

Decision Tree and Concept Learning

CS3025, Knowledge-Based
Systems
Lecture 18

Yuting Zhao
Yuting.zhao@gmail.com

2017-11-16

Outline

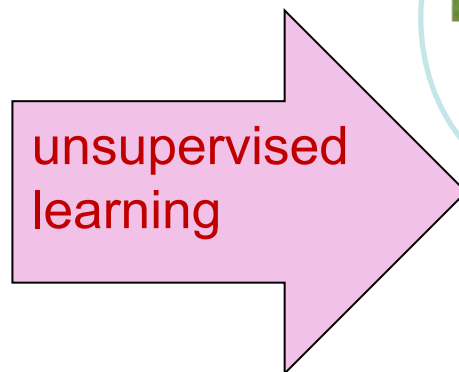
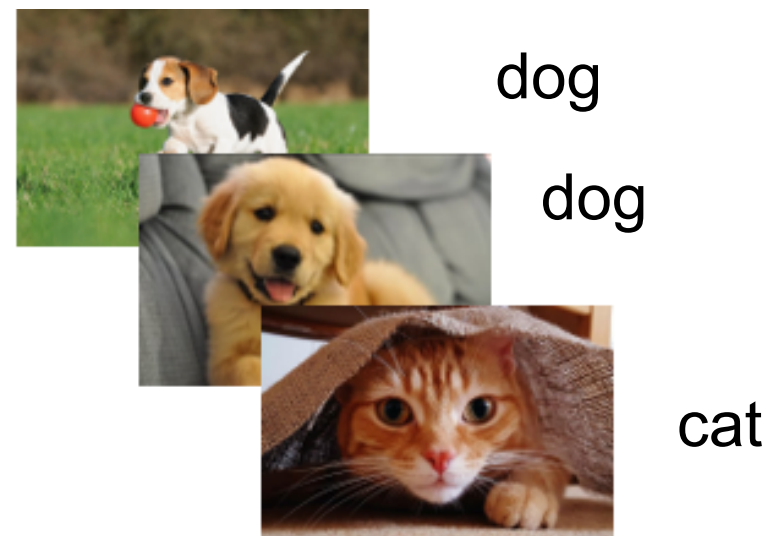
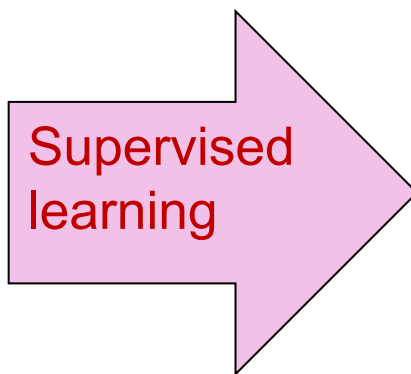
- Types of learning
 - Supervised learning
 - Unsupervised learning
- Decision trees

Why should programs learn?

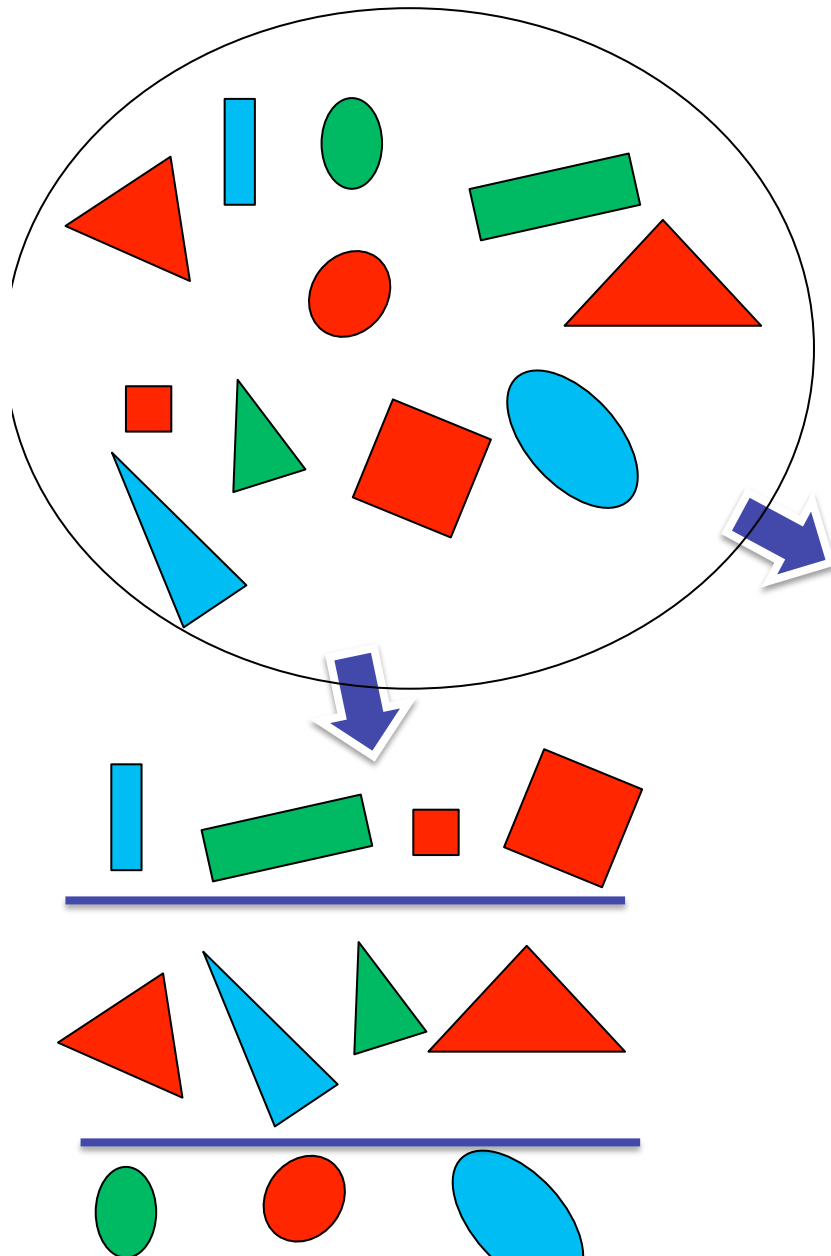
- All the programs seen up to now have been **static**:
 - if we run the programs again on the same data, they will do exactly the same as before
- We would like programs to learn from their experience.
- We would like the possibility of learning being continuous.

How do **you** learn?

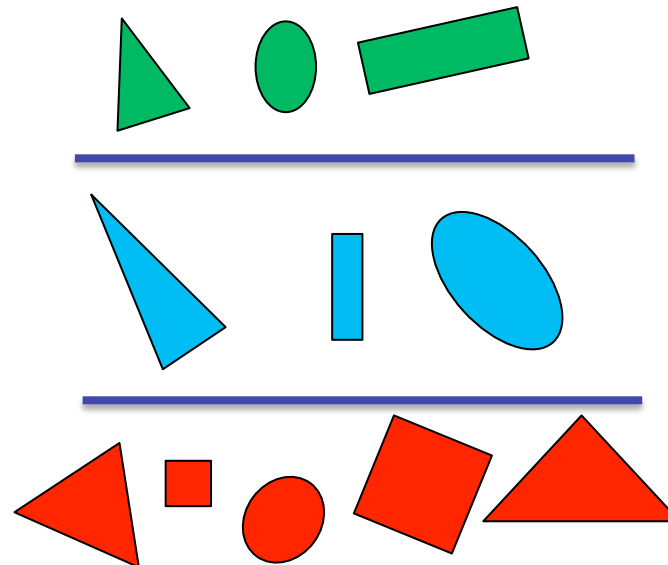
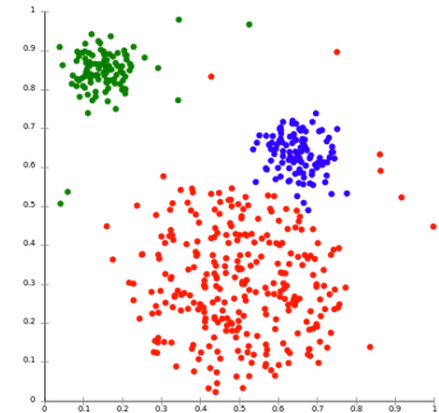
- By being told – relies on:
 - someone to do the telling
 - something to tell!
- By finding a teacher, who provides a set of pre-classified examples (i.e. I/O pairs), or taking action and obtaining correct answer from observations: **Supervised learning**
- By searching for regularities in unclassified data (e.g. clusters): **Unsupervised learning**



Unsupervised Learning



- Cluster analysis or clustering: putting “similar” together.
- How to define similarity



Supervised Learning

- ‘Learning by example’ (**inductive learning**)
 - teacher provides good training instances and the learner is expected to generalise
 - in knowledge acquisition it is often easier to get an expert to give examples than to give rules
 - this is how experimental science works
(the 'teacher' is the natural world)

Supervised Classification

- **Supervised learning:**

- the machine learning task of inferring a function from **labeled training data**
- Knowledge is represented as **function** (input – output)

- Given:

- **Target**: a fixed set of **classes**: $Y = \{y_1, y_2, \dots, y_n\}$, e.g. {sports, politics, ..., music}
- **Training data**: a collection of data objects X with known classes Y , i.e. $(X, Y) = \{(x_1, y_1), (x_2, y_2) \dots (x_n, y_n)\}$. E.g {(doc1, sports), (doc2, sports), (doc3, music) ...}.
- **Testing data**: a description of an unseen instance, D_{new} e.g. a new document without class label information

- Goal:

- Predict the category/class of D_{new} : $y(x) \in Y$, where $y(x)$ is a classification function, aka trained model, whose domain is X and whose range is Y .

What can you do with classification?

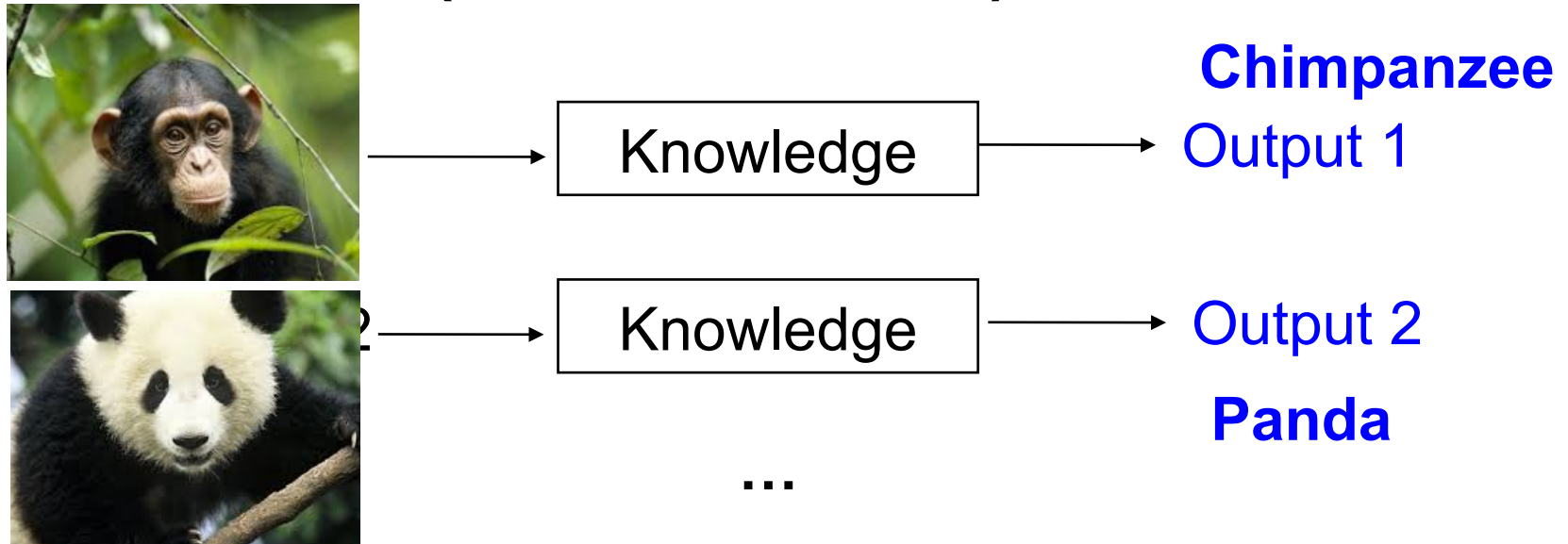
Applications:

- Topic classification
 - Given a news article, predict the topic of the article, e.g., finance vs. sports
- Spam Classification
 - Given an email, predict whether it is a spam or not
- Medical Diagnosis
 - Given a list of symptoms, predict whether a patient has cancer or not
- Weather
 - Based on temperature, humidity, etc... predict if it will rain tomorrow

Knowledge as a function

- Knowledge (**performance element**) can be described as a function:
 - given a description, x , for a given object or situation, the output is given by $f(x)$ where f embodies the knowledge contained in the performance element.
 - could be:
 - analytical mathematical function
 - lookup table
 - rule set (including STRIPS rules)
 - neural network
 - decision tree
 - etc.

Application of knowledge (deduction)



We have

- inputs
- knowledge

We get

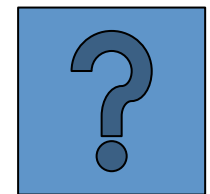
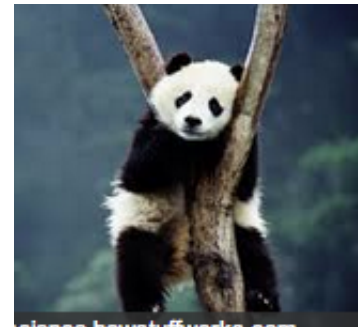
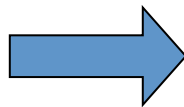
- outputs

An Important Assumption

- Training and unseen (test) data
 - in the **same feature space**
 - follow the **same distribution**



Training data



Unseen data

Supervised Classifier

★★★★★ Some flaws, but overall, GREAT, 25 Oct 2011

★★★★★ The best? Maybe so....., 26 Oct 2011

★★★☆☆ A limited device, 29 Oct 2011
By [A reviewer](#) (United Kingdom) - [See all my reviews](#)
This review is from: **Apple iPhone 4S 16GB Black (Electronics)**
I'm not "an Apple fanatic with the ethos 'if it aint Apple don't bother'", so you will get something balanced here, but I will say that I purchased an iPhone 4S with a strong desire to like it. I really tried my best and intended to use it exclusively, but due to me having already experienced Android, it had to go back to the shop.

I don't care who makes a product or what their marketing is like, I care about how versatile and useful the product is and in this respect I just couldn't avoid the obvious conclusion that this device is deficient. Shock, horror, Apple?! Yes, they don't walk on water, they just have slick marketing.

What were the problems? I'll just list those I discovered in the few days using the phone. Some of these I suppose are going to be subjective but I'll just tell you how I found it:

Training set (corpus)

By [M. Bond](#) (London) - [See all my reviews](#)
REAL NAME

By [Dr. W. E. Allen "wallen200"](#) (Belfast, UK) - [See all my reviews](#)
REAL NAME

This review is from: **Apple iPhone 4S 16GB Black (Electronics)**
The first thing I need to say is that the Apple iPhone 4S is the best smart phone in the market at present, and unless something radical happens will probably be the best smart phone until the iPhone 5 is released. I am not going to labour all the features, these have been well covered in the description and the previous reviews. However I will say that this phone is definitely not worth upgrading to from the iPhone 4 and even if you have an iPhone 3GS I would say it would be better to wait until the next generation iPhone comes out. The reason I say this is that this phone has really only two differences from the iPhone 4 - Siri and a higher resolution camera. I will discuss these first.

Test set

Naïve Bayes, SVM,
MaxEnt , etc

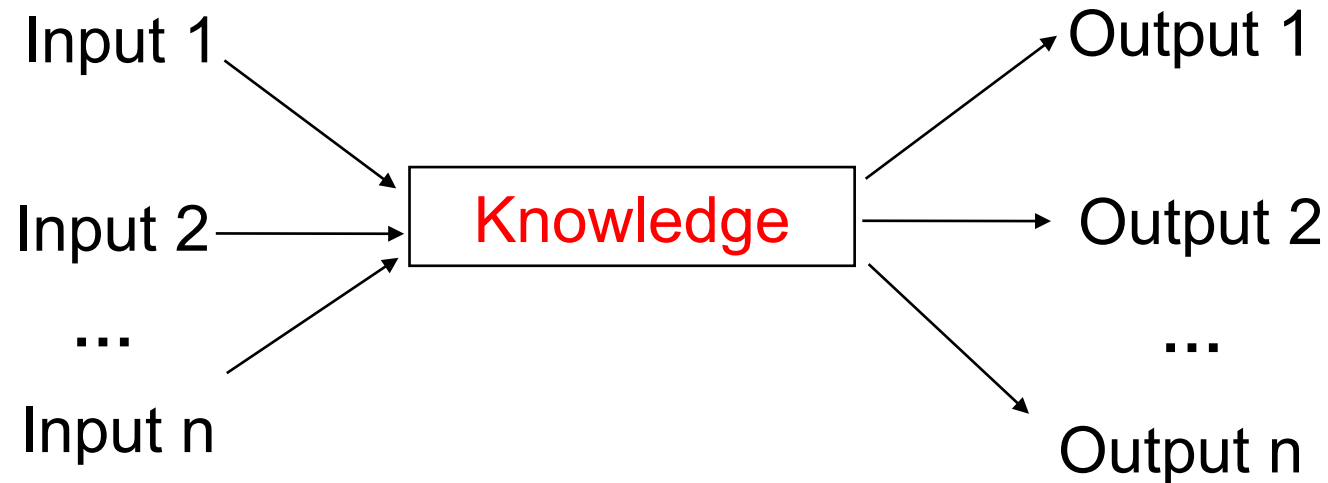
Learn
Model

Model

Apply
Model

- Rely on syntactic or co-occurrence patterns in large text corpora

Inductive learning (**induction**)



We have

- inputs
- outputs

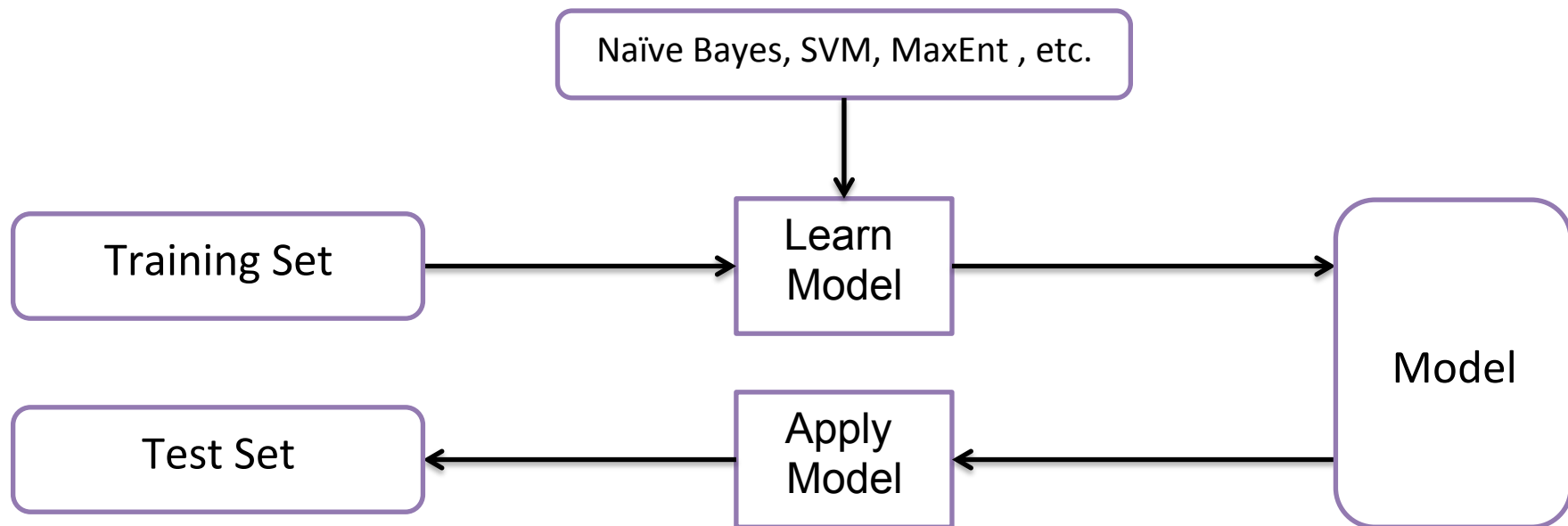
We get

- **knowledge**

Definition of inductive learning

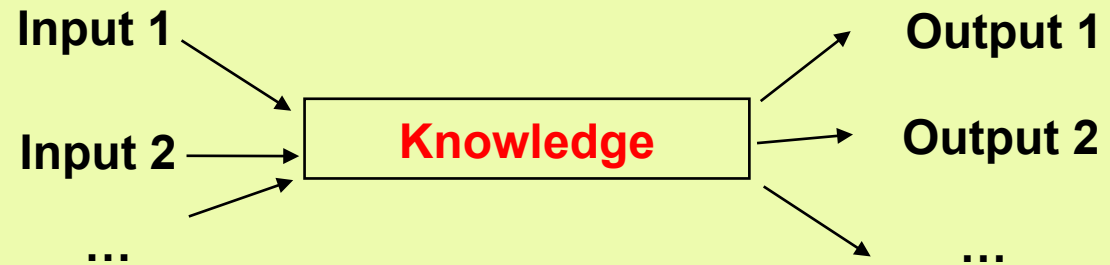
- Given a set of input/output pairs (examples):
 $(x, f(x))$
 - where f is unknown, but the output $f(x)$ is provided with its corresponding input, x .
 - **find a function**, $h(x)$ (hypothesis) which best approximates $f(x)$.
 - ‘finding’ implies *searching* in a space of different possible hypotheses.

Supervised Classification

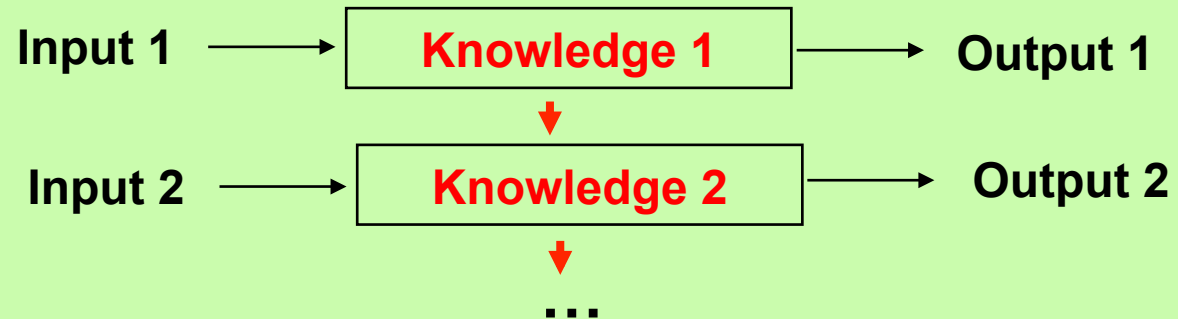


Inductive learning

Non-incremental



Incremental



Assume that **Knowledge 2** is more complete than **Knowledge 1**, etc.

Non-incremental vs. Incremental

- Non-incremental
 - learning from all examples at once
- Incremental
 - learning by **refining** from successive examples
- If you haven't seen **all** possible examples, you can never *know* that the system is going to give the correct answer for a previously unknown example.
- You may never see all possible examples.



Outline

- Types of learning
 - Supervised learning
 - Unsupervised learning
- **Decision trees**

The Supervised Classification Task

- **Input:** Collection of instances with a set of **pairs** (attributes x and a special nominal attribute Y called class attribute)
- **Output:** A model that accurately predicts y from x on previously unseen instances
 - Previously unseen instances are called test set
- Usually input collection is divided into
 - Training set for building the required model
 - Test set for evaluating the model built

Supervised Learning for Classification

- Many commercial systems (partly) rely on classification techniques (MSN, Verity, Enkata, Yahoo!, ...)
 - Naive Bayes (simple, common method)
 - **Decision trees** (intuitive, powerful)
 - Support-vector machines (new, more powerful)
 - plus many other methods ...
- No free lunch: requires hand-classified training data
- But data can be built up (and refined) by amateurs, e.g., Amazon Mechanical Turk
- Note that many commercial systems use a mixture of methods

Example Data

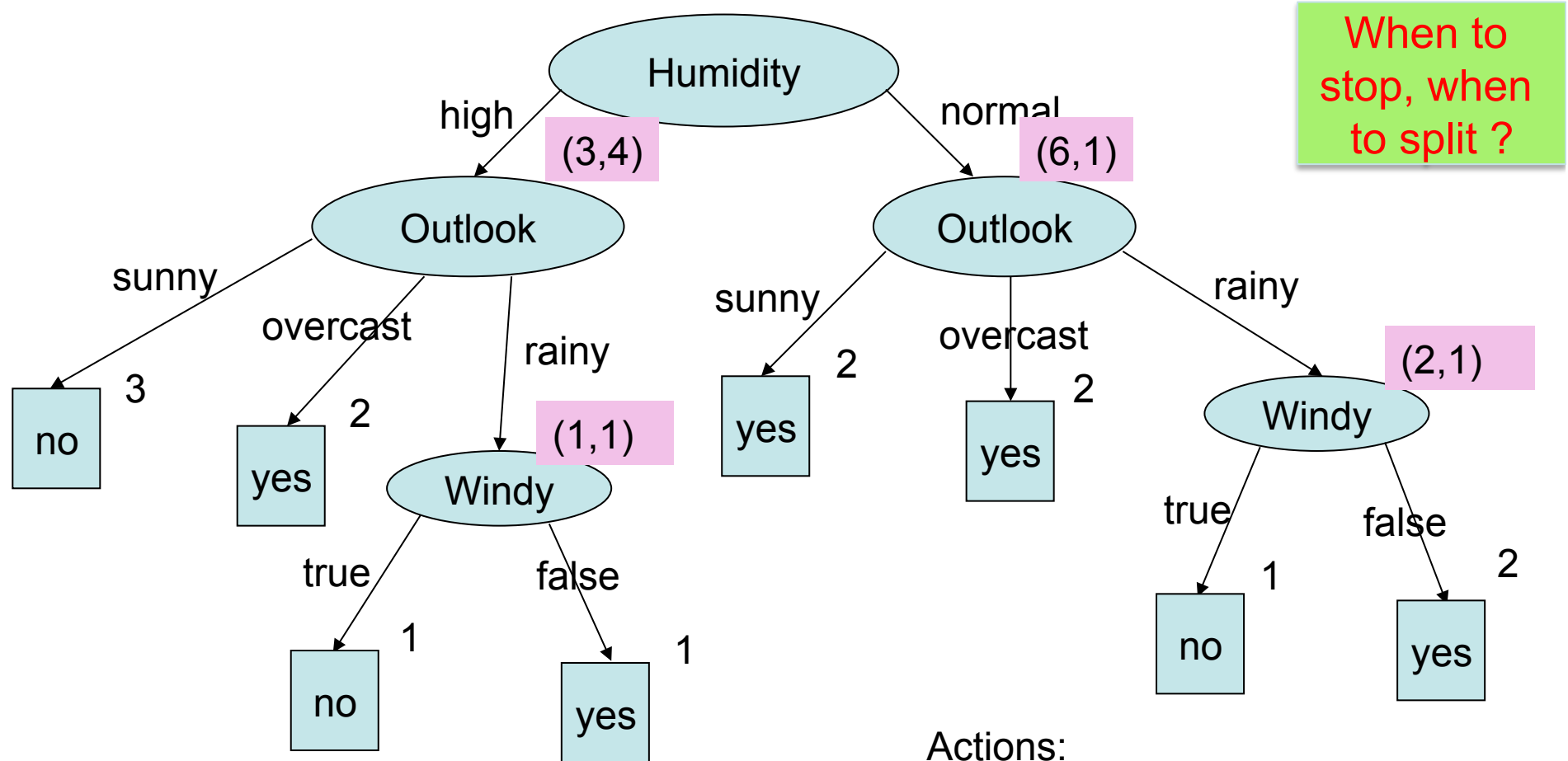
Class
Attribute

Outlook	Temperature	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

Decision Tree Construction

- Recursive procedure
 - Select an attribute to place at the **root** node
 - Make one **branch** for each possible value of the selected attribute
 - For each branch repeat the above two steps recursively
 - Using only those instances that actually reach the branch
 - **Stop** developing a branch if it has instances belonging to the same class, otherwise split the branch
- Several decision trees are possible
 - Based on the order of the selection of the attributes

Example Decision Tree 1



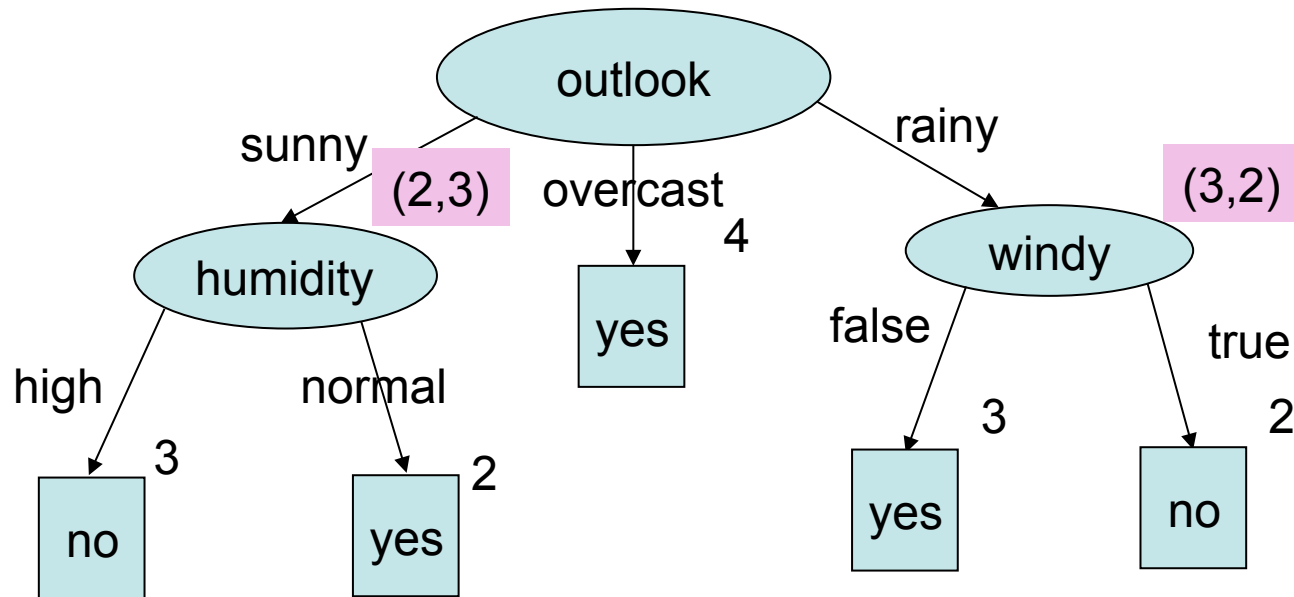
Observations:
 Outlook and windy repeated in this tree
 Windy tested only when outlook is rainy

Actions:

Start the tree with Outlook

Test windy when outlook is rainy

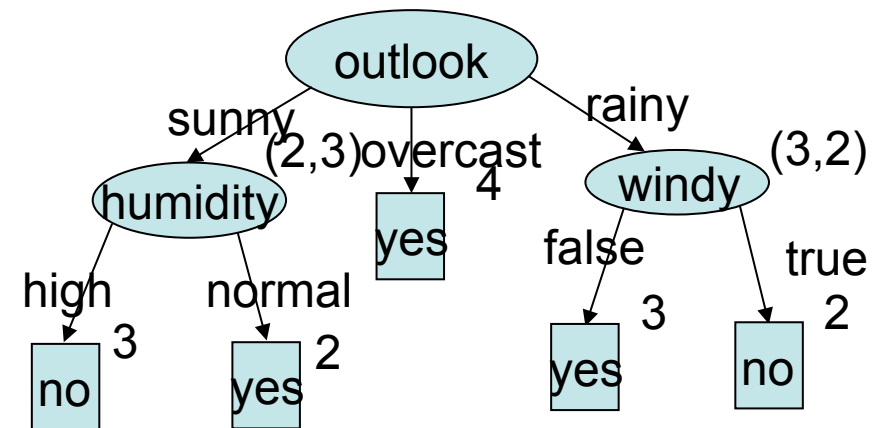
Example Decision Tree 2



When to
stop, when
to split ?

```
(defrule decision-tree
  (weather (outlook ?o) (humidty ?h) (windy ?w))
  =>
  (if ( or (and (= sunny ?o) (= high ?h) )
          (and (= rainy ?o) (= true ?w) )
      )
    then ((printout t "decision: NO" crlf))
    else ((printout t "decision: YES" crlf))
  )
)
```

Jess code (1)



```
(defrule decision-tree-1
  (weather (outlook sunny) (humidity high))
  =>
  (printout t "decision: NO" crlf)) )
```

```
(defrule decision-tree-2
  (weather (outlook sunny) (humidity normal))
  =>
  (printout t "decision: YES" crlf)) )
```

```
(defrule decision-tree-3
  (weather (outlook overcast) )
  =>
  (printout t "decision: YES" crlf)) )
```

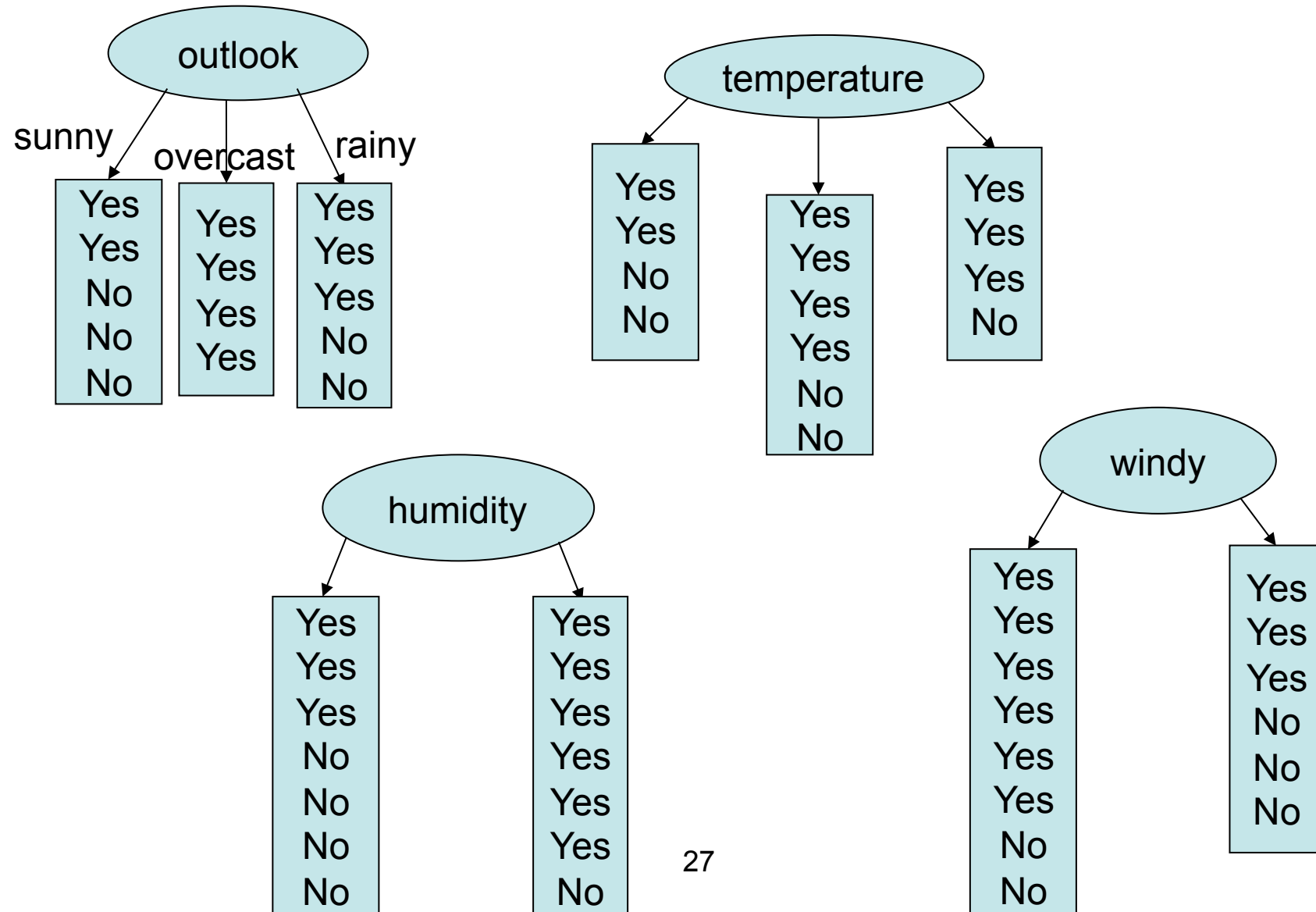
```
(defrule decision-tree-4
  (weather (outlook rainy) (windy false))
  =>
  (printout t "decision: YES" crlf)) )
```

```
(defrule decision-tree-5
  (weather (outlook rainy) (windy true))
  =>
  (printout t "decision: NO" crlf)) )
```

Jess code (2)

Which code is more efficient?

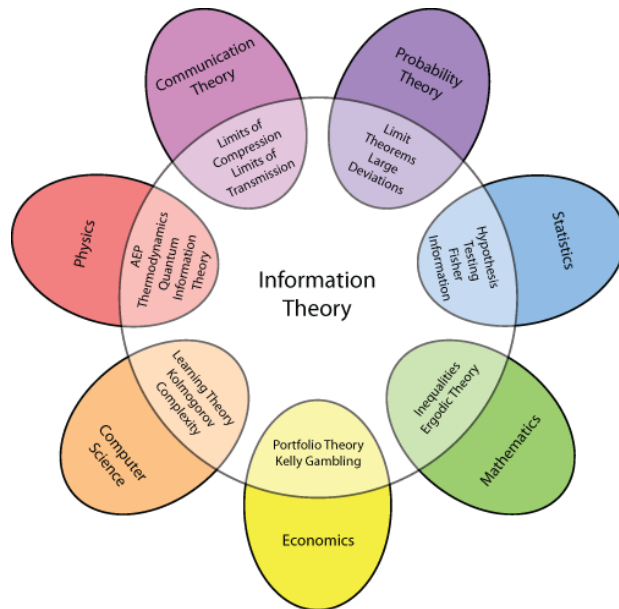
Tree stumps for the weather data



Splitting heuristic

- Select the node which ‘**provides the most information**’ first
- Based on information theoretic measure (Shannon)
- Aims to **minimise** the number of tests needed to provide a classification
- Applied to our two decision trees describing the weather data
 - Decision tree 2 is preferable to decision tree 1
- Small decision trees are better
 - **Attribute ordering** makes the difference

Information Theory (Shannon)

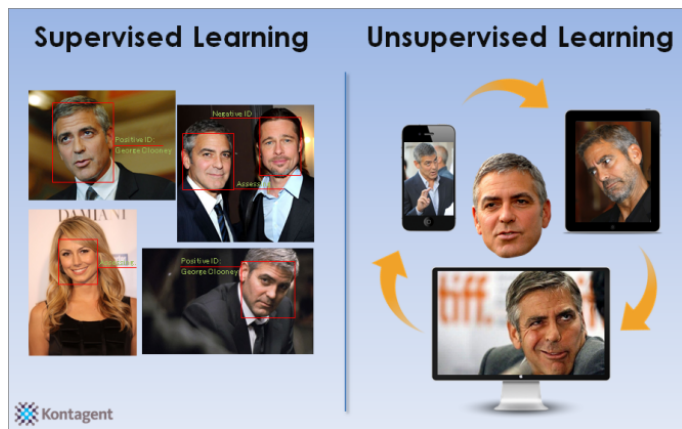


- Want a numerical measure for each attribute
 - Maximum when attribute is perfect (provides perfect separation)
 - Minimum when attribute is useless (no separation)
- Suppose attribute has n possible values; the i^{th} value has prior probability P_i
- Information content of the attribute is:
$$I(P_1, P_2, \dots, P_n) = \sum -P_i \log_2 P_i$$
- Chose attribute with **highest information content**.
- [Bit more complex – see R&N]

Attribute Selection for Splitting

- In our case, outlook is a better attribute at the root than others
 - Because when outlook is at the root, one of its branches (overcast) immediately leads to a ‘pure’ daughter node which terminates further tree growth
 - **Pure nodes** have the same class label for all the instances reaching that node
- We need a generic function that helps to rank order attributes
 - The function should reward attributes that produce ‘pure’ daughter nodes

What you should know



- Two different types of learning
 - Supervised learning (inductive learning)
 - Unsupervised learning

- Decision Tree
 - supervised learning algorithm
 - Attribute selection for splitting

