

Introduction to Machine Learning

Lecture 6

Albert Orriols i Puig
aorriols@salle.url.edu

Artificial Intelligence – Machine Learning
Enginyeria i Arquitectura La Salle
Universitat Ramon Llull

Recap of Lecture 4

□ ID3 is a strong system that

- Uses hill-climbing search based on the information gain measure to search through the space of decision trees
- Outputs a single hypothesis.
- Never backtracks. It converges to locally optimal solutions.
- Uses all training examples at each step, contrary to methods that make decisions incrementally.
- Uses statistical properties of all examples: the search is less sensitive to errors in individual training examples.
- Can handle noisy data by modifying its termination criterion to accept hypotheses that imperfectly fit the data.

Recap of Lecture 4

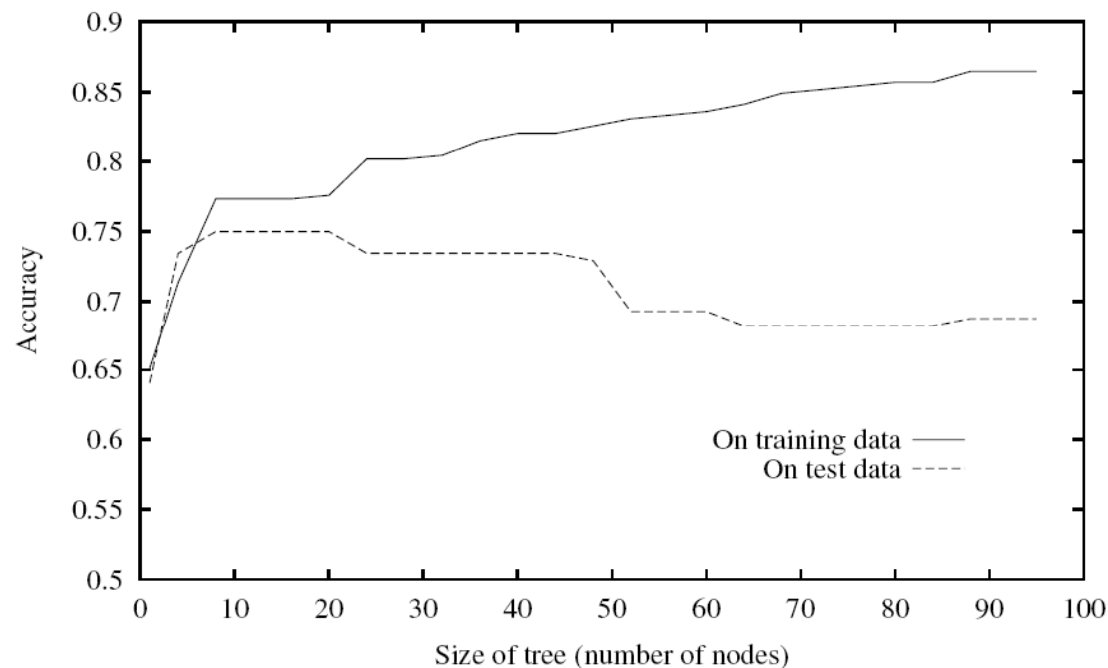
- **However, ID3 has some drawbacks**
 - It can only deal with nominal data
 - It is not able to deal with noisy data sets
 - It may be not robust in presence of noise

Today's Agenda

- Going from ID3 to C4.5
- How C4.5 enhances ID3 to
 - Be robust in the presence of noise. Avoid overfitting
 - Deal with continuous attributes
 - Deal with missing data
 - Convert trees to rules

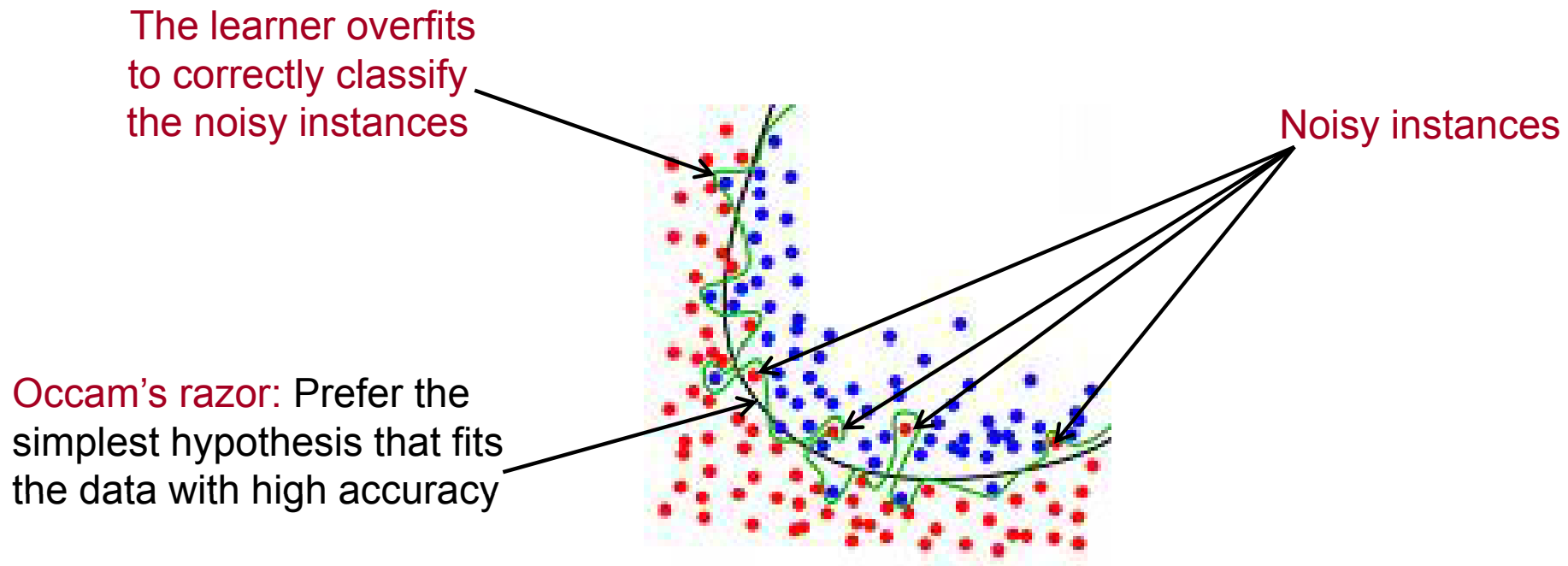
What's Overfitting?

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



Why May my System Overfit?

- In domains with noise or uncertainty
 - the system may try to decrease the training error by completely fitting all the training examples



How to Avoid Overfitting?

- **Ok, my system may overfit... Can I avoid it?**
 - Sure! Do not include branches that fit data too specifically
- **How?**
 1. **Pre-prune:** Stop growing a branch when information becomes unreliable
 2. **Post-prune:** Take a fully-grown decision tree and discard unreliable parts



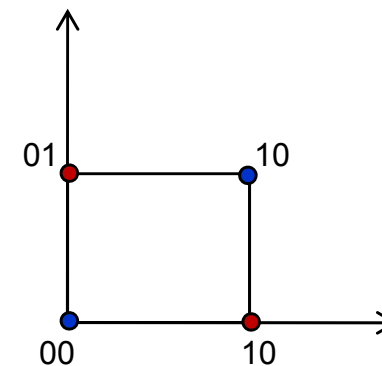
Pre-pruning

- **Based on statistical significance test**
 - Stop growing the tree when there is no statistically significant association between any attribute and the class at a particular node
 - Use all available data for training and apply the statistical test to estimate whether expanding/pruning a node is to produce an improvement beyond the training set
- **Most popular test: chi-squared test**
- **ID3 used chi-squared test in addition to information gain**
 - Only statistically significant attributes were allowed to be selected by information gain procedure

Pre-pruning

- **Early stopping:** Pre-pruning may stop the growth process prematurely
- **Classic example: XOR/Parity-problem**
 - No individual attribute exhibits any significant association to the class
 - Structure is only visible in fully expanded tree
 - Pre-pruning won't expand the root node
- **But:** XOR-type problems rare in practice
- **And:** pre-pruning faster than post-pruning

	x1	x2	Class
1	0	0	0
2	0	1	1
3	1	0	1
4	1	1	0

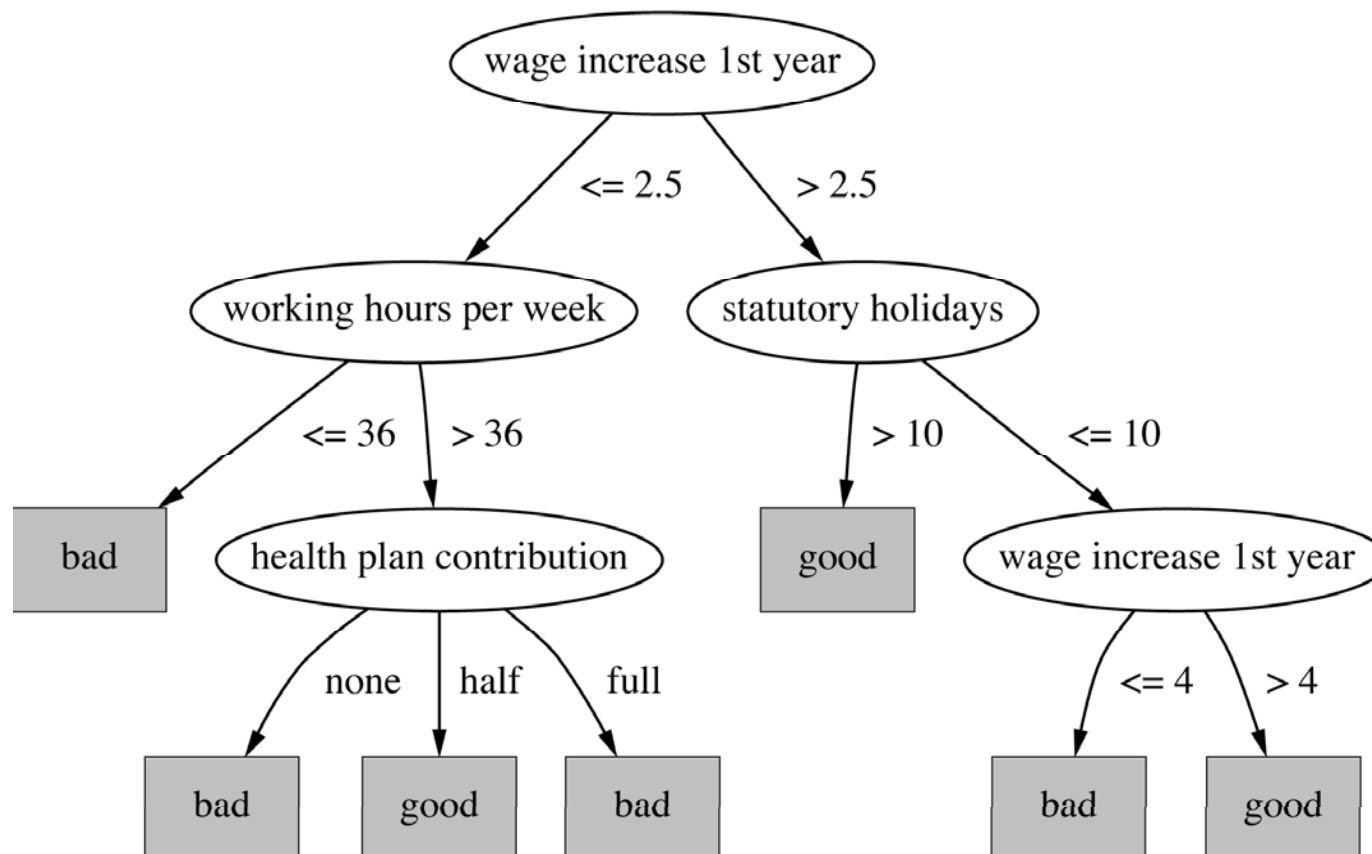


Post-pruning

- **First, build the full tree**
- **Then, prune it**
 - Fully-grown tree shows all attribute interactions
- **Problem:** some subtrees might be due to chance effects
- **Two pruning operations:**
 1. *Subtree replacement*
 2. *Subtree raising*
- **Possible strategies:**
 - error estimation
 - significance testing
 - MDL principle

Subtree Replacement

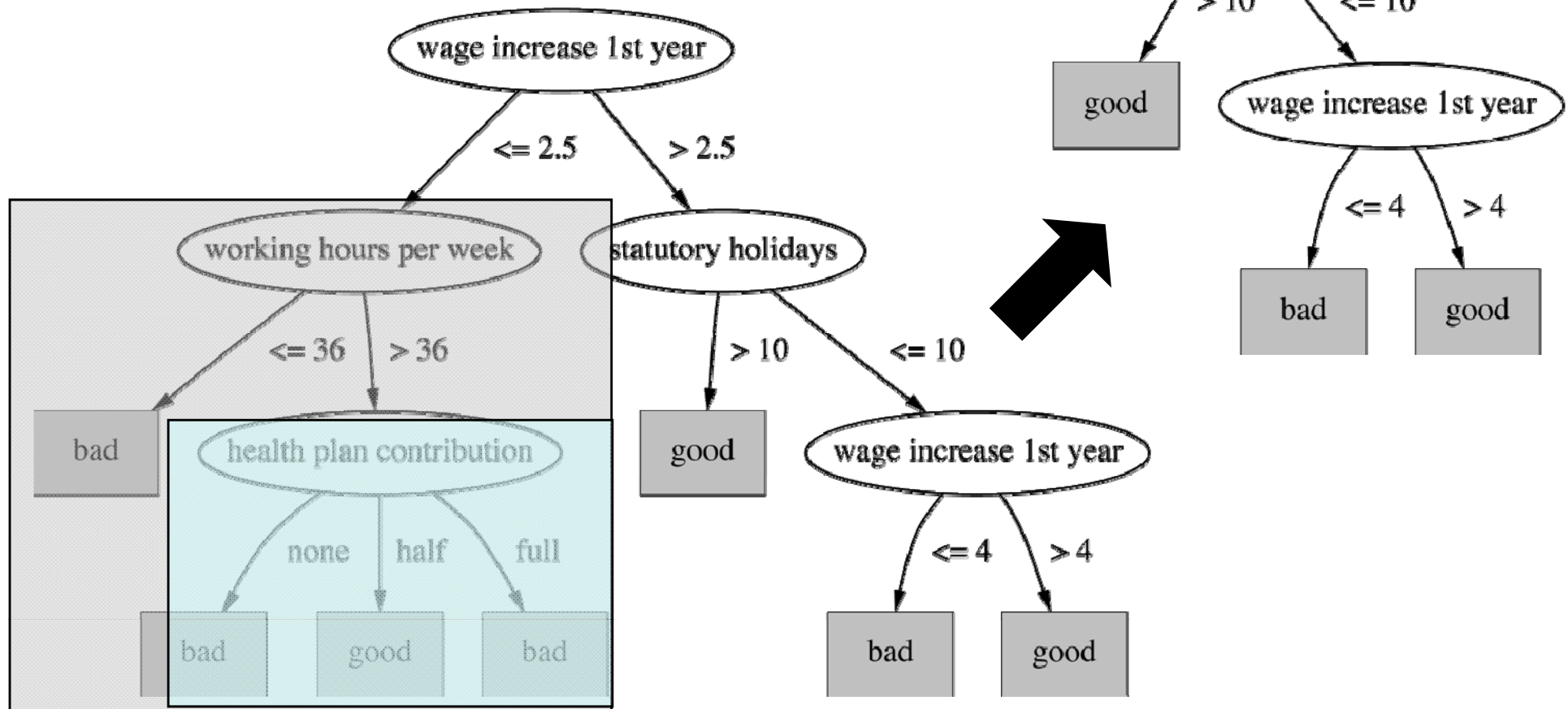
- Bottom up approach
- Consider replacing a tree after considering all its subtrees
- Ex: labor negotiations



Subtree Replacement

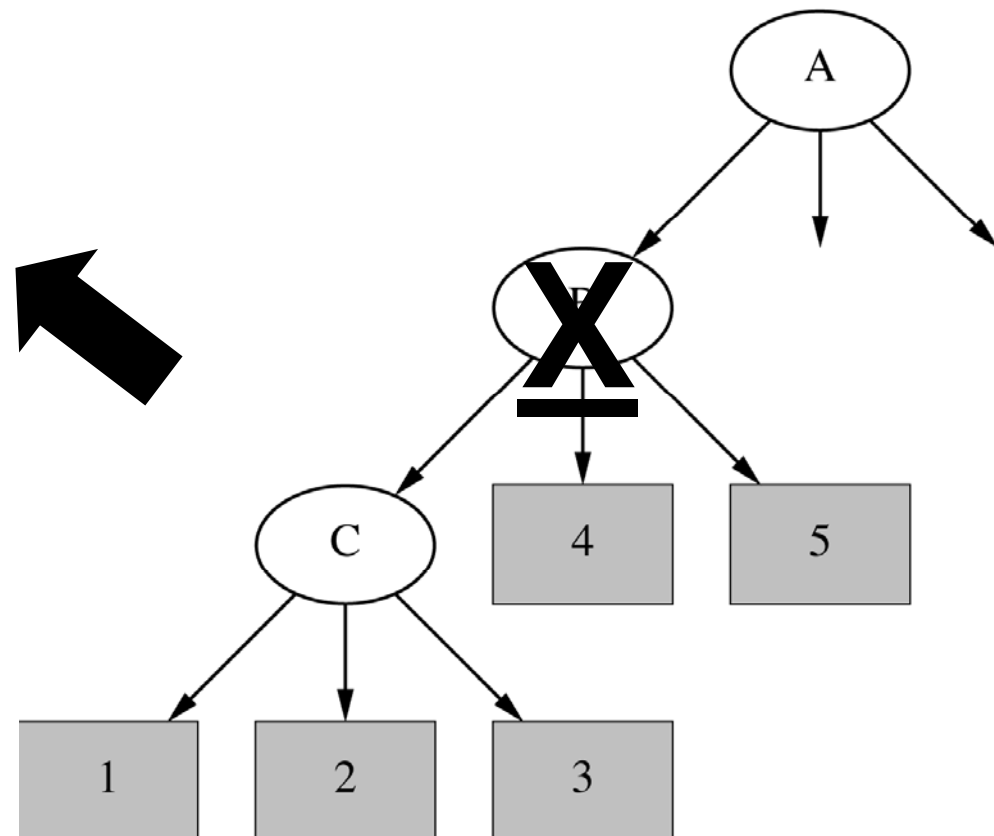
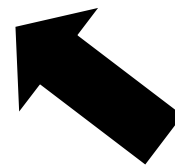
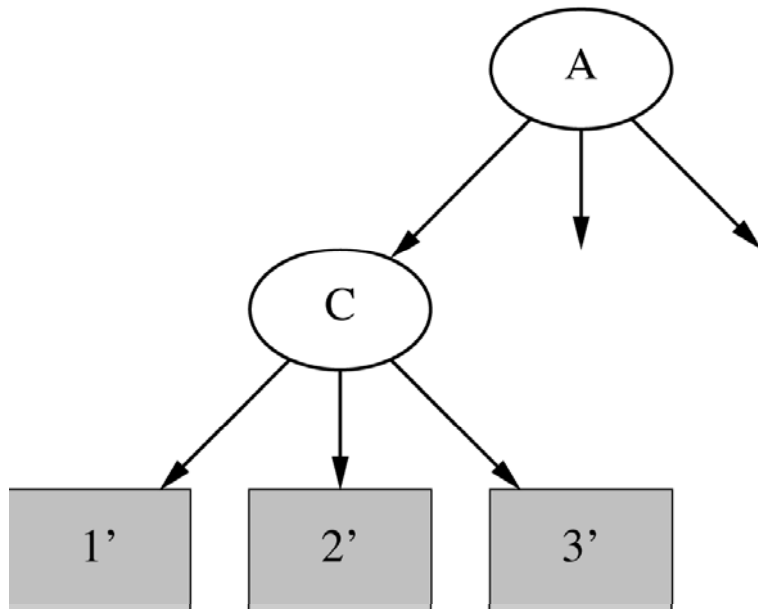
Algorithm:

1. Split the data into training and validation set
2. Do until further pruning is harmful:
 - a. Evaluate impact on the validation set of pruning each possible node
 - b. Select the node whose removal most increases the validation set accuracy



Subtree Raising

- Delete node
- Redistribute instances
- Slower than subtree replacement
(*Worthwhile?*)



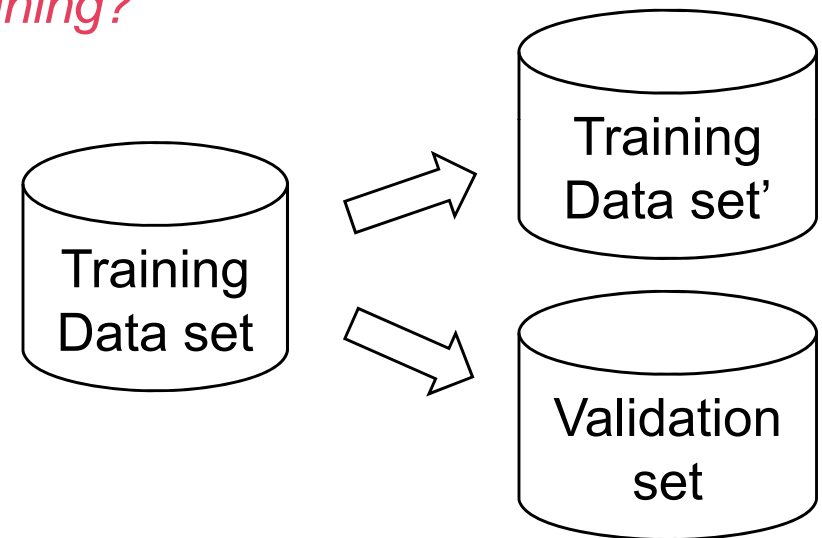
Estimating Error Rates

□ Ok we can prune. But when?

- Prune only if it reduces the estimated error
- Error on the training data is NOT a useful estimator

Q: Why it would result in very little pruning?

- Use hold-out set for pruning
 - **Separate a validation set**
 - **Use this validation set to test the improvement**



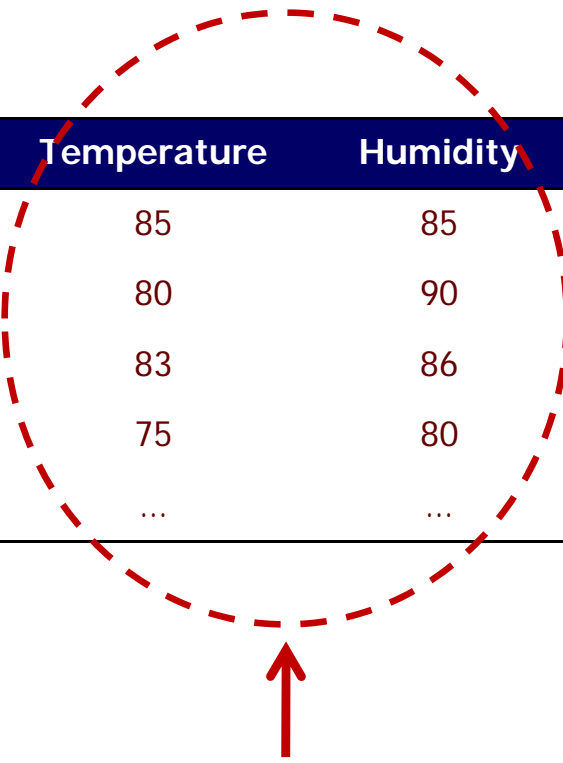
- C4.5's method
 - **Derive confidence interval from training data**
 - **Use a heuristic limit, derived from this, for pruning**
 - **Standard Bernoulli-process-based method**
 - **Shaky statistical assumptions (based on training data)**

Deal with continuous attributes

- **When dealing with nominal data**
 - We evaluated the gain for each possible value
- **In continuous data, we have infinite values.**
- **What should we do?**
 - Continuous-valued attributes may take infinite values, but we have a limited number of values in our instances (at most N if we have N instances)
 - Therefore, simulate that you have N nominal values
 - **Evaluate information gain for every possible split point of the attribute**
 - **Choose the best split point**
 - **The information gain of the attribute is the information gain of the best split**

Deal with continuous attributes

□ Example



Outlook	Temperature	Humidity	Windy	Play
Sunny	85	85	False	No
Sunny	80	90	True	No
Overcast	83	86	False	Yes
Rainy	75	80	False	Yes
...

Continuous attributes

Deal with continuous attributes

□ 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

- $\text{Info}([4,2],[5,3]) = 6/14 \text{ info}([4,2]) + 8/14 \text{ info}([5,3]) = 0.939 \text{ bits}$

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

Deal with continuous attributes

□ To speed up

- Entropy only needs to be evaluated between points of different classes

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

Potential optimal breakpoints

Breakpoints between values of the same class cannot be optimal

Deal with Missing Data

- **Treat missing values as a separate value**
 - Missing value denoted “?” in C4.X
 - Simple idea: treat missing as a separate value
 - **Q: When this is not appropriate?**
 - **A:** When values are missing due to different reasons
 - **Example 1:** gene expression 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)

Deal with Missing Data

- **Split instances with missing values into pieces**
 - A piece going down a branch receives a weight proportional to the popularity of the branch
 - weights sum to 1
- **Info gain works with fractional instances**
 - Use sums of weights instead of counts
- **During classification, split the instance into pieces in the same way**
 - Merge probability distribution using weights

From Trees to Rules

- **I finally got a tree from domains with**

- Noisy instances
- Missing values
- Continuous attributes

- **But I prefer rules...**

- No context dependent

- **Procedure**

- Generate a rule for each tree
- Get context-independent rules

From Trees to Rules

- **A procedure a little more sophisticated: C4.5Rules**
 - C4.5rules: greedily prune conditions from each rule if this reduces its estimated error
 - **Can produce duplicate rules**
 - **Check for this at the end**
 - Then
 - **look at each class in turn**
 - **consider the rules for that class**
 - **find a “good” subset (guided by MDL)**
 - Then rank the subsets to avoid conflicts
 - Finally, remove rules (greedily) if this decreases error on the training data

Next Class

- **Instance-based Classifiers**

Introduction to Machine Learning

Lecture 6

Albert Orriols i Puig
aorriols@salle.url.edu

Artificial Intelligence – Machine Learning
Enginyeria i Arquitectura La Salle
Universitat Ramon Llull