

Marcel Gehrke

Data Understanding vs. Machine Training

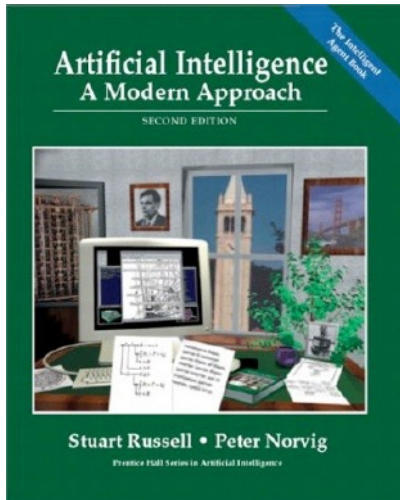
Inductive Learning

Inductive Learning

Version Space

Inductive Learning

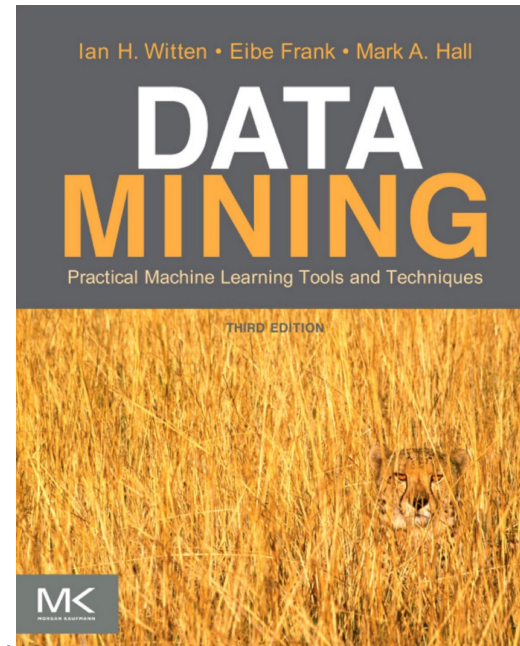
Chapter 18/19



Material adopted from
Yun Peng, Chuck Dyer,
Gregory Piatetsky-Shapiro & Gary Parker

Inductive Learning

Chapters 3 and 4



Marcel Gehrke

Card Example: Guess a Concept

- Given a set of examples
 - Positive: e.g., 4♣ 7♣ 2♠
 - Negative: e.g., 5♥ j♠
- What cards are accepted?
 - What concept lays behind it?

Card Example: Guess a Concept

■ $(r=1) \vee \dots \vee (r=10) \vee (r=J) \vee (r=Q) \vee (r=K) \Leftrightarrow \text{ANY-RANK}(r)$

$(r=1) \vee \dots \vee (r=10) \Leftrightarrow \text{NUM}(r)$

$(r=J) \vee (r=Q) \vee (r=K) \Leftrightarrow \text{FACE}(r)$

$(s=\spadesuit) \vee (s=\clubsuit) \vee (s=\diamondsuit) \vee (s=\heartsuit) \Leftrightarrow \text{ANY-SUIT}(s)$

$(s=\spadesuit) \vee (s=\clubsuit) \Leftrightarrow \text{BLACK}(s)$

$(s=\diamondsuit) \vee (s=\heartsuit) \Leftrightarrow \text{RED}(s)$

A hypothesis is any sentence of the form:

$R(r) \wedge S(s)$, where:

- $R(r)$ is ANY-RANK(r), NUM(r), FACE(r), or $(r=x)$
- $S(s)$ is ANY-SUIT(s), BLACK(s), RED(s), or $(s=y)$

Simplified Representation

For simplicity, we represent a concept by rs , with:

- $r \in \{a, n, f, 1, \dots, 10, j, q, k\}$
- $s \in \{a, b, r, \clubsuit, \spadesuit, \diamondsuit, \heartsuit\}$

For example:

- $n\spadesuit$ represents: $\text{NUM}(r) \wedge (s=\spadesuit)$
- aa represents: $\text{ANY-RANK}(r) \wedge \text{ANY-SUIT}(s)$

Extension of a Hypothesis

The **extension** of a hypothesis h is the set of objects that satisfies h

Examples:

- The extension of $f♠$ is: $\{j♠, q♠, k♠\}$
- The extension of aa is the set of all cards

Inductive Learning

Most General/Specific Hypothesis

More General/Specific Relation

- Let h_1 and h_2 be two hypotheses in H
- h_1 is **more general** than h_2 iff the extension of h_1 is a proper superset of the extension of h_2

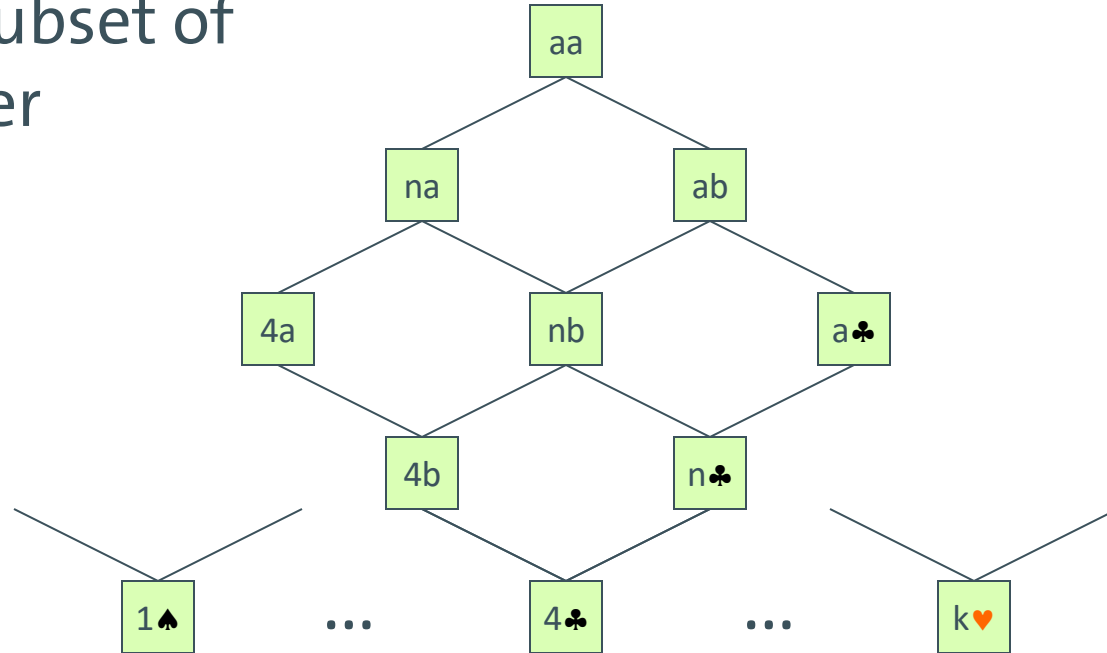
Examples:

- aa is more general than f ♦
- f♥ is more general than q♥
- fr and nr are not comparable

More General/Specific Relation

- Let h_1 and h_2 be two hypotheses in H
- h_1 is **more general** than h_2 iff the extension of h_1 is a proper superset of the extension of h_2
- The inverse of the “more general” relation is the “**more specific**” relation
- The “more general” relation defines a **partial ordering** on the hypotheses in H

Example: Subset of Partial Order

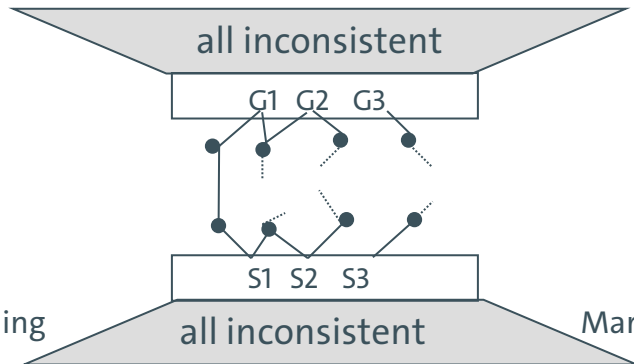


G-Boundary / S-Boundary of V

- A hypothesis in V is **most general** iff no hypothesis in V is more general
- **G-boundary** G of V : Set of most general hypotheses in V

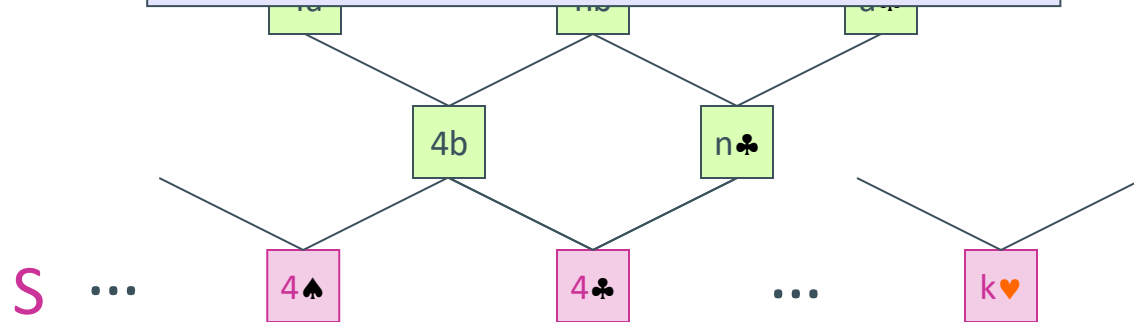
G-Boundary / S-Boundary of V

- A hypothesis in V is **most general** iff no hypothesis in V is more general
- **G-boundary** G of V: Set of most general hypotheses in V
- A hypothesis in V is **most specific** iff no hypothesis in V is more specific
- **S-boundary** S of V: Set of most specific hypotheses in V



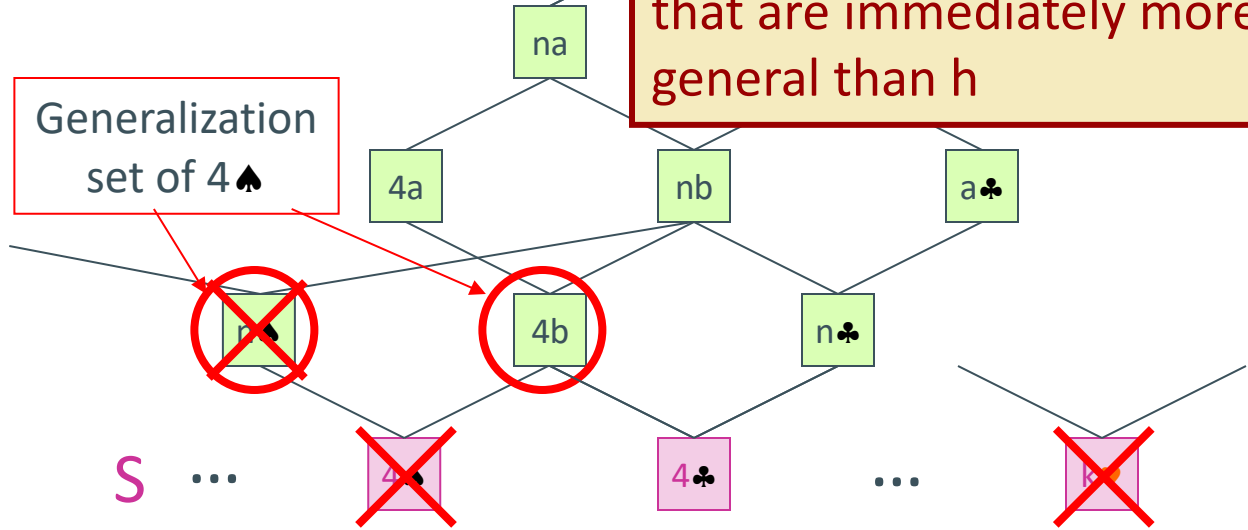
Example: **G**
G-/S-Boundaries

We replace every hypothesis in S whose extension does not contain 4♣ by its generalization set



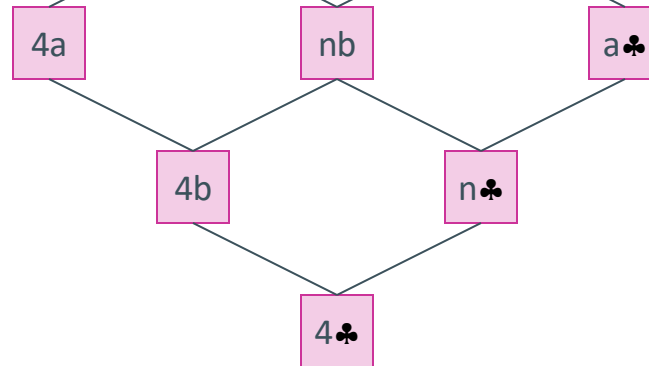
Example: G G-/S-Boundaries of V

The **generalization set** of an hypothesis h is the set of the hypotheses that are immediately more general than h



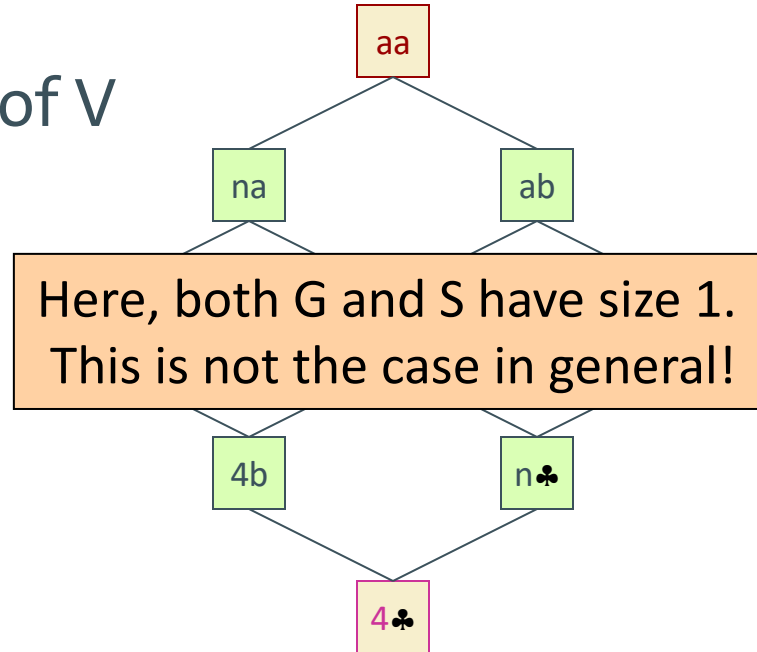
Example: **G**
G-/S-Boundaries

We remove every hypothesis in S that is more general than another hypothesis in S



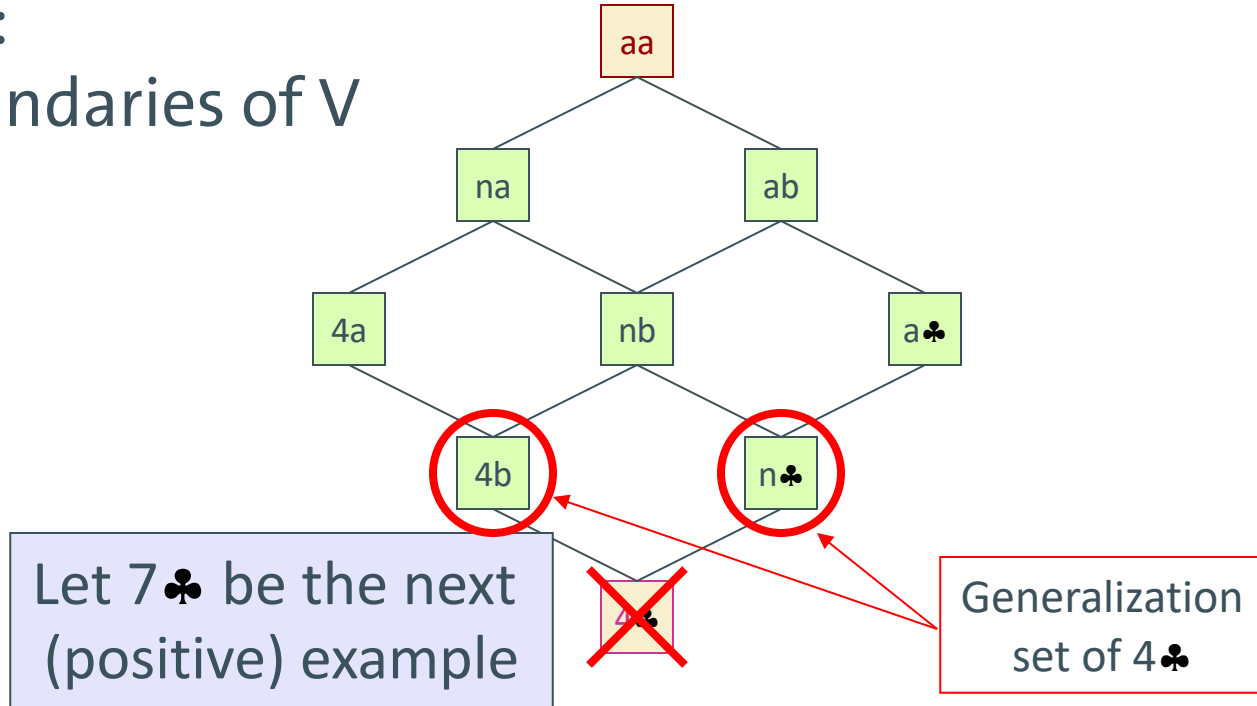
S

Example: **G** G-/S-Boundaries of V



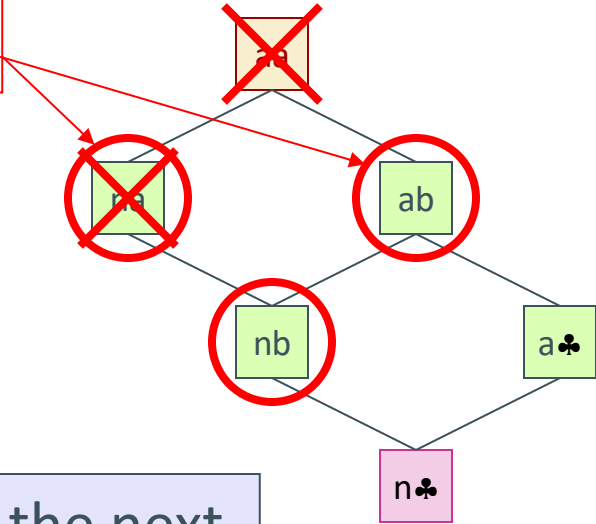
S

Example: G-/S-Boundaries of V



Example: G-/S-Boundaries of V

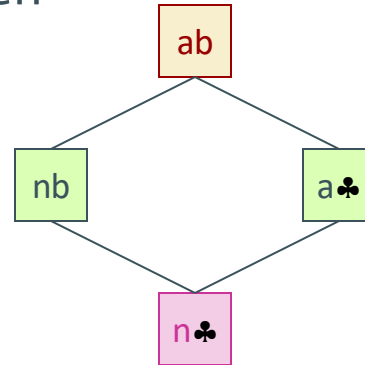
Specialization
set of aa



Let 5♥ be the next
(negative) example

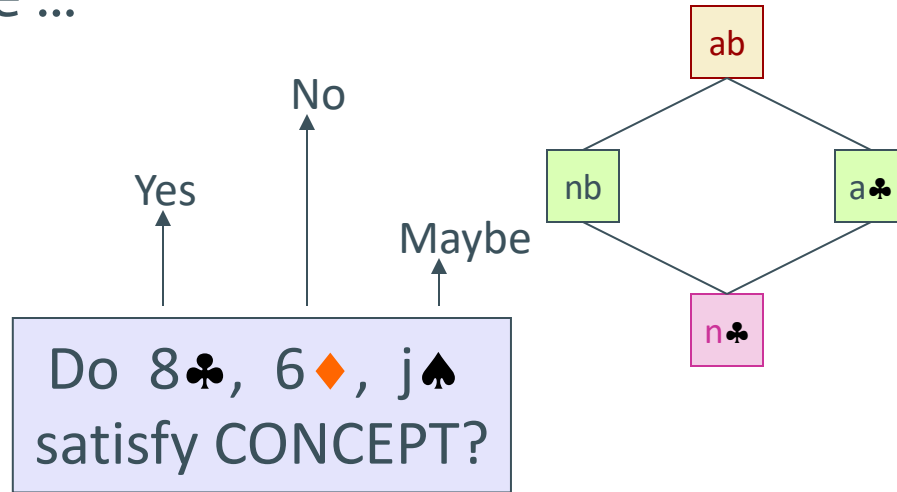
Example: G-/S-Boundaries of V

G and S, and all hypotheses in between form exactly the version space

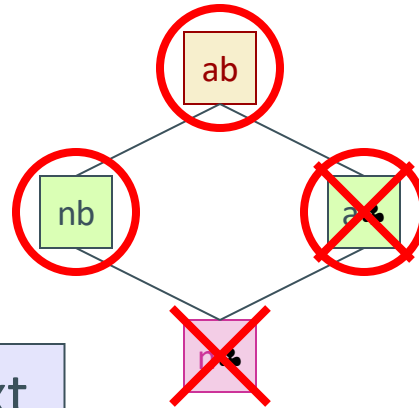


Example: G-/S-Boundaries of V

At this stage ...

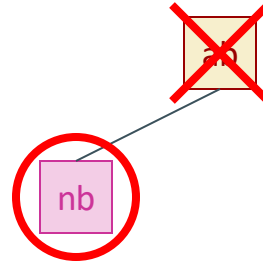


Example: G-/S-Boundaries of V



Let $2\spadesuit$ be the next
(positive) example

Example: G-/S-Boundaries of V



Let j^{\spadesuit} be the next
(negative) example

Example: G-/S-Boundaries of V

+ 4♣ 7♣ 2♠

− 5♥ j♠

nb

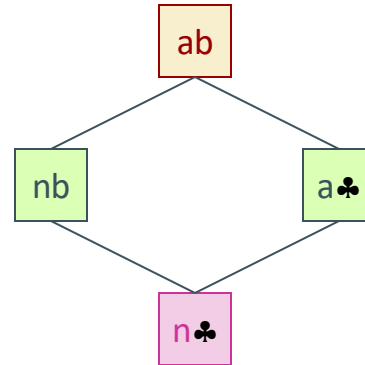
$\text{NUM}(r) \wedge \text{BLACK}(s)$

Example: G-/S-Boundaries of V

Let us return to the
version space ...

... and let $8♣$ be the next
(negative) example

The only most specific
hypothesis disagrees with
this example, so no
hypothesis in H agrees with
all examples

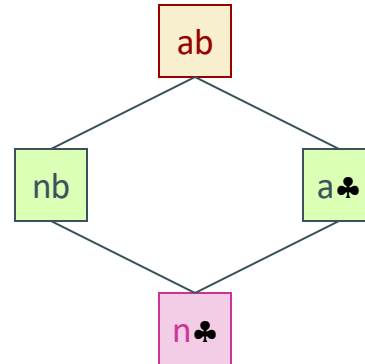


Example: G-/S-Boundaries of V

Let us return to the
version space ...

... and let $j \heartsuit$ be the next
(positive) example

The only most general
hypothesis disagrees with
this example, so no
hypothesis in H agrees with
all examples



Inductive Learning

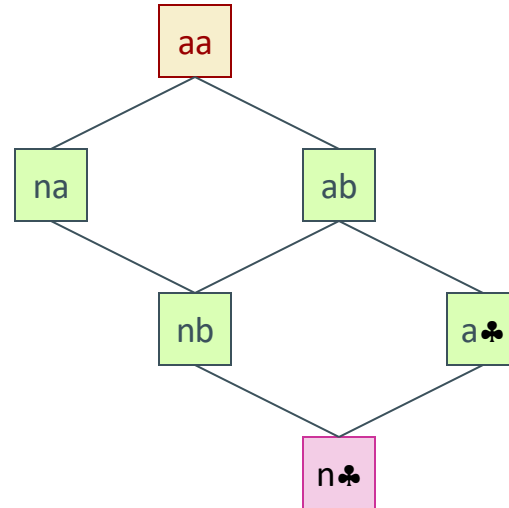
Selection Strategy

Example-Selection Strategy

- Suppose that at each step the learning procedure has the possibility to select the object (card) of the next example
- Let it pick the object such that, whether the example is positive or not, it will eliminate one-half of the remaining hypotheses
- Then a single hypothesis will be isolated in $O(\log |H|)$ steps

Example

- 9♣?
- j♥?
- j♣?



Example-Selection Strategy

- Suppose that at each step the learning procedure has the possibility to select the object (card) of the next example
- Let it pick the object such that, whether the example is positive or not, it will eliminate one-half of the remaining hypotheses
- Then a single hypothesis will be isolated in $O(\log |H|)$ steps
- But picking the object that eliminates half the version space may be expensive

Noise

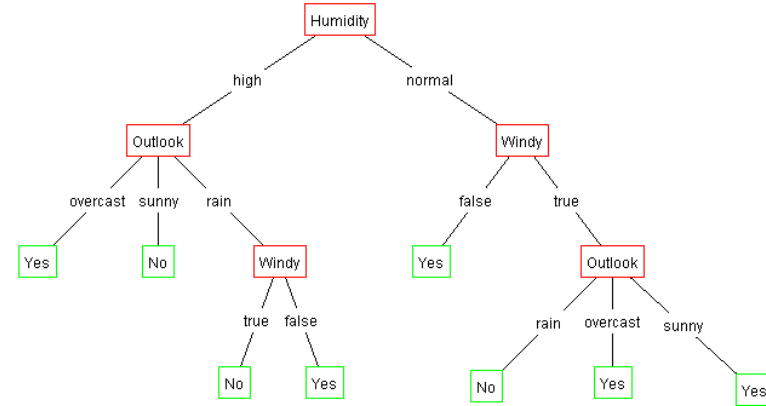
- If some examples are **misclassified**, the version space may collapse
- **Possible solution:**
Maintain several G- and S-boundaries, e.g., consistent with all examples, all examples but one, etc...

Inductive Learning

Decision Trees

Decision Trees

Outlook	Temperature	Humidity	Windy	Play?
sunny	hot	high	false	No
sunny	hot	high	true	No
overcast	hot	high	false	Yes
rain	mild	high	false	Yes
rain	cool	normal	false	Yes
rain	cool	normal	true	No
overcast	cool	normal	true	Yes
sunny	mild	high	false	No
sunny	cool	normal	false	Yes
rain	mild	normal	false	Yes
sunny	mild	normal	true	Yes
overcast	mild	high	true	Yes
overcast	hot	normal	false	Yes
rain	mild	high	true	No



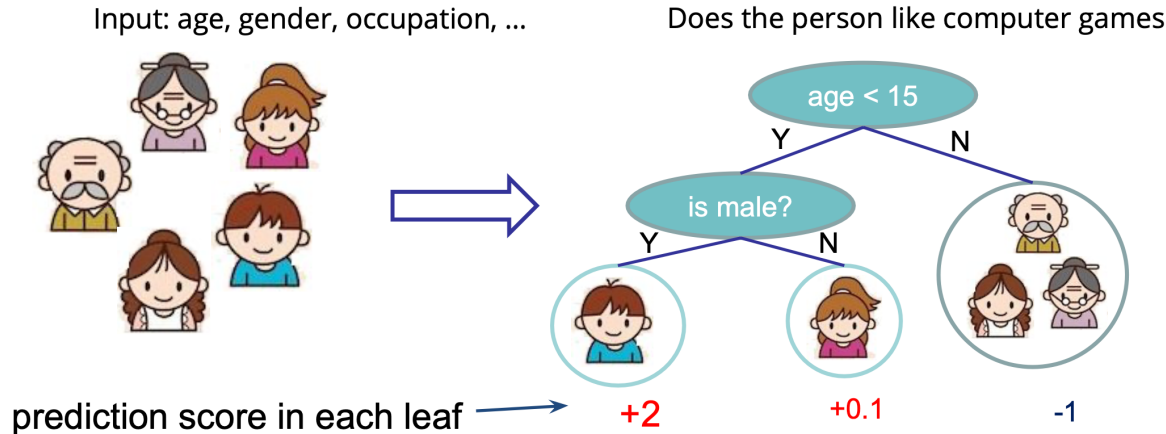
Decision Trees

- An **internal node** is a **test on an attribute**.
- A **branch** represents an **outcome of the test**, e.g., Color=red.
- A **leaf node** represents a **class label**

- At each node, one attribute is chosen to split training examples into distinct classes as much as possible
- A new case is classified by following a matching path to a leaf node.

Regression Trees

- Regression tree (also known as CART)
- This is what it would look like for a commercial system



Building Decision Trees

1. Top-down tree construction

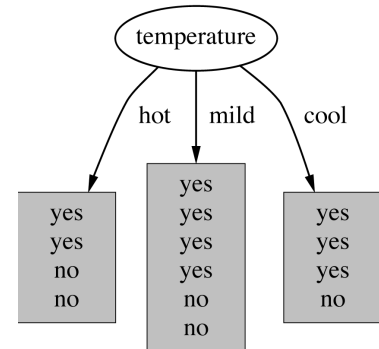
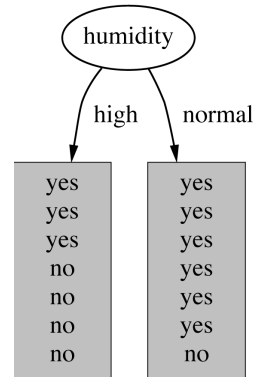
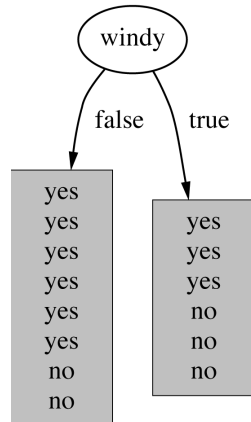
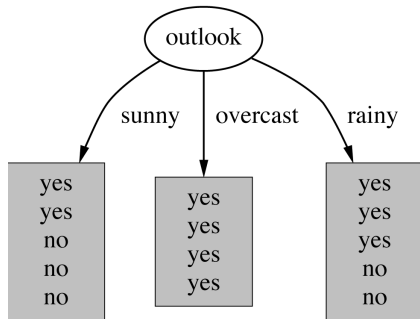
- At start, all training examples are at the root.
- Partition the examples recursively by choosing one attribute each time.

2. Bottom-up tree pruning

- Remove subtrees or branches, in a bottom-up manner, to improve the estimated accuracy on new cases.

R. Quinlan, Learning efficient classification procedures, Machine Learning: an artificial intelligence approach, Michalski, Carbonell & Mitchell (eds.), Morgan Kaufmann, p. 463-482., 1983

Which attribute to select?



Choosing the Best Attribute

- The key problem is choosing which attribute to split a given set of examples.
- Some possibilities are:
 - **Random:** Select any attribute at random
 - **Least-Values:** Choose the attribute with the smallest number of possible values
 - **Most-Values:** Choose the attribute with the largest number of possible values
 - **Information gain:** Choose the attribute that has the largest expected information gain, i.e., select attribute that will result in the smallest expected size of the subtrees rooted at its children.

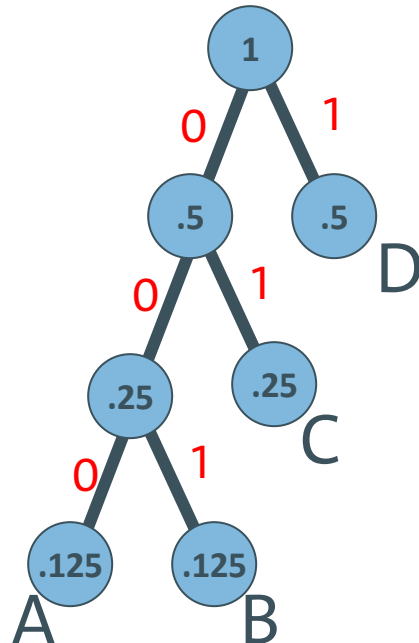
Inductive Learning

Information Theory

Information Theory

- Assume you can bet 1\$ for a coin flip (10000 bets), if your bet is right, you get back 2\$ otherwise you get nothing
- You know that the coin used is rigged and comes up heads with probability 0.99, so you bet heads - obviously (but find somebody arranging this bet :))
- The expected value for the bet is 1.98\$
- How much will you be willing to pay for the advance information about the actual outcome of the flip? What the value of the advance information?
- Less than 0.02\$!
- If the coin were fair, your expected value would 1\$ and you would be willing to pay up to 1\$
- The less you know, the more valuable the information
- Information theory does not measure the value of information in \$ but the information content of a message in bits.

Huffman Code Example



M	code	length	prob	Exp. len
A	000	3	0,125	0,375
B	001	3	0,125	0,375
C	01	2	0,250	0,500
D	1	1	0,500	0,500

average message length

1,750

If we need to send many messages (A,B,C or D) and they have this probability distribution and we use this code, then over time, the average bits/message should approach **1.75**

Information Theory Background

- If there are n equally probable possible messages, then the probability p of each is $1/n$
- Information (number of bits) conveyed by a message is $\log(n) = -\log(p)$
- Eg, if there are 16 messages, then $\log(16) = 4$ and we need 4 bits to identify/send each message.
- In general, if we are given a probability distribution

$$P = (p_1, p_2, \dots, p_n)$$

- the information conveyed by the distribution (aka **entropy** of P) is:

$$I(P) = -(p_1 \cdot \log(p_1) + p_2 \cdot \log(p_2) + \dots + p_n \cdot \log(p_n)) = - \sum_i p_i * \log(p_i)$$

Information Theory Background

- Information conveyed by distribution (aka **entropy** of P) is:

$$I(P) = -(p_1 \cdot \log(p_1) + p_2 \cdot \log(p_2) + \dots + p_n \cdot \log(p_n))$$


- Examples:

- if P is $(0.5, 0.5)$ then $I(P)$ is 1
 - if P is $(0.67, 0.33)$ then $I(P)$ is 0.92,
 - if P is $(1, 0)$ or $(0, 1)$ then $I(P)$ is 0.
- The more uniform is the probability distribution, the greater is its information
 - The **entropy** is the **average number of bits/message** needed to represent a stream of messages

Example: Attribute “Outlook”, 1

Outlook	Temperature	Humidity	Windy	Play?
sunny	hot	high	false	No
sunny	hot	high	true	No
overcast	hot	high	false	Yes
rain	mild	high	false	Yes
rain	cool	normal	false	Yes
rain	cool	normal	true	No
overcast	cool	normal	true	Yes
sunny	mild	high	false	No
sunny	cool	normal	false	Yes
rain	mild	normal	false	Yes
sunny	mild	normal	true	Yes
overcast	mild	high	true	Yes
overcast	hot	normal	false	Yes
rain	mild	high	true	No

Example: attribute “Outlook”, 2

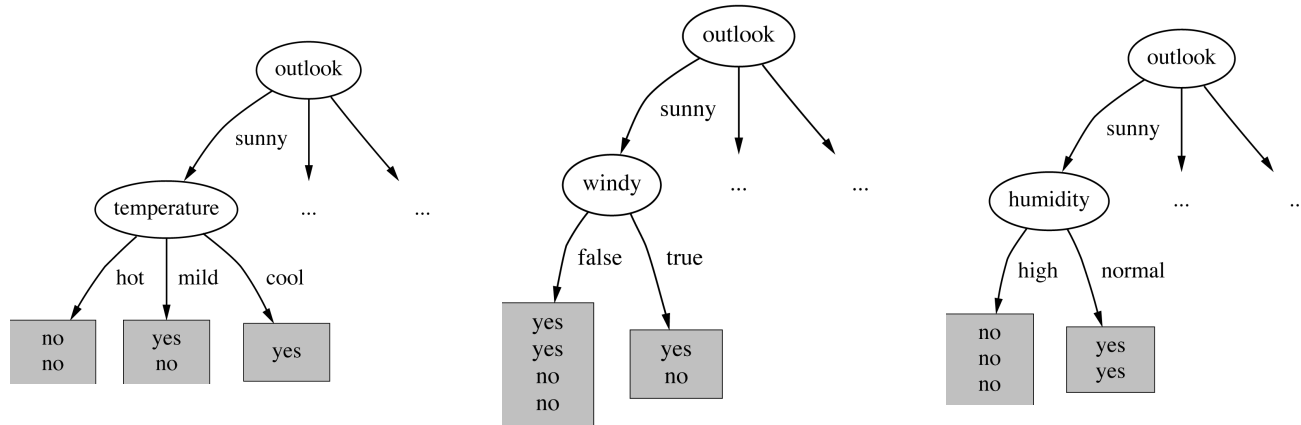
- “Outlook” = “Sunny”:
 $info([2,3]) = entropy\left(\frac{2}{5}, \frac{3}{5}\right) = -\frac{2}{5}\log\left(\frac{2}{5}\right) - \frac{3}{5}\log\left(\frac{3}{5}\right) = 0.971 \text{ bit}$
- “Outlook” = “Overcast”:
 $info([4,0]) = entropy(1,0) = -1\log(1) - 0\log(0) = 0 \text{ bits}$

- “Outlook” = “Rainy”:
 $info([3,2]) = entropy\left(\frac{3}{5}, \frac{2}{5}\right) = -\frac{3}{5}\log\left(\frac{3}{5}\right) - \frac{2}{5}\log\left(\frac{2}{5}\right) = 0.971 \text{ bits}$
- Expected information for attribute:
 $info([2,3], [4,0], [3,2]) = \left(\frac{5}{14}\right) \cdot 0.971 + \left(\frac{4}{14}\right) \cdot 0 + \left(\frac{5}{14}\right) \cdot 0.971 = 0.693 \text{ bits}$

Note: $\log(0)$ is not defined, but we evaluate $0 \cdot \log(0)$ as zero

Computing the Information Gain

- Information gain:
(information before split) – (information after split)
 $gain(\text{"Outlook"}) = info([9,5]) - info([2,3], [4,0], [3,2]) = 0.940 - 0.693 = 0.247 \text{ bits}$
- Information gain for attributes from weather data:
 $gain(\text{"Outlook"}) = 0.247 \text{ bits}$
 $gain(\text{"Temperature"}) = 0.029 \text{ bits}$
 $gain(\text{"Humidity"}) = 0.152 \text{ bits}$
 $gain(\text{"Windy"}) = 0.048 \text{ bits}$

Continuing to Split



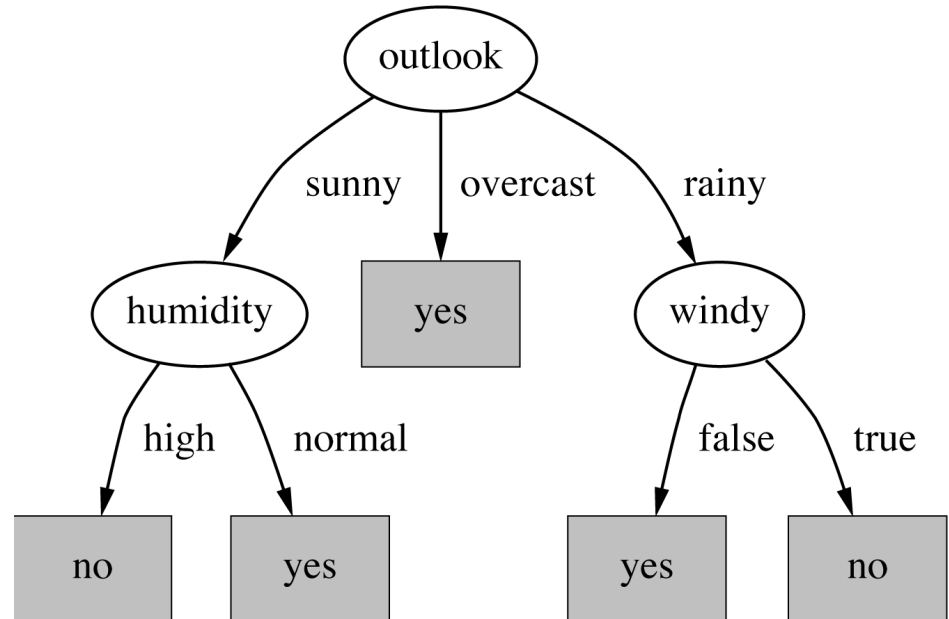
$gain("Temperature") = 0.571 \text{ bits}$

$gain("Humidity") = 0.971 \text{ bits}$

$gain("Windy") = 0.020 \text{ bits}$

The Final Decision Tree

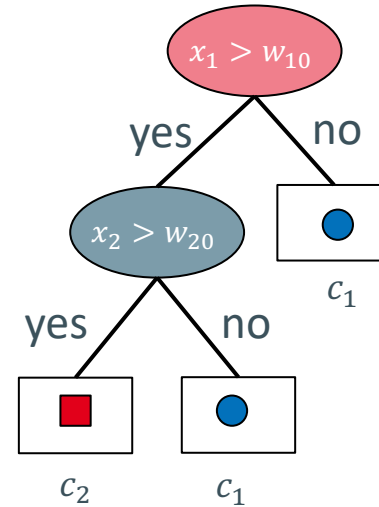
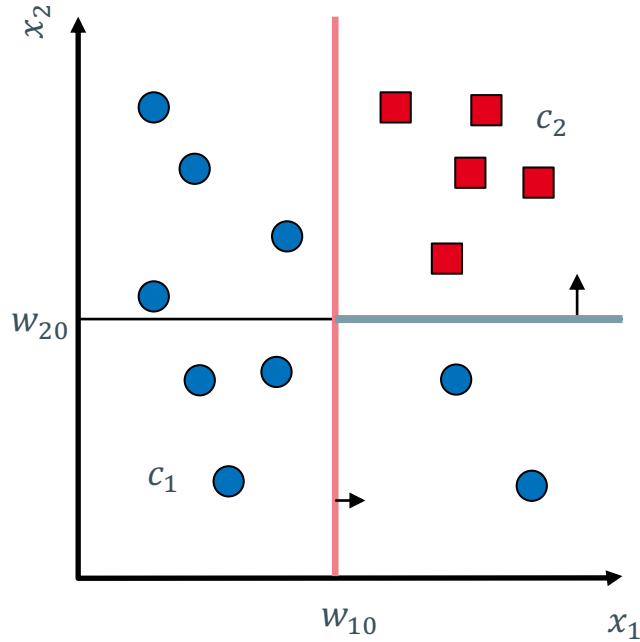
- Note: not all leaves need to be pure; sometimes identical instances have different classes
⇒ Splitting stops when data can't be split any further



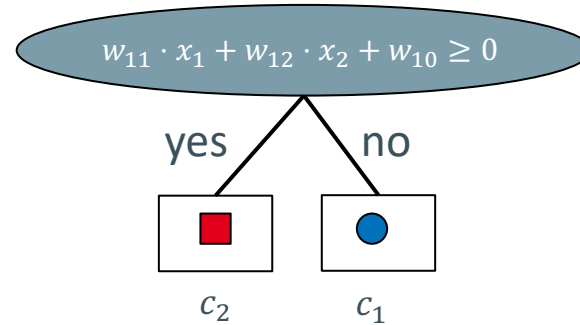
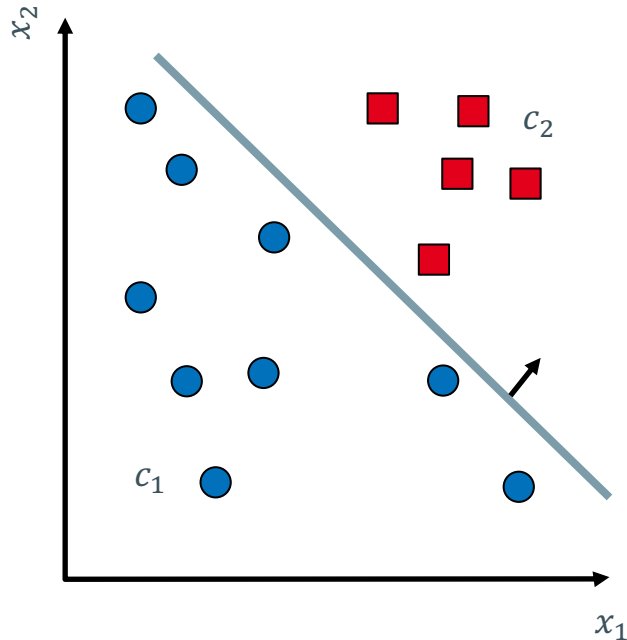
Inductive Learning

Evaluating Decision Trees

Univariate Splits



Multivariate Splits



Classification

1R – Simplicity First!

Given: Table with data

Goal: Learn decision function

- Based on rules that all test one particular attribute
- One branch for each value
- Each branch assigns most frequent class
- Error rate: proportion of instances that do not belong to the majority class of their corresponding branch
- Choose attribute with lowest error rate

(Assumes nominal attributes)

Outlook	Temp	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

Evaluating the Weather Attributes

Classification

Outlook	Temp	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

Attribute	Rules	Errors	Total errors
Outlook	Sunny → No	2/5	4/14
	Overcast → Yes	0/4	
	Rainy → Yes	2/5	
Temp	Hot → No*	2/4	5/14
	Mild → Yes	2/6	
	Cool → Yes	1/4	
Humidity	High → No	3/7	4/14
	Normal → Yes	1/7	
Windy	False → Yes	2/8	5/14
	True → No*	3/6	

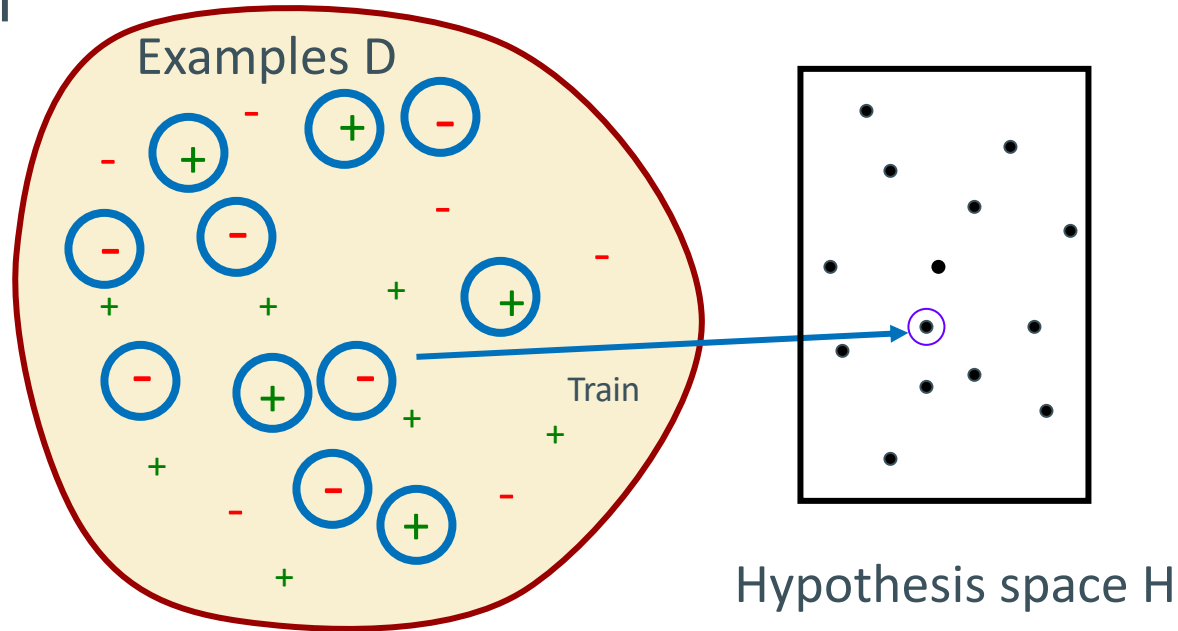
* indicates a tie

Assessing Performance of a Learning Algorithm

- Take out some of the training set
 - Train on the remaining training set
 - Test on the excluded instances
 - *Cross-validation*

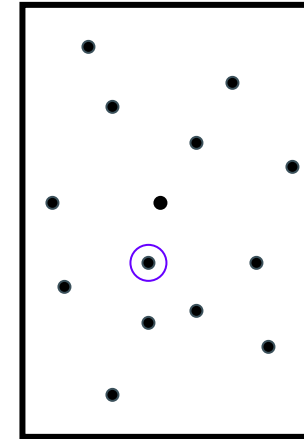
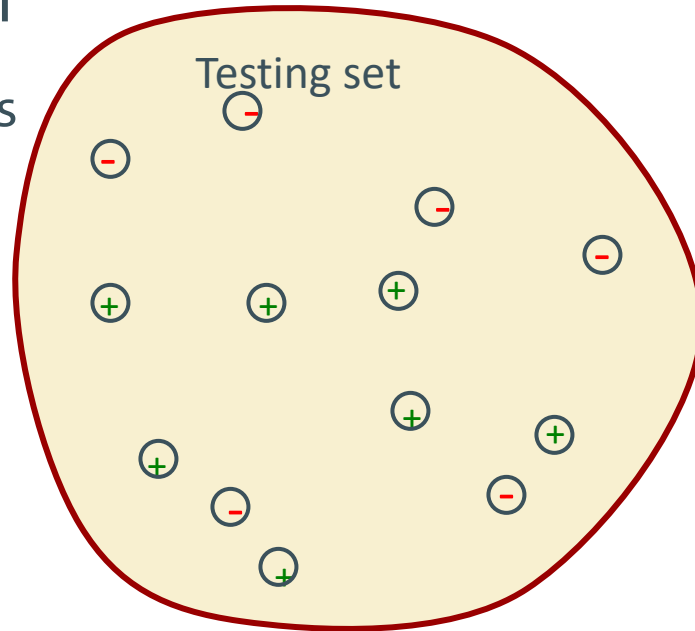
Cross-Validation

- Split original set of examples, train



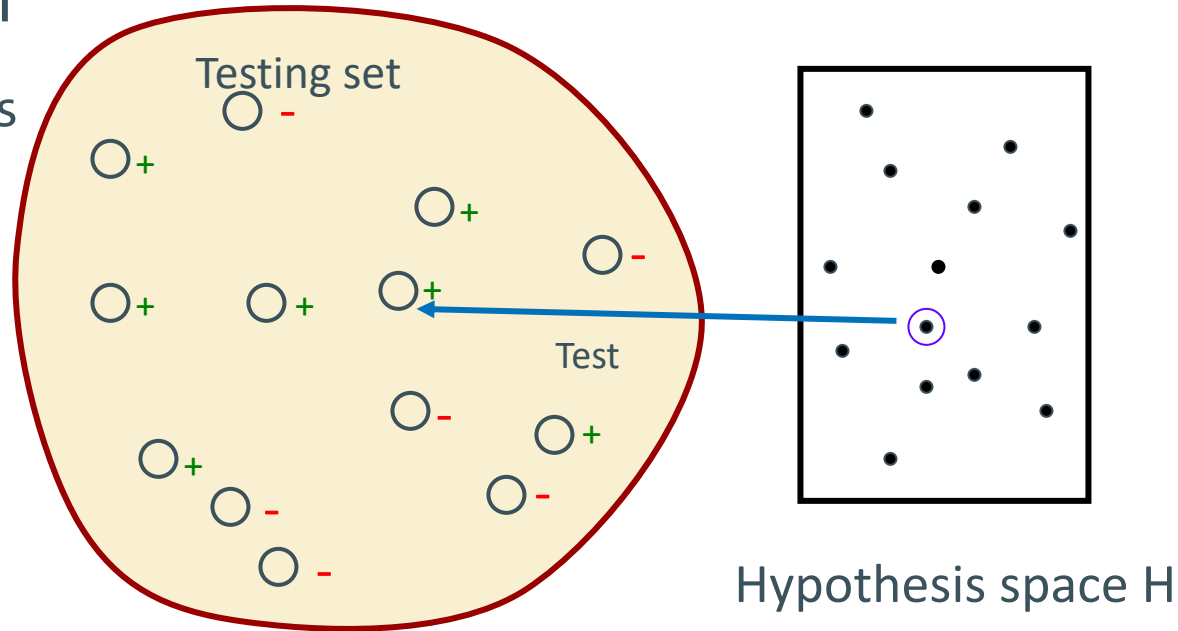
Cross-Validation

- Evaluate hypothesis on testing set



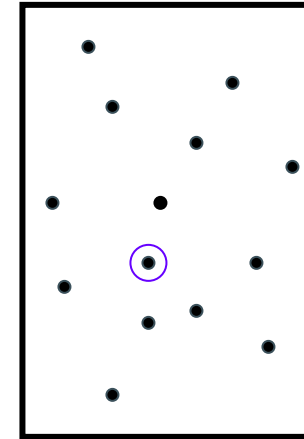
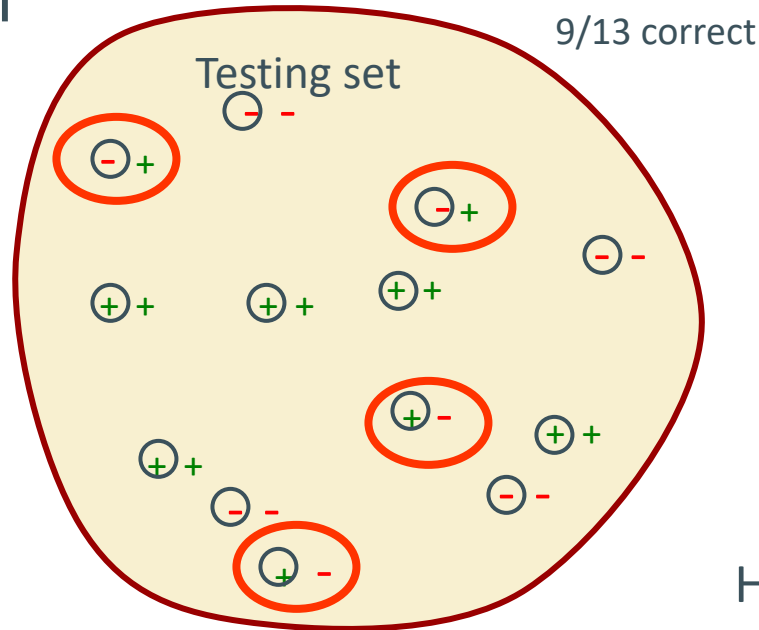
Cross-Validation

- Evaluate hypothesis on testing set



Cross-Validation

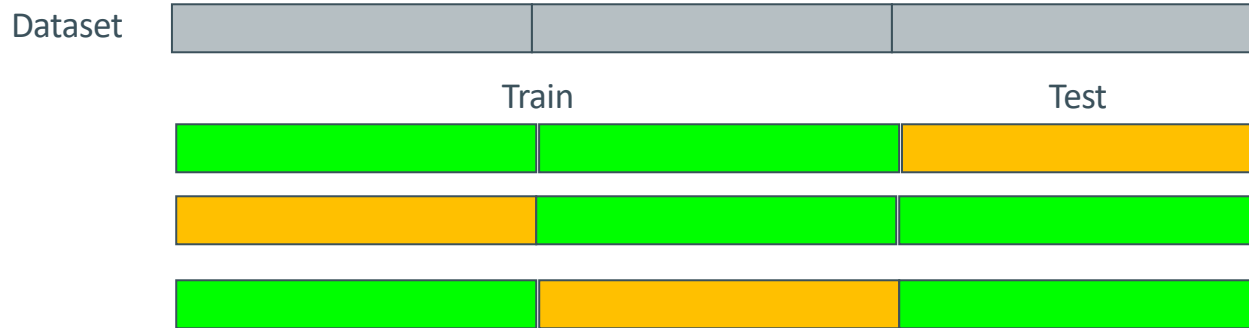
- Compare true concept against prediction



Hypothesis space H

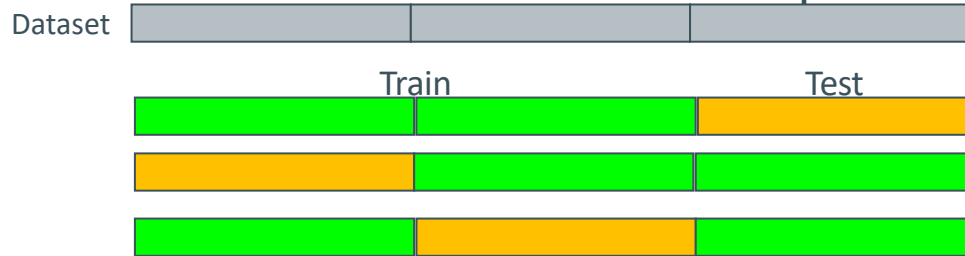
Common Splitting Strategies

- k-fold cross-validation: k random partitions



Common Splitting Strategies

- k-fold cross-validation: k random partitions



- Leave-p-out: all possible combinations of p instances



Discussion of 1R

- 1R was described in a paper by Holte (1993)
 - Contains an experimental evaluation on 16 datasets (using *cross-validation* so that results were representative of performance on future data)
 - Minimum number of instances was set to 6 after some experimentation
 - 1R's simple rules performed not much worse than much more complex classifiers
- Simplicity first pays off!

Robert C. Holte, *Very Simple Classification Rules Perform Well on Most Commonly Used Datasets*, Journal Machine Learning Volume 11, Issue 1, pp 63-90, 1993

From ID3 to C4.5: History

- ID3 (Quinlan) – 1960s
- CHAID (Chi-squared Automatic Interaction Detector) – 1960s
- CART (Classification And Regression Tree)
 - Uses another split heuristics (Gini impurity measure)
- C4.5 innovations (Quinlan):
 - Permit numeric attributes
 - Deal with missing values
 - Pruning to deal with noisy data
- C4.5 - one of best-known and most widely-used learning algorithms
 - Last research version: C4.8, implemented in Weka as J4.8 (Java)
 - Commercial successor: C5.0 (available from Rulequest)

Dealing with Numeric (Metric) Attributes

Discretize numeric attributes

Divide each attribute's range into intervals

- Sort instances according to attribute's values
 - Place breakpoints where the class changes
- This minimizes the total error

Outlook	Temperature	Humidity	Windy	Play
Sunny	85.3	85	False	No
Sunny	80.2	90	True	No
Overcast	83.8	86	False	Yes
Rainy	75.2	80	False	Yes
...

64 65 68 69 70 71 72 72 75 75 80 81 83 85

The Problem of Overfitting

- This procedure is very sensitive to noise
 - One instance with an incorrect class label will probably produce a separate interval
- Also: *time stamp* attribute will have zero errors
- Simple solution:
enforce minimum number of instances in majority class per interval

Discretization Example

- Example (with $\text{min} = 3$):

64	65	68	69	70	71	72	72	75	75	80	81	83	85
Yes	No	Yes	Yes	Yes	No	No	Yes	Yes	Yes	No	Yes	Yes	No







Same decision for both intervals

- Final result for temperature attribute

64	65	68	69	70	71	72	72	75	75	80	81	83	85
Yes	No	Yes	Yes	Yes	No	No	Yes	Yes	Yes	No	Yes	Yes	No



With Overfitting Avoidance

- Resulting rule set:

Attribute	Rules	Errors	Total errors
Outlook	Sunny → No	2/5	4/14
	Overcast → Yes	0/4	
	Rainy → Yes	2/5	
Temperature	≤ 77.5 → Yes	3/10	5/14
	> 77.5 → No*	2/4	
Humidity	≤ 82.5 → Yes	1/7	3/14
	> 82.5 and ≤ 95.5 → No	2/6	
	> 95.5 → Yes	0/1	
Windy	False → Yes	2/8	5/14
	True → No*	3/6	

Numeric Attributes – Advanced

- Standard method: binary splits
 - E.g. **temperature** < 45
- Unlike nominal attributes, every attribute has many possible split points
- Solution is straightforward extension:
 - Evaluate info gain (or other measure) for every possible split point of attribute
 - Choose “best” split point
 - Info gain for best split point is info gain for attribute
- Computationally more demanding

Example

- Split on temperature attribute:

64	65	68	69	70	71		72	72	75	75	80	81	83	85
Yes	No	Yes	Yes	Yes	No		No	Yes	Yes	Yes	No	Yes	Yes	No

- E.g. **temperature < 71.5: yes/4, no/2**
temperature ≥ 71.5: yes/5, no/3

▪

- Place split points halfway between values
- Can evaluate all split points in one pass!

Missing as a Separate Value

- Missing value denoted “?” in C4.X (Null value)
- Simple idea: treat missing as a separate value
- **Q: When is this not appropriate?**
- **A:** When values are missing due to different reasons
 - Example 1: blood sugar value could be missing when it is very high or very low
 - Example 2: field **IsPregnant** missing for a male patient should be treated differently (no) than for a female patient of age 25 (unknown)

Missing Values – Advanced

Questions:

- How should tests on attributes with different unknown values be handled?
- How should the partitioning be done in case of examples with unknown values?
- How should an unseen case with missing values be handled?

Missing Values – Advanced

- Info gain with unknown values during learning
 - Let T be the training set and X a test on an attribute with unknown values and F be the fraction of examples where the value is known
 - Rewrite the gain:
$$\begin{aligned} \text{Gain}(X) &= \text{probability that } A \text{ is known} * (\text{info}(T) - \text{info}_x(T)) + \\ &\quad \text{probability that } A \text{ is unknown} * 0 \\ &= F \cdot (\text{info}(T) - \text{info}_x(T)) \end{aligned}$$
- Consider instances w/o missing values
- Split w.r.t. those instances
- Distribute instances with missing values proportionally

Inductive Learning

Pruning

Pruning

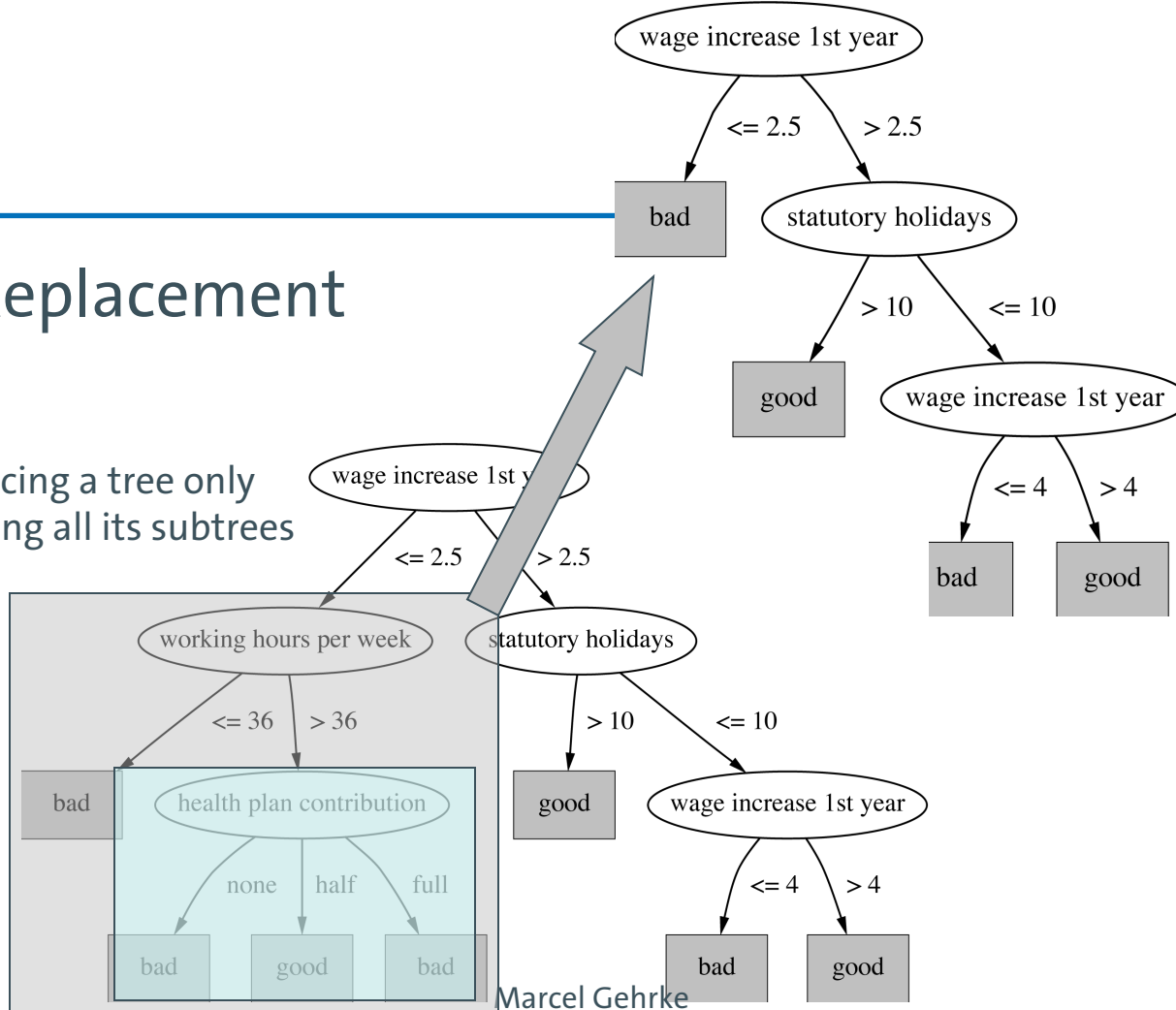
- Goal: Prevent overfitting to noise in the data
- Two strategies for “pruning” the decision tree:
 - Postpruning - take a fully-grown decision tree and discard unreliable parts
 - Prepruning - stop growing a branch when information becomes unreliable
- Postpruning preferred in practice—prepruning can “stop too early”

Post-pruning

- First, build full tree
- Then, prune it
 - Fully-grown tree shows all attribute interactions
- Two pruning operations:
 1. *Subtree replacement*
 2. *Subtree raising*

Subtree Replacement

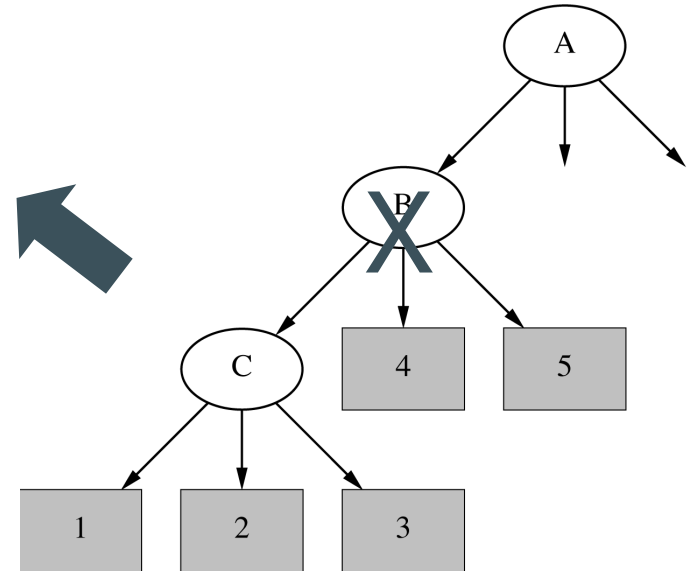
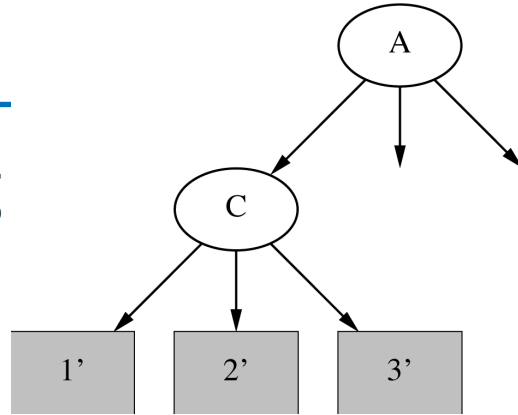
- *Bottom-up*
- Consider replacing a tree only after considering all its subtrees



*Subtree Raising

- Delete node
- Redistribute instances
- Slower than subtree replacement

(Worthwhile?)



Post-pruning

- First, build full tree
 - Then, prune it
 - Fully-grown tree shows all attribute interactions
- Expected Error Pruning

Estimating Error Rates

- Prune only if it reduces the estimated error
- Error on the training data is NOT a useful estimator
 - *Q: Why would it result in very little pruning?*
- Use hold-out set for pruning (“reduced-error pruning”)

Expected Error Pruning

- Approximate expected error assuming that we prune at a particular node.
- Approximate backed-up error from children assuming we did not prune.
- If expected error is less than backed-up error, prune.

Static Expected Error

- If we prune a node, it becomes a leaf labeled C
- What will be the expected classification error at this leaf?

$$Error(S) = \frac{N-n+k-1}{N+k}, \text{ where:}$$

S is the set of examples in a node

k is the number of classes

N examples in S

C the majority class in S

n out of N examples in S belong to C

Laplace error estimate – based on the assumption that the distribution of probabilities that examples will belong to different classes is uniform.

Backed-up Error

- For a non-leaf node *Node*
- Let children of *Node* be *Node*₁, *Node*₂, etc.
 - Probabilities can be estimated by relative frequencies of attribute values in sets of examples that fall into child nodes

$$\text{BackedUpError}(\text{Node}) = \sum_i P_i \cdot \text{Error}(\text{Node}_i)$$
$$\text{Error}(\text{Node}) = \min(\text{Error}(\text{Node}), \text{BackedUpError}(\text{Node}))$$

Example Calculation

- Static Expected Error of b

$$Error([4,2]) = \frac{N - n + k - 1}{N + k}$$

- Left child of b

$$Error([3,2]) = \frac{5 - 3 + 2 - 1}{5 + 2} = 0,429$$

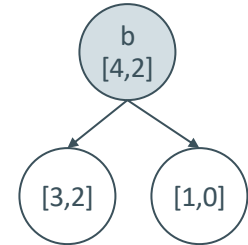
- Right child of b

$$Error([1,0]) = \frac{1 - 1 + 2 - 1}{1 + 2} = 0,333$$

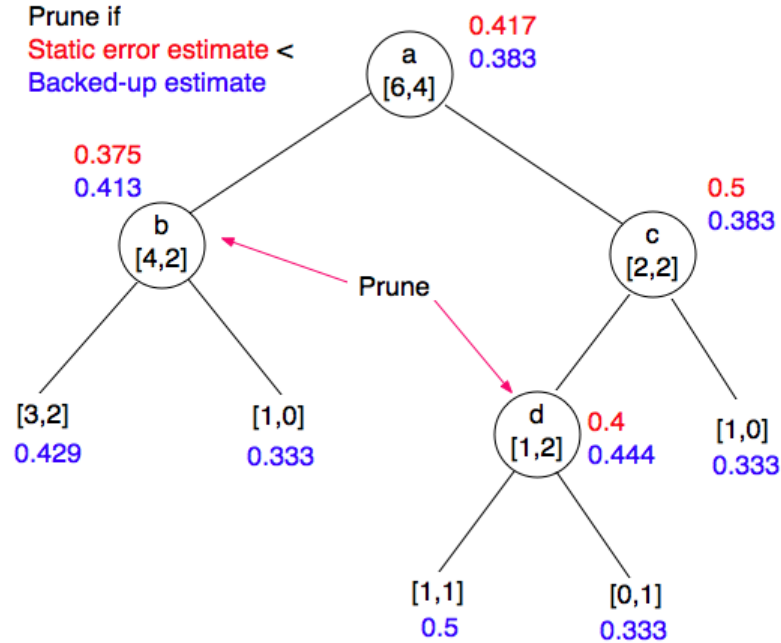
- Backed Up Error of b

$$BackedUpError(b) = \frac{5}{6}Error([3,2]) + \frac{1}{6}Error([1,0]) = 0,413$$

- $0,375 < 0,413 \rightarrow$ Prune tree.



Example



Regression Trees

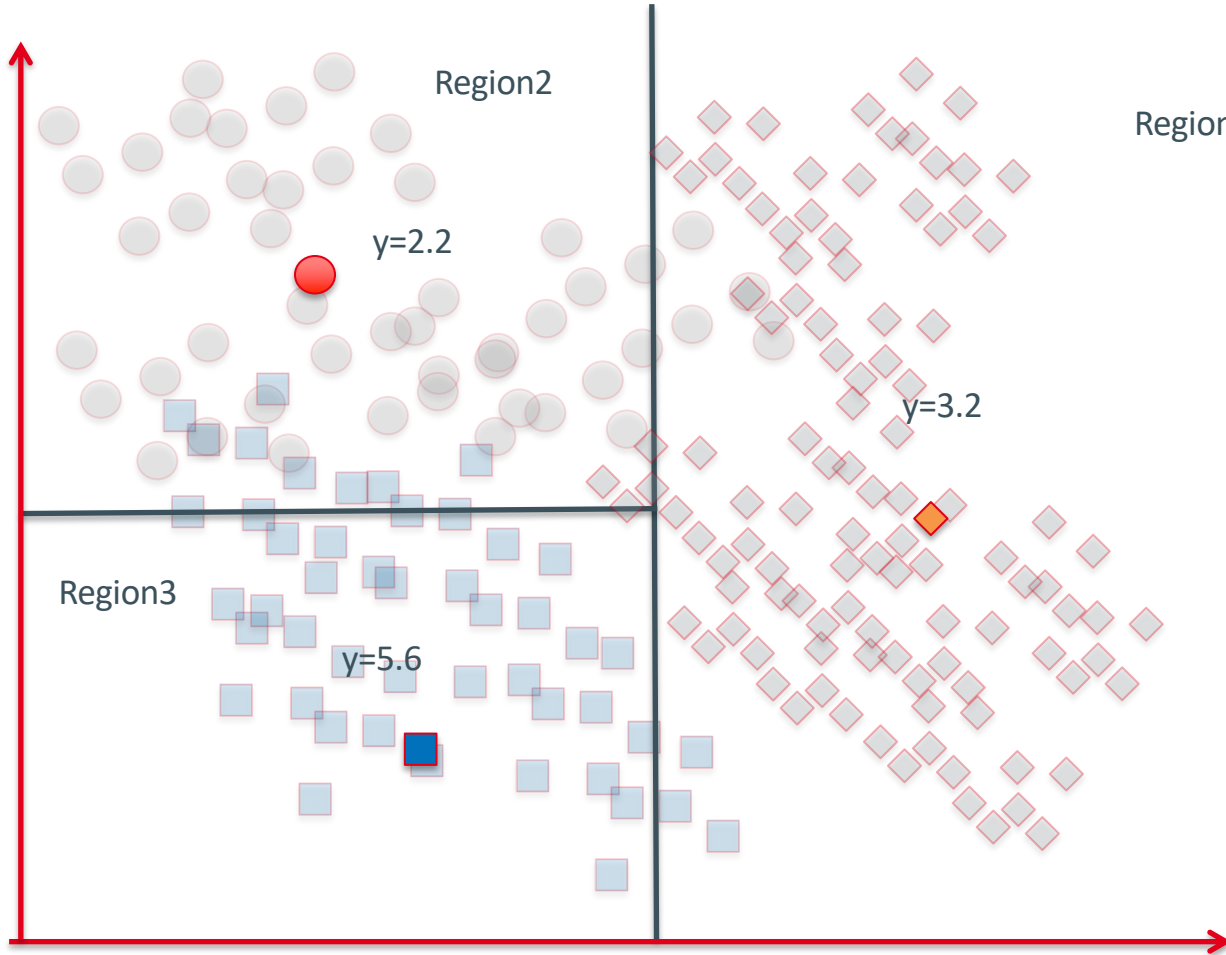
Build a regression tree:

Divide the predictor space into J distinct not overlapping regions

$$R_1, R_2, R_3, \dots, R_J$$

We make the same prediction for all observations in the same region;
use the mean of responses for all training observations that are in the
region

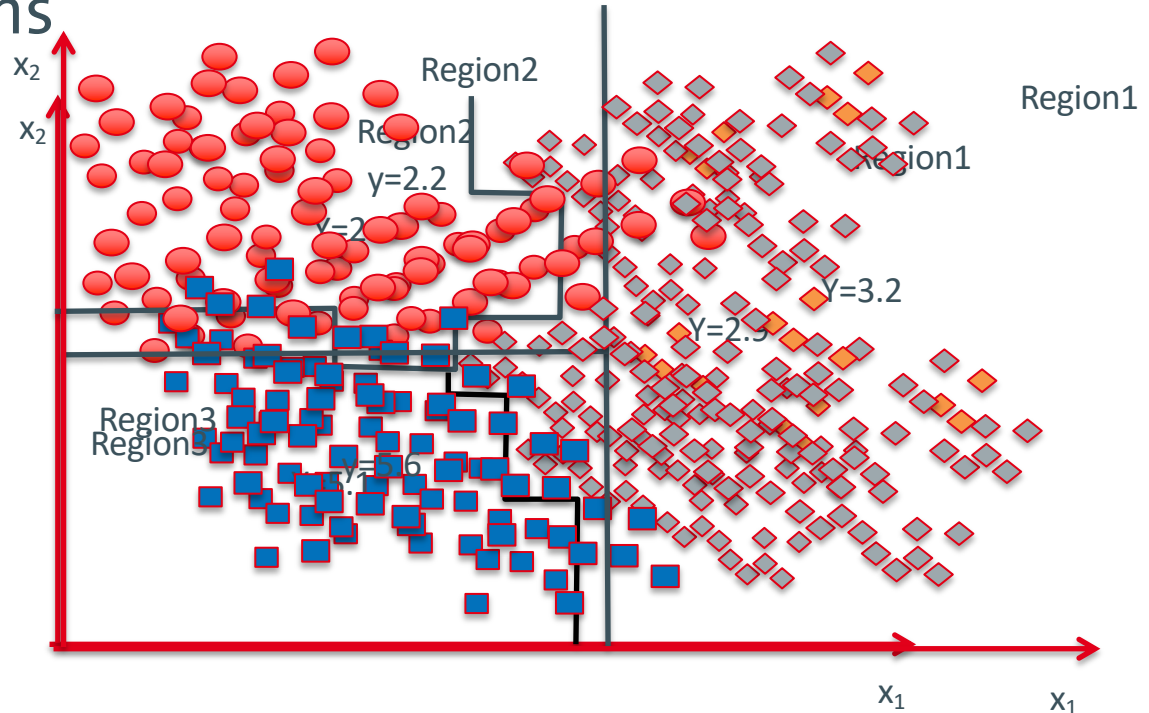
x_2



Finding the Sub-regions

The regions could have any shape.

But we choose just rectangles



Find Boxes R_1, \dots, R_J that Minimise the RSS

$$RSS = \sum_{j=1}^J \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$$

where \hat{y}_{R_j} is the mean response value of all training observations in the R_j region

This is computationally very expensive!

Solution: Top down approach, greedy approach
recursive binary splitting

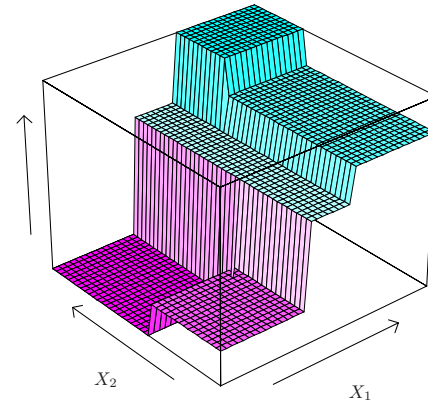
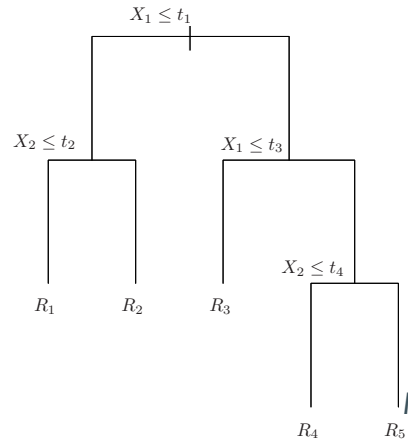
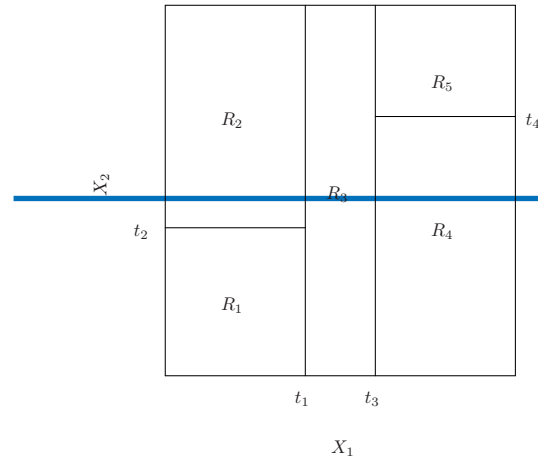
Recursive Binary Splitting

1. Consider all predictor X_p and all the all possible values of the cutpoints s for each of the predictors. Choose the predictor and cutpoint s.t. it minimizes the RSS

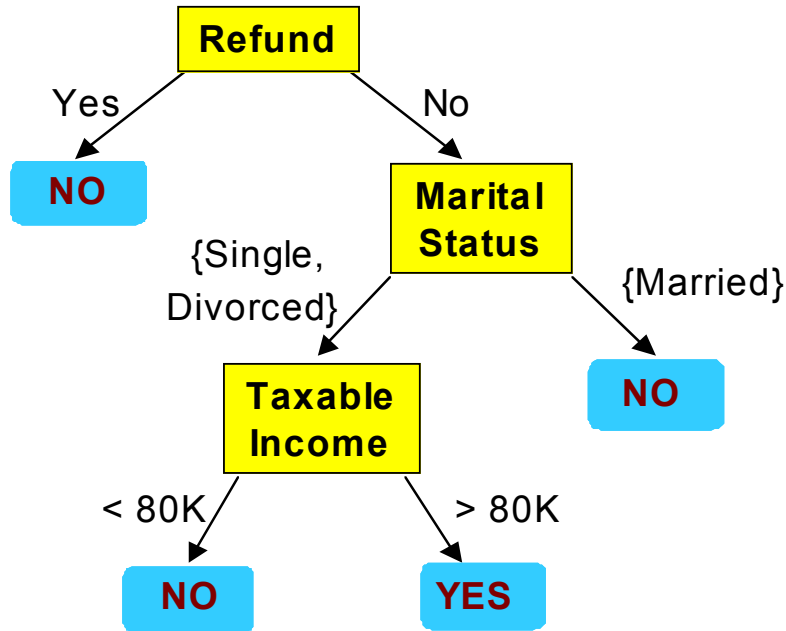
$$\sum_{i:x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i:x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2$$

This can be done quickly, assuming number of predictors is not very large

2. Repeat #1 but only consider the sub-regions
3. Stop: node contains only one class or node contains less than n data points or max depth is reached



From Decision Trees to Rules



Refund = Yes \rightarrow No

Refund = No \wedge

Marital Status = {Single, Divorced} \wedge
Taxable Income < 80k \rightarrow No

Refund = No \wedge

Marital Status = {Single, Divorced} \wedge
Taxable Income > 80k \rightarrow Yes

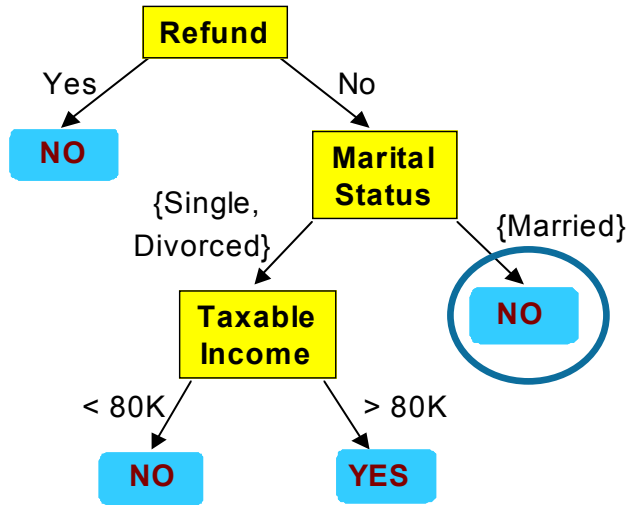
Refund = No \wedge

Marital Status = Married \rightarrow No

From Decision Trees to Rules

- Derive a rule set from a decision tree:
Write a rule for each path from the root to a leaf.
 - The left-hand side is easily built from the label of the nodes and the labels of the arcs.
- Rules are mutually exclusive and exhaustive.
- Rule set contains as much information as the tree

Rules Can Be Simplified



Initial Rule:

$$(\text{Refund}=\text{No}) \wedge (\text{Status}=\text{Married}) \rightarrow \text{No}$$

<i>Tid</i>	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Rules Can Be Simplified

- The resulting rules set can be simplified:
 - Let LHS be the left hand side of a rule.
 - Let LHS' be obtained from LHS by eliminating some conditions.
 - We can certainly replace LHS by LHS' in this rule if the subsets of the training set that satisfy respectively LHS and LHS' are equal.
 - A rule may be eliminated by using meta-conditions such as "if no other rule applies".

VSL vs DTL

- Decision tree learning (DTL) is more efficient if all examples are given in advance; else, it may produce successive hypotheses, each poorly related to the previous one
- Version space learning (VSL) is incremental
- DTL can produce simplified hypotheses that do not agree with all examples
- DTL has been more widely used in practice