Chapter 13
Machine Learning

Abstract To adapt to the environment it necessary that intelligent machines must have the capability to learn. This chapter presents the basic concepts and techniques of learning found in humans, as well as their implementation aspects for machines. The chapter presents the challenges of building learning capabilities in machines, types of machine learning, and the relative efforts needed to build these learning capabilities. The philosophy of the discipline of machine learning is presented. The basic model of learning is discussed, followed by the classes of learning—supervised and unsupervised—then various techniques of inductive learning—argument based learning, online concept learning, propositional and relational learning, and learning through decision trees—are presented in sufficient details. Other techniques like discovery-based learning, reinforced learning, learning and reasoning through analogy, explanation-based learning are presented, with some worked examples. Finally, the potential applications of machine learning, the basic research problems in machine learning, followed by chapter summary, and a set of exercises are appended.

Keywords Machine learning · Machine learning classes · Machine learning model · Supervised learning · Unsupervised learning · Inductive learning · Decision-tree-based learning · Argument-based learning · Propositional learning · Discovery-based learning · Learning through analogy · Reinforced learning · Explanation-based learning · Machine learning applications · Research problems

13.1 Introduction

Learning is one of the most important activities of human beings and living beings in general, which help us in adapting to the environment. Humans learn from nature as well as from special learning environments through different techniques that vary in complexity. The learning requires transformations of ideas and information structures in the human mind.

This chapter presents the basic concepts and examples of learning techniques found in humans, and their implementations aspects for learning in machines. A common view holds that learning involves making changes in the learning system

K. R. Chowdhary, *Fundamentals of Artificial Intelligence*,
https://doi.org/10.1007/978-81-322-3972-7_13

that will improve it in some way. These changes are *adaptive*, because it is found that the system does some task or tasks more effectively the next time when they are drawn from the same population of tasks.

Most of the knowledge of the world we have around us, and about the explicit set of tasks, is not formalized, and even not available in text form. The latter is a necessary requirement for it to be understood by any computer program. This is the reason it is not easy to write a program for a computer to do many tasks that we humans do so easily, such as understanding spoken sentences, images, languages, or driving a car. Any attempt to achieve this, i.e., organizing sets of facts in elaborate data structures to enable computers to understand it, has achieved very little success.

The concept of *learning* is based on the principles of training the computing machines, and enabling them to teach themselves. Tenet of these machines/programs is related to what we consider as good decision. For humans and animals, evolutionary principles dictate that any decision made by them should lead to such behaviors that optimize the chances of *survival* and *reproduction*. In the societies of human beings, a good decision is one that might include social interactions that bring *status*, or a sense of *well-being*. For a machine such as a self-driving car, such criteria cannot exist! So the quality of decision-making for a self-driving car is measured based on the criteria as to how close the autonomous vehicle imitates the behavior of trained human drivers.

It is important to note that the knowledge required to make a good decision in a particular situation is not necessarily clear to the extent to code it into a computer program. Consider, for example, that a mouse has knowledge about its surroundings, and has an innate sense of where to sniff and how to find food or mates, and at the same time save itself from predators. It is not easy for any programmer to specify step-by-step procedures or a set of instructions to produce such behaviors for some artificial mouse. However, this all remains encoded in the brain of a mouse in the form of knowledge.

Before we think about creating computers (programs) that can train themselves, it is important to answer the fundamental question as to how we humans acquire knowledge. Partly, the knowledge in humans may be innate, but most of it is acquired through experience and observation. What we know intuitively has been often learned from examples and practice—the process which cannot be turned into a clear sequence of steps as a computer program. Since 1950 people have looked for ways and means and tried to refine general principles that allow animals or humans, to acquire knowledge through experience. These are the subjects of discussion in this chapter. Machine learning aims to create theories and procedures—*learning algorithms*—that allow machines to learn. One such is learning from examples presented to them, called learning by examples [2].

Learning Outcomes of This Chapter:

1. List the differences among the three main styles of learning: supervised, reinforcement, and unsupervised. [Familiarity]

2. Identify examples of classification tasks, including the available input features and output to be predicted. [Familiarity]
3. Explain the difference between inductive and deductive learning. [Familiarity]
4. Implement simple algorithms for supervised learning, reinforcement learning, and unsupervised learning. [Usage]
5. Determine which of the three learning styles is appropriate to a particular problem domain. [Usage]
6. Evaluate the performance of a simple learning system on a real-world data set. [Assessment]
7. Characterize the state of the art in learning theory, including its achievements and its shortcomings. [Familiarity]

13.2 Types of Machine Learning

The field of machine learning is concerned with the development of computational theories for learning processes and building learning machines. Because the ability to learn is a fundamental to any intelligent behavior, the concerns and goals of machine learning are central to the discipline of Artificial Intelligence. In the learning process, the learner transforms the information provided by a teacher (or environment) into some new form, and these are stored in that form for use in the future. The nature of this knowledge and transformation are the deciding factors for the type of learning strategy used. Following are the fundamental *strategies of learning*.

Rote Learning
Learning by Instruction
Learning by Deduction
Learning by Analogy
Learning by Induction (or Similarity)
Reinforcement Learning
Discovery-based Learning

These strategies are in increasing order of complexity of transformation required in the initial information, which is stored as knowledge base. The complexity orders of these methods are such that, an increasing difficulty level on the part of the student (learner) results in a correspondingly decreasing complexity on the part of the teacher (environment), and vice versa. Following is the brief introduction about each of these methods.

Rote Learning

The *rote* learning is the simplest among all the learning, and requires the least amount of inferencing from the knowledge. To accomplish the inferencing later, the original knowledge is copied in the same form in which it will be used later. Hence, the information from the teacher is more or less directly accepted and memorized by the

learner. This kind of learning is used when we learn the tables in the early school classes. A major concern in this learning is how to index the stored knowledge for future retrieval.

Learning by Instruction

The learning which is next higher in efforts after rote learning is learning by *instruction*. This approach requires the knowledge to be transformed into an operational form before it can be integrated into the knowledge base. This learning takes place among the students in a class when the class teacher presents a number of facts directly in a properly organized form to the students.

In learning by instruction (or *learning by being told*), the basic transformations performed by a learner are *selection* and *reformation* (mainly at a syntactic level) of information provided by the teacher.

Note that, depending on how the teachers presents the knowledge in the class, different types of learning take place, and learning by instructions is not the only learning induced by a teacher into his/her students.

Learning by Deduction

Deductive learning is carried out through a sequence of deductive inference steps using known facts. Using this, new facts and knowledge are logically derived. For example, if we have the knowledge of a rule, say "a father's father is a grand father," and we have the case that a is the father of b and b is the father of c, then we can say that a is the grand father of c. Here we have deduced the result, hence the name deductive learning.

Learning by Analogy

Analogical learning provides learning of new concepts through the use of similar known concepts or solutions. This is used, for example, when we solve some problems in our college examination, based on the solution of somewhat similar solutions we have already done at home. Also, for example, learning to drive a bike when we already know how to drive a bicycle is an example of analogical learning.

Learning by Induction

We make use of an inductive learning when after seeing a number of instances or examples of a concept we formulate a general concept.

In any human learning activity, mostly the strategies out of those discussed above are involved. It is important to understand the difference between these strategies on the part of designing a learning system. Though most current artificial systems focus on any single learning strategy, one may expect that machine learning research will give increasing attention to a multi-strategy approach, the one which is close to the human learning system. We as a learner use almost always a combination of strategies in our daily life. For example, remembering a telephone number, we use rote learning; seeing someone wearing a new dress every day, we learn that so-and-so is rich through inductive learning. Learning that those students who complete their

homework in time get more Cumulative Percentile Index (CPI) is also an example of inductive learning.

Another example, when we learn Java language having learned *C++* is analogical learning. When we are asked to solve a new puzzle, it is *discovery-based learning*. Whenever, a teacher appreciates the student's efforts by saying "very good", for asking an interesting question, or answering a question, it is *reinforcement-based learning* for the student.

When you have concluded based on some clues that a teacher is not in a mood of teaching, most likely you have used deductive learning (or reasoning).

13.3 Discipline of Machine Learning

Any field of scientific study is best defined by the central question it asks. That way, the field of Machine Learning seeks the answer to the following questions:

> 1. How to build a computer system that can automatically improve its performance with experience, and 2. What are the fundamental laws that govern the learning processes?

The faculty of *Learning*, like the faculty of *Intelligence*, involves such a vast range of processes that it is difficult to define it precisely. A dictionary definition of learning includes phrases such as "to gain knowledge, or understanding of, or skill in, by study, instruction, or experience." Another dictionary definition says, learning is "modification of a behavioral tendency by experience." Zoologists and psychologists also study the learning processes in animals and humans.

This chapter focuses on learning in machines. There are several similarities between learning in animals and in machines. Many techniques in machine learning are derived from the theories of animal and human learning and are presented in the form of computational/cognitive models by psychologists. It is quite likely that the concepts and techniques being explored by researchers in machine learning may illuminate certain aspects of biological learning.

Regarding the machines, it is roughly said that a machine learns whenever it changes in its structure or program or data, based its input or in response to external information. This change takes place in such a manner that its expected future performance improves. A change like addition of a record in a database, though makes a change in data structure, does not fall in the discipline of learning. As another example, the performance of a speech-recognition machine improves after hearing several samples of a person's speech—we feel convinced that the machine has learned. This process is obviously a learning process.

Machine learning usually refers to the changes in systems that perform tasks such as recognition, diagnosis, planning, robot control, prediction, and their combinations. These tasks are usually associated with artificial intelligence (AI). The "changes" mentioned above is in one of the two types: 1. enhancements in already performed task by a system, or 2. creating special ability in the system.

One obvious question related to machine learning is "Why should machines be built to learn, and why not to build them to perform the desired task in the first place?" There are several reasons why machine learning is important. The one we have already mentioned is that understanding the learning process will help us in building theories which will help us understand how the animals and humans learn. It is like when we learned optics, we were able to better understand human vision system. However, there are the following important reasons why we should study machine learning, and why there should be machines that learn:

(a) There are many tasks that cannot be defined well in advance, except by examples. Like many times we are able to specify input/output relations as pairs of data but a precise relationship between inputs and desired outputs cannot be specified. For example, the weight-to-height ratio of a male as input and health as output. We would like that based on these input–output sample examples, the machine be able to adjust its internal structure, so that for new inputs it produces correct outputs, and thus suitably constrain its input/output function to approximate the relationship implicit in the given examples.

(b) It is possible that among large sets of data, like—credit cards, medical records, weather data, astronomical data, etc., there are hidden important relationships and correlations. The machine learning methods can often be used to extract these relationships and patterns through the process of *data mining*, a field of machine learning.

(c) The machines produced by the human designers often do not work as desired in certain environments. In fact, when the machine was designed, certain characteristics of the working environment might not be completely known at design time, hence could not be incorporated in the design. This often happens in space explorations—like missions to Moon, Mars, Saturn, and even to Sun. Machine learning methods can improve/modify the performance of the system on the site in the existing machine design.

(d) The amount of knowledge available and required for certain tasks may be too large for explicit encoding by humans. But machines can be designed that learn this knowledge gradually with time, and might be able to capture it continuously on the fly as it needs.

(e) There is no need to redesign the machines if there are machines that can adapt to a changing environment. Such machines, once designed, can learn with situations, and functions for a long time—for example in long planetary missions.

With the continuous redesign of AI, the changes are taking place in the world knowledge as well as vocabulary.

The questions in the domain of AI cover a broad range of learning tasks, like: How to design an autonomous mobile robot that learns to navigate using its own

experience? How to mine the data of historical medical records to learn that a given patient X will respond best to what treatments? How to build search engines that automatically customize to the user's interests and other similar questions.

The following is a more precise definition of machine learning.

Definition 13.1 (*Machine Learning*) A machine learns with respect to a particular task T, performance metric P, and type of experience E, if it reliably improves its performance P at task T, following the experience E.

Depending on how we specify T, P, and E, the learning task might also be called by names such as *data mining*, *autonomous discovery*, and *database updating*, *programming by example*.

The field of machine learning focuses on how to make computers that program themselves through their experience, with provision of some initial structures. This is in contrast to the field of computer science that so far focused primarily on manually programming computers.

Other fields that are closely related to Machine Learning are Psychology and Neuroscience, which are concerned with the study of human and animal learning, and are collectively called *Cognitive Science*. The questions—How do animals learn, and how can computers be made to learn—most probably have highly intertwined answers. The insights into machine learning far outweigh due to contributions from the fields of statistics and computer science than due to studies in human learning behavior. The main reason behind this is the poor state of understanding about human learning processes. However, the relation between machine learning and human learning behavior is continuously becoming stronger, mainly due to the new machine learning algorithms, such as *temporal difference learning*. These algorithms are being suggested as the explanations for *neural signals* observed in learning in the animals.

The primarily goal of machine learning is to design computers that can adapt and learn from their experience without intervention from programmers. Following is an example: the present software engineering, data-intensive software construction and testing, is aimed to construct programs that are correct and robust. It is data-intensive software construction, i.e., given a set of input data to be processed, code functions that produce output (data), such that the code should correctly function even when input data changes, and even on the face of incorrect data, should reject them rather than producing wrong results. A machine can be designed to learn from the pattern of data and continuously improve its performance until it reaches the desired performance. The error (difference) between the expected performance and the actual performance can be fed back to act as a tuning to the software, that is, to make it learn. This latter is the automation of human-intensive task of software engineering that we call testing.

Machine learning methods for software engineering are appropriate whenever *hand engineering* of software is difficult and yet data is available to be analyzed by learning algorithms. Following are the situations where machine learning algorithms are expected to outperform humans:

(i) human expertise is lacking compared to the machine intelligence in genome programs—the genome programs are basically evolution process, which can be used in applications, like drug design;
(ii) the characteristics of the task changes frequently, like in airline seat sales strategies, where the price is decided by supply-versus-demand ratio;
(iii) humans have limited capability of introspection, like required in computer vision, speech recognition, and motor control;
(iv) a learning program can be customized for individual users, e.g., information filtering, user interfaces, language, etc.

However, there is yet one limitation of machine learning—the field of machine learning is inherently stochastic—hence its application may not be appropriate for tasks where completely error-free performance can be guaranteed, e.g., in life-support control in intensive care.

13.4 Learning Model

The process of learning requires that new knowledge structures be created due to some form of input stimuli. Then, this new knowledge is assimilated in the existing knowledge and is tested for its utility and efficiency. That is, it will be used in performing some task. Figure 13.1 shows a general learning model. The environment is part of the overall system for learning. The environment may be taken as a teacher or the one provided by nature which produces random stimuli, or a room with obstacles where a robot needs to navigate from one wall to another, or may be a classroom with a teacher and students who are taught using a blackboard and chalk. The communication language between a teacher and a learner may be the same in which a knowledge base was created or may be different.

The input to the learner component may be physical stimuli or some kind of description or is symbolic. The information by the learner component creates/modifies the knowledge structures in the knowledge base, which is used by the performance component to carry out some tasks, like solving some problems, game playing, etc. The response produced by the performance component is evalu-

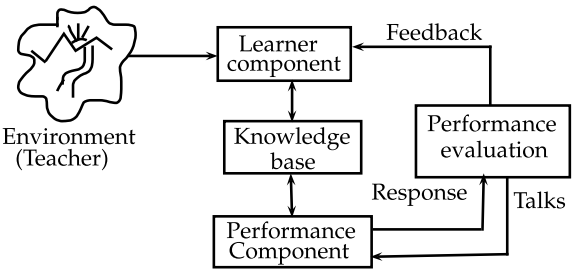


Fig. 13.1 A learning model

ated by the performance evaluator component, and a feedback generated by this is sent to the learner to decide if the performance is acceptable, based on which the learner updates its knowledge base. This cycle is repeated until the performance of the system is reached to the desired level.

In the sections, we present specific learning techniques in more detail. We present first the fundamental categories of learning; a learning may be basically classified as belonging to supervised or unsupervised learning.

13.5 Classes of Learning

When there is essentially a teacher present for learning to take place, it is *supervised learning*, and a learning achievable without a teacher is called unsupervised learning. Teacher, here, means not necessarily a physical teacher, but even nature is a teacher. For example, we put fingers in hot water and we get it burnt. So here, feedback is from hot water, which teaches us that if you again put the fingers in hot water any time in future, it would be burnt in similar way.

13.5.1 Supervised Learning

Supervised learning methods require an oracle to be made, such that it forecasts the classes (categories) to which the given examples would belong and then classifies the examples in those categories. These methods assume

(a) Existence of some teacher (environment),
(b) A fitness function to measure the fitness of an example for a class, and
(c) Some external method of classifying the training instances.

A supervised learning method uses machine learning techniques to induce a classifier from sense-annotated data sets. As an example of supervised learning for text classification, a classifier picks one word at a time, perform its word sense disambiguation (WSD), and then performs a classification task in order to assign the appropriate sense to each instance of the word.

A classifier typically learns with the help of a training set containing examples in which any given target word has been manually tagged with the sense form the sense inventory of some reference dictionary.

In general, a learning method is called *supervised* learning if the learning process requires some intervention from the user. Some approaches in supervised learning require the user to provide training examples. This is done as a pre-process to appropriately tag the examples occurring in the training set. Alternatively, it can be done as an online process where examples are dynamically tagged as needed during the learning process.

Propositional learning is a category of supervised learning in which examples of concepts are represented in terms of either *zero order logic* or *attribute-value logic*.

This representation has equivalent expressiveness in a strict mathematical sense. The rules are often learned from positive examples. The examples of concepts are represented as sets of attributes in the form of slots in knowledge frames, whose values are heads of syntactic phrases occurring within the training documents. The rules learned through these approaches are used in performing the tasks, like *Information Extraction (IE)* from parsed free text, *text classification*, *question-answering*, etc. The learning process identifies the syntactic phrases that contain the heads to fill these slots. Often, there is a partial match between these phrases and the exact slot filler, which is sufficient when the aim is to extract only an approximate of the relevant information. If an approximate match fails, there is necessity of post-processing.

Other class of supervised learning is *relational learning*, which is based on representing the examples of concepts in terms of *First-Order Predicate Logic* (FOPL)—as attributes and relations between textual elements. More details of propositional and relational learning are covered in sections that follow.

13.5.2 Unsupervised Learning

This learning eliminates the need of a teacher, and the learner is solely responsible to form his own concepts and evaluate these for learning. In fact, the unsupervised methods have to discover the concept classes to which the given examples belong, i.e., mapping the examples to concept classes.

For example, scientists are usually not blessed to have a teacher to help them pursue the research, instead they propose hypothesis to explain the observations made by them, and evaluate their hypotheses based on criteria like generality, simplicity, and elegance, and test these hypotheses through experiments designed by themselves.

Another example of unsupervised learning is *unsupervised WSD*, which is based on unlabeled corpora, and do not exploit any manually sense-tagged corpus to provide a sense choice for a word in the context.

The *discovery-based learning* is also a category of unsupervised learning.

13.6 Inductive Learning

The ability of human beings to master meeting the complex demands is mainly based on the human ability to exploit previous experiences, like if we have previous experience of walking through the Thar desert unaided for five miles from 10 years ago, we will not hesitate to repeat the next adventure now even for still more distance. Based on our previous experiences, we are able to predict the characteristics or relations of man-made or natural objects, we can reason about possible outcomes of actions, and we can apply the previous successful routines and strategies to new tasks and problems, as in the case of journey through a desert example. It is understood that the basic process to expand the knowledge is—first construct a hypothesis such that

we can transfer the knowledge from previous experience/patterns to new situations. This approach of learning is called *inductive inference* [6].

Inductive inference is a mechanism of generalization over the observed regularities in the examples. Every learning algorithm must make some a priori assumptions to allow for the generalization, called *inductive bias*, which also provides a rational basis to allow for the transfer of learned hypothesis to the new situation. For this transfer, the *language bias* or *restrictions* characterizes the language in which the induced hypothesis is represented. A bias for search or preference characterizes the method of selecting the proper hypothesis. Every machine learning algorithm, be it symbolic, or statistical, or neural, makes such a priori assumptions. This basic learning approach is also valid for humans, whose learning is taken as a base mechanism, that takes place at all levels, from concept acquisition to learning high-level schemata.

The acquired concepts and strategic rules need to be communicated, so that they are available as declarative structures of knowledge. Hence, inductive learning programming should be *symbolic programming*.

In the abstract, we can view the output of inductive learning as a set of *Production rules* of the form,

In *situation* **do** *action*

where *action* may be something overt, or an internal action, or even an inference. We use the *concept* sometimes for the right-hand side (i.e., in place of action), and the word *concept definition* or *pattern* for the left-hand side (i.e., in place of situation), to formalize the inductive learning from one or more instances in which $action_i$ was an appropriate action response to $situation_i$. Based on these instances, we infer that the general version $action_{gen}$ is the appropriate type of action in response to the general situation type $situation_{gen}$.

$situation_1$	$action_1$
$situation_2$	$action_2$
…	…
$situation_{gen}$	$action_{gen}$.

The above example shows that inductive learning is the generalization from a set of examples; and with little pondering we note that this is one of the most fundamental learning types among humans. Learning of concepts is a typical inductive learning approach—given the examples of some concepts, such as “cat”—it will allow us to correctly recognize future instances of this concept; similarly, learning the concept that something is a “good stock investment”, it will allow us to correctly perform instances of this type in the future, i.e., if an investment has been done intelligently, and has been found rewarding, there are good chances that, in future also we may adopt a similar strategy. Such instances are very common in life, hence, the most common application of inductive learning is in *concept learning*.

Definition 13.2 (*Concept Learning*) Concept learning is, given the number of positive and negative examples of some concepts with some background knowledge, to find a general description or hypothesis of the concepts describing all the positive examples only, and ignoring all the negative examples. □

Definition 13.3 (*Deductive System*) A logical system whose conclusions *logically follow* from a set of input facts, and the system is *sound*, then the system is called a deductive system. □

To contrast the deduction with an induction in logic-based systems, consider that we have the *training set* with the following formulas:

$$Ball(Obj_1), Round(Obj_1), Ball(Obj_2), Round(Obj_2), \\ Ball(Obj_3), Round(Obj_3). \tag{13.1}$$

Using these formulas if a learning system draws the conclusion $(\forall x)[Ball(x) \rightarrow Round(x)]$ then that system is an *inductive* learner. Note that this conclusion does not logically follow from the facts. Hence, conclusion is useful when there are no rules of the form $[Ball(x), Ball(x) \rightarrow Round(x)]$.

Unlike deduction, which is based on general axioms, induction is based on specific facts (examples). Induction has two goals: 1. formulate acceptable general assertions, that explain the given facts, and 2. to predict unseen facts. Thus, an inductive inference is aimed to provide a complete and correct description about a given phenomenon using specific/partial observations about it. Such inductively obtained description is true for at least already seen examples, but there is no guarantee of its correctness about the new examples.

Inductive inference methods can be used in two types of applications: 1. as an interactive tool for acquiring knowledge using examples, or 2. the method is used as part of some learning system. In the first case, a user has strong control through the examples provided by him/her, which decides the type of knowledge going to be acquired. Whereas in the second application, the inductive procedures (methods) are activated when some system component wants to learn using positive/negative examples. These examples constitute the feedback from which the system can achieve the completion of the current task.

In any relevant domain, the *background knowledge* is a deciding factor for assumptions and constraints imposed on examples and descriptions generated. The background knowledge is represented using either *declarative format* or *procedural format*. Concept learning requires searching through a large space of hypotheses (descriptions), each as a sequence of instructions for executing a specific task, with a goal to finding a hypothesis that best explains the examples. The instruction in the hypothesis are implicitly defined using the hypothesis representation language.

Inductive Programming

Inductive learning (also called concept learning) can be further classified according to the following perspectives, depending on whether it is based on

- Supervised learning,
- Unsupervised learning,
- Concept hierarchies, or
- Single/multi-label learning.

Inductive programming is concerned with *learning computer programs*, designed to work with incomplete specifications—samples of desired input/output behavior, along with some constraints. These programs are called *induced programs*, and are represented in the declarative form using functional programming or logic programming languages. To be used in machine learning, inductive programming creates program hypotheses in the form of generalized recursive programs. In contrast to the *classification-based learning*, inductive program hypotheses are supposed to cover all the given examples correctly. This is due to the fact that for a program it is expected that the desired input/output relation holds for legal inputs.

The inductive programming is used with two general approaches: 1. *generate-and-test*, and 2. *analytical* approach. The first enumerates syntactically correct programs and tests each against the given examples, with search guided through some search strategy. It is obvious that a cognitive system does not learn rules by the generate-and-test approach.

13.6.1 Argument-Based Learning

A challenge with machine learning is to deal with large spaces of possible hypotheses, and the search associated with that space. This problem is better addressed using expert domain knowledge to constrain the search. Further, the problem is somewhat simplified, as machine learning allows to use to some extent the general domain knowledge, which holds true for the entire domain. For constraining the search, it allows the use of "local" expert's knowledge relating to specific situations, which is valid for chosen learning examples. For this, a domain expert provides some learning examples, these together with other examples are input to the learning algorithms. These examples are explained in the form of arguments: *for* and *against*, and called as *argumented* examples; hence the learning using such examples is called *Argument-Based Learning* (ABL) [5].

Definition 13.4 (*Learning from examples*) Having given some examples (arguments), find a theory that is consistent with them. □

To explain the idea of argument-based learning, we consider a simple learning problem called "Learning about approval of credit limit." Each example consists of information, as shown in Table 13.1, which comprises, apart from other things, the decision of the manager (Yes/ No), indicting whether the credit limit is approved or not.

Table 13.1 Examples for credit limit approval

Customer's name	Repayments' regularity	Rich	Height	Credit approved
Mrs. Red	No	Yes	Tall	Yes
Mr. Green	No	No	Short	No
Miss Blue	Yes	No	Medium	No

An algorithm after learning from these examples will *induce* the following rule:

$$IF\ Height = Tall\ THEN\ Credit\ approved = Yes. \tag{13.2}$$

Now, we want to see how this rule will modify when it faces the arguments.

Definition 13.5 (*Arguments-based learning*) Given the examples and supporting arguments, find a theory that explains the examples using arguments. □

To understand what it means to "explain the examples using given arguments," see the examples in Table 13.1. Let us assume that an expert gave an *argument*: "Mrs. Red was allowed credit because she is rich." Next, we refer to the rule (13.2), which states that all tall people were approved credit. Note that this rule correctly classifies Mrs. Red (as tall), but does not include in the classification the argument (of rich) for Mrs. Red. Neither the rule mentions Mrs. Red's property, namely she is rich. Hence, we say that this rule (13.2) does not "AB-cover" Mrs. Red. Therefore, an ABL algorithm induces other rule,

$$IF\ Rich = yes\ THEN\ Credit\ approved = Yes \tag{13.3}$$

which explains Mrs. Red's example, using the given argument: credit was approved because *Rich* = *Yes*; now, this rule AB-covers Mrs. Red. Based on this discussion we note that the use of arguments in learning contributes to the following advantages:

1. They impose constraints over the space of possible hypotheses, thus reducing both time and space complexity of each search, enabling faster and more efficient induction of theories;
2. An induced theory is more meaningful for an expert system since it is consistent with given arguments as well as examples.

In fact, there may be many possible hypotheses from the perspective of a machine learning method that may explain the given examples sufficiently well. But these hypotheses should be simple enough to be understandable to expert (systems). Using arguments should lead to hypotheses that explain given examples in similar terms to that used by an expert, and should correspond to full justifications.

The argument-based examples is a resource for domain expert, which is added as partial domain knowledge in a learning system in advance. This prior knowledge

in the form of arguments as individual examples is simple to explain, concrete, and specific, in contrast to the general theories, which do not possess these properties. Since an argument is for a specific example, it is not required to be true for the entire domain, and suffices that it is true in the context of the given argumented example. As a test, consider a possible expert's argument to reject credit approval to Mr. Green: "Mr. Green did not get credit approval because he does not repay regularly" (see Table 13.1). This is a reasonable argument, however it does not hold for the entire domain, as Mrs. Red was approved credit, although she did not pay regularly. This generalization of the context of an argumented example is carried out by an argument-based learning algorithm only to the extent that it preserves the consistency with other examples.

13.6.2 Mutual Online Concept Learning

A mutual *online concept learning* system is an agent-based system (see more details in Chap. 16) with communication channel for exchange of information among the agents (see Fig. 13.2). It is assumed that the communication channel is an ideal one without any noise. A one-agent framework includes the basic elements like *concept*, *instance producing mechanism*, *instances*, *instance interpreting mechanism*, and *concept adaption mechanism* (or *online concept learning*), with communication channel shared among agents.

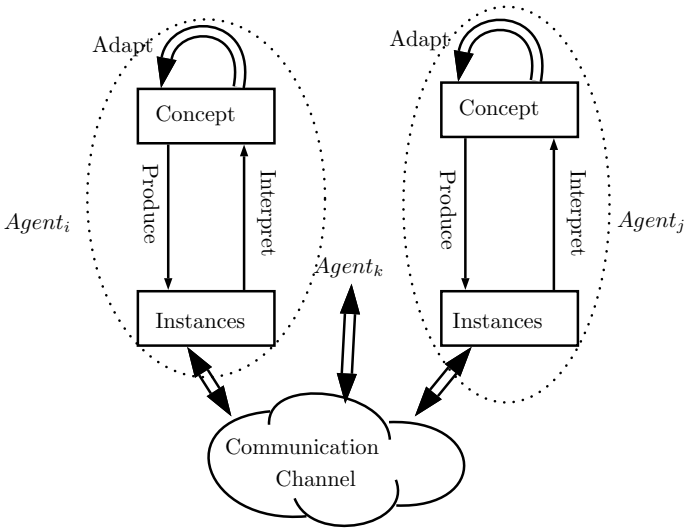


Fig. 13.2 Mutual online concept learning framework

In simple terms, a *concept* is a mapping from inputs to outputs. When taken as a function, it is a deterministic function, i.e., if the input is repeated, the output will follow the same sequence. When we think of it as a mapping with some probability distribution associated, then the output is also probabilistic in nature. As a deterministic function f, it can be formalized as $f : X \rightarrow Y$. Considering it a *Boolean concept*, the domain is $X = \{0, 1\}^n$, and its range is $Y = \{0, 1\}$. For implementing a concept, we can choose from a variety of representations, such as *propositional logic*, *first-order logic*, or *neural networks*. When a concept is viewed as a function, an *instance* (x) is just a specific case in the function's domain X, i.e., $x \in X$ [9].

When it is required for an agent to express a concept to the outside world, it uses an *inverse function* to generate an instance using a concept, shown as *produce* operation in Fig. 13.2. There be many instances that can be used to express the same concept if the mapping from input to output is not one-to-one.

An agent may receive an instance from another agent through a communication channel, and may *adapt* its concept like probability, network connection weights, etc. This adaptation may become helpful from the point of view that the receiving agent may perform its task better next time due to updated concept or knowledge, in terms of getting more benefits/payoff. Let us assume that, for an agent i, an associated concept is $\mathscr{C}_i^t$ at time t, and for agent j it is $\mathscr{C}_j^t$. Let the instance $instance(\mathscr{C}_j^t)$ be generated by j at time t, and this instance be received by agent i. At the next time $t + 1$, the agent i updates its concept to $\mathscr{C}_i^{t+1}$ by adapting from its previous concept (at time t) together with the instance of concept of agent j at time t, i.e.,

$$\mathscr{C}_i^{t+1} = adapt(\mathscr{C}_i^t, instance(\mathscr{C}_j^t)), \quad j \neq i. \tag{13.4}$$

The representation in the learning processes, as well as the knowledge representation, is an important consideration for system performance. For real values-based instances, the space in concept learning can be of n-dimensions: $\mathbb{R}^n$, and $\{0, 1\}^n$ for Boolean values. The concepts learned from instances are based on linear threshold values, such that there is a hyperplane in $\mathbb{R}^n$ that separates the points on which the function is 1 from the points on which it is 0.

Figure 13.2 indicates the logic of concept based on a broad sense, but does not specify the time order of agent interactions between the agents. This interaction can be performed in any of the two modes: *serial* or *concurrent*. In the first case, at each time step only one agent is generating instances or updating concepts, and other agents should be idle. In the second (concurrent) case, two or more agents may simultaneously generate the instances and update the concepts. There is no difference between these agents while working in the concurrent mode, however when in the serial mode, the two agents are not strictly equivalent since there must be one agent that starts the mutual process first, and so makes an initial impact on the direction of formation of a concept.

13.6.3 Single-Agent Online Concept Learning

For the sake of simplicity, we consider a single-agent for online concept learning from a teacher/environment, that is static. In contrast to a single-agent system, the environment is continuously changing (not fixed) in a *multi-agent* online concept learning system, because the agents act on the environment shared by themselves.

The learning task to be performed is as follows: obtain the target *concept* or *function* $f^* : \{0, 1\}^n \rightarrow \{1, -1\}$, that maps each *instance*[1] to the correct *label* or *class* in response to the learning instances and feedback from the teacher [9].

A sequence of trials is needed for the online learning task to take place. In a trial, the following order is followed by the events:

1. an instance is received by a learner from a teacher;
2. the instance is labeled as 1 or -1 by the teacher;
3. the teacher tells the learner whether the label is correct or not;
4. this feedback is used by the learner to update its concept.

In the above steps, each new trial begins when the previous trial ends.
A learning algorithm called the *Perceptron algorithm* takes real-valued inputs in the form of a vector, and outputs as 1 if the result of computation is greater than a threshold (θ), otherwise -1 is the output [3].

To be precise, if an input instance is $\overrightarrow{x} = (x_1, \ldots, x_n)$, the output computed by the perceptron is $f(\overrightarrow{x})$:

$$f(\overrightarrow{x}) = \begin{cases} +1 & \text{if } \sum_{i=1}^{n} w_i x_i > \theta \\ -1 & \text{otherwise.} \end{cases} \tag{13.5}$$

In the above, $\overrightarrow{w} = (w_1, \ldots, w_n) \in \mathbb{R}^n$ is called the current weight vector of the perceptron. For the sake of convenience we take threshold θ as 0, and instead add a constant $x_0 = 1$ with a weight w_0. For the sake of compactness the perceptron function in Eq. (13.5) is written as

$$f(\overrightarrow{x}) = sgn(\overrightarrow{w} . \overrightarrow{x}).$$

Let $\overrightarrow{w} . \overrightarrow{x} = y$. Now, we can rewrite Eq. (13.5) as

$$sgn(y) = \begin{cases} +1 & \text{if } y > 0 \\ -1 & \text{otherwise.} \end{cases}$$

It is to be noted that each weight vector $\overrightarrow{w}$ defines a *perceptron function* or concept, hence updating a weight vector is equivalent to updating a concept.

Learning of a concept requires choosing values for the weight vector $\overrightarrow{w} = (w_1, \ldots, w_n)$. At the start of the algorithm, the initial weights are chosen as

[1] Instance and example are the same here, which are input to the function.

$\overrightarrow{w} = (0, \ldots, 0)$. On receiving an input instance $\overrightarrow{x} = (x_1, \ldots, x_n)$, the learning algorithm predicts the label of $\overrightarrow{x}$ to be $f(\overrightarrow{x})$. For the sake of compactness we represent $f(\overrightarrow{x})$ by y.

If it is found that the predicted label is correct, then there are no changes in the weight vector. But if the predicted label is wrong, the weight vector is updated using the perceptron learning rule given below. Here, λ is the learning rate.

$$\overrightarrow{w} = \overrightarrow{w} + \lambda y.\overrightarrow{x}. \tag{13.6}$$

Multiple Agent Concept Learning

It is an extension of the single-agent concept learning. In the multi-agent system, every agent behaves both as a teacher and a learner, with a dynamic environment. This is due to the reason that the other agents being part of the learner environment also change their concepts as they learn from each other. Since there is no teacher or environment existing initially, there is no fixed concept to be learned. Accordingly, at the begin, the concept to be learned is dynamically formed due to the interaction among the agents.

13.6.4 Propositional and Relational Learning

The learning methods making use of "propositional calculus" are called *attribute-value/propositional* learners. They use objects with a fixed set of attributes, selected from a predefined set of values. The learning based on first-order relational descriptions—the *relational learning*—induces descriptions of relations and uses the objects described in terms of their components and relations among the components. As an example, if an object X has attributes a, b, c then this can be represented as a tree with X as the root and a, b, c as leaves, showing X having relation with each leaf. This relation is the *propositional* relation. The background knowledge of relations is formed using the language of examples and concept descriptions, which is typically a first-order logic. Particularly, the learners that induce hypotheses in the form of logic programs, i.e., *Horn Clauses*, are called inductive logic programming systems.

The inductive learning has application in automatic construction of knowledge base in expert systems, which are an alternative to classic knowledge acquisition systems. The inductive techniques are also used for refinement of existing knowledge base, since they facilitate the detection of inconsistencies, redundancies, lack of knowledge, and also are helpful in the simplification of the rules provided by the domain expert. Due to their capability to detect patterns present in the input examples, the inductive methods are also used in the domains of biology, psychology, medicine, and genetics.

13.6.5 Learning Through Decision Trees

This is a combination of *induction-based* learning as well as *supervised* learning. In decision trees, every record is associated with a unique leaf node of the decision tree. The criteria for choosing a unique leaf node is to start from the root node, and repeatedly choose a child node based on the splitting criteria, which evaluates a condition on the input record at the current node.

The decision tree is a predictive model for learning, which represents the classification rules using a tree structure that recursively partitions the training data set into two subsets. Each internal node of the tree (all, except the leaf nodes) performs a *test* on the feature value, which results in two outcomes, one of those is selected, and ultimately reaches the terminal node where the prediction is performed.

An algorithm for decision-tree construction comprises two states: 1. construction of decision tree, and 2. pruning the tree. In the first step, the decision tree grows top-down in the greedy way. The data is examined by selecting the split condition at each node, starting with the root node. The data is then partitioned and procedure applied recursively. In state 2, the tree already constructed is pruned to control its size. Sophisticated pruning methods select the tree in such a way that it minimizes the prediction errors.

The decision trees tests one feature at each internal node, with one branch for each feature value, and the class predictions is carried out at the leaves.

Algorithm 13.1 is an algorithm for *decision-tree learning* (*relational learning*). It recursively calls the function $DT(T, E_{curr})$ representing knowledge that has been learned using decision trees. The variable T represents the decision tree constructed so far, and E_{curr} is the set of input examples. If all input examples in E_{curr} are in the same class C (i.e., representing the same concept), the decision tree will contain only one leaf corresponding to C. For example, the proposition $E_1 =$ "A smiling person is a friend," can be represented as a single leaf tree as shown in Fig. 13.3, with $C = friendly$. If the examples in the input belong to n number $C_1, \ldots, C_n$, an attribute is chosen by the algorithm and based on this attribute, E is divided into sets $E_1, \ldots, E_n$, which are disjoint. Each partition E_i consists of the examples with the same value in the attribute selected. Each set E_i is applied to the algorithm until the sets which have been obtained are left with elements belonging to the unique class. Finally, terminal nodes of the tree are the resultant disjoint classes $C_1, \ldots, C_n$ into which the examples are classified.

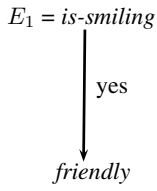


Fig. 13.3 A decision tree with single proposition

Algorithm 13.1 Algorithm for Decision tree

1: Initialize: $E_{curr} = E, T = nil$
2: *function* $DT(T, E_{curr})$
3: **if** all examples in $E_{curr} \in Class\ C_s$ **then**
4: generate a leaf labeled C_s
5: **else**
6: ; {generate new node}
7: select the most informative attribute A with values $\{v_1, \ldots, v_n\}$
8: Split E_{curr} into subsets $E_1, \ldots, E_n$ according to values $v_1, \ldots, v_n$
9: **for** $i = 1$ to n **do**
10: $DT(T_i, E_i)$
11: **end for**
12: **end if**
13: Output: Decision-tree T with the root A and subtrees $T_1, \ldots, T_n$
14: End

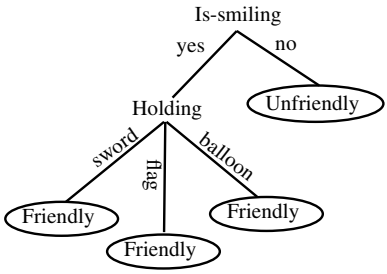


Fig. 13.4 A decision tree for root domain

Any unseen example is classified as follows: start from the root node, attributes are tested at internal nodes, repeat the process and proceed to a terminal node. It is assumed that in an example that belong to a different class, at least one of the attributes has a different value.

In a decision tree, each internal node is labeled by an attribute and links. The attribute is selected from a set of possible values. In the example of Fig. 13.4, one of the attributes is "is smiling" and the set of its values is $\{yes, no\}$. Another attribute is "holding" with the set of values as $\{sword, flag, balloon\}$. The decision-tree construction has two important objectives: one is selection of the most informative attribute having the closest sense to the context in the example, and the second is to minimize the number of tests. Note that the number of tests is height (or depth) of the tree—paths length from the root to terminal node, and it directly defines the difficulty level of the problem [4].

Example 13.1 Decision tree for financing a client.

Consider financing an individual by a financing company. The finance may be based on the good credit history or other factors.

Figure 13.5 shows the details, which causes learning. The learning helps in classifying the individual loan applications 1 . . . 16 into various categories shown in this figure. □

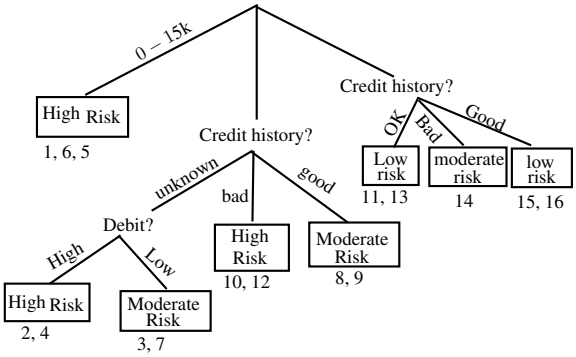


Fig. 13.5 Learning through decision tree

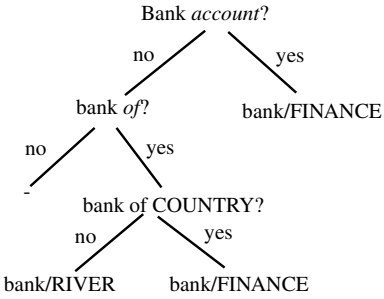


Fig. 13.6 Disambiguation-tree for *bank*

Example 13.2 Decision tree for Word Sense Disambiguation (WSD).

Consider an example of a decision tree for Word Sense Disambiguation (WSD), i.e., to find out the correct sense of a given word in a phrase. We are interested in the word "bank" in a sentence: "we sat on the bank of Ganges." The process of disambiguation is demonstrated through a decision tree shown in Fig. 13.6.

The figure shows that the tree is traversed such that for every internal node it asks a question, having the answer. The answer to each branching is provided from the hidden knowledge in the sentence, as well as through commonsense knowledge. We note that it follows the path of *no-yes-no* from the root node until the terminal node, and the choice of sense *bank/RIVER* is made. In the tree, the node "bank of COUNTRY" stands for, for example, "bank of India", "Bank of America", etc. The terminal nodes for which no choice can be made based on specific feature values are indicated with empty values by "–". □

Construction of the decision tree requires extensive access to the training database. Sometimes the training database is so huge that it does not fit the memory, in that case some efficient data access method is needed to achieve the scalability.

During the construction of a decision tree it requires construction of statistics in memory for each node, and is carried out in a single scan. The split at any node is based on some database partition that satisfies the split criteria.

The statistics for splits is updated in the memory at each node in a path starting from root node up to terminal node. The partitioning at each node should satisfy the splitting criteria corresponding to that node, and sufficient statistical data in the form of corpus is necessary for the construction of any learning decision tree.

13.7 Discovery-Based Learning

The discovery-based learning requires the maximum learning efforts among all types of learning. One of the earliest discovery-based programs is *Automated Mathematician* (AM), which derives a number of concepts in mathematics. It is based on the concepts of set theory. The AM discovers the natural numbers by modifying a concept called "bag", which is a generalization of the concept of a set in *set theory*. The concept "bag"permits multiple occurrences of set elements, e.g., $\{a, a, a, b, c, c, d, d\}$. Hence, the natural number 4 is represented as $\{1, 1, 1, 1\}$. An addition of the natural numbers is shown as the union of sets: $\{1, 1, 1, 1\} \cup \{1, 1\} = 6$. The operation of multiplication is the series of additions: $3 * 4 = \{1, 1, 1\} \cup \{1, 1, 1\} \cup \{1, 1, 1\} \cup \{1, 1, 1\} = 12$. The operation of subtraction is the difference of the sets: $4 - 3 = \{1, 1, 1, 1\} - \{1, 1, 1\} = \{1\}$. The division is obtained by repeated the set differences. A prime number is an integer having only two dividers, the unity and the number itself.

One approach for discovery-based learning is learning by concept hierarchies, as explained in the following.

Learning by Discovering the Concept Hierarchies

One of the most general approaches to solving a complex problem is to decompose the problem into smaller, less complex, and manageable subproblems. This principle is the foundation for *structured induction* in machine learning, where a concept hierarchy is defined and rules are learnt for each of the (sub)concepts. In this process, the concept hierarchy can be manually constructed or it can be automated, instead of learning a single complex classification rule from examples.

Consider that given five Boolean attributes $x_1, \ldots, x_5$, it is required to learn Boolean function y, expressed as

$$y = (x_4 \wedge x_5) \oplus (x_3 \wedge (x_1 \oplus x_2)). \tag{13.7}$$

For five attributes there is space of 32 points. Consider that out of these, there are total 24 randomly selected points (i.e., 75% of the total) are to be selected as *examples* for learning. These are called as attribute-value vectors, and are useful for hiding the actual structure of the original expression. In Eq. 13.7 it is possible to justify that the best possible structure of sub-concepts will be as given in Fig. 13.7, with definition of functions as $f_1 = \oplus, f_2 = \wedge, f_3 = \wedge$, and $f_4 = \oplus$.

An approach for learning by concept hierarchies is based on function decomposition. Consider decomposing a function $y = f(X)$ into $y = g(A, h(B))$. The orig-

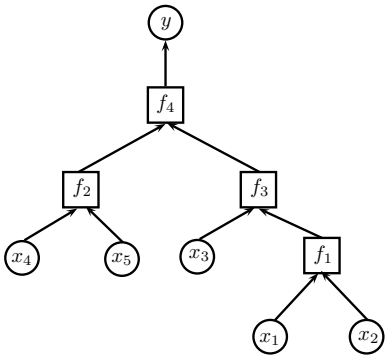


Fig. 13.7 Hierarchy of concepts as a tree

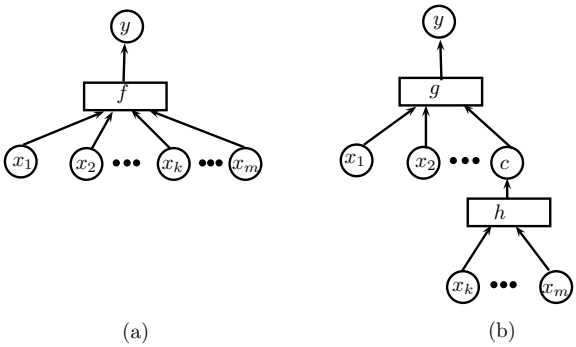


Fig. 13.8 **a** Original decision tree, **b** Decision tree after decomposing

inal function is shown in Fig. 13.8a and b shows the decomposed function. In the original function, $X = \{x_1, x_2, ..., x_k, ..., x_m\}$, and after decomposition, $A = \{x_1, x_2\}$, $B = \{x_k, ..., x_m\}$ are sets of input attributes, and y is variable, which represent the class. Also, input attributes $X = A \cup B$. The f, g, h are functions, which are partially specified as examples, i.e., by sets of attribute-value vectors with assigned classes. The functions g and h are determined in the decomposition process, such that their joint complexity is lower than the complexity of F. The attribute c is called intermediate concept and defined as $c = h(B)$, which is discovered in the decomposition process.

The function decomposition can be applied recursively on the functions g and h, which should result in a hierarchy of concepts, as shown in Fig. 13.8b. For each concept in the hierarchy, like $h(B)$, there is a corresponding function (h here), that decides the dependency of the concept, e.g., concept c, on its immediate descendants in the hierarchy [11].

13.8 Reinforcement Learning

Our learning is through feedback of our actions in the real world. Human beings always learn by interactions with the environment/teacher, which are *cause* and *effect* relations. The environment or the world around us is the teacher, but its lessons are often difficult to detect or grasp by the mind or analyze, hence hard to learn. The best example is learning by a dog or a young child, where good actions are rewarded and bad actions are discouraged. It is not defined as a more general example, but the actions with the world.

The programs that improve their performance at some task by rewards and punishments from the environment are examples of *Reinforcement Learning* (RL). For many automatic learning procedures for real-world tasks, like job-shop scheduling and elevator scheduling, RL has been found to be very successful. The total *discounted* reward the learner receives is optimized in the reinforcement learning, i.e., a reward that is received after one time step is considered equivalent to a fraction of the same reward received immediately before [1, 8].

There are four components of RL:

1. a policy,
2. a reward function,
3. a value mapping, and
4. a model of environment.

Learning agent's choices and methods of action are defined under a given policy. The policy is represented using a set of production rules. Sometimes the policy could also be probabilistic in nature. The reward function is a relationship between the state and goal, which maps each state into a reward measure, and indicates the need of that action to achieve the goal.

A reinforcement-based learning differs from supervised learning in the sense that a reinforcement system may not have a teacher to respond to each action, instead the learner itself creates a policy for interpreting the feedback. For example, playing in a particular style leads to winning a chess game for a player is reinforcement learning. The RL is a commonly used framework for problems requiring a number of agents, but their presence does not contribute to much of complexity as RL needs only limited feedback for learning. With the objective to maximize the expected reward, RL algorithms attempt to learn policies that help in taking optimal sequential decisions. However, this learning may be intractably slow as the methods scale up to more real-world problems with large state spaces.

A commonly used approach to solving RL problems makes use of a function, called *value function approximation*, using which an agent tries to learn a value for each state. This value is a long-term expected reward from entering a particular state, and such values are used for deriving a policy.

The state space for real-world problems contains infinitely a large number of possible states features. Hence, the designer of any such task must pick up only the most relevant features. Consider the task: Travel to Mumbai for work, and the

weather report for New Delhi is not likely to be relevant. From these we should construct a feature set. In most real-world situations, the feature set for a problem is broken down into a number of subsets, such that each subset can learn a specific concept of the domain. In this case, some concepts/feature-set subsets may be more important than the others.

Example 13.3 Formulate a task of navigation to be carried out by an agent, with a goal to investigate the best plan to go from point A to point B, and may choose a path and transport method of walking, driving, taxi, bus, train, or air.

The feature set of the agent's state space may include

- Positions of A and B,
- Raining (yes/no),
- Type of shoes of agent,
- Agent is with umbrella (Yes/no),
- Current time, and
- Day of week.

Using positions as features, the agent can learn the concept of position and basic path planning. The features raining, shoes, and umbrella are useful in learning as to how the weather governs the policy, and the features of time, and day in a week may be useful in learning to handle traffic, for example. A conventional approach to solving this problem through RL is to learn in a 6D space (positions, raining, shoes, umbrella, time, weekday) when all the features are taken into account. □

It is important to note that, the order in which features have been selected plays a significant role, in terms of difficulty level of the problem, and robustness of its solution.

13.8.1 Some Functions in Reinforcement Learning

In general, in a RL system, an agent recognizes itself in some state $p \in S$, then takes some action $a \in A$, and then recognizes itself in a new state q. The new state q in which the agent enters from its previous state p is decided by the agent's *transition function*:

$$T(S \times A) \rightarrow S. \tag{13.8}$$

In addition to the state change, the agent receives a reward r for arriving to state q, based on the *reward function*:

$$R(S \times A) \rightarrow \mathbb{R}. \tag{13.9}$$

A function called *value function* $V^{\pi}(p)$ is based on the average sum-of-rewards received when an agent starts in state p, and enters state q following a policy π. In this condition, we express the relation between the value function and *optimal policy* $V^{*}(p)$ as

$$\forall \pi, p : V^{*}(p) \geq V^{\pi}(p). \tag{13.10}$$

The RL is found successful in many domains, which includes many real-world domains, where it is used to optimize discounted total rewards. However, the most natural criteria in many other domains are based on optimizing the average reward per time step.

13.8.2 Supervised Versus Reinforcement Learning

The learning tasks are generally divided into two classes:

- *One-shot decisions* are tasks like classification and prediction, and
- *Sequential decision tasks* are like control, optimization, and planning.

The one-shot tasks are decision-based tasks that are usually formulated as *supervised learning* tasks. The learning algorithm in these tasks is provided with a set of input–output pairs, which act as relations between input and output. The input is the description of information used for making the decision, while the output is the description of the correct decision. For example, in handwriting recognition, the input is an image of the handwritten text of a digit/letter and the output is the identity of the digit/letter. Once such an algorithm is trained, it can recognize continuous handwritten text comprising sequence of letters and digit combinations, with good accuracy depending on how well the learning algorithm has been trained.

The sequential decision tasks are often formulated as *reinforcement learning* tasks. Learning algorithm in such tasks is part of an agent that interacts with the external environment. At each step, the agent observes the current state of the environment, then selects and executes some action, and on execution of this action the environment changes its state. Due to the state–action combination, the environment provides some feedback in the form of immediate reward to the agent, and the process goes on.

The reinforcement learning process maximizes the long-term rewards received by the agent. For example, when this learning is used in a setting in manufacturing, the current state is equal to the list of orders to be fulfilled along with the current status of manufacturing facility. The actions may change the settings of various production machines to maximize the overall profit. The latter can be achieved, for example by reducing energy consumption by machines as well as the idle times of machines. The immediate rewards include, saving in cost of each action, and revenue received when the order is completed.

13.9 Learning and Reasoning by Analogy

For analogical reasoning, an example in the form of a known solution is sufficient for learning a new solution. We use analogy when we learn about electric currents (e.g., Kirchhoff's current law, $I_3 = I_1 + I_2$ (see Fig. 13.9) by thinking about water pipes flow rates $Q_3 = Q_1 + Q_2$. While learning the subjects, like medicine, law, and economics, we also make use of analogy. Other examples are learning to play a card game of bridge from the knowledge of hearts, or programming a new algorithm using previously learned programming examples and concepts.

Some specific competence is necessary in using analogies to do certain kinds of learning and reasoning. The learning takes place when a constraint description is generated in one domain using analogy, having given a description with constraints in another domain. For example, we can better learn the Kirchhoff's current law using the knowledge about water flows in pipes, as illustrated in Fig. 13.9. Learning takes place when an analogy is used to answer a question about one situation, having given another situation that has preceded the current situation [7].

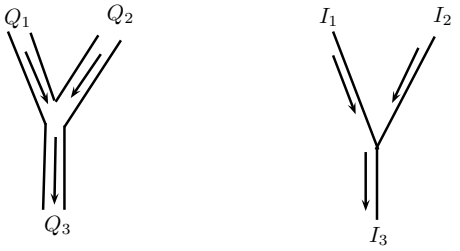
Analogy plays an important role in our reasoning process. We frequently explain or justify one phenomenon with another, where a previous experience often serves as framework or pattern for a new *analogous* experiences. A familiar experience is often in dealing with new experiences. Since so many of our acts are near repetitions of our previous acts, analogical-based learning is predominant in our living style. An implemented system for analogical learning has ingredients [10], discussed below.

Representation of Extensible Relations

The situations for analogical learning are represented based on relations between pairs of parts. In addition, more descriptions can be attached to the relations when needed.

Importance-Dominated Matching

Best possible match is found based on what is important in the situations, and then the similarity is measured between two situations. The importance is computed based on various kinds of constraints, and cause is a common importance-determining constraint.



From known flow rates:
$Q_3 = Q_1 + Q_2$

Infer Kirchoffs current Law:
$I_3 = ??$

Fig. 13.9 Analogical problem-solving

Analogy-Driven Constraint Learning

A constraint such as the one based on Kirchhoff's law is learned as a result of mapping parts of a situation belonging to a well-understood domain into the parts of another situation in a poorly understood domain.

Analogy-Driven Reasoning

Many times we are interested to find out if a particular relation holds. The *causes* found in a remembered situation can supply suggestive precedents.

Analogical reasoning shall best work subject to the following restrictions:

- *Symbolic sufficiency*. A situation to a previous instance can be represented using certain classes, properties, actions, etc. However, these collections should be simple, so that matching can be performed efficiently.
- *Description-based similarity*. A situation can be said to be similar to a previous one, if it can be mapped.
- *Constraint-based similarity*. Two situations are said to be similar if they are governed by the same constraint.

A program written for implementing reasoning by analogy should function based on identifiable principles, and must do the reasoning and learning by analogy, as in the following situations:

1. A class teacher explains to a student that a voltage drop across a resistance is calculated similar to the way we find the pressure drop across a section of a pipe having known the water flow rate and friction in it. With this analogy, a student can find the voltage drop across a resistance without the knowledge of Ohm's law.
2. A class teacher tells about two different resistances r_1, r_2, their voltage drops v_1, v_1, and currents i_1, i_2, respectively, and the student will find out the Ohm's law.
3. A class teacher suggests to generalize the principle using the flow of water in a pipe and the flow of current in a resistance, and the student concludes that there is a linear relation between the forces (potential/pressure) and flows (current/water flow rate).

From the above examples of analogical reasoning, we understand that a practice with some specific situations in one domain enables the invention of a law. Alternatively, once several examples of one law are known, the comparison results in the generalization of the law, like in the second example above. Analogies are likenesses or similarities between things that are otherwise different. The participants or things in analogies are unlimited: they may be concepts, objects, problems, solutions, plans, situations, episodes, and so on.

Analogies play a dominant role in human reasoning and learning processes—previously remembered experiences are transformed and extended to fit into new situations; old experiences may provide the explanations or scenarios that tend to

match with the new situations in some aspects and, therefore, may offer the promise of suggesting a solution to the new situation.

The difference between *inductive* learning and *analogical* learning is that, induction or similarity-based learning is based on the observation of a number of training examples, but analogical and explanation-based learning requires only one example in the form of a known solution or a past experience, as a sufficient criteria for learning to take place.

Analogical Reasoning

Consider the statement: "I have a Pomeranian dog, who is friendly with the children. Whenever I see another Pomeranian dog, I assume that he too will be friendly with the children." This type of reasoning we usually perform in our daily life. But it also appears that it does not guarantee the truth. The reason being that is the existence of one or few shared characteristics does not mean that all the remaining characteristic are identical.

For analogical reasoning to work, it is necessary that some relevant and known similarities between two objects or situations be give a sufficient base to a reason to believe that there are further similarities between them, and that should help to answer an open question. However, it is not always that way. At the most, analogical thinking can give the base to think of some probabilities, and these may be the base for probabilistic judgment. The kind of thinking that takes place in analogical reasoning has a simple structure:

1. Object A possesses a set of characteristics, say X,
2. Object B shares the characteristics X,
3. The object A has characteristic Y also,
4. Because the object A and B share characteristic X, based on this we conclude what is not known yet, that is, B shares Y as well, which is not necessarily correct.

In Fig. 13.10a, the analogical reasoning will work, while in (b) it may not.

Definition 13.6 (*Reasoning by Analogy*) It is the process of inferring that a *conclusion property* Y holds of a particular object or situation B (called target), given that B shares a property or set of properties X with another object/situation A (called source), which has the set of properties X.

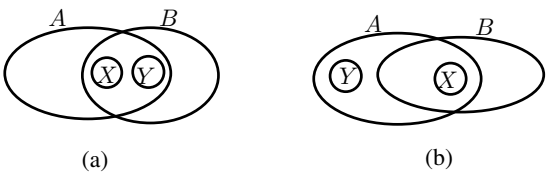


Fig. 13.10 Similarities computing for analogical reasoning

$$\begin{aligned} &A(X) \wedge B(X) \\ &A(Y) \\ &\_ \_ \_ \_ \_ \_ \\ &\therefore B(Y). \end{aligned} \tag{13.11}$$

The set of common properties X is the similarity between A and B, and the conclusion property Y is projected from A onto B. The process may be summarized by the following statements.

Justification. The analogical reasoning method defined in Eq. (13.11) can be justified by a two-step process: 1. from the first premise $A(X) \wedge B(X)$, conclude a *generalization* $\forall X[A(X) \rightarrow B(X)]$, and 2. instantiate the property X and generalize it to Y, then apply the inference rule of modus ponens to obtain the conclusion $B(Y)$. Due to this process, only the first step is non-deductive. Hence, it appears as if the problem of justifying the analogy has been reduced to the problem of justifying a single-instance inductive generalization. □

A criteria for the evaluation of strength of an (enumerative) induction states that, as the number of instances confirming the generalizations increases, an inference increases in plausibility. If this is the only criteria used for analogical inference, then all the conclusions projected by any analogy without counterexamples should also be plausible. But, this fails. Consider the example, if the inspection of an Indian peacock reveals that its tail is more colorful than its legs, a projection of this conclusion onto an unseen peacock is plausible. But, projecting that a hanging feather on the peacock's tail will be observed on a second peacock is not plausible. Anyone who has closely looked at the tail of even a single peacock will have no counterexamples to these conclusions. Also, both conclusion properties can be projected, so the difference in agreeableness is accounted for some other criterion. Hence, the problem of analogy is different from (enumerative) induction because the analogy requires a stronger criterion for plausibility.

The analogical learning is a *nontrivial* form of inference, i.e., the conclusions based on the analogies are not necessarily logical entailments of the previous (source) knowledge. Analogical is a form of plausible reasoning for the current situation.

The steps in Eq. (13.11) require many other things, like the criteria for similarity measure, some form of indexing, efficient recall mechanism, a flexible mapping between base and object domain and, various levels of abstraction.

Following are the *steps* to perform the analogical reasoning:

1. *Analogue Recognition*. A new situation or problem is encountered and recognized as being similar to the previously known situation.
2. *Criteria for competence*. It is concerned with the knowledge required for any particular set of algorithms to exhibit the specified behavior.
3. *Representation of situation by extensible relations*. Extending the mapped experience, i.e., the newly mapped analogous are modified and extended to fit the target situation.

4. *Determining Analogy and Matching*. Here, two patterns are selected for their similarities and mapped from the base to the target domain.
5. *Similarity measures*. Various similarity measures are defined, and matchings are evaluated as per those.
   The *Access and Recall* mechanisms are useful features, as per which the similarity feature of a new problem serves as an index to the previously solved problem. This will help us in recalling the previous problem when required.
6. *Abstraction*. The matchings are abstracted away to generalize it.

The newly formulated solution is validated for its applicability through some trial process, like theorem provers or simulation. If the validation is supported, a generalized solution is formed, which accounts for old and new, both, which is equal to a new learning.

When a case is inferred analogous to another on the basis of a unifying principle such that it is accepted without having been tested against other possibilities, such inference due to the analogical reasoning will be faulty. In addition, when some similarities between two cases are considered decisive on the basis of insufficient investigation of relevant differences, such analogical reasoning will also go wrong.

13.10 A Framework of Symbol-Based Learning

Given the data, one way to characterize the learning problem is based on the goal/target of the learner algorithm, which may be a concept, or description of a class of objects. A learning algorithm may also acquire plans, heuristics for problem solution, or other form of procedural knowledge in any structure discussed so far, including the predicate logic.

The set of operations performed by the algorithm may include generalizations rule, heuristics rule, or a plan that satisfies the goal. The typical operations are generalizations or specializations of symbolic expressions, adjusting the weight of a neural network, etc. In concept learning, a learner may generalize a definition as follows:

$color(bird, black) \rightarrow is(bird, crow).$
$\exists x\ color(X, black) \rightarrow is(X, crow).$

There is a potential concept space, which is searched by the learner algorithm and to find the concept. The complexity of this concept space is the main complexity of the problem solution. The learning program/algorithm must be committed to order and the direction of the search, as well as it should make use of the available training data and heuristics, to carry out the search efficiently. The learner may make use of the heuristics, and, for example, learn the concept *ball*, by using the first example as a candidate concept:

$size(object_1, small) \wedge color(object_1, red) \wedge shape(object_1, round)$

and generalize using the second example:

$$size(object_2, small) \wedge color(object_2, red) \wedge shape(object_2, round).$$

Here we have used *heuristics* and not *induction* for learning.

13.11 Explanation-Based Learning

One type of *deductive learning* is Explanation-based Learning (EBL). Following types of information are required to implement the explanation-based learning:

1. A formal statement of the *goal concept* to be learned;
2. *Domain theory*: It relates to the concept and the training example;
3. *A training example*: At least one positive training example of the concept is needed. It is an instance of the target, what the training program sees in the world;
4. *A Domain History*: It is a set of facts and rules used for explaining, how a training example is an instance of the good concept; and
5. *Operational criteria*: These are some means of describing the form, which a concept definition may assume.

The EBL makes use of an explicitly represented domain theory to construct an explanation of a training example, which is usually a proof that logically follows the knowledge base. An EBL begins with following prerequisites:

– A target Concept
– Learning by Analogy
– Learning by Instruction
– Learning by Induction
– Learning by Deduction
– Reinforcement Learning
– Discovery-based Learning, which determines the effective definition of this concept, called *local concept*, which requires a high-level description.

Consider using EBL to learn that a given specific object "this cup" is a cup. The target concept is a *rule* in predicate form, which is to be used to infer whether a given object is in fact a cup. We express this as

$$premise(X) \rightarrow cup(X). \tag{13.12}$$

In this implication, the premise is a *conjunctive expression* with variable X, and predicate expression $part(X, Y)$ indicates that $X's$ part is Y. Let the *domain knowledge* include

$$canbelifted(X) \wedge holdsliquid(X) \rightarrow cup(X)$$

$light(X) \wedge part(X, handle) \rightarrow canbelifted(X)$
$part(X, Y) \wedge concave(X) \wedge pointsup(X) \rightarrow up(X)$
$small(X) \rightarrow light(X)$

where predicate $holdsliquid(X)$ means that object X holds liquid, $canbelifted(X)$ means X can be lifted, and $pointsup(X)$ means object X points upwards.

A training example should be an instance of the concept for the goal concept, as represented by the following facts of predicate logic.

$cup(thiscup)$.
$small(thiscup)$.
$put(thiscup, handle)$.
$own(bob, thiscup)$.
$part(thiscup, bottom)$.
$part(thiscup, bowl)$.
$color(thiscup, red)$.
$pointsup(bowl)$.
$concave(bowl)$.

The operational criteria should be such that the target concept can be defined in terms of observable structured properties of object, like it points up and its parts:bottom, handle, bowl, etc. The domain rules are provided in such a way that these enable the learner (algorithm) to infer whether a description is operational or not. Making use of this approach, a theorem prover can construct an explanation as to why the given example is in fact an instance of the training concept—a proof that confirms that the target concept logically follows from the example. This is shown in Fig. 13.11, as a generalized proof tree.

The EBL has the following advantages:

1. The training examples often contain many irrelevant information (or noise), such as "make of the cup". However, the domain theory allows the learner to select only the relevant aspects of training example.

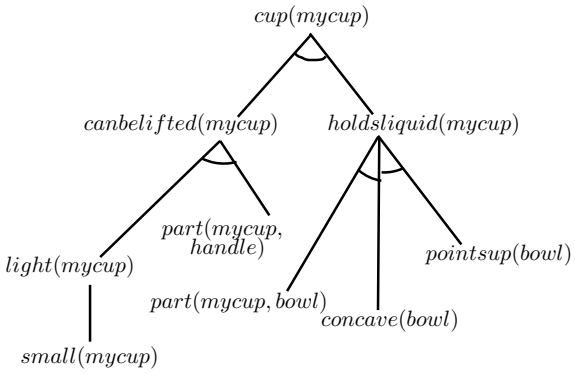


Fig. 13.11 Explanation-based learning to identify the object cup

2. Usually, an example may allow for many possible generalizations, and most of them may not be useful. However, since the EBL inference logically follows the example, the generalization of EBL are logically consistent with the domain theory.

13.12 Machine Learning Applications

The domain of machine learning applications is continuously expanding. In the following we present some of the major application areas of machine learning:

Machine Perception

These approaches are used for learning to perform segmentation, feature extraction, and classification. All these provide higher performance and greater ease of implementation than traditional systems built from individual hand-built components.

Large-Scale Information Management and Data Analysis

In the present time, the data are being gathered in businesses and scientific applications far faster than ever before. This volume of data cannot be disseminated by any manual method. The machine learning methods are helpful in finding patterns and relationships in these data sets.

Information Filtering and Information Retrieval

Finding out useful information from an abundant size of collections is impossible using any manual methods. Models of the information needs of users are constructed based on machine-learning methods for various available information sources. Using these models it is possible to process the huge volume of information, both offline and online, to filter the undesired information and deliver to users only the information they need. However, for this the users should first express their need in natural language or symbolic form.

Control and Optimization

The *adaptive control methods* are more of futuristic systems, which are aimed to provide large-scale optimization, e.g., over the entire enterprises for planning, scheduling, manufacturing, inventory management, and logistics. The machine learning methods in controls and optimizations may also make it possible for new kinds of applications, such as smart buildings and smart vehicles.

Speech Recognition

All the presently available speech recognition systems in the market, make use of machine learning in some form, to train the system to recognize speech of unknown users as well of specific users. Before shipping, such systems (programs) are trained in speaker-independent mode, and after it is delivered to the individual users, it is

trained in speaker-dependent mode to accurately recognize his/her voice. Chapter 20 presents the speech-recognition techniques in more detail.

Computer Vision

The computer vision (or machine vision) applications range from face-recognition systems to sophisticated systems that automatically classify microscope images of cells. Many of the computer systems are more accurate than hand-crafted programs. Some of the important applications of computer vision are handwriting recognition, fingerprint recognition, and face recognition, which are available at a commercial scale.

Model Building

Many areas in science and engineering require construction of complex models, to support monitoring, diagnosis, repair, prediction, and extrapolation. Such models are a combination of programming and adaptation. Programming is used for the construction of the model, while adaptation is aimed to modify the model and calibrate it for data. The designers of such models need better tools for construction, calibration, and managing the models. These models have applications in medical science, strategic systems, space research, etc.

Machine Learning Techniques to Make Computers Easier to Use

Computers are still not easier to use due to ignorance of the user. While using a computer, each user has certain goals (about tasks, resources, etc.) and different preferences (that include, styles, habits, and abilities). Computer systems have very small features for these yet. Some examples are spell checker, command repetition, exception handling in high-level languages, in operating system, and in databases. However, these things are progressively appearing in plenty in smart phones, which include suggesting the next word while sending email or message, suggesting navigation path for drivers through map applications, storing the schedules and appointments and reminding through alarms, progressive learning of languages, etc. There is trend now of smart-watches which are claiming to provide solution for many things related to health and health monitoring.

13.13 Basic Research Problems in Machines Learning

There are wide-ranging machine-learning algorithms, from *general*—"off-the-shelf" methods for a variety of problems to *custom methods* suited for specific tasks. For example, general methods are learning through decision trees, rule sets, probabilistic networks, and feedforward neural networks, which have been applied to many different problems. On the other side, speech recognition algorithms based on hidden Markov models (HMMs), and methods for inferring using binding-site geometry for drug design are application-specific.

Many fundamental research problems in machine learning are concerned about the entire range from general to specific spectrum.

1. It is not known yet, as to what all the possible general methods are, as so far only the four techniques exist in the general category: algorithms for decision trees, rules, Bayes nets, and neural nets. It is not clear as to what other convenient representational formalism should be explored.
2. It is not known as to what extent these four general methods are optimal, or if there is scope for improvement in these algorithms.
3. What are truly the theoretical and engineering principles, that can be used as guidelines for the development of application-specific machine learning algorithms? The development of such algorithms is currently expensive, time-consuming, and mostly ad hoc in nature.
4. What are the efficient and convenient methods that can be used to acquire, represent, and incorporate the application-specific knowledge into general learning methods?

In addition, there are several other challenging questions of fundamental nature in the sequential decision-making tasks.

1. What are the situations in which it is preferable to represent and learn a *policy* directly, instead of learning a *value function* first? The difference between learning a policy versus value function is a policy specifies what action should be chosen in each state, but a value function provides accumulated long-term reward for each state. The best action can be selected based on the criteria: the one that will lead to the state with the highest value as per the value function. Some methods, like genetic algorithms, genetic programming, and parameterized policies, search directly in policy space. However, the other methods, like temporal difference learning and real-time dynamic programming, construct value functions.
2. What should be the properties of an ideal value function approximator?
3. What are the best ways to resolve the trade-off between *exploitation* (i.e., execution of the current policy) and *exploration* (i.e., search for better policies)?
4. What are the ways to solve large and partially observable Markov decision problems?

13.14 Summary

In every learning situation, a learner transforms the information received from an environment/teacher into some new form, and stores in that format for future use. The type of learning strategy used decides the type and magnitude of this transformation. Following are the basic learning strategies:

1. Rote Learning.
2. Learning by Instruction.

3. Learning by Deduction.
4. Learning by Analogy.
5. Learning by Induction.
6. Reinforcement Learning.
7. Discovery-based Learning.

These strategies are in increasing order of complexity of learning as well as the magnitude of transformation required.

A general learning model comprises: learner unit, knowledge base, performance evaluation, and the teacher, which may or may not include a physical teacher.

The learning techniques are fundamentally classified as *supervised learning* and *unsupervised learning*. The major difference between these is that the supervised learning requires the presence of a teacher, while the unsupervised technique does not require the teacher component.

Inductive Learning is a generalization carried out using a set of examples. It is one of the most commonly used learning techniques used by humans. Learning through *concepts* is a typical inductive learning process: given the examples of some concepts, such as "cat", we attempt to conclude a definition such that it will be helpful to correctly recognize future instances of the concept "cat". Note that, inductive learning makes use of specific facts/examples rather than general axioms as in the deductive processes.

While searching for possible hypotheses in the large space, the space can be constrained using the domain knowledge of the experts. *Learning from examples* is given the examples find a theory that is consistent with the examples. The explanation examples are called *argumented* examples, and these are in the form of arguments: *for* and *against*.

Based on the idea, whether it is a single concept or multiple concepts that are to be learned, we call it *single-concept* or *multiple-concept learning*. A *concept* is a function (a mapping), such that the learning task is supposed to discover this function, called *target function* $f^* : \{0, 1\}^n \rightarrow \{0, 1\}$. The function maps each *instance* to the correct *label* (also called *class*). In the learning process, the system responds to learning instances and also to feedback received from the teacher.

The *attribute-value* learners or *propositional learners* are the methods that use a formalism of propositional calculus, where objects are defined with a fixed set of attributes. The learning methods based on first-order relational descriptions are called *relational learners*; they induce descriptions of relations, and make use of objects described in terms of their components and relations among components.

Automated Mathematician (AM) is one of the earliest *discovery-based learning* programs, deriving a number of concepts in mathematics. It is based on the concepts of set theory. AM discovers the natural numbers by modifying concept of "bag", which is a generalization of set. An approach to solving a problems, which is among the most general, is to decompose the problem into smaller, less complex, and better manageable subproblems. This principle is the foundation for learning by *concept hierarchies*. The rules for each concepts and sub-concepts are learned as a *discovery learning*.

Reinforcement Learning (RL) is an example of programs, which improve their performance for some task based on the criteria of rewards and punishments from the environment. Most approaches to reinforcement learning optimize the total *discounted* reward received by the learner.

For *analogical reasoning*, an example in the form of a known solution is sufficient for learning a new solution. It is the process of inferring that a *conclusion property* Y holds a particular object or situation B (called target): from the fact that B shares a property or set of properties X with another object/situation A (called source), which has the set of properties X. The set of common properties X is the similarity between A and B, and the conclusion property Y is projected from A onto B.

Goal of most learning algorithms is a concept, or a general description of a class of objects, represented in predicate logic. There is a potential concept space, which the learner searches to find the concept. A learner may generalize a definition as follows:

$$color(bird, black) \rightarrow is(bird, crow)$$
$$\exists x\ color(X, black) \rightarrow is(X, crow).$$

The above is *symbol-based* learning.

One type of *deductive learning* is Explanation-based Learning (EBL). The following information is required to implement it: a *goal concept*, *domain theory*, *a training example*, *a domain history*, and *operational criteria*.

Potential Applications of machine learning are machine perception, information management and large-scale data analysis, information filtering and retrieval, control and optimization, software engineering, speech recognition, computer vision, model building, etc.

Exercises

1. At the end of the day, try to recollect any five important events you have witnessed this day. Try to associate the type of learning you have used in these events.
2. Analyze and propose the type of structures you consider are appropriate in the following learning processes:

   a. Rote learning
   b. Inductive learning
   c. Supervised learning
   d. Unsupervised learning
   e. Learning by example
   f. Analogical learning

   Answer the above questions, in reference to some programming language, e.g., *C*. And, describe the estimated algorithmic steps. For example, for rote learning

you may take the example of the number tables, and indexing you need, so that one can speak the table of say "5", and also one can answer $5 \times 6 = 30$.

3. What is the difference between the *learning by induction* versus *learning by examples*.
4. Explain the major difference between analogical learning and inductive learning in respect of the approach used for learning.
5. Suggest a learning method for each of the following, explaining why the suggested method is appropriate, and provide logical steps to learn using it.

    a. To learn how to drive a car after having observed a trained driver, while you are riding along with the same.
    b. To learn how to drive a car after having known the driving of a bullock-cart.
    c. To learn how to drive a car after having learned the driving of a tractor.
    d. To learn how to fly an aircraft after having very closely observed the birds flying, like eagle, crane, and hawk.
    e. Learning to keep your wallet protected after having lost it due to theft.

6. The helicopter takes off straight, instead of having a run before taking off. It has similarity with peacock in the birds. Explain a mathematical model, which supports the learning by analogy of a copter with a peacock.

References

1. Ahmadi M et al (2007) IFSA: incremental feature-set augmentation for reinforcement learning tasks. In: The sixth international joint conference on autonomous agents and multi-agent systems, pp 1128–1135
2. Bengio Y (2016) Machines who learn. Sci Am 6:38–43
3. Domingos P (2012) A few useful things to know about machine learning. Commun ACM 55(10):78–87
4. Mantaras RL, Armengol E (1998) Machine learning from examples: inductive and Lazy methods. Data Knowl Eng 25(1998):99–123
5. Mozina M et al (2007) Argument based machine learning. Artif Intell 171:922–937
6. Schmid U, Kitzelmann E (2011) Inductive rule learning on the knowledge level. Cogn Syst Res 12:237–248
7. Sunstein CR (1993) On analogical reasoning. Harv Law Rev 106:741–791
8. Tadepalli P, OK D (1998) Model-based average reward reinforcement learning. Artif Intell 100:177–224
9. Wang J, Gasser L (2002) Mutual online concept learning for multiple agents. In: AAMAS'02, July 15–19, Bologna, Italy
10. Winston PH (1980) Learning and reasoning by analogy. Commun ACM 23(12):689–703
11. Zupan B et al (1999) Learning by discovering concept hierarchies. Artif Intell 109:211–242