Artificial Intelligence

Open Elective

Module 5: Learning CH15

Dr. Santhi Natarajan
Associate Professor
Dept of AI and ML
BMSIT, Bangalore

Contents

What is learning

- Denotes changes in the system that enable a system to do the same task more efficiently next time.
- Learning is constructing or modifying representations of what is being experienced.
- Learning is making useful changes in our minds.
- Learning improves understanding and efficiency
- Discover new things or structures which were previously unknown (data mining, scientific discovery)
- Fill in skeletal or incomplete observations or specifications about a domain (this expands the domain of expertise and lessens the brittleness of the system)
- Build software agents that can adapt to their users or to other software agents
- Reproduce an important aspect of intelligent behaviour.

Learning Systems

- Machine learning systems perform the following iteratively:
  - Produce a result
  - Evaluate it against expected result
  - Tweak a system

- Machine learning systems also discover patterns without prior expected results

- Open box: changes are clearly visible in the knowledge base and clearly interpretable by the human users.

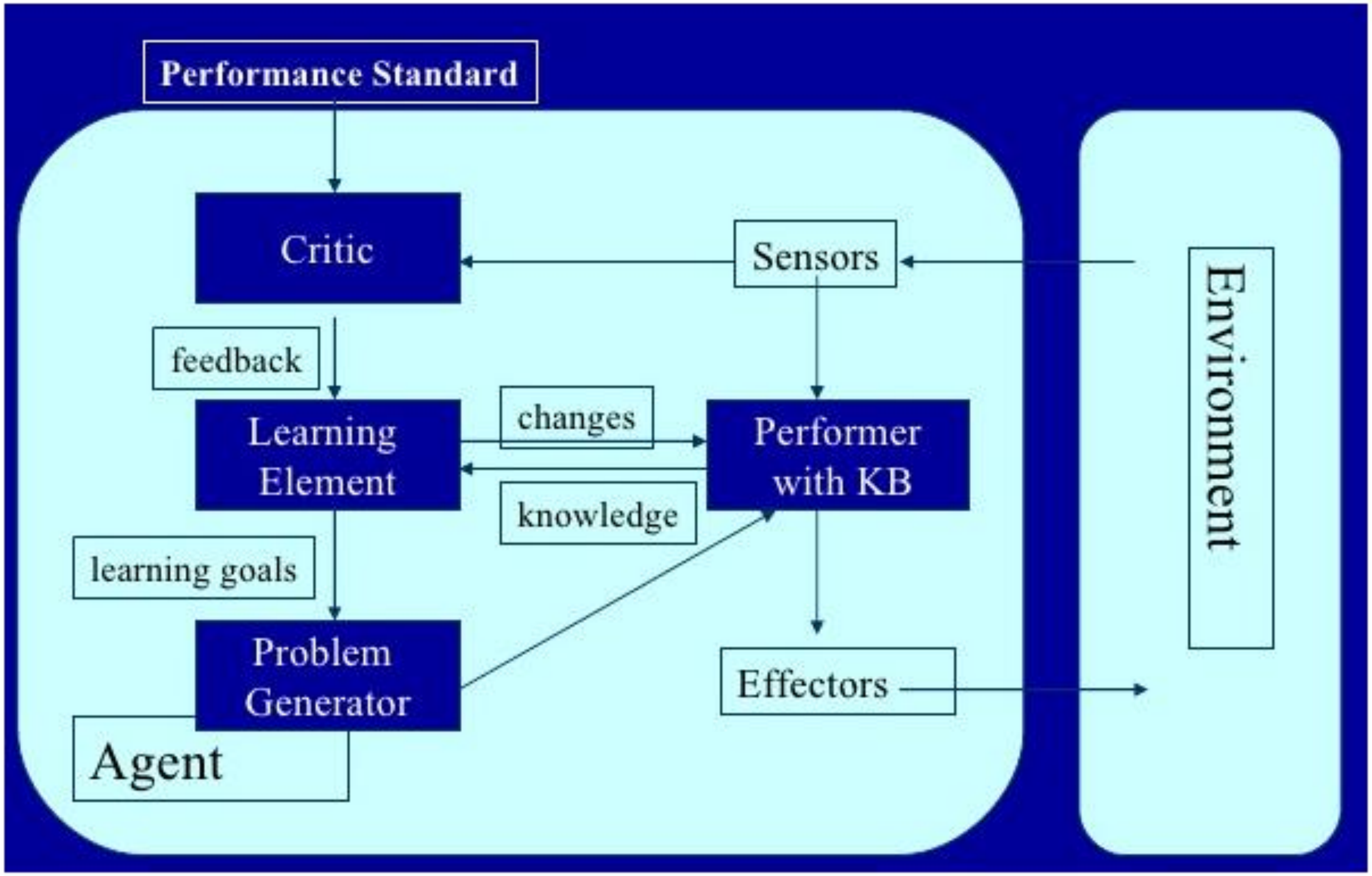
- Black box: changes done to the system are not readily visible or understandable.

Learner Architecture

- Machine learning systems has the four main components:
  - **Knowledge Base (KB)**:
    - ✓ what is being learnt
    - ✓ Representation of domain
    - ✓ Description and representation of problem space
  - **Performer**: does something with the knowledge base to produce results.
  - **Critic**: evaluates results produced against expected results
  - **Learner**: takes output from the critic and modifies something in the KB or the performer.
  - They may also need a problem generator to test performance against.

.

Learning Agent Architecture



Learning Examples

Problem	Representation	Performer (interacts with human)	Critic (human player)	Learner(elicits new questions to modify KBP
Animal guessing game	Binary decision tree	Walk the tree and ask associated questions	Human feedback	Elicit a question from the user and add it to the binary tree
Playing chess	The board layout, game rules, moves	Chain through the rules to identify move, use conflict resolution to choose one, output the move.	Who won (credit assignment problem)	Increase the weight for some rules and decrease for others.
Categorizing documents	Vector of word frequencies, corpus of documents	Apply appropriate functions to identify which category the file belongs to	A set of human-categorized documents	Modify the weights on the function and improve categorization
Fixing computers	Frequency matrix of causes and symptoms	Use known symptoms to identify potential causes.	Human input about symptoms and cause observed for a specific case	Update the frequency matrix with actual symptoms and outcomes
Identifying digits in Optical Character Recognition	Probability of digits, matrix of pixels, percentage of light, no: of straight lines	Input the features for a digit, output probability that it is one in the set from 0 to 9.	Human categorized training set	Modify the weights on the network of associations

Learning Paradigms

Paradigm	Description
Rote learning	Knowledge engineering: direct entry of rules and facts
Learning by taking advice	Human/system interaction producing explicit mapping
Learning in problem solving	Parameter adjustments
Learning from examples: induction	Using specific examples to reach general conclusions
Explanation based learning	Learn from a single example and later generalize. More analytical and knowledge intensive approach.
Learning through discovery	Unsupervised, specific goal not given
Learning through analogy	Determining correspondence between two different representations: case-based reasoning
Formal learning theory	Formal mathematical model of learning
Neural net learning and genetic learning	Evolutionary search techniques, based on an analogy to the "survival of the fittest"

Rote Learning

- Rote learning is the basic learning activity.
- It is also called memorization because the knowledge, without any modification is, simply copied into the knowledge base. Direct entry of rules and facts
- Knowledge base is captured knowledge.
- Traditional approach to develop ontologies.
- Data caching to improve performance.
- As computed values are stored, this technique can save a significant amount of time.
- Rote learning technique can also be used in complex learning systems provided sophisticated techniques are employed to use the stored values faster and there is a generalization to keep the number of stored information down to a manageable level.
- Checkers-playing program, for example, uses this technique to learn the board positions it evaluates in its look-ahead search.

Rote Learning

- Depends on two important capabilities of complex learning systems:
  - **Organized storage of information**: need sophisticated techniques for data retrieval. It will be much faster than recomputing the data.
  - **Generalization**: The number of distinct objects that might potentially be stored can be very large. To keep the number of stored objects down to a manageable levels.

Learning by taking advice

- This type is the easiest and simple way of learning.
- In this type of learning, a programmer writes a program to give some instructions to perform a task to the computer. Once it is learned (i.e. programmed), the system will be able to do new things.
- Also, there can be several sources for taking advice such as humans(experts), internet etc.
- However, this type of learning has a more necessity of inference than rote learning.
- As the stored knowledge in knowledge base gets transformed into an operational form, the reliability of the knowledge source is always taken into consideration.
- The programs shall operationalize the advice by turning it into a single or multiple expressions that contain concepts and actions that the program can use while under execution.
- This ability to operationalize knowledge is very critical for learning. This is also an important aspect of Explanation Based Learning (EBL).

Learning in Problem Solving

- When the program does not learn from advice, it can learn by generalizing from its own experiences.
  - Learning by parameter adjustment
  - Learning with macro-operators
  - Learning by chunking
  - The unity problem

Learning in Problem Solving

Learning by parameter adjustment

- Here the learning system relies on evaluation procedure that combines information from several sources into a single summary static.
- For example, the factors such as demand and production capacity may be combined into a single score to indicate the chance for increase of production.
- But it is difficult to know a priori how much weight should be attached to each factor.
- The correct weight can be found by taking some estimate of the correct settings and then allow the program modify its settings based on its experience.
- Features that appear to be good predictors of overall success will have their weights increases, while those that do not will have their weights decreased.
- This type of learning systems is useful when little knowledge is available.
- In game programs, for example, the factors such as piece advantage and mobility are combined into a single score to decide whether a particular board position is desirable. This single score is nothing but a knowledge which the program gathered by means of calculation.

Learning in Problem Solving

Learning by parameter adjustment

- Programs do this in their static evaluation functions, in which a variety of factors are combined into a single score. This function as a polynomial form is given below:

$$c_1t_1 + c_2t_2 + c_3t_3 + \ldots$$

- The t terms are the values of the features that contribute to the evaluation. The c terms are the coefficients or weights that are attached to each of these values. As learning progresses, the c values will change.

- In designing programs it is often difficult to decide on the exact value to give each weight initially. So the basic idea of idea of parameter adjustment is to:

  - ➢ Start with some estimate of the correct weight settings.
  - ➢ Modify the weight in the program on the basis of accumulated experiences.
  - ➢ Features that appear to be good predictors will have their weights increased and bad ones will be decreased.

Learning in Problem Solving

Learning by parameter adjustment

- Important factors that affect the performance are:
  - When should the value of a coefficient be increased and when should it be decreased
    - ✓ The coefficients of terms that predicted the final outcome accurately should be increased, while the coefficients of poor predictors should be decreased.
    - ✓ The problem of appropriately assigning responsibility to each of the steps that led to a single outcome is known as credit assignment system.
  - By how much should be value be changed.
- Learning procedure is a variety of hill-climbing.

- This method is very useful in situations where very little additional knowledge is available or in programs in which it is combined with more knowledge intensive methods.

Learning in Problem Solving

Learning with Macro-Operators

- Sequence of actions that can be treated as a whole are called macro-operators.
- Once a problem is solved, the learning component takes the computed plan and stores it as a macro-operator.
- The preconditions are the initial conditions of the problem just solved, and its post conditions correspond to the goal just achieved.
- The problem solver efficiently uses the knowledge base it gained from its previous experiences.
- By generalizing macro-operators the problem solver can even solve different problems. Generalization is done by replacing all the constants in the macro-operators with variables.
- The STRIPS, for example, is a planning algorithm that employed macro-operators in it's learning phase. It builds a macro operator MACROP, that contains preconditions, post-conditions and the sequence of actions. The macro operator will be used in the future operation.
- The set of problems for which macro-operators are critical are exactly those problems with non-serializable sub goals.(working on one subgoal will necessarily interfere with the previous solution to another subgoal).
- One macro operator can produce a small global change in the world, even though the individual operators that make it up produce many undesirable local changes.
- Domain specific knowledge we need can be learnt in the form of macro operators.

Learning in Problem Solving

Learning by chunking

- Chunking is similar to learnig with macro-operators. Generally, it is used by problem solver systems that make use of production systems.

- A production system consists of a set of rules that are in if-then form. That is given a particular situation, what are the actions to be performed. For example, if it is raining then take umbrella.

- Production system also contains knowledge base, control strategy and a rule applier. To solve a problem, a system will compare the present situation with the left hand side of the rules. If there is a match then the system will perform the actions described in the right hand side of the corresponding rule.

- Problem solvers solve problems by applying the rules. Some of these rules may be more useful than others and the results are stored as a **chunk**.

- Chunking can be used to learn general search control knowledge.

- **Several chunks may encode a single macro-operator and one chunk may participate in a number of macro sequences**.

Learning in Problem Solving

Learning by chunking

- Chunks learned in the beginning of problem solving, may be used in the later stage. The system keeps the chunk to use it in solving other problems.

- **Soar** is a general cognitive architecture for developing intelligent systems. Soar requires knowledge to solve various problems. It acquires knowledge using chunking mechanism. The system learns reflexively when impasses have been resolved. An impasse arises when the system does not have sufficient knowledge. Consequently, Soar chooses a new problem space (set of states and the operators that manipulate the states) in a bid to resolve the impasse. While resolving the impasse, the individual steps of the task plan are grouped into larger steps known as chunks. The chunks decrease the problem space search and so increase the efficiency of performing the task.

- In Soar, the knowledge is stored in long-term memory. Soar uses the chunking mechanism to create productions that are stored in long-term memory. A chunk is nothing but a large production that does the work of an entire sequence of smaller ones. The productions have a set of conditions or patterns to be matched to working memory which consists of current goals, problem spaces, states and operators and a set of actions to perform when the production fires. Chunks are generalized before storing. When the same impasse occurs again, the chunks so collected can be used to resolve it.

Learning in Problem Solving

The Utility Problem

- The utility problem in learning systems occurs when knowledge learned in an attempt to improve a system's performance degrades it instead.
- The problem appears in many AI systems, but it is most familiar in speedup learning. Speedup learning systems are designed to improve their performance by learning control rules which guide their problem-solving performance. These systems often exhibit the undesirable property of actually slowing down if they are allowed to learn in an unrestricted fashion.
- Each individual control rule is guaranteed to have a positive utility (improve performance) but, in concert, they have a negative utility (degrade performance).
- One of the causes of the utility problem is the serial nature of current hardware. The more control rules that speedup learning systems acquire, the longer it takes for the system to test them on each cycle.
- One solution to the utility problem is to design a parallel memory system to eliminate the increase in match cost. his approach moves the matching problem away from the central processor and into the memory of the system. These so-called active memories allow memory search to occur in "nearly constant-time" in the number of data items, relying on the memory for fast, simple inference and reminding.

Learning in Problem Solving

The Utility Problem

- PRODIGY program maintains a utility measure for each control rule. This measure takes into account the average savings provided by the rule, the frequency of its application and the cost of matching it.

- If a proposed rule has a negative utility, it is discarded or forgotten.

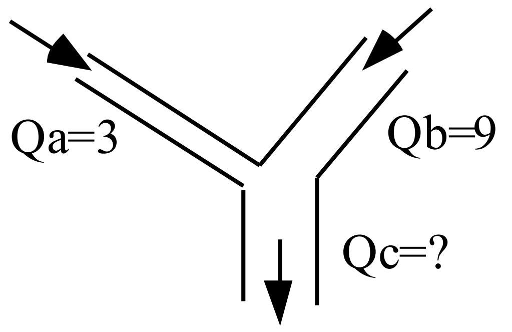
- If not, it is placed in long term memory with the other rules. It is then monitored during subsequent problem solving.
- If its utility falls, the rule I discarded.

- Empirical experiments have demonstrated the effectiveness of keeping only those control rules with high utility.

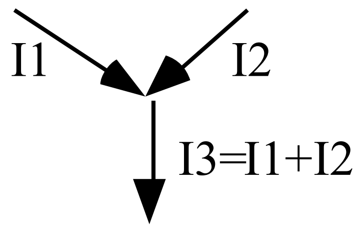
- Such utility considerations apply to a wide range of learning problems.

Learning by Analogy

Learning by analogy means acquiring new knowledge about an input entity by transferring it from a known similar entity.



Simple Hydraulics Problem



Kirchoff's First Law

One may infer, by analogy, that hydraulics laws are similar to Kirchoff's laws, and Ohm's law.

Learning by Analogy

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:

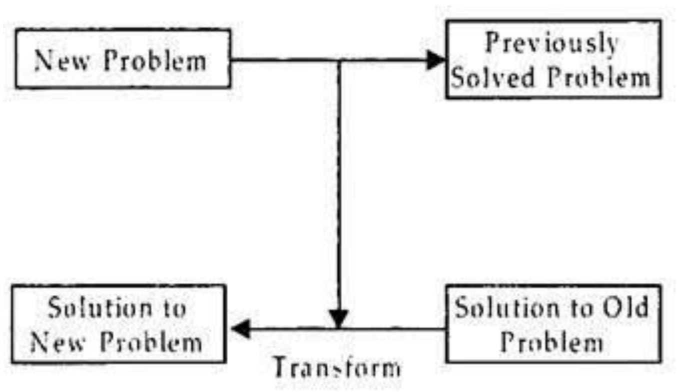
Pressure Drop is like Voltage Drop

A variable in a programming language is like a box.

Transformational Analogy

Look for a similar solution and copy it to the new situation making suitable substitutions where appropriate.

E.g. Geometry.
If you know about lengths of line segments and a proof that certain lines are equal then we can make similar assertions about angles.



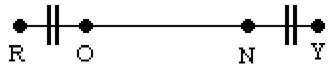
Transformational Analogy.

We know that lines *RO* = *NY* and angles *AOB* = *COD*

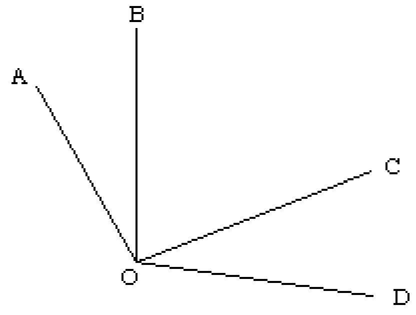
We have seen that *RO* + *ON* = *ON* + *NY* - additive rule.

So we can say that angles *AOB* + *BOC* = *BOC* + *COD*

So by a transitive rule line *RN* = *OY*
So similarly angle *AOC* = *BOD*



(a) Old Proof



(b) New Proof

Derivational Analogy

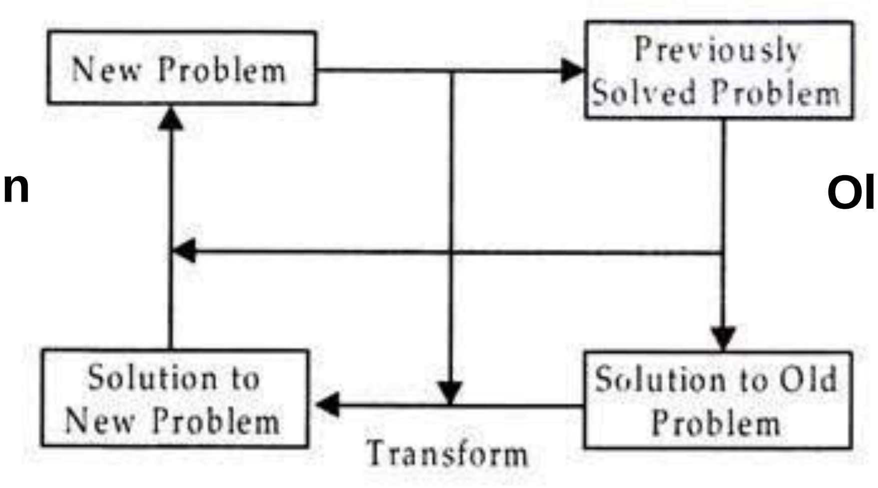
Two problems share significant aspects if they match within a certain threshold, according to a given similarity metric.

The solution to the retrieved problem is perturbed incrementally until it satisfies the requirements of the new problem.

Transformational analogy does not look at how the problem was solved -- it only looks at the final solution. The history of the problem solution - the steps involved - are often relevant.

New derivation

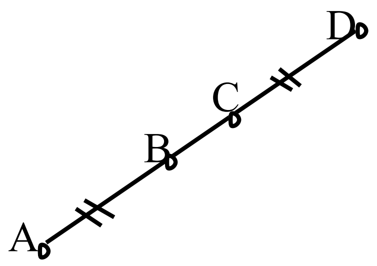
Old derivation



DevirationalAnalogy.

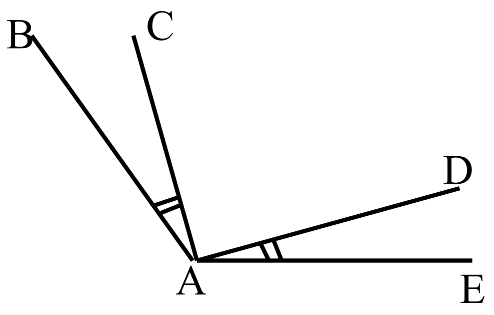
Derivational Analogy

GIVEN: AB = CD
PROVE: AC = BD



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

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



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

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

Explanation based Learning

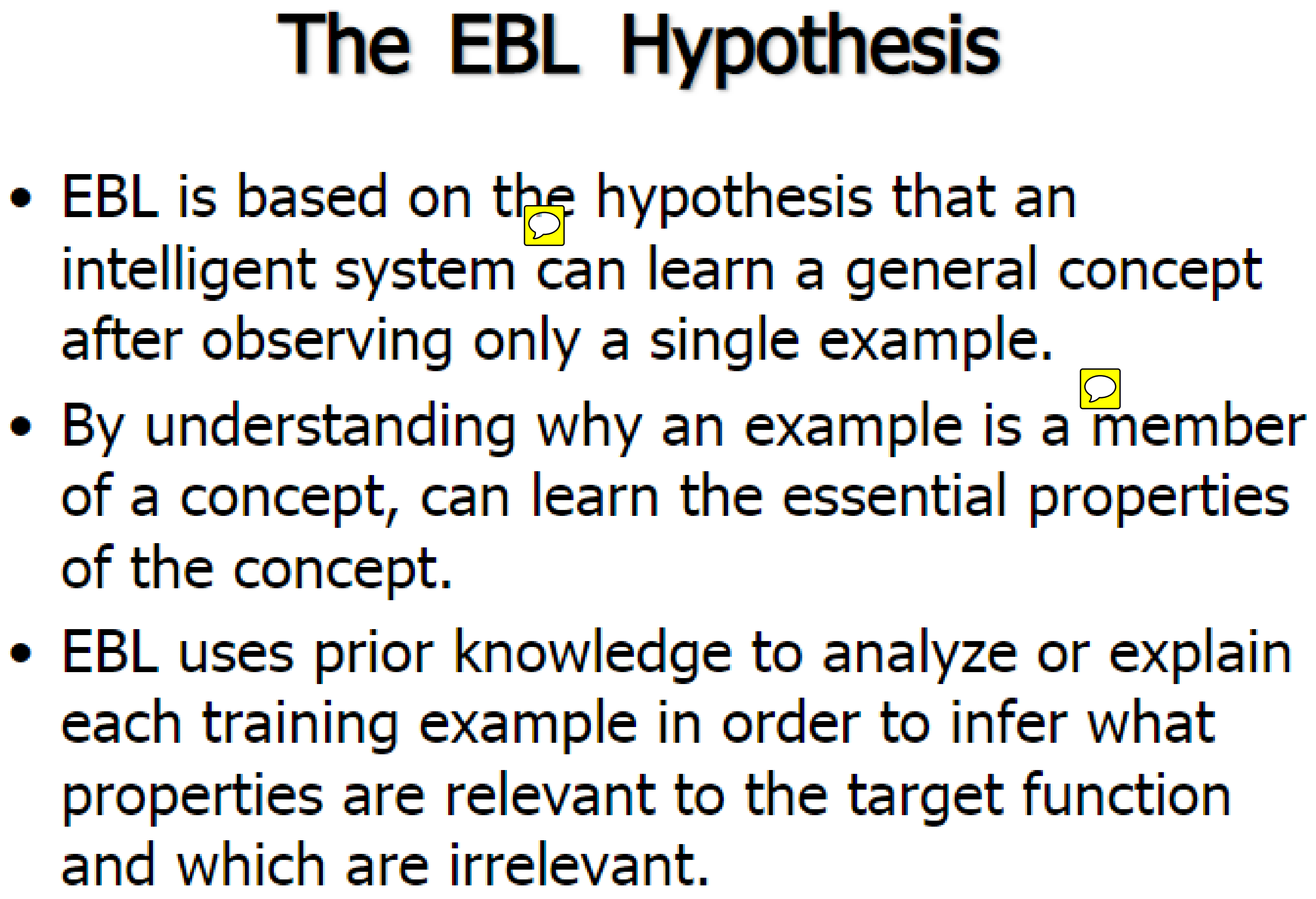
What is EBL ?

- Learning *general* problem-solving techniques by observing and analyzing human solutions to *specific* problems.
- EBL attempts to formulate a *generalization* after observing only a single example.
- Introduced by Gerald De Jong in 1981.

"Hey! Look what Zog do!"

(drawn by Gary Larson)

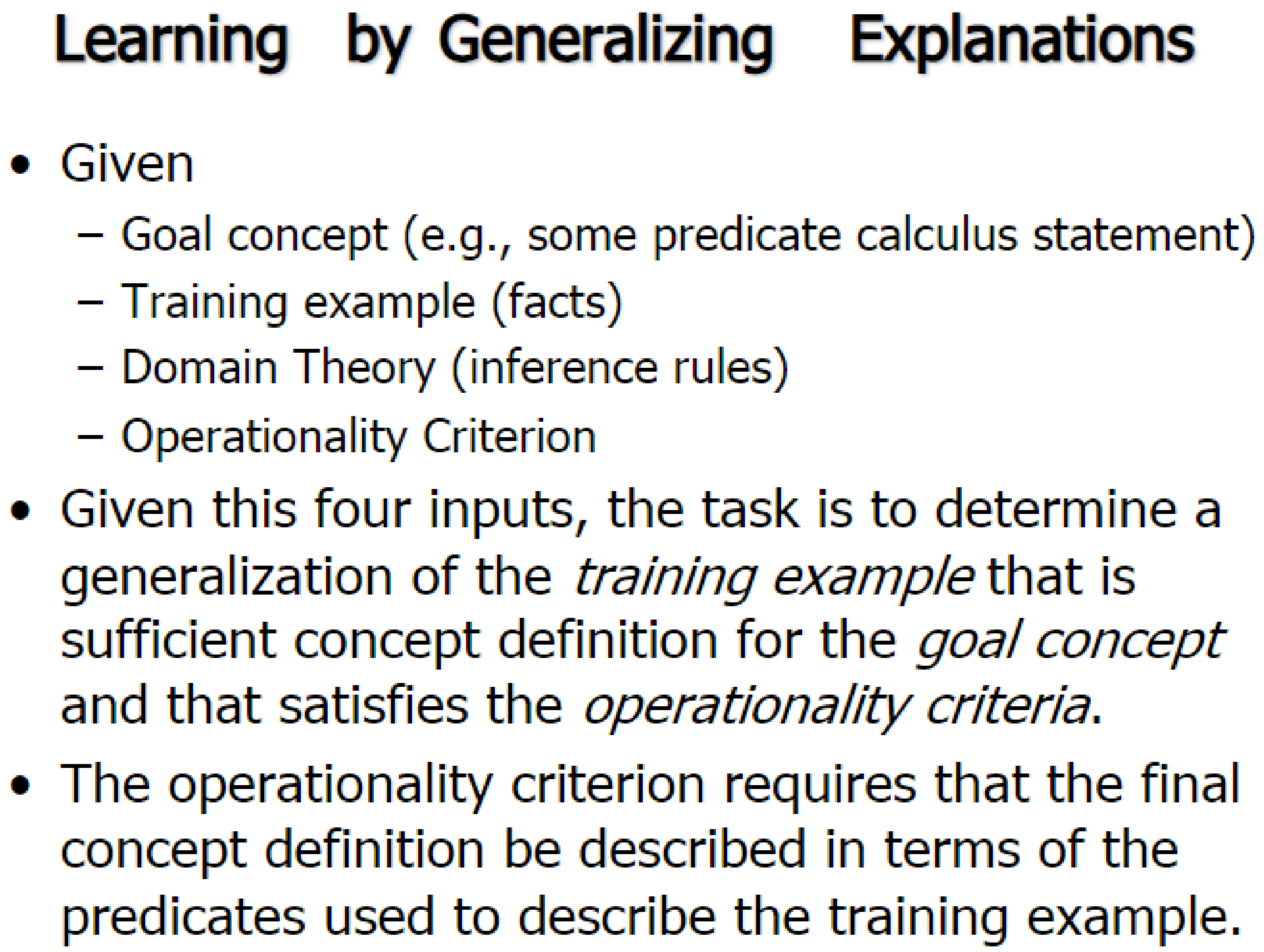
Explanation based Learning



The EBL Hypothesis

- EBL is based on the hypothesis that an intelligent system can learn a general concept after observing only a single example.
- By understanding why an example is a member of a concept, can learn the essential properties of the concept.
- EBL uses prior knowledge to analyze or explain each training example in order to infer what properties are relevant to the target function and which are irrelevant.

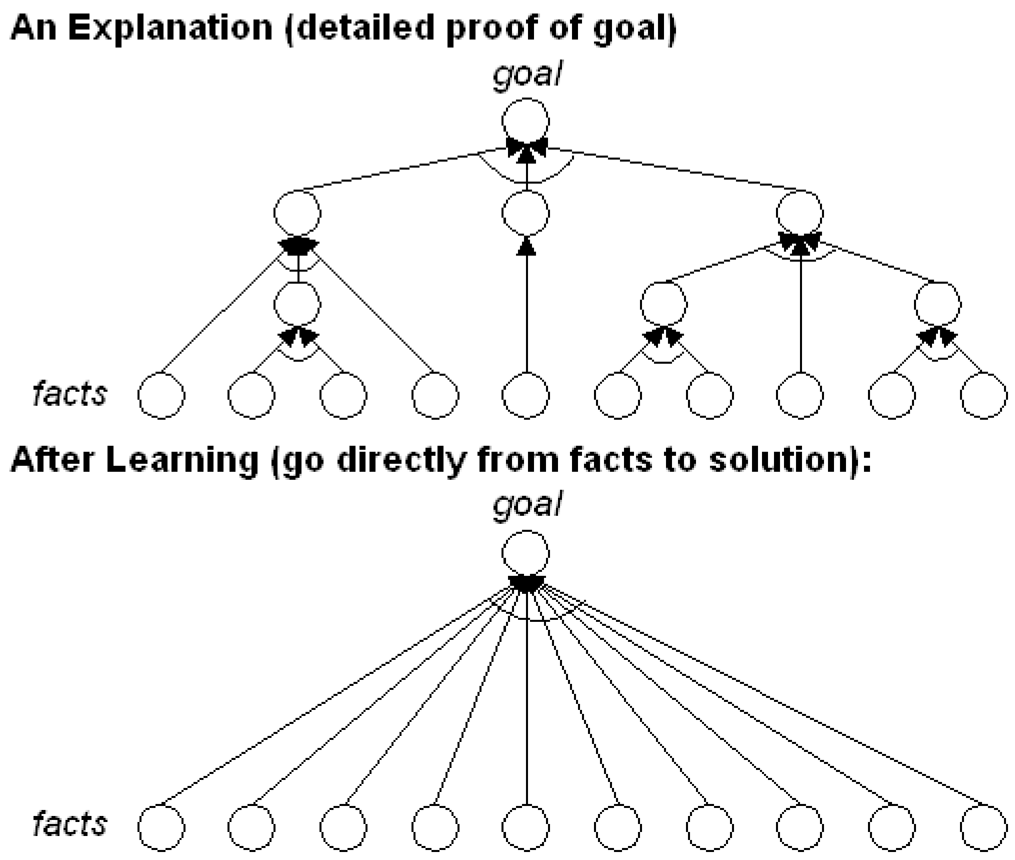
Explanation based Learning



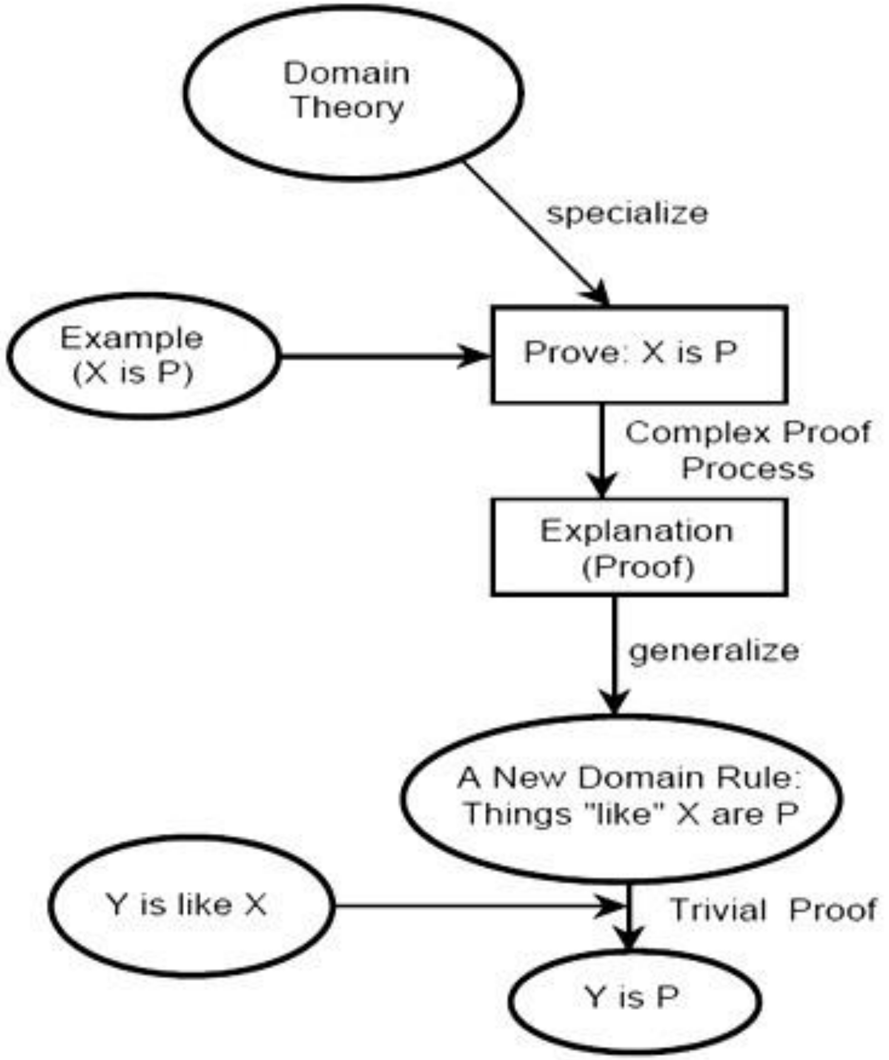
Learning by Generalizing Explanations

- Given
  - Goal concept (e.g., some predicate calculus statement)
  - Training example (facts)
  - Domain Theory (inference rules)
  - Operationality Criterion
- Given this four inputs, the task is to determine a generalization of the *training example* that is sufficient concept definition for the *goal concept* and that satisfies the *operationality criteria*.
- The operationality criterion requires that the final concept definition be described in terms of the predicates used to describe the training example.

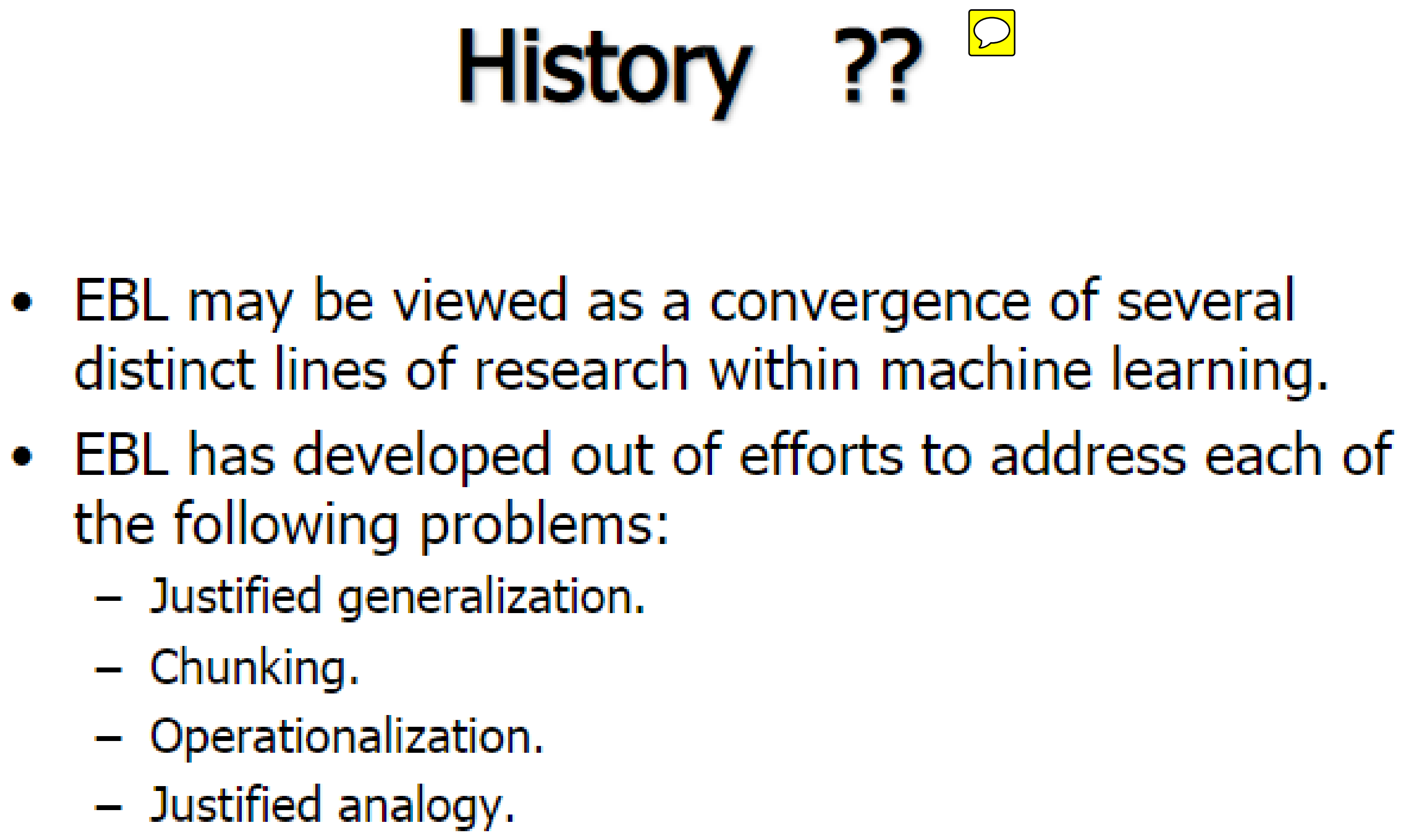
Explanation based Learning


An Explanation (detailed proof of goal)
goal
facts
After Learning (go directly from facts to solution):
goal
facts

Explanation based Learning



Explanation based Learning



History ??

- EBL may be viewed as a convergence of several distinct lines of research within machine learning.
- EBL has developed out of efforts to address each of the following problems:
  - Justified generalization.
  - Chunking.
  - Operationalization.
  - Justified analogy.

Learning by Discovery

An entity acquires knowledge without the help of a teacher.

Theory Driven Discovery - AM (1976)

AM is a program that discovers concepts in elementary mathematics and set theory.

AM has 2 inputs:

- A description of some concepts of set theory (in LISP form). *E.g.* set union, intersection, the empty set.
- Information on how to perform mathematics. *E.g.* functions.

How does AM work?

AM employs many general-purpose AI techniques:

- A frame based representation of mathematical concepts.
  AM can create new concepts (slots) and fill in their values.
- Heuristic search employed
  250 heuristics represent *hints* about activities that might lead to interesting discoveries.
  How to employ functions, create new concepts, generalisation *etc.*
- Hypothesis and test based search.
- Agenda control of discovery process.

Learning by Discovery

AM *discovered*:

- **Integers**-- it is possible to count the elements of this set and this is an the image of this counting function -- the integers -- interesting set in its own right.

- **Addition**-- The union of two disjoint sets and their counting function

- **Multiplication**-- Having discovered addition and multiplication as laborious set-theoretic operations more effective descriptions were supplied by hand.

- **Prime Numbers**-- factorisation of numbers and numbers with only one factor were discovered.

- **Golbach's Conjecture**-- Even numbers can be written as the sum of 2 primes. *E.g.* 28 = 17 + 11.

- **Maximally Divisible Numbers**-- numbers with as many factors as possible. A number *k* is maximally divisible is *k* has more factors than any integer less than *k*. *E.g.* 12 has six divisors 1,2,3,4,6,12.

Learning by Discovery

Data Driven Discovery -- BACON (1981)

Many discoveries are made from observing data obtained from the world and making sense of it -- *E.g.* Astrophysics - discovery of planets, Quantum mechanics - discovery of sub-atomic particles.

BACON is an attempt at providing such an AI system. BACON system outline:

- Starts with a set of variables for a problem.
  - *E.g.* BACON was able able to derive the *ideal gas law*. It started with four variables p - gas pressure, V -- gas volume, n -- molar mass of gas, T -- gas temperature. Recall $pV/nT = k$ where k is a constant.

- Values from experimental data from the problem are inputted.

- BACON holds some constant and attempts to notice trends in the data.

- Inferences made.

BACON has also been applied to Kepler's 3rd law, Ohm's law, conservation of momentum and Joule's law.

Learning by Discovery

Clustering

- Clustering involves grouping data into several new classes.
- It is a common descriptive task where one seeks to identify a finite set of categories or clusters to describe the data. For example, we may want to cluster houses to find distribution patterns.
- Clustering   is the process of grouping a set of physical or abstract objects into classes of similar objects.
- A cluster is a collection of data objects that are similar to one another within the same cluster and are dissimilar to the objects in other clusters. Clustering analysis helps construct meaningful partitioning of a large set of objects.

Learning by Discovery

Clustering

The task of clustering is to maximize the intra-class similarity and minimize the interclass similarity.

- Given N k-dimensional feature vectors, find a "meaningful" partition of the N examples into *c* subsets or groups
- Discover the "labels" automatically
- *c* may be given, or "discovered“
- much more difficult than classification, since in the latter the groups are given, and we seek a compact description

Learning by Discovery

AutoClass

- AutoClass is a clustering algorithm based upon the Bayesian approach for determining optimal classes in large datasets.

.

- Given a set *X={X1, …, Xn}* of data instances *Xi* with unknown classes, the goal of Bayesian classification is to search for the best class description that predicts the data in a model space.

.

- Class membership is expressed probabilistically.

- An instance is not assigned to a unique class, but it has a probability (expressed as weight values) of belonging to each of the possible classes.

- AutoClass calculates the likelihood of each instance belonging to each class *C* and then calculates a set of weights *wij=(Ci / SjCj)* for each instance.

- Weighted statistics relevant to each term of the class likelihood are calculated for estimating the class model.
- The classification step is the most computationally intensive. It computes the weights of every instance for each class and computes the parameters of a classification.

Formal Learning

Formal learning theory

- Theory of the learnable by Valiant: classifies problems by how difficult they are to learn.

- Formally, a device can learn a concept if it can, given positive and negative examples, produce an algorithm that will classify future examples correctly with probability 1/h.

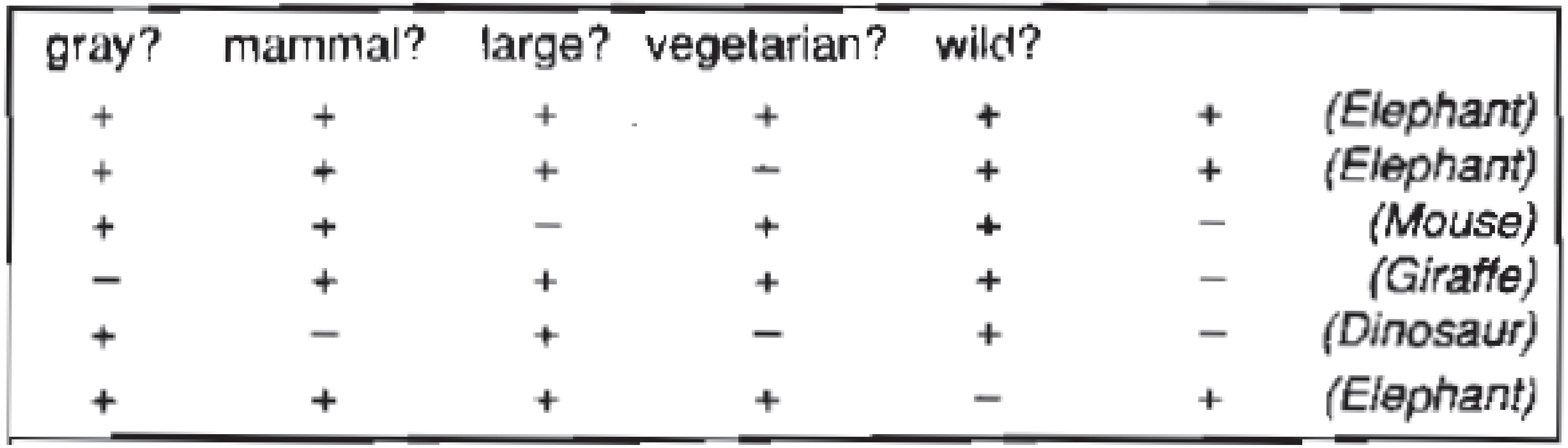
- Complexity of learning a function is decided by three factors:
  - ➢ The error tolerance (h)
  - ➢ The number of binary features present in the example (t)
  - ➢ Size of rules necessary to make the discrimination (f)

- If the number of training examples is a polynomial in h,t, f, then the system is said to be trainable.

- Example: learning feature descriptions

- Mathematical theory will be used to quantify the use of knowledge

Formal Learning



gray?	mammal?	large?	vegetarian?	wild?		
+	+	+	+	+	+	*(Elephant)*
+	+	+	−	+	+	*(Elephant)*
+	+	−	+	+	−	*(Mouse)*
−	+	+	+	+	−	*(Giraffe)*
+	−	+	−	+	−	*(Dinosaur)*
+	+	+	+	−	+	*(Elephant)*

Other Learning Models

Neural net learning and genetic learning

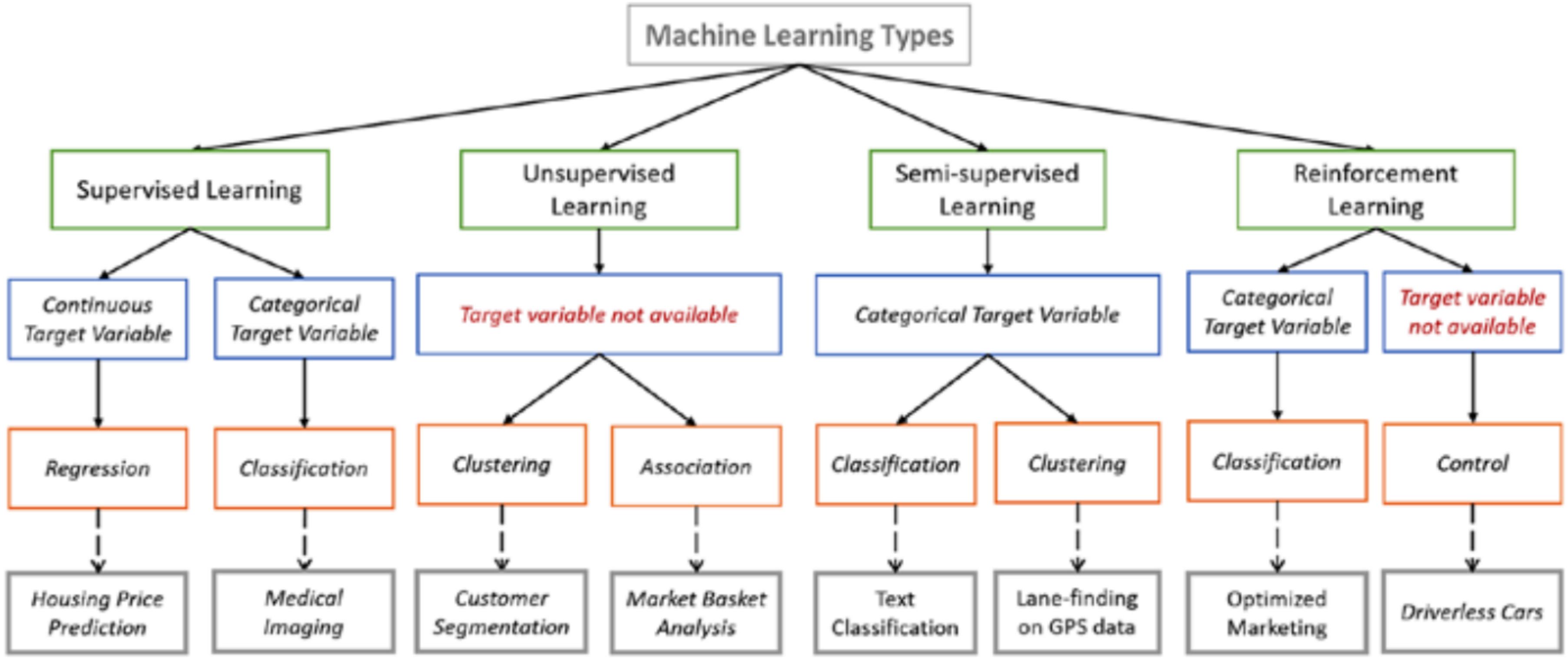
- Neural networks
- Population genetics and selection

Learning in Problem Solving

Neural net learning and genetic learning

- Neural networks
- Population genetics and selection

What is learning



What is learning