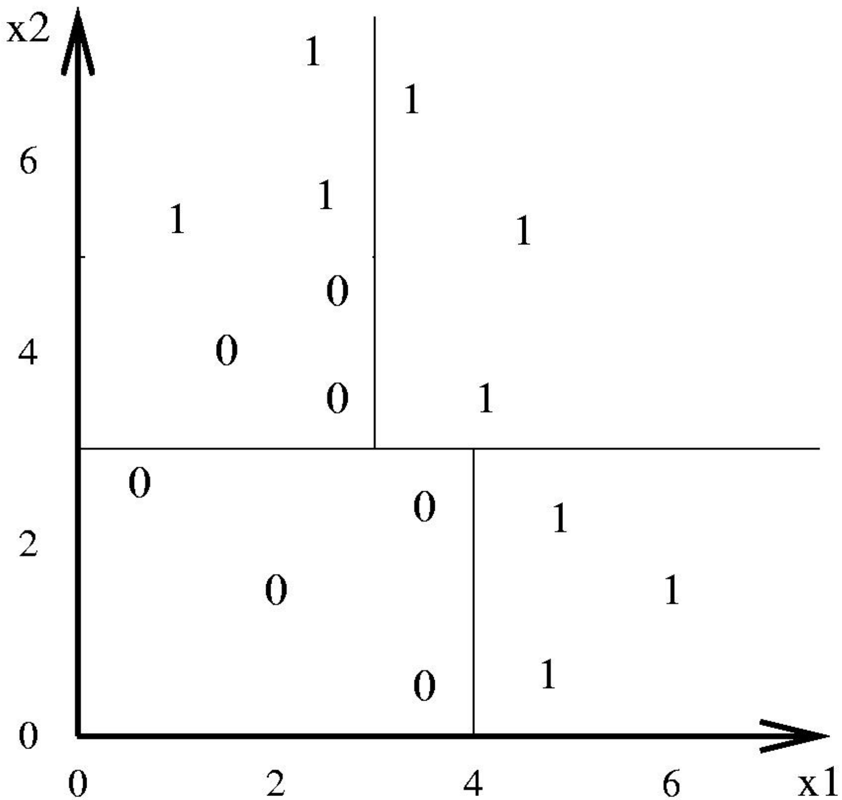
Decision trees divide the feature space into axis-parallel rectangles, and label each rectangle with one of the K classes.

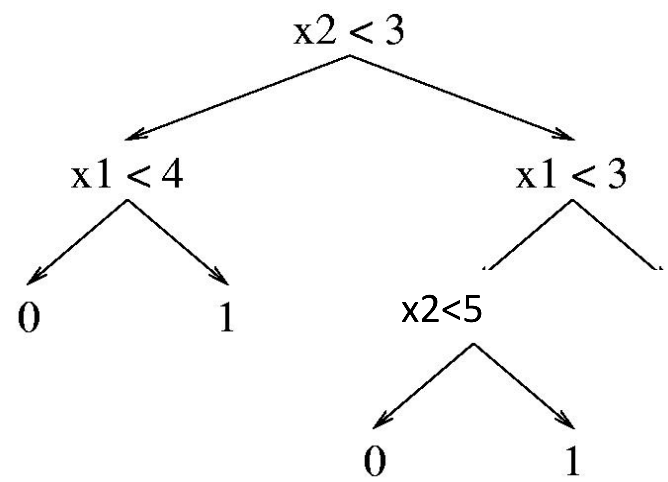




Decision Tree Decision Boundaries

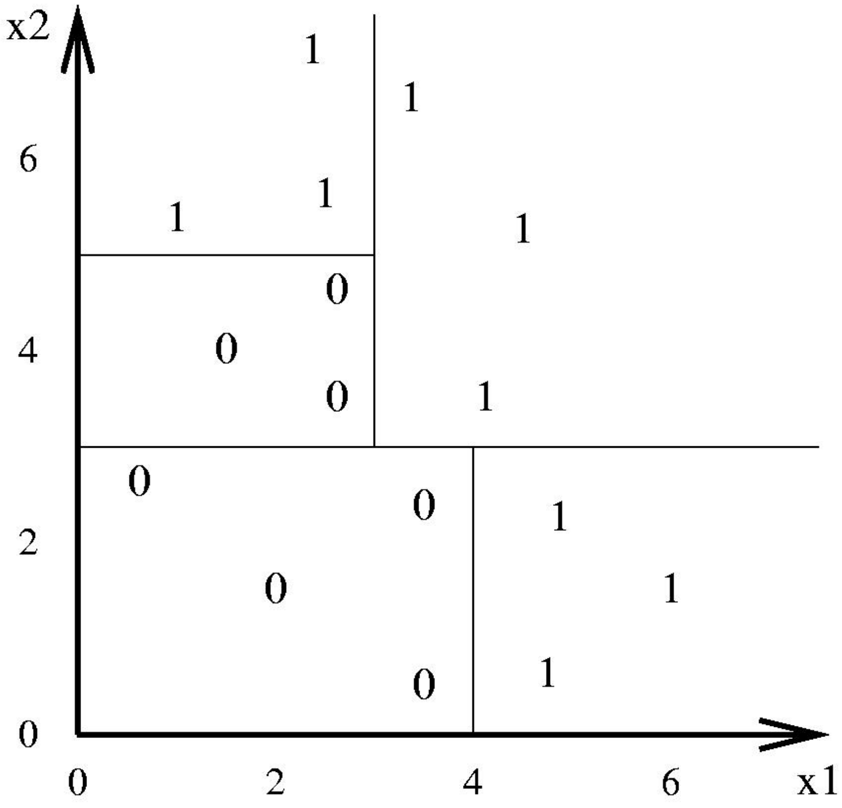
Decision trees divide the feature space into axis-parallel rectangles, and label each rectangle with one of the K classes.

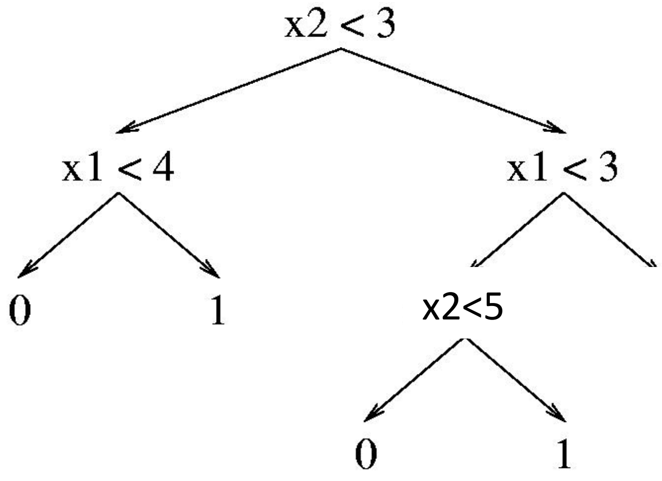




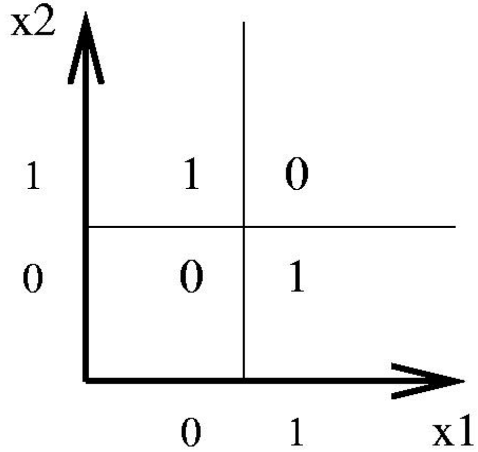
Decision Tree Decision Boundaries

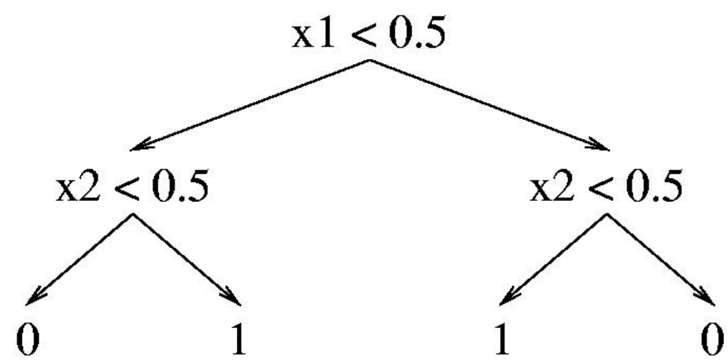
Decision trees divide the feature space into axis-parallel rectangles, and label each rectangle with one of the K classes.





Decision Trees can represent any boolean function





The tree will in the worst case require exponentially many nodes, however.

Can you put a bound on the number of leaf nodes?

Decision Tree Hypothesis Spaces

As the number of nodes (or depth) of tree increases, the hypothesis space grows

- **depth 1** ("decision stump") can represent any boolean function of one feature.
- **depth 2** Any boolean function of two features; some boolean functions involving three features (e.g., $(x_1 \wedge x_2) \vee (\neg x_1 \wedge \neg x_3)$)
- **etc.**

Decision Tree Learning Algorithm

The same basic learning algorithm has been discovered by many people independently:

GrowTree(S)
if ($y = 0$ for all $\langle \mathbf{x}, y \rangle \in S$) **return** new leaf(0)
else if ($y = 1$ for all $\langle \mathbf{x}, y \rangle \in S$) **return** new leaf(1)
else
    choose best attribute x_j
    $S_0 =$ all $\langle \mathbf{x}, y \rangle \in S$ with $x_j = 0$;
    $S_1 =$ all $\langle \mathbf{x}, y \rangle \in S$ with $x_j = 1$;
    return new node(x_j, GrowTree(S_0), GrowTree(S_1))

The following questions may arise in your mind!

- How to choose the best attribute?
  - Which property to test at a node
- When to declare a particular node as leaf?
- What types of trees should we prefer, smaller, larger, balanced, etc?
- If a leaf node is impure (has both positive and negative classes), what should we do?
- What if some attribute value is missing?

Choosing the best Attribute?

- Fundamental principle underlying tree creation
  - Simplicity (prefer smaller trees)
  - Occam's Razor: Simplest model that explains the data should be preferred
- Each node divides the data into subsets
  - Heuristic: Make each subset as **pure as possible**.

Choosing the best Attribute: Information Gain Heuristic

$$Gain(S, A) = H(S) - \sum_{v \in Values(A)} P(v) H(S_v)$$

- Entropy, denoted by H is a measure of impurity
- Gain = Current impurity – New impurity
  - Reduction in impurity
  - Maximize gain
- Second term actually gives expected entropy (weigh each bin by the amount of data in it)

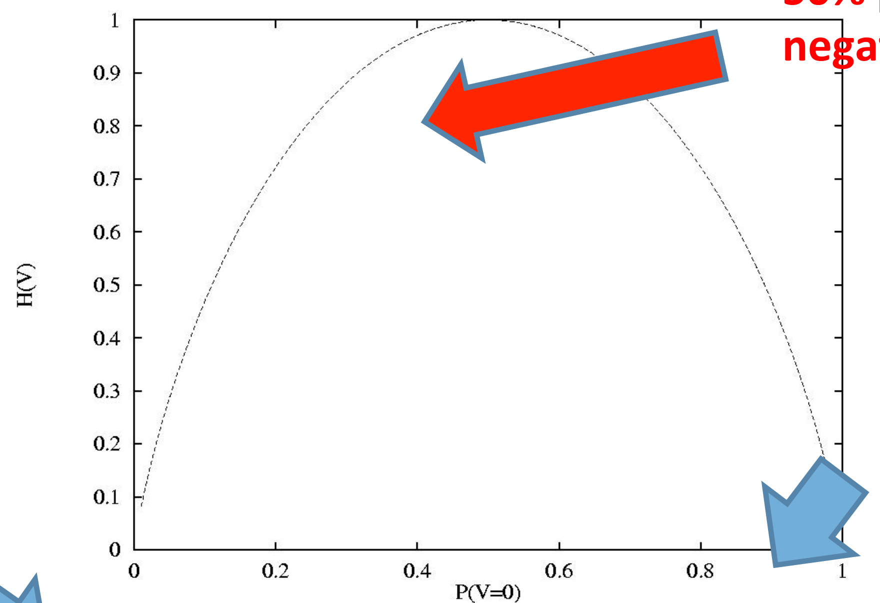
Entropy

The *entropy* of V, denoted $H(V)$ is defined as follows:

$$H(V) = \sum_{v=0}^{1} -P(H = v) \lg P(H = v).$$

This is the average surprise of describing the result of one “trial” of V (one coin toss).

50% positive and 50% negative examples



All examples are positive

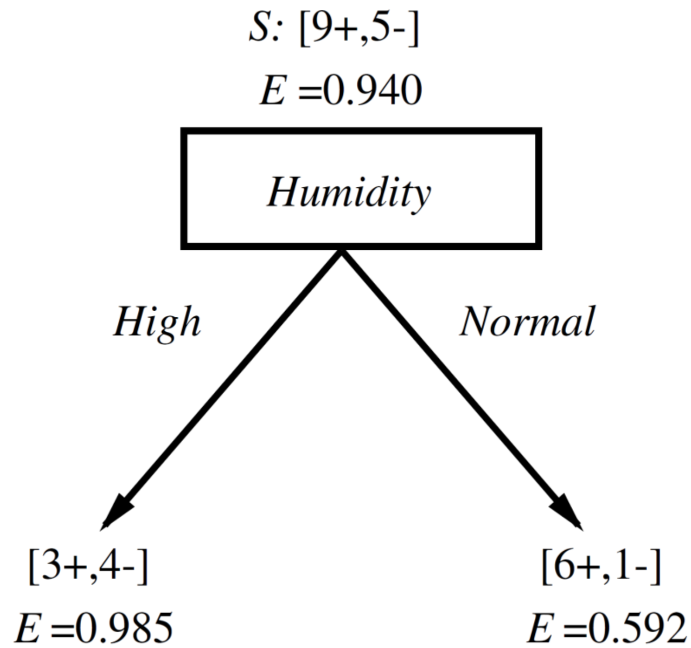
All examples are negative

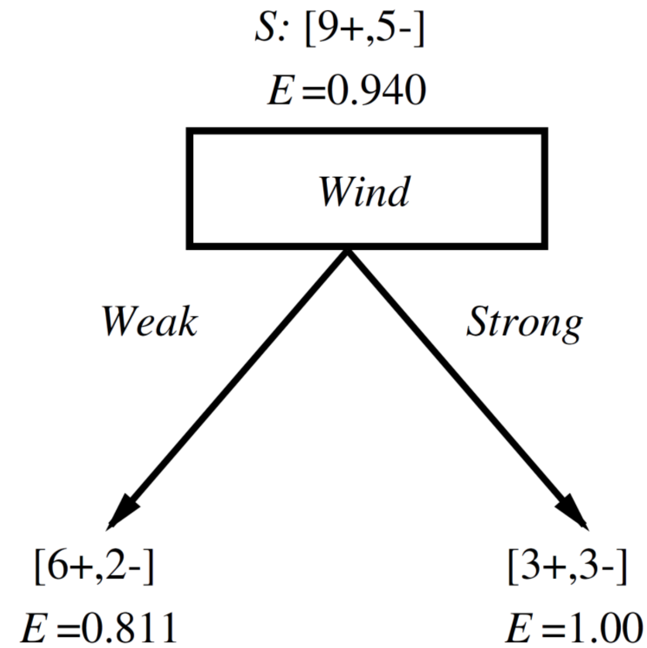
Entropy can be viewed as a measure of uncertainty.

When do I play tennis?

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
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

When do I play tennis?





Gain (S, Humidity)

$= .940 - (7/14).985 - (7/14).592$
$= .151$

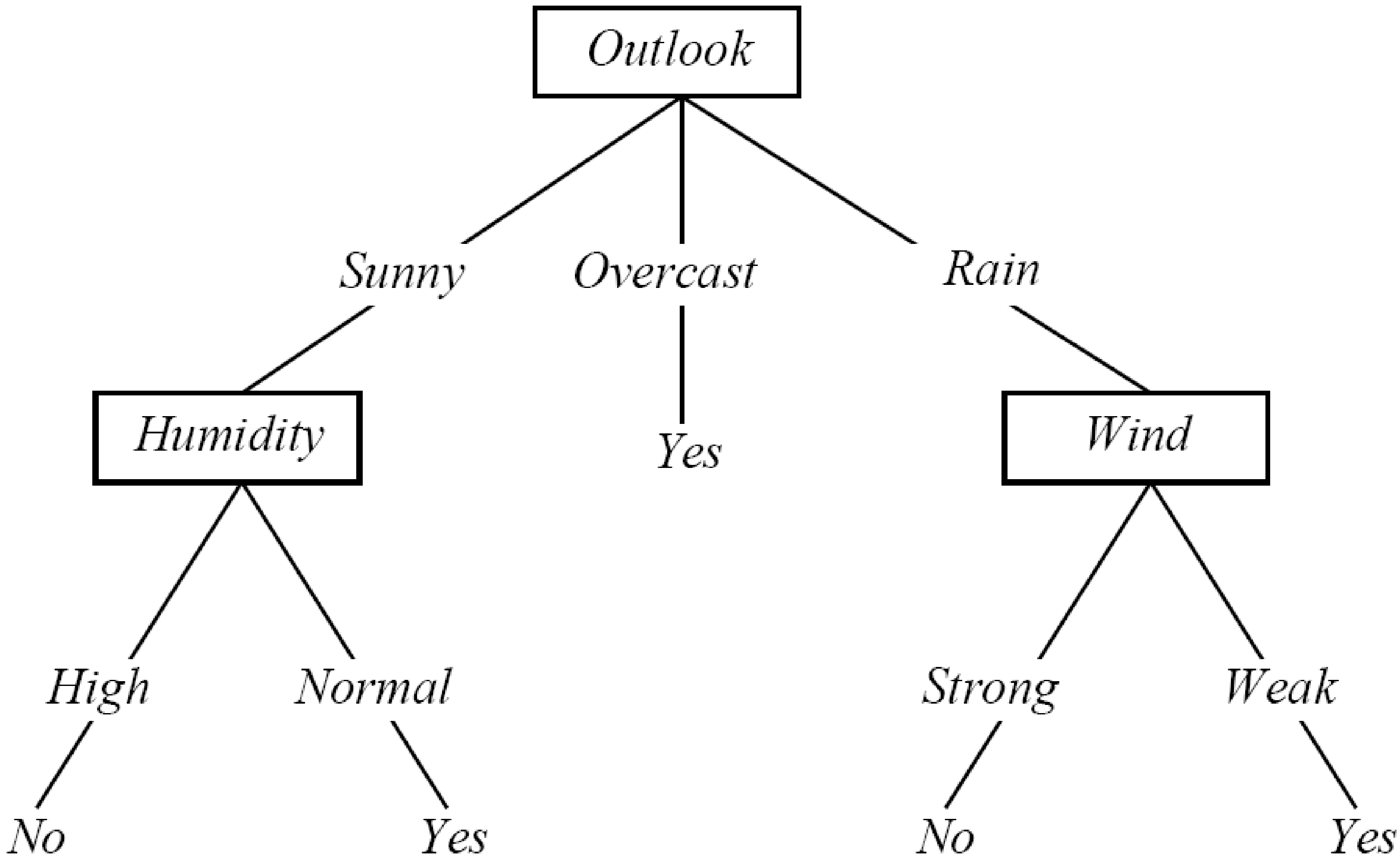
Gain (S, Wind)

$= .940 - (8/14).811 - (6/14)1.0$
$= .048$

When do I play tennis?

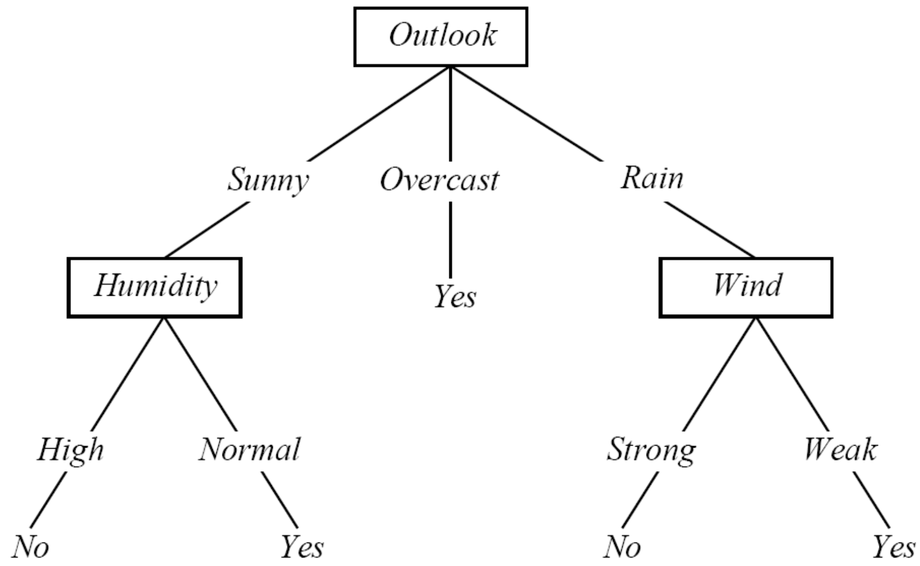
Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	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

Decision Tree



Is the decision tree correct?

- Let’s check whether the split on Wind attribute is correct.
- We need to show that Wind attribute has the highest information gain.



When do I play tennis?

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
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

Wind attribute – 5 records match

Note: calculate the entropy only on examples that got "routed" in our branch of the tree (Outlook=Rain)

Day					PlayTennis
D1					No
D2					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

ID3 Decision Tree Algorithm

ID3(Examples, Target_Attribute, Attributes)

- Create a *Root* node for the tree.
- If all Examples are positive, Return the single-node tree Root, with label = +.
- If all Examples are negative, Return the single-node tree Root, with label = 1.
- If Attributes is empty, Return the single-node tree Root, with label = most common value of Target_attribute in Examples.
- Otherwise
  - A <- attribute from Attributes that best classifies examples
  - The decision attribute for Root <- A
  - For each possible value, v_i of A,
    - Add a new tree branch below Root, corresponding to the test A = v_i.
    - Let Examples_{vi} be the subset of Examples that have value v_i for A
    - If Examples_{vi} is empty,
      - Then add a new leaf node with label = most common value of Target_attribute in Examples.
    - Else, below this new branch add the subtree
      - ID3(Examples_{vi}, Target_attribute, Attributes – {A})
- End
- Return Root

ID3 Decision Tree ALgorithm

- 1) Establish Classification Attribute
- 2) Compute Classification Entropy.
- 3) For each attribute in R, calculate Information Gain using classification attribute.
- 4) Select Attribute with the highest gain to be the next Node in the tree (starting from the Root node).
- 5) Remove Node Attribute, creating reduced table R_S.
- 6) Repeat steps 3-5 until all attributes have been used, or the same classification value remains for all rows in the reduced table.

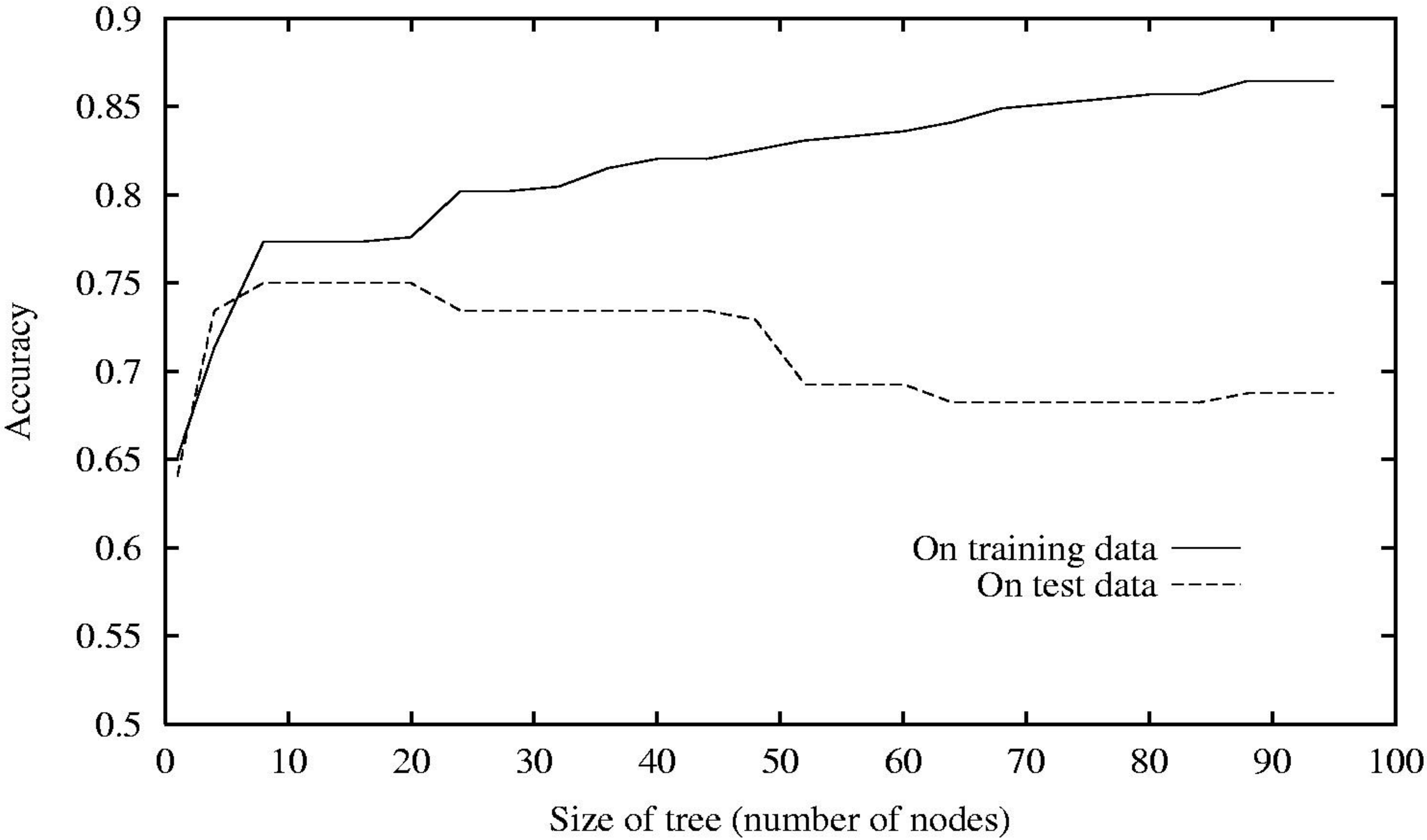
Decision Tree Example

Model	Engine	SC/Turbo	Weight	Fuel Eco	Fast
Prius	small	no	average	good	no
Civic	small	no	light	average	no
WRX STI	small	yes	average	bad	yes
M3	medium	no	heavy	bad	yes
RS4	large	no	average	bad	yes
GTI	medium	no	light	bad	no
XJR	large	yes	heavy	bad	no
S500	large	no	heavy	bad	no
911	medium	yes	light	bad	yes
Corvette	large	no	average	bad	yes
Insight	small	no	light	good	no
RSX	small	no	average	average	no
IS350	medium	no	heavy	bad	no
MR2	small	yes	average	average	no
E320	medium	no	heavy	bad	no

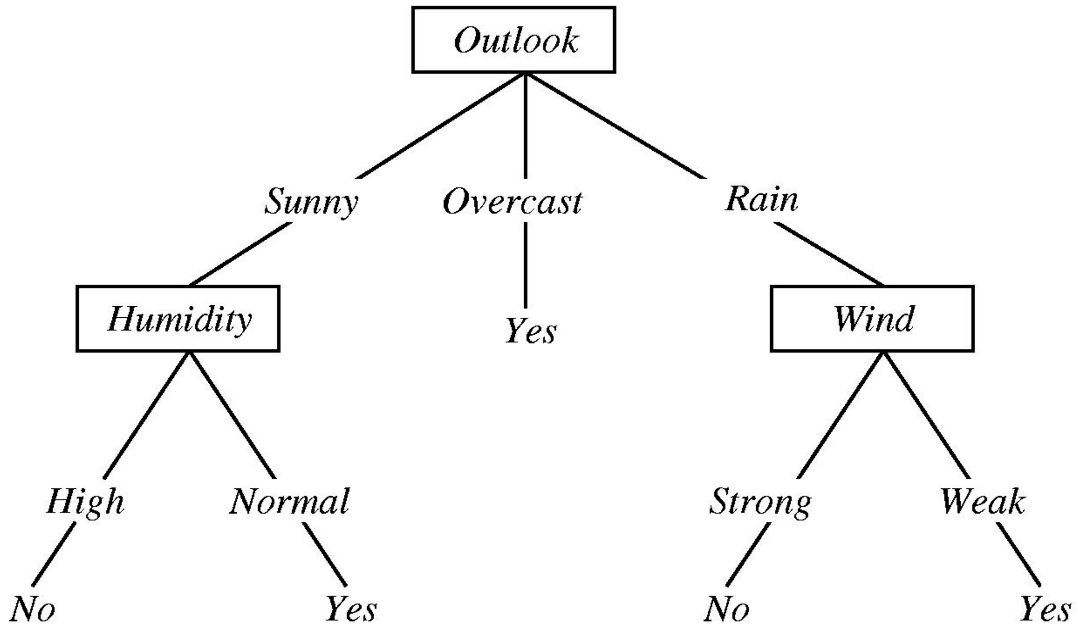
Practical Issues in Decision Tree Learning

- Overfitting
- When to stop growing a tree?
- Handling non-Boolean attributes
- Handling missing attribute values

Overfitting in Decision Tree Learning



Overfitting in Decision Trees



Consider adding a noisy training example:

Sunny, Hot, Normal, Strong, PlayTennis=No

What effect on tree?

Overfitting

Consider error of hypothesis h over

- training data: $error_{train}(h)$
- entire distribution $\mathcal{D}$ of data: $error_{\mathcal{D}}(h)$

Hypothesis $h \in H$ **overfits** training data if there is an alternative hypothesis $h' \in H$ such that

$$error_{train}(h) < error_{train}(h')$$

and

$$error_{\mathcal{D}}(h) > error_{\mathcal{D}}(h')$$

Sources of Overfitting

- Noise
- Small number of examples associated with each leaf
  - What if only one example is associated with a leaf. Can you believe it?
  - Coincidental regularities
- **Generalization** is the most important criteria
  - Your method should work well on examples which you have not seen before.

Avoiding Overfitting

- Two approaches
  - Stop growing the tree when data split is not statistically significant
  - Grow tree fully, then post-prune
- Key Issue: What is the correct tree size?
  - Apply statistical test to estimate whether expanding a particular node is likely to produce an improvement beyond the training set
    - E.g., chi-squared test
  - Add a complexity penalty
  - Divide data into training and validation set

Chi-square Pruning

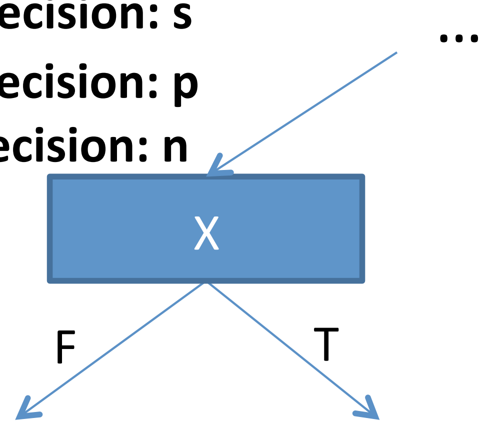
- Build entire tree
- Consider each leaf decision and perform the chi-squared test (label vs. split variable).

Chi-square Pruning

of instances entering this decision: s

of + instances entering this decision: p

of - instances entering this decision: n



 # instances here: s_f

of + instances here: p_f

of - instances here: n_f

 # instances here: s_t

of - instances here: p_t

of - instances here: n_t

Hypothesis: **X is uncorrelated with the decision**

Then p_f should be "close" to (s_f * p/s)

And p_t should be "close" to (s_t * p/s)

Expected counts

Similarly for n_f and n_t

Training and Validation Split

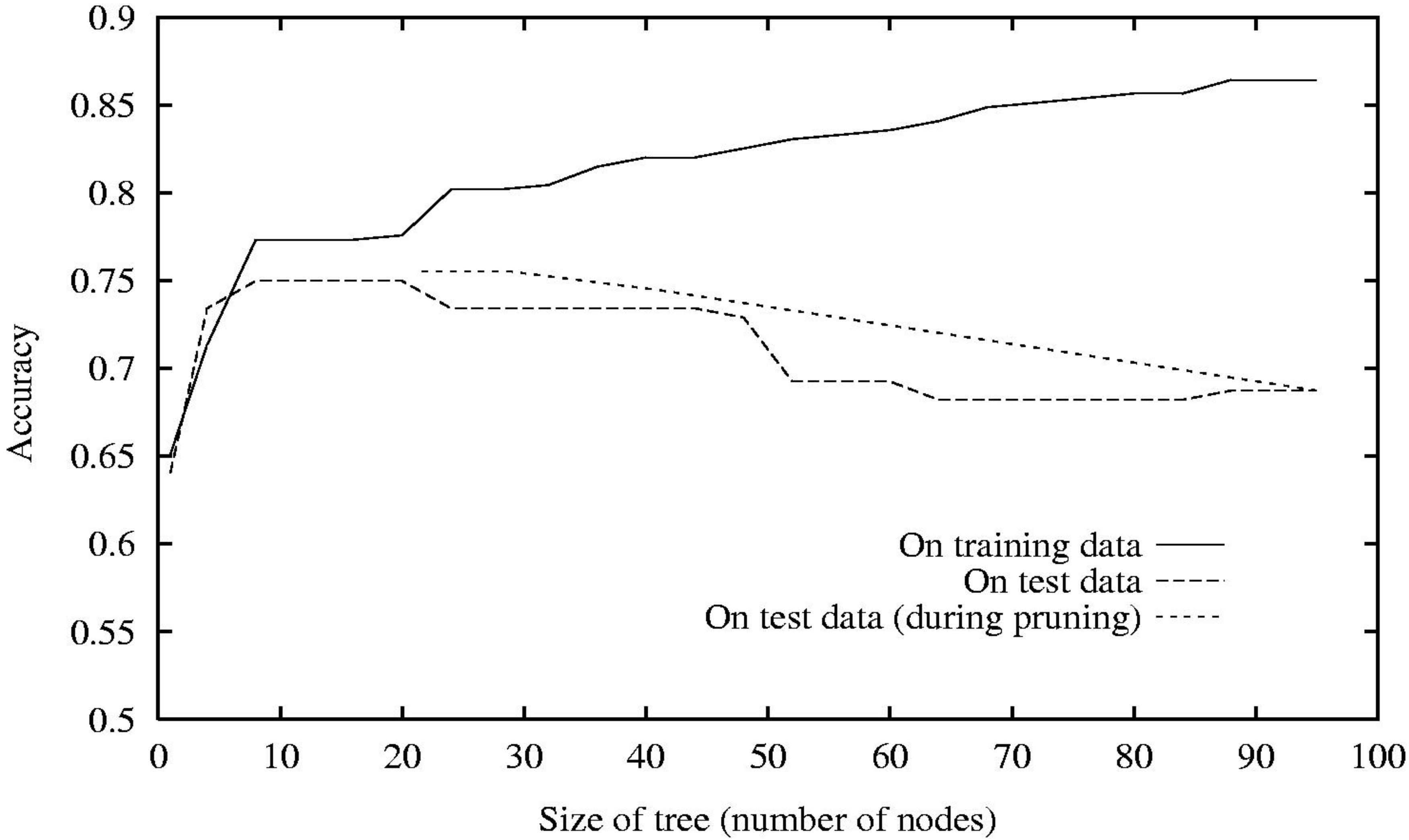
- Typical approach: 2/3rd of the data for training and 1/3rd for validation
- Learn on the training data. Evaluate impact of pruning on the validation set
- Why this works?
  - Training and validation set are unlikely to exhibit the same random fluctuations caused by co-incidental regularities and random errors
  - Must be large enough to serve as a safety check and therefore the 2/3rd -1/3rd split!

Pruning Approach 1

Reduced-Error Pruning

- Split data into training and validation set
- Do until further pruning is harmful:
  - Evaluate impact on validation set of pruning each possible node (plus those below it)
  - Greedily remove the one that most improves validation set accuracy

Effect of Reduced-Error Pruning



Rule Post Pruning

- Induce the decision tree using the full training set (allowing it to overfit)
- Convert the decision tree to a set of rules
- Prune each rule by removing pre-conditions that improve the estimated accuracy
  - Estimate accuracy using a validation set
- Sort the rules using their estimated accuracy
- Classify new instances using the sorted sequence

Rule Post Pruning

IF (Outlook == Sunny) $\wedge$ (Humidity == High)

THEN PlayTennis = NO

- Each rule pruned by removing antecedent/precondition.
- Preconditions: (Outlook == Sunny), (Humidity == High)
- Select whichever pruning results in improved accuracy

Rule Post Pruning

- Allows distinguishing between different contexts
  – Each path, distinct rule.
  – Pruning decisions regarding attributes can be performed differently for each path
- Removes distinction between attribute tests near the root and those near the leaves.
  – Reduces bookkeeping issues regarding re-organizing tree

Non-Boolean Features

- Features with multiple discrete values
  - Construct a multiway split?
  - Test for one value versus all of the others?
  - Group the values two disjoint subsets?
- Real-valued features
  - Consider a threshold split using each observed value of the feature

Attributes with Many Values

Problem:

- If attribute has many values, $Gain$ will select it
- Imagine using $Date = Jun\_3\_1996$ as attribute

One approach: use $GainRatio$ instead

$$GainRatio(S, A) \equiv \frac{Gain(S, A)}{SplitInformation(S, A)}$$

$$SplitInformation(S, A) \equiv - \sum_{i=1}^{c} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}$$

where S_i is subset of S for which A has value v_i

Non-Boolean Attributes

- Continuous Attributes
  - Split using a threshold (Boolean split)
  - What threshold to use?
    - Sort the examples according to the attribute and consider all mid-way points at which the classification changes as a threshold

Temp	40	48	60	72	80	90
Play Tennis?	NO	NO	YES	YES	YES	NO

Two features: Threshold=54; Threshold=85

- Other option: Split into multiple intervals

Handling Missing Values

- Some attribute-values are missing
  - Example: patient data. You don't expect blood test results for everyone.
- Treat the missing value as another value
- Ignore instances having missing values
  - Problematic because throwing away data
- Assign it the most common value
- Assign it the most common value based on the class that the example belongs to.

Handling Missing Values: Probabilistic Approach

- Assign a probability to each possible value of attribute "A"
- Let us assume that A(x=1)=0.4 and A(x=0)=0.6
  - A fractional 0.4 of instance goes to branch A(x=1) and 0.6 to branch A(x=0)
  - Use fractional instances to compute gain

Summary: Decision Trees

- Representation
- Tree growth
- Choosing the best attribute
- Overfitting and pruning
- Special cases: Missing Attributes and Continuous Attributes
- Many forms in practice: CART, ID3, C4.5