Artificial Intelligence

Learning

The act, process, or experience of gaining knowledge or skill

Outline

1. Learning
2. Learning Agent
3. Types of Learning
4. Rote Learning
5. Learning by Analogy
6. Learning by Example

What is Learning?

- *Psychology:*

Learning is the process of acquiring new understanding, knowledge, behaviors, skills, values, attitudes, and preferences

-[Richard Gross, Psychology: The Science of Mind and Behavior 6E, Hachette UK, ISBN 978-1-4441-6436-7]

- Examples
  - Walking (motor skills)
  - Riding a bike (motor skills)
  - Telephone number (memorizing)
  - Playing Cards Game (strategy)
  - Develop scientific theory (abstraction)
  - Language
  - Recognize fraudulent credit card transactions
  - Etc.

What is Machine Learning?

- Learning denotes “changes in the system that are adaptive in the sense that they enable the system to do the same task or tasks drawn from the same population *more efficiently and more effectively* the next time”.

  \- Herbert A. Simon, 83

- Agents can *improve their performance* through diligent study of their own experiences.

  \- Russell & Norvig

(One more) Definition of Machine Learning

- Definition [Tom Mitchell, 1997]:

  A computer program is said to learn from

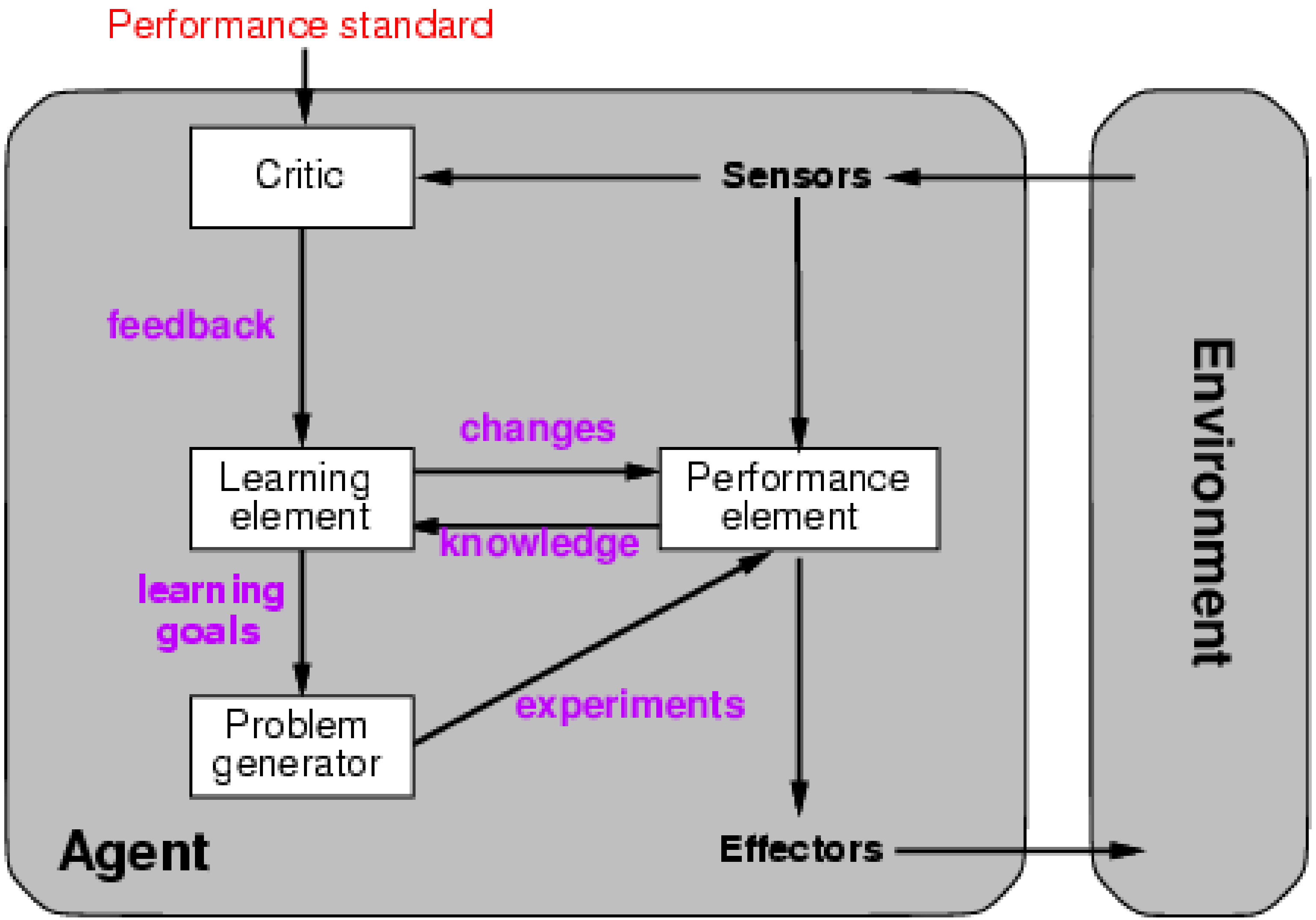
  - Experience **E** w.r.t. some class of
  - Tasks **T** and
  - Performance measure **P**

  if its performance at tasks in **T**, as measured by **P**, improves with experience **E**.

- Given : A task **T** performance measure **P** Some experience **E** with the task
- Goal: Generalize the experience **E** in a way that allows to improve your performance **P** on the task **T**.

- Spam Filtering
  - **T**: Classify emails HAM / SPAM
  - **E**: Examples $(e_1,HAM),(e_2,SPAM),(e_3,HAM),(e_4,SPAM)$, ...
  - **P**: Prob. of error on new emails
- Personalized Retrieval
  - **T**: Driving on four-lane highways using vision sensors
  - **E**: Sequence of image and steering commands recorded while observing a human driver
  - **P**: Average distance traveled before a human-judged error
- Play Checkers
  - **T**: Play checkers
  - **E**: games against self
  - **P**: percentage wins

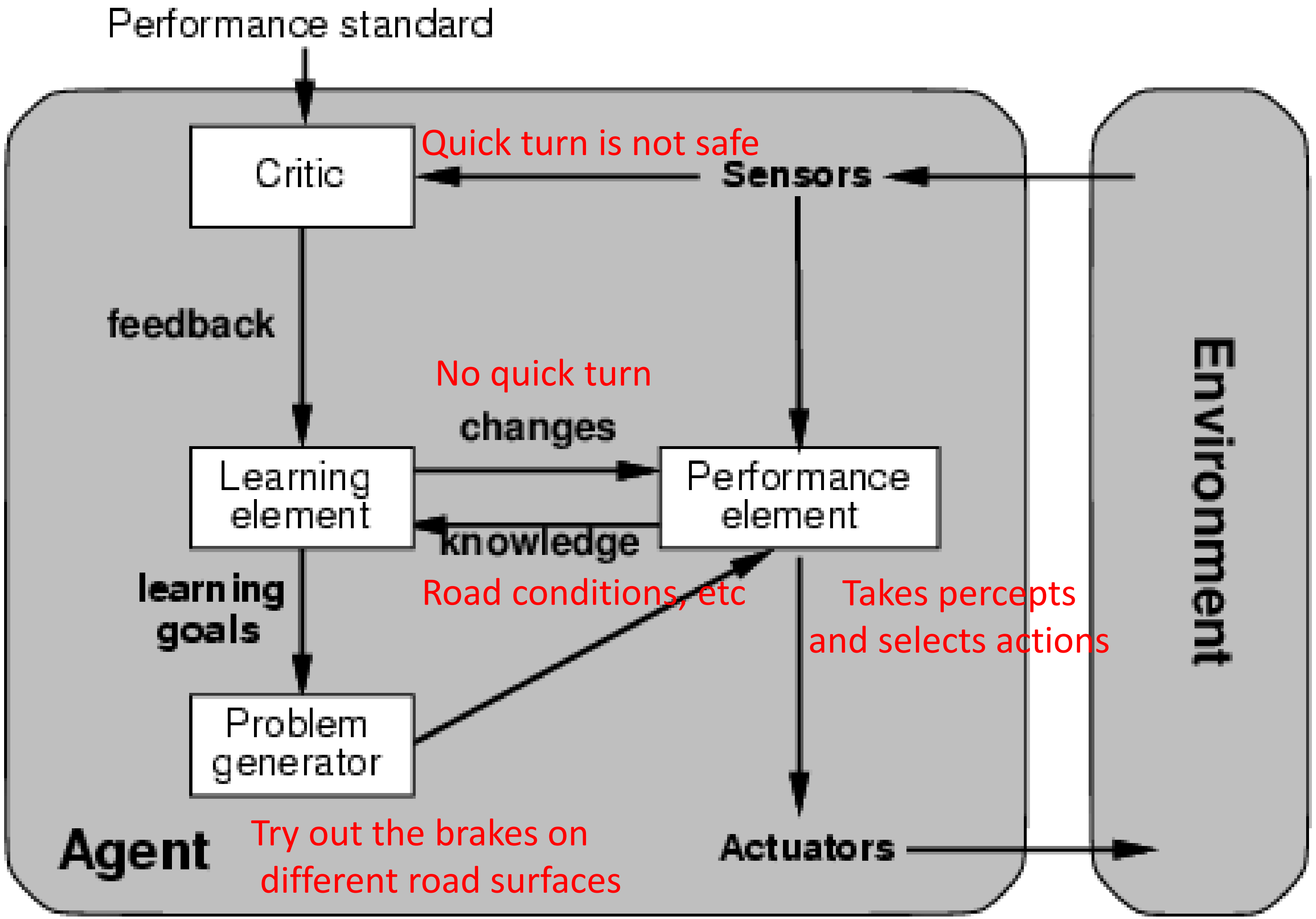
Learning Agents



Learning agents

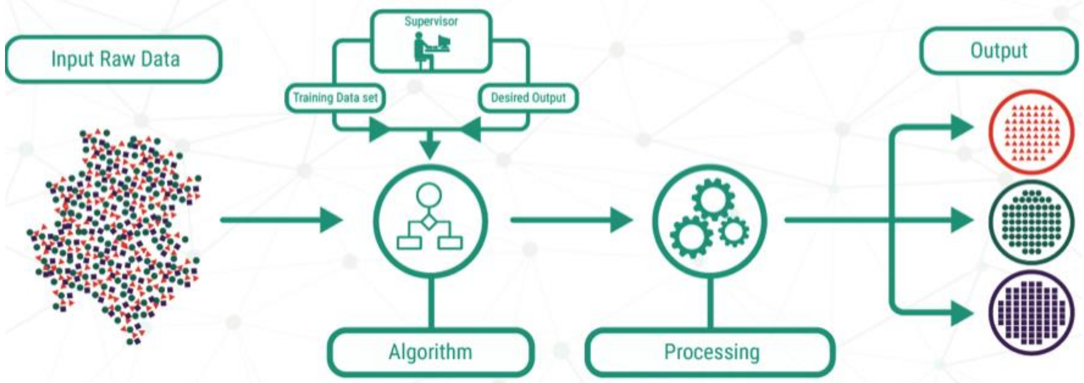
Learning enables an agent to modify its decision mechanisms to improve performance

More complicated
when agent needs to learn
utility information → Reinforcement learning
(reward or penalty: e.g., high tip or no tip)

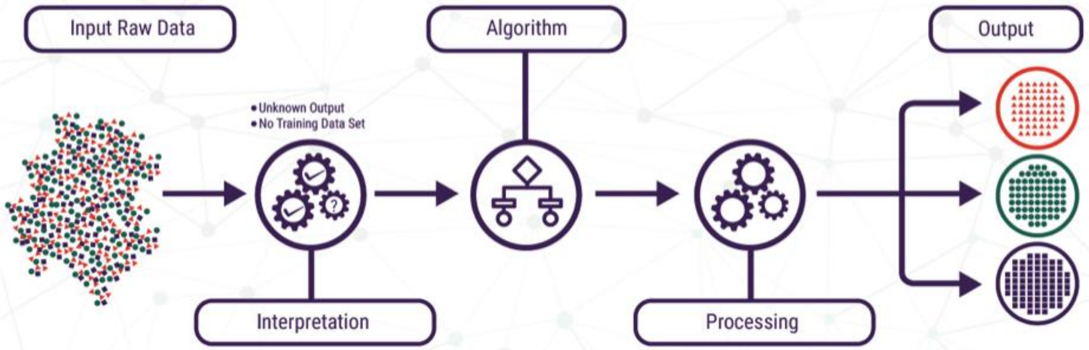


Types of Feedback

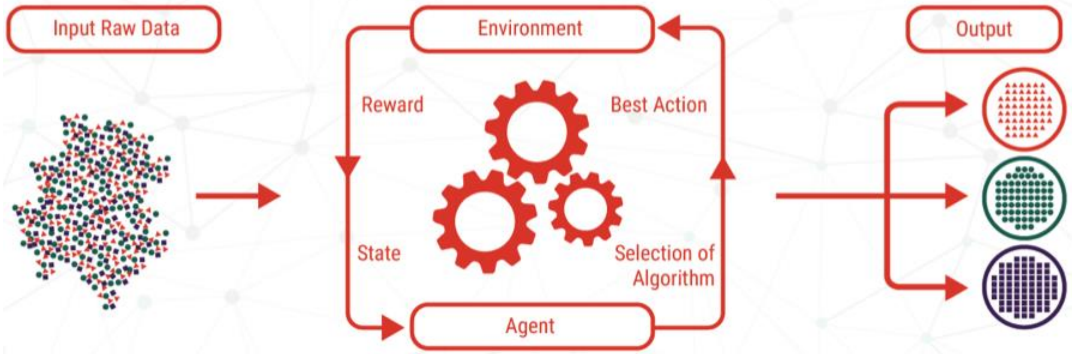
- Supervised learning: (Labeled examples)
  - Agent is given correct answers for each example
  - Agent is learning a function from examples of its inputs and outputs



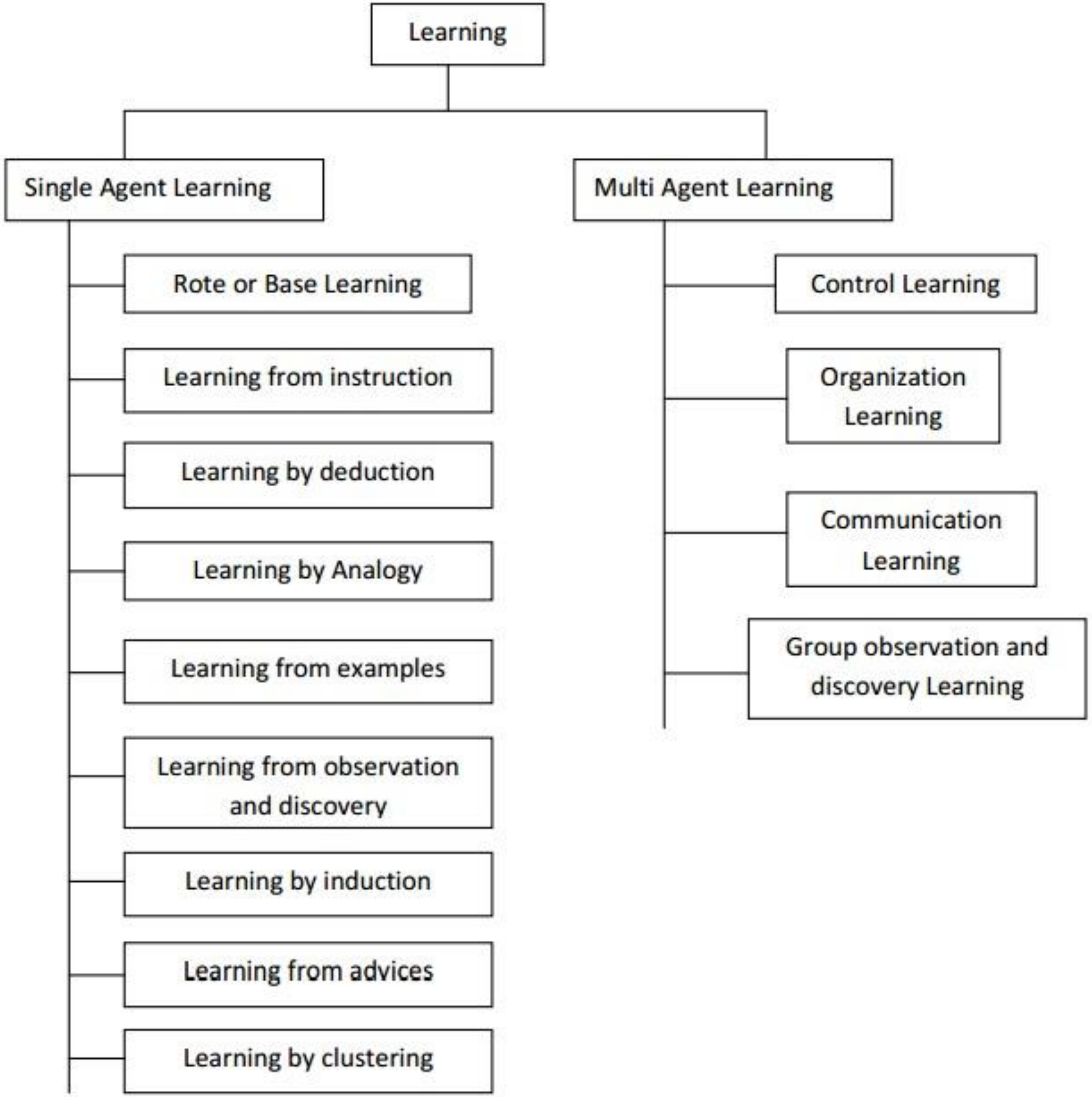
- Unsupervised learning: (Unlabeled examples)
  - Agent must infer correct answers
  - Completely unsupervised learning is impractical, since agent has no context



- Reinforcement learning: (Rewards)
  - Agent is given occasional rewards for correct
  - Typically involves subproblem of learning “how the world works”



Methods of Learning



- Rote Learning (Memorization):
  - Simple storage of computed information (facts/results)
  - No inference
- Direct instruction (by being told)
  - Teach a robot how to hold a cup
  - Required inference
- Analogy
  - Transform existing knowledge to new situation
  - Learn how to hold a cup then learn to how to hold object with handle
- Induction
  - Learning from examples
  - Studied in machine learning
- Deduction
  - Deductive inference steps using known facts

Rote learning

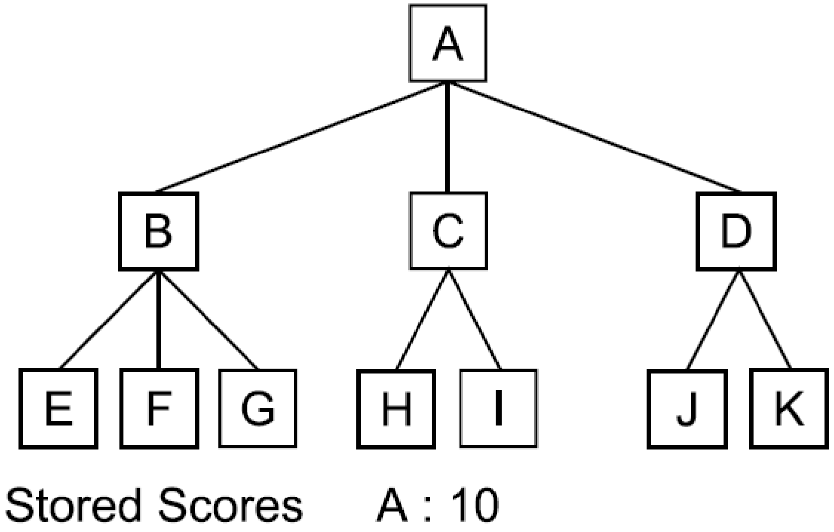
- Rote learning consists of memorizing the solutions of the solved problems so that the system needs not to solve them again:

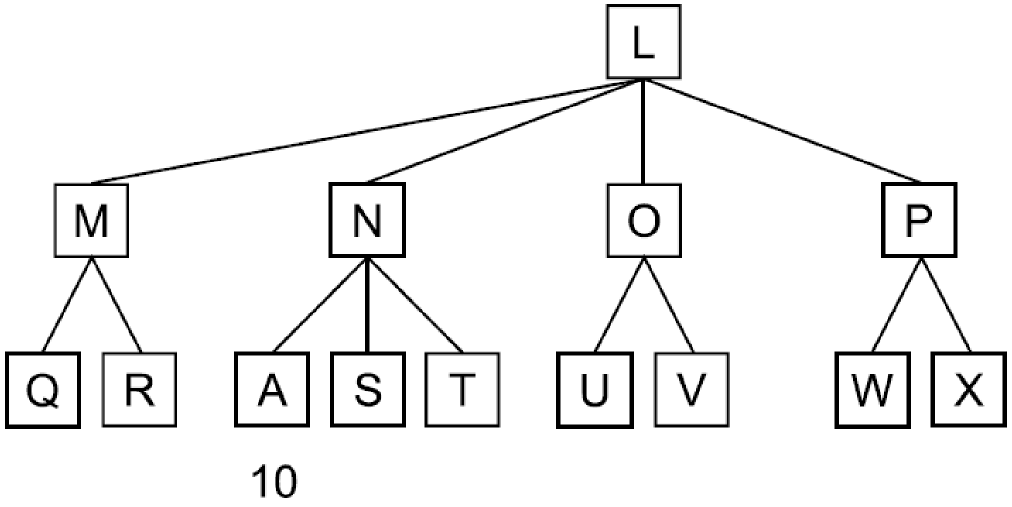
(X1, ... , Xn)	$\xrightarrow{f}$	(Y1, ... , Yp)	$\xrightarrow{\text{store}}$	[(X1, ... , Xn), (Y1, ... , Yp)]
Input pattern	Performance function	Output value of computation		Associated pair

- During subsequent computations of $f(X_1, \ldots X_n)$, the performance element can simply retrieve $(Y_1, \ldots , Y_p)$ from memory rather than recomputing it.

Rote learning

- Memory organizations
  - Rote learning requires useful organization of the memory so that the retrieval of the desired information will be very fast.
- Stability of the environment
  - The information stored at one time should still be valid later.
- Store-versus-compute trade-off
  - The cost of storing and retrieving the memorized information should be smaller than the cost of recomputing it.





Direct Instruction

- Direct instruction is a complex form of learning.
- This type of learning requires more inference than rote learning
  - Since the knowledge must be transformed into an operational form before learning when a teacher presents several facts directly to us in a well-organized manner

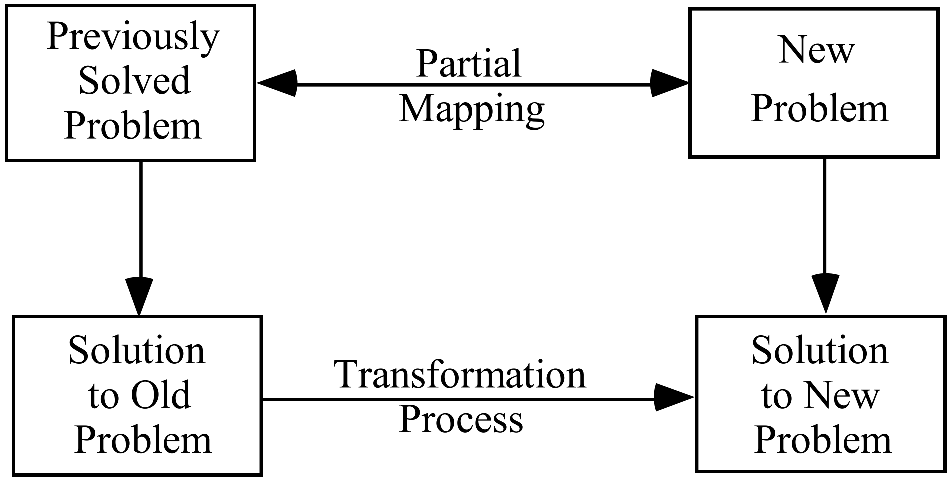
Learning By Analogy

- Learning by analogy means acquiring new knowledge about an input entity by transferring it from a known similar entity.
- Example:
  - Last month, the stock market was a roller coaster.
- Central intuition supporting learning by analogy:
  - If two entities are similar in some respects, then they could be similar in other respects as well.
- Examples of analogies:
  - Pressure Drop is like Voltage Drop
  - A variable in a programming language is like a box.

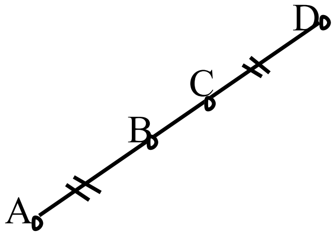
Methods of Analogical Problem Solving

Analogy

Transformational



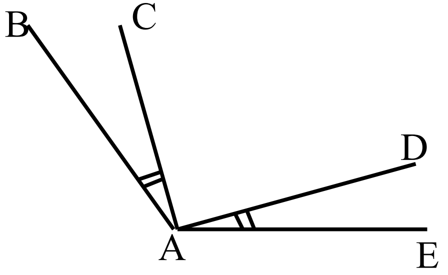
GIVEN: AB = CD
PROVE: AC = BD



AB = CD
BC = BC
AB + BC = BC + CD
AC = BD

σ = (AB <- <BAC
CD <- <DAE
AC <- <BAD
BD <- <CAE)

GIVEN: <BAC = <DAE
PROVE: <BAD = <CAE



<BAC = <DAE
<CAD = <CAD
<BAC + <CAD= <CAD + <DAE
<BAD = <CAE

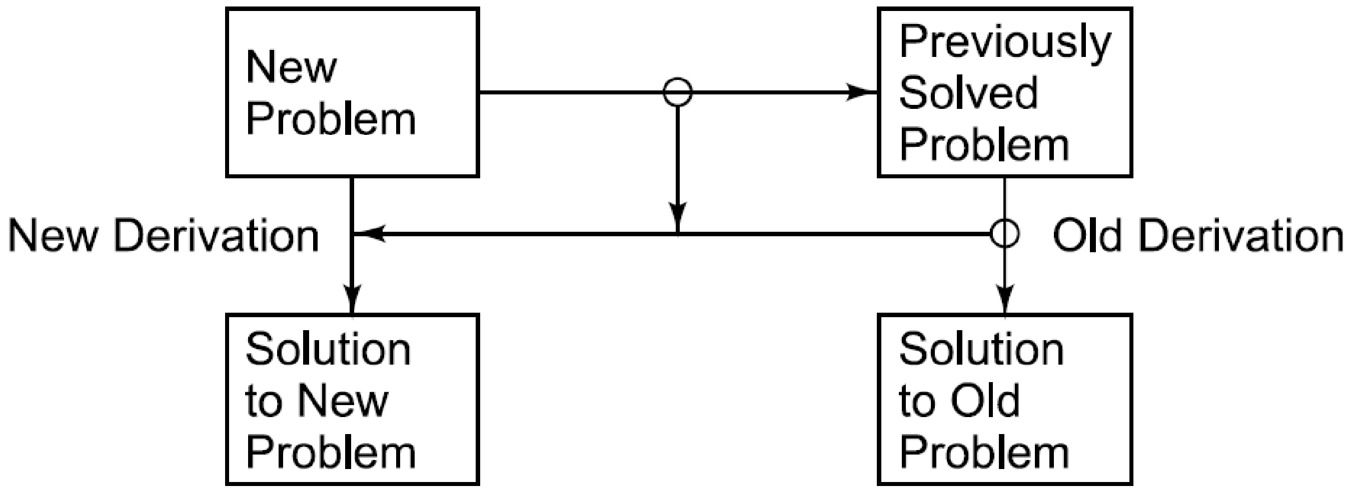
Methods of Analogical Problem Solving

Analogy

- Transformational
- Derivational

```c
void bubbleSort(int arr[], int n)
{
   int i, j;
   for (i = 0; i < n-1; i++)
       // Last i elements are already in place
       for (j = 0; j < n-i-1; j++)
           if (arr[j] > arr[j+1])
              swap(&arr[j], &arr[j+1]);
}
```

```python
def bubbleSort(arr):
    n = len(arr)
    for i in range(n):
        # Last i elements are already in place
        for j in range(0, n-i-1):
            if arr[j] > arr[j+1] :
                arr[j], arr[j+1] = arr[j+1], arr[j]
```



Learning by Examples: Inductive

Basic Problem: Induce a representation of a function (a systematic relationship between inputs and outputs) from examples.

- **target function** $f: X \rightarrow Y$
- **example** $(x, f(x))$
- **hypothesis** $h: X \rightarrow Y$ *such that* $h(x) = f(x)$

x = set of attribute values (***attribute-value representation***)
x = set of logical sentences (*first-order representation*)

Y = set of discrete labels (***classification***)
Y = Continuous values (***regression***)

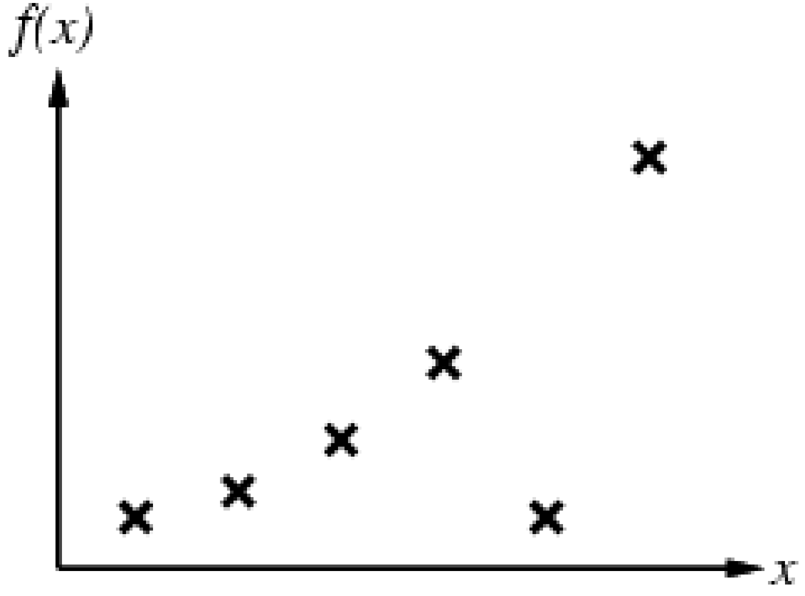
Learning by Examples: Inductive

- Simplest form: learn a function from examples
  - ***f*** is the target function
  - An example is a pair (***x***, ***f(x)***)
- **Pure induction task:**
  - Given a collection of examples of ***f***, return a function ***h*** that approximates ***f*.**
  - **Problem:** find a hypothesis ***h,*** such that ***h* ≈ *f*,** given a training set of examples
- This is a highly simplified model of real learning:
  - Ignores prior knowledge
  - Assumes examples are given
- Learning a discrete function is called classification learning.
- Learning a continuous function is called regression learning.

Inductive learning method

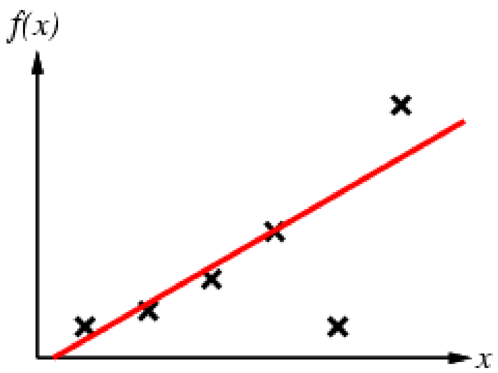
- Fitting a function of a single variable to some data points
  - Examples are (x, f(x) pairs;
  - Hypothesis space H – set of hypotheses we will consider for function **f**, in this case **polynomials of degree at most k**

- Construct/adjust *h* to agree with *f* on training set
- *h* is consistent if it agrees with *f* on all examples
- E.g., curve fitting:

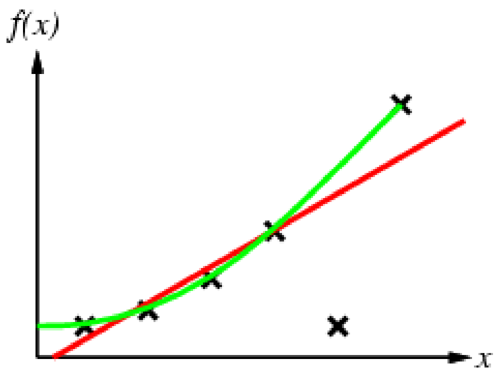


Multiple consistent hypotheses?

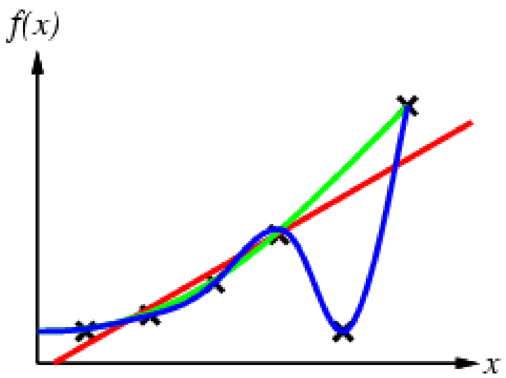
Polynomials of degree at most k



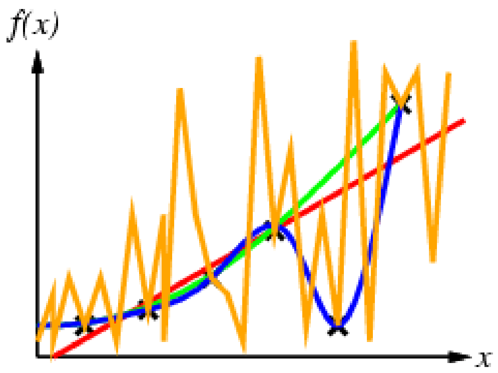
Linear hypothesis



Degree 1 polynomial hypothesis



Degree 3 polynomial and approximate linear fit



Sinusoidal hypothesis

How to choose from among multiple consistent hypotheses?

Ockham's razor:

- Prefer the simplest hypothesis consistent with data
- Maximize a combination of consistency and simplicity

Learning Decision Trees (LDT)

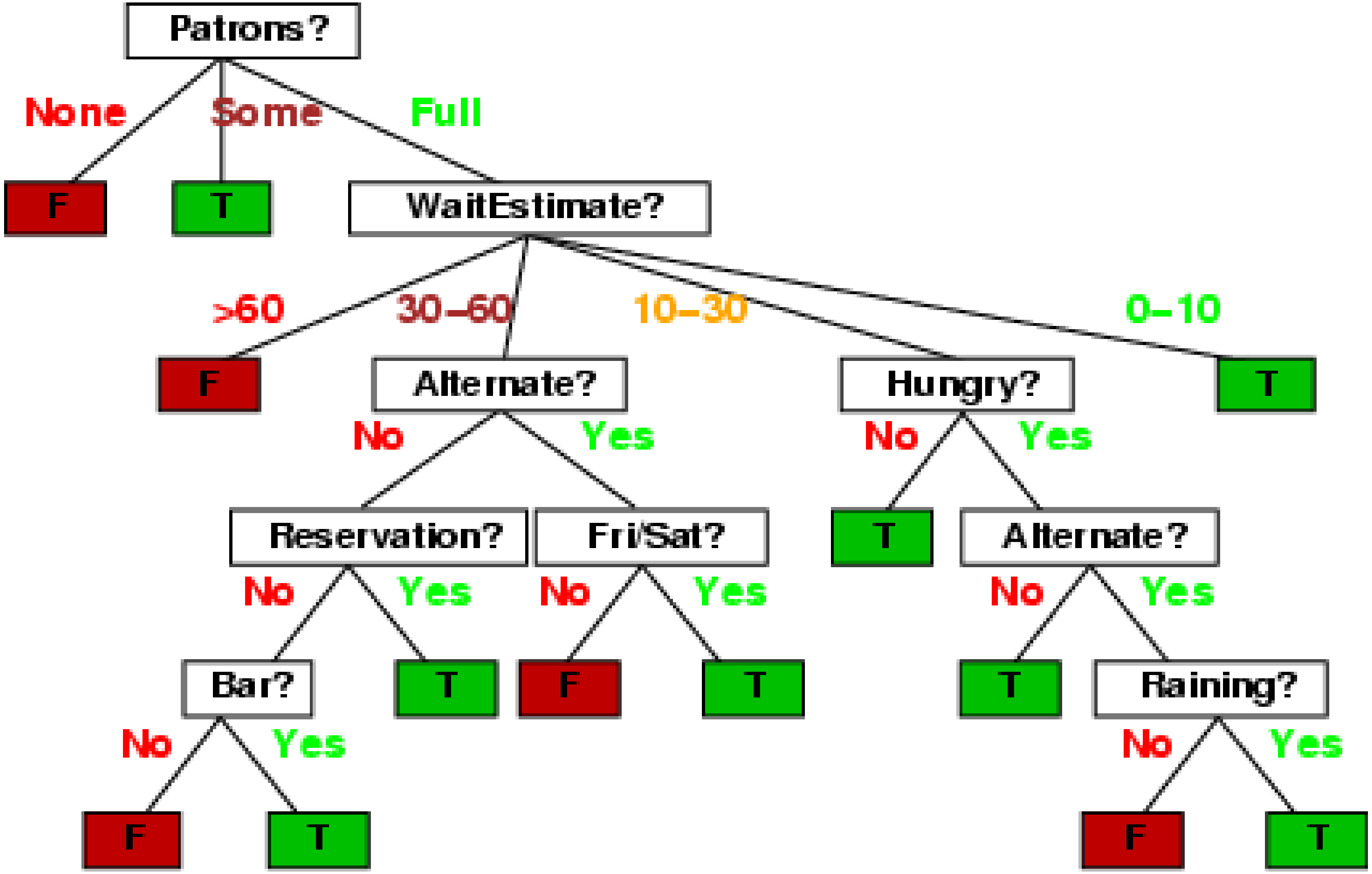
- Decision tree branch on values of a set of input attributes, leading to answers at the leaves.
- Construction (and optimization) of the tree is a learning problem.
  - Classification learning: learning a discrete function
  - Regression: learning a continuous function

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T

- Problem: Wait for a table at a restaurant?
- Attributes:

Alternate: is there an alternative restaurant nearby?
Bar: is there a comfortable bar area to wait in?
Fri/Sat: is today Friday or Saturday?
Hungry: are we hungry?
Patrons: number of people in the restaurant (None, Some, Full)
Price: price range ($, $$, $$$)
Raining: is it raining outside?
Reservation: have we made a reservation?
Type: kind of restaurant (French, Italian, Thai, Burger)
WaitEstimate: estimated waiting time (0-10, 10-30, 30-60, >60)

DT Example

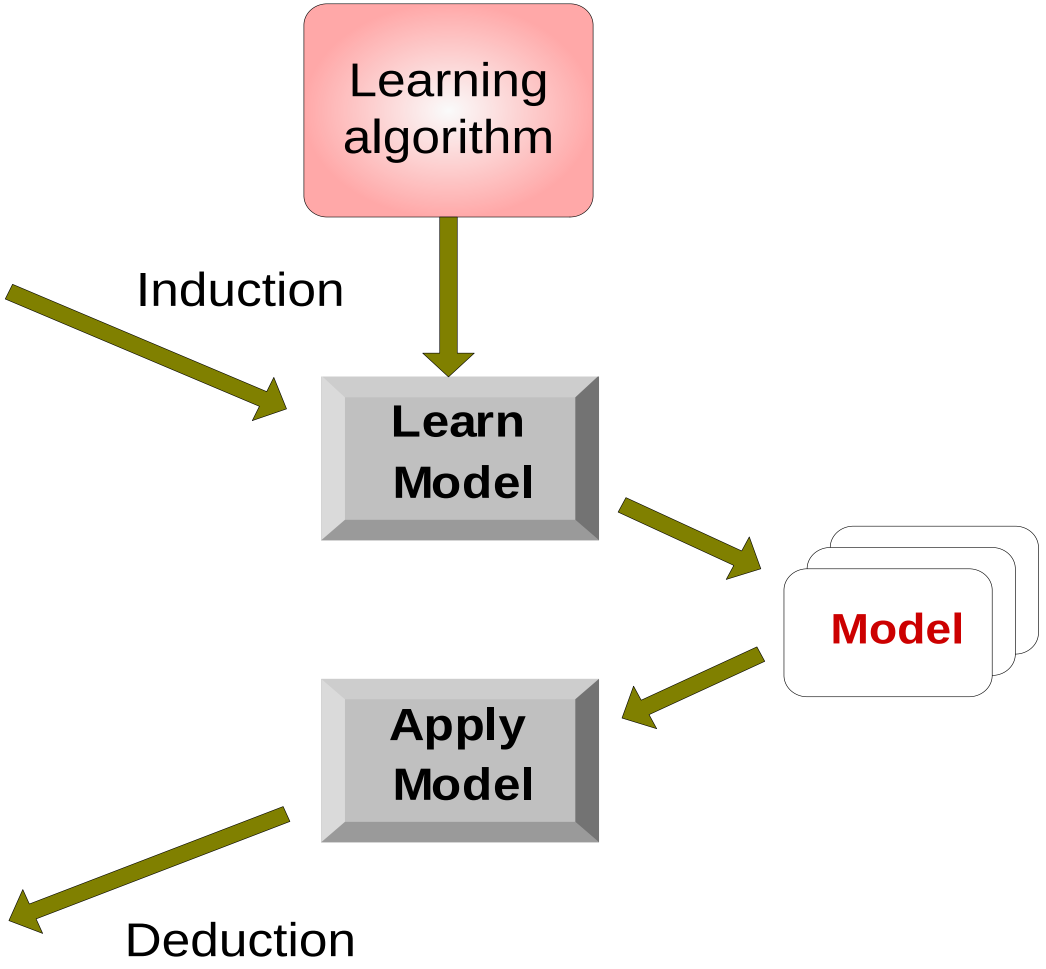


Patrons?
None
Some
Full
F
T
WaitEstimate?
>60
30-60
10-30
0-10
F
Alternate?
Hungry?
T
No
Yes
No
Yes
Reservation?
Fri/Sat?
T
Alternate?
No
Yes
No
Yes
No
Yes
Bar?
T
F
T
T
Raining?
No
Yes
No
Yes
F
T
F
T

Illustrating Classification Task

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set



Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set