Different Conceptions of Learning: Function Approximation vs. Self-Organization

Pei Wang(✉) and Xiang Li

Department of Computer and Information Sciences,
Temple University, Philadelphia, USA
{pei.wang,xiang.li003}@temple.edu

Abstract. This paper compares two understandings of "learning" in the context of AGI research: *algorithmic learning* that approximates an input/output function according to given instances, and *inferential learning* that organizes various aspects of the system according to experience. The former is how "learning" is often interpreted in the machine learning community, while the latter is exemplified by the AGI system NARS. This paper describes the learning mechanism of NARS, and contrasts it with canonical machine learning algorithms. It is concluded that inferential learning is arguably more fundamental for AGI systems.

1 Learning: Different Conceptions

Learning is always considered an important aspect of intelligence. *Machine learning* has grown into one of the most active fields in AI, and it gives computers the ability to work without being explicitly programmed with problem-specific skills. Learning algorithms have been used in many applications in various domains [4,14]. In recent years, deep learning techniques have dramatically improved the state-of-the-art in fields like vision, speech recognition, natural language processing, and game playing [6,19].

However, it does not mean that the existing machine learning techniques have satisfied the needs for learning in AGI systems. In cognitive science, "learning" is usually taken as having multiple forms, such as associative learning, skill learning, inductive learning, etc. [12]. Though machine learning study also covers supervised learning, unsupervised learning, reinforcement learning, etc., many of them can still be considered "algorithmic learning", consisting of two steps, each following an algorithm [4]:

1. the learning process follows an algorithm that takes the training data as input, and produces a model as output;
2. then, the model serves as an algorithm that carries out the domain task.

A comparison between an early (1984) collection on machine learning [7] and a recent (2012) textbook on the topic [4] shows clearly that algorithmic learning has become the dominate paradigm, while the other forms of learning have

B. Steunebrink et al. (Eds.): AGI 2016, LNAI 9782, pp. 140–149, 2016.
DOI: 10.1007/978-3-319-41649-6_14

largely faded out. Of course, as a highly diverse field, not all machine learning techniques fit this description, but it is still a widely adopted framework.

Algorithmic learning has its roots in the foundation of computer science, where a problem-solving process is normally specified as following a problem-specific algorithm. In conventional systems, the problem-specific algorithms are programed by human beings and followed by computers, so consequently a computer system serves as a function that maps any given problem instance into the corresponding solution. In algorithmic learning, the domain task is still taken to be a function to be carried out, except that this function is not specified in its general form by an algorithm, but exemplified by the training data, so "learning" becomes the problem of generalizing the training data to an approximate function. This process is taken to be a meta-problem (function approximation) to be solved by a meta-algorithm (the learning algorithm).

A typical way to realize algorithmic learning is with a feedforward neural networks [4,6]. It has been proved that such networks are universal function approximators. In such a system, the input and output data, as well as the intermediate results, are normally represented as vectors. Each instance corresponds to a point in a multidimensional space. The domain task is to produce an output vector according to the input vector, under the assumption that similar inputs will yield similar outputs. In other words, the function to be approximated (or learned) must be *smooth*.

The training instances are usually assumed to be randomly and independently selected from a sample space. The learning algorithm takes the training data as input, and produces a model that maps every instance in the sample space to output in a way that is mostly consistent with the training data. The model should be confirmed using certain testing data, also randomly selected from the sample space. After that the model is ready to be used as an algorithm for the domain problem, where the data come from the same sample space, too.

Advances in learning algorithms, an increase in available training data, and ever more powerful hardware have allowed algorithmic learning techniques, especially *deep neural networks* that have multiple intermediate layers, to produce surprisingly good results.

In spite of its successes, "function approximation" is not the only meaningful interpretation of "learning". Even if we ignore the relevant works in cognitive psychology and developmental psychology, and focus on machine learning only, there is still a tradition of *inferential learning*. This tradition came from the logical study of non-deductive inference (*induction*, *abduction*, and *analogy*), with the work of Michalski as the best-known example [8,9,21].

This approach represents a understanding of learning that is very different from algorithmic learning (as exemplified by feedforward neural networks), as shown in the following aspects:

Knowledge representation: The system's knowledge base is represented as a set of beliefs, which usually can be seen as a conceptual network or hierarchy.

Learning problem specification: Learning corresponds to the modification of the knowledge base, such as adding, deleting, and revising beliefs. In terms

of the conceptual network, learning means the modification of the topological structure, as well as the adjustment of the parameters of the network.

Learning process specification: The learning process follows a set of inference rules, each triggered under certain conditions and cause specific effects. Learning process is not necessarily separated from the other reasoning processes, and there may be no specific "learning algorithm".

Though this conception of learning appeared early in the history of machine learning, it has gradually lost favor in that community, and is even rarely mentioned in recent textbooks [4]. What we would like to do in this paper is to provide a more advanced model of inferential learning, and to argue for its advantages over algorithmic learning in the context of AGI projects. Some of the arguments have been presented in our previous publications [21–23], though here they are revised to take the recent progresses into account.

2 Learning in NARS

NARS is a reasoning system based on the theory that "intelligence" is the ability for a system to *adapt* given *insufficient knowledge and resources*. That is, the system must depend on *finite* resources to make *real-time* response while being *open* to unanticipated problems and events. In other words, an intelligent system must be able to *learn* from its experience in a problem domain. Consequently, the system's solutions are usually not absolutely optimal, but the best the system can find at the time, and the system could always do better if it had more knowledge and resources. The project has been described in detail in two books [23,24] and many papers, which, and the source code of the current implementation, can be accessed at http://cis-linux1.temple.edu/~pwang/. In this paper, we only briefly introduce the learning mechanism of the system.

The domain knowledge of NARS is represented as a collection of *beliefs*. These summarize the system's experience in the form of a sentences according to a formal grammar. The grammar of NARS is distinct from most other reasoning systems where the representation is a first-order predicate calculus or variant thereof. The language of NARS is based in the term-logic tradition, where a sentence has a "subject-copula-predicate" format.

A typical statement in NARS has the form of "$S \rightarrow P \langle t \rangle$", where '$S$' is the subject term, '$P$' the predicate term, '$\rightarrow$' the "inheritance copula" indicating that the subject is a specialization of the predicate (or, equivalently, the predicate is a generalization of the subject), and 't' the truth-value of the belief, which is a pair of numbers "frequency-confidence" that measures the evidential support the belief gets from the system's experience. The frequency value is defined as w^+/w (the proportion of positive evidence among all current evidence), and the confidence value is defined as $w/(w + k)$ (the proportion of current evidence among all future evidence after a constant amount of evidence arrives). Detailed explanations of the NARS truth-value can be found in [23], while for the current discussion, it is enough to know that no belief is absolutely certain, and all truth-values can be revised by new evidence, though those with higher confidence will

be more stable. This definition of truth-value is one implication of the assumption of insufficient knowledge and resources.

Using this language, the beliefs of the system can be naturally visualized as a weighted graph, with terms as vertices, statements as edges, and truth-values as weights. When implemented in a computer system, a "concept" is a data structure identified by a term and referring to all the statements with that term as a component (e.g., subject or predicate). The "content" or "meaning" of a concept consists of what the system knows about it at the moment. Since concepts are summaries of the system's experience, there will be new concepts introduced or generated from time to time, and the meaning of the existing concepts may also change. Therefore NARS has an "experience-grounded" semantics.

The assumption of insufficient resources has several implications. Due to storage restriction, some contents in a concept may be removed to release space for new contents, and even whole concepts may be deleted. This mechanism is responsible for "absolute forgetting", by which some information is permanently lost. Due to the demand of producing real-time responses to inference tasks, the beliefs within a concept have a priority distribution maintained dynamically among them, and so do the concepts in the memory. At any moment, the system allocates its processing time among its concepts according to their priority values. Within each concept, high-priority beliefs are accessed more often.

According to the changes in the environment, as well as the feedback after each inference step, the priority values of the relevant concepts and beliefs are adjusted, partly to reflect their relevance to the current situation and their usefulness in past problem-solving processes. This mechanism is responsible for "relative forgetting", by which some information gradually becomes less accessible. Though forgetting is often undesired, it is inevitable in a system that is always open to new information and must make real-time responses.

Though the above description only provides a highly simplified description of the memory of NARS, it still shows that the domain knowledge of NARS can be roughly divided into three kinds:

1. The terms and the concepts named by them,
2. The beliefs that relate the terms to each other, with their truth-values,
3. The priority values of the concepts and beliefs.

NARS has *complete* learning capability, in the sense that all of the three kinds of knowledge can be generated or modified by the system itself, given proper experience.

Experiences in NARS consists of a stream of input sentence, and each of them is a task to be processed. If an input task contains a novel term or statement that does not exist in the system at the moment, the corresponding new concept and statement will be generated and added into the memory. This is the simplest form of learning: learning by being told.

New concept and statements can also be generated by the system's inference rules. NARS is designed in the *term logic* framework, mainly because this framework naturally supports the unification of several types of inference. Here *deduction* takes the syllogistic form as in Aristotle's Syllogistic [1], then following the

approach of Peirce [11], *abduction* and *induction* can be obtained from *deduction* by exchanging a premise and the conclusion.

	deduction	**abduction**	**induction**
first premise	$M \rightarrow P$	$P \rightarrow M$	$M \rightarrow P$
second premise	$S \rightarrow M$	$S \rightarrow M$	$M \rightarrow S$
conclusion	$S \rightarrow P$	$S \rightarrow P$	$S \rightarrow P$

Different from Aristotle and Peirce, we define these types of inference in a multi-valued logic, so there is a truth-value function attached to each rule to calculate the truth-value of the conclusion from those of the premises, and different rules have different functions. Among the above three, *deduction* may generate high-confidence conclusions, while the other two can only generate low-confidence conclusions, which are usually considered as hypotheses.

Beside the above *syllogistic rules* that generate new statements among given terms, NARS also has *compositional rules* that generate new terms to summarize given premises. The following are some examples.

	union	**intersection**	**difference**
first premise	$M \rightarrow P$	$M \rightarrow P$	$M \rightarrow P$
second premise	$M \rightarrow S$	$M \rightarrow S$	$M \rightarrow S$
conclusion	$M \rightarrow (S \cup P)$	$M \rightarrow (S \cap P)$	$M \rightarrow (S - P)$

These rules also have their truth-value functions. In these rules, the predicate terms of the conclusions are "compound terms" composed from the predicate terms of the premises, so as to provide a more efficient representation of the same information. There are also rules where the subject terms of the conclusions are compounds.

Beside the above compounds that are similar (though not identical) to set operations, compound terms also include Cartesian products (i.e., ordered sequences) of terms and statements themselves (so there can be statements of statements). Furthermore, compounds can be used recursively to form more complicated terms, as well as used in inference in the same way as the atomic (i.e., non-compound) terms. In this way, the system can learn the patterns that it noticed in experience, and then reason on them in the same way as atomic terms.

Here what the syllogistic rules and compositional rules do can be considered both as *reasoning* and as *learning*, from different perspective. Such a step is learning because the conclusion represents the experience provided by the premises in a different form, and the inference step modifies the memory in an enduring manner.

If an input or generated statement or term already exists in the memory, the new item will be merged with the existing item, which may lead to the change in the truth-value of the statement or the content of the concept identified by the term. These changes also let the system gradually learn from its experience.

Finally, the priority-value adjustments correspond to the learning of a type of domain knowledge, too, even though the results of this learning is embedded in

the memory structure, rather than explicitly expressed by terms and statements. It is well known that in problem solving, it is often crucial to know which concepts and beliefs should be considered first among all alternatives, and this type of knowledge usually comes from the accumulated feedback of problem-solving practices.

Putting the above learning mechanisms together, NARS can start with an empty memory and learn all domain knowledge from its experience. Even though for practical considerations the system can also start with a preloaded memory, all contents in the memory can still be revised and adjusted by future experience. The primary function of learning in NARS is to organize its experience to meet the need of adaptation under the restriction of insufficient knowledge and resources. The only knowledge that cannot be learned is the meta-knowledge embedded in the system's grammar rules, inference rules, resources management policy, etc. Since all meta-knowledge is domain-independent, NARS can be put to learn and work in any domain.

3 Comparison and Discussion

Though the above descriptions about the two conceptions of learning are relatively simple, they nevertheless show enough differences. In the following they are compared, using NARS and the deep learning (DL) architecture described in [6] as examples. The evaluation will be based on in the requirements of AGI, rather than that of a specific domain problem.

In knowledge representation, the two approaches follow different traditions: NARS uses sentences in a formal language, while DL uses vectors in a feature space. In principle, the two are equivalent, in the sense that any knowledge base in one can be converted into the other. However, in practice there are several factors to be considered, such as naturalness, flexibility, and efficiency. Vector-based representation is natural for sensory input, but much less so for symbolic input. Though "word vectors" and "thought vectors" have been introduced, they are either inefficient (when the number of dimension is high) or hard to interpret (when the number of dimension is low). To explicitly represent conceptual structure is also difficult. On the other hand, NARS allows vectors to be used as a special type of compound term (called *product* in [23,24]), which, plus the other types of compound terms, make the representation of symbolic structure very easy and natural.

In DL, the vectors are connected in a layered network with a fixed topological structures, with links between layers corresponding to generalization. As explained above, the conceptual structure in NARS is dynamically generated and modified, and the conceptual relation *inheritance* also corresponds to generalization. NARS also has other copulas, including *similarity*, *implication*, and *equivalence*, so can directly represent various conceptual relations. Furthermore, NARS has no problem to introduce new terms at run time, while DL normally requires a constant vocabulary.

NARS is designed according to an experience-grounded semantics, which is intuitively located between the *localized* representation of symbolic approaches

and the *distributed* representation of neural networks. In NARS, the meaning of a concept is defined by its relations with other concepts, not an "object" it refers to, and each relation is true to a degree. In this aspect, its knowledge representation is also "distributed" to an extent. However, all the relations in the network correspond to copulas that have well-defined meaning, rather than merely as associations that spread activations. For this reason, it is possible to carry out various types of inference, which can be justified as valid according to the semantic theory. The representation also has a "localized" aspect, as each individual concept and belief can be interpreted meaningfully by itself, rather than only has a holistic interpretation as in DL – in DL only the input and output vectors are meaningful outside the network, while the intermediate results represented by the hidden nodes are not always easy to understand.

In DL, the task of a learning system is usually taken to be the approximation of a function defined between the input and output of the whole system. This focus on end-to-end relation allows the system to be trained and evaluated as a blackbox, without the need to interpret the intermediate results. Though such a treatment is desired for systems designed for single tasks, it has difficulty in handling multiple types of tasks. The explorations in transfer learning [20], multitask learning [17], and multi-strategy learning [2] are pushing machine learning in this direction, though the progress so far is limited. On the contrary, NARS is not designed to answer any specific type of question. Instead, it can be asked any question expressible in its language. Consequently, any concept and belief in the system can be asked, so it can be learned by the system, and after that it can also be used to answer questions about other concepts and beliefs. In this way, the distinction of "input" and "output" terms/statements is relative and temporary, and there is no separately defined "task domains".

Besides being restricted to a single function exemplified by the training data, the blackbox nature of DL has other issues, such as the interpretation of its successes and mistakes. It has been found that learning algorithms can make mistakes that never happen to human beings [5,10]. One possible reason is because the intermediate levels of generalization are not directly verified, but only judged by their contributions to the ultimate output. Since after each generalization the feature space is transferred into another one, similar points in the former and those in the latter are not necessarily the same, as long as they are not sufficiently close to a training instance. On the contrary, in NARS every level of generalization is produced by inference rules and verified independently. Even though the system may make mistakes due to insufficient knowledge and resources, it will not make incomprehensible mistakes.

NARS allows multiple levels of generalization of the same experience. For example, the observation "Tweety flies" can be generalized by the belief "Tweety is a canary" to "Canaries fly", by the belief "Tweety is a bird" to "Birds fly", and by the belief "Tweety is a animal" to "Animals fly", all by the same *induction rule*. It is the other experience that will gradually differentiate these results: over-generalization ("Animals fly") will lose priority due to its low frequency (from more negative evidence), and under-generalization ("Canaries fly") will lose priority due to its

low confidence (from less total evidence), compared to the proper generalization ("Birds fly"). Of course the system will not try all possible generalizations, but only those that get the system's attention in resource competition. This possibility of multiple-generalization suggests solutions to the over-fitting problem and the inductive biases problem, as the system can keep several generalizations for different needs, instead of choosing one at the beginning. Similarly, in concept learning the result is not necessarily a partition of the instances, but can allow multiple inheritance or overlapping concepts with graded membership. For example, Tweety can be learned to be a canary, a yellow thing, and a cartoon character, at the same time.

The learning mechanism in NARS covers several processes that adjust various aspects of the memory. So there is no single "learning algorithm" in the system, but multiple algorithms for different purposes. Furthermore, learning is unified with reasoning, and is carried out by a set of inference rules, each implemented by a lightweight algorithm that can be finished within a constant amount of time. Due to insufficient resources, NARS does not attempt to use all relevant beliefs on each problem, but only accesses the high-priority ones within the currently available time for the problem. Consequently, a system-level learning problem, such as to digest a piece of new knowledge or to acquire a complicated concept, is carried out by many inference steps linked together at the run time according to many ever-changing factors, so may not be accurately repeatable. This result also means that in NARS there is no problem-specific algorithm or stable input–output mapping, as the output depends on history and context.

In NARS, this "non-algorithmic" learning mechanism provides a unified solution to several issues that are being explored in machine learning study:

One-shot learning [3]**:** Inference rules like *induction* and *abduction* can use a single example to generate a *hypothesis*, i.e., a belief with low confidence value. More examples will increase the confidence value, though are not required for the generation of hypotheses.

Online learning [16]**:** As an open system, NARS does not require all training data to be available at the very beginning, but allows them to come from time to time. New evidence will lead to incremental revision of existing beliefs and concepts, rather than demanding a complete retraining.

Real-time learning [13]**:** Since each *type* of learning task in NARS does not follow a fixed algorithm, its processing has no determined time requirement. Instead, each task *instance* can have its only time requirement attached, and the system will process it accordingly, similar to an anytime algorithm.

Active learning [15]**:** NARS uses backward inference to generate derived questions, so has the ability to actively collect information to answer questions, rather than passively using whatever input the environment provides to it. Even the existing beliefs will be selectively used.

Life-long learning [18]**:** In NARS, learning is not a separate process, but a cognitive function carried out by almost all the processes in the system. As long as the system runs, multiple self-organizing activities will happen in

various parts. In no time will a belief or concept be finally "learned" so that no further revision is possible.

In NARS, all these properties are natural features of the inferential learning processes, rather than additional attributes to be realized separately.

4 Conclusions

In machine learning study, "learning" is often formalized as a computational process following an algorithm, and the process produces an approximate function according to training cases, which then is used to solve a domain problem. Though this "algorithmic learning" surely cannot cover all types of machine learning techniques, it nevertheless is a representative paradigm.

Such an algorithm is designed to carry out a specific type of learning in isolation, while in an AGI system the learning function and the other cognitive functions need to be unified or closely integrated. Furthermore, learning in an AGI system needs to be carried out under the restriction of available knowledge and resources. For instance, very often the system does not have the amount of training instances demanded by most learning algorithms, and the system often needs to work in real time, without a dedicated training period.

In "inferential learning", as implemented in NARS, "learning" is unified with reasoning and other cognitive functions, carried out using finite processing capability in real time, and open to novel tasks. Here learning does not follow a specific algorithm, but serves multiple self-organizing roles in various ways to make the system's behaviors dependent on history and context.

Though algorithmic learning, especially deep neural network, has achieved great successes in various domains, in AGI research it still have many challenges. Compared to it, inferential learning may provide a better alternative as the main learning paradigm in AGI.

Acknowledgments. The authors thank the anonymous reviewers for their helpful comments and suggestions.

References

1. Smith, R.: Aristotle : Prior Analytics. Hackett Publishing Company, Indianapolis (1989)
2. Estlin, T.A.: Using multi-strategy learning to improve planning efficiency and quality. Ph.D. thesis, Department of Computer Sciences, The University of Texas at Austin, Austin, TX (1998)
3. Fei-Fei, L., Fergus, R., Perona, P.: One-shot learning of object categories. IEEE Trans. Pattern Anal. Mach. Intell. **28**, 594–611 (2006)
4. Flach, P.: Machine Learning: The Art and Science of Algorithms That Make Sense of Data. Cambridge University Press, New York (2012)
5. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. In: International Conference on Learning Representations (2015)

6. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**, 436–444 (2015)
7. Michalski, R., Carbonell, J., Mitchell, T. (eds.): Machine Learning: An Artificial Intelligence Approach. Springer, Heidelberg (1984)
8. Michalski, R.S.: A theory and methodology of inductive learning. Artif. Intell. **20**, 111–116 (1983)
9. Michalski, R.S.: Inference theory of learning as a conceptual basis for multistrategy learning. Mach. Learn. **11**, 111–151 (1993)
10. Nguyen, A., Yosinski, J., Clune, J.: Deep neural networks are easily fooled: high confidence predictions for unrecognizable images. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (2015)
11. Peirce, C.S.: Collected Papers of Charles Sanders Peirce, vol. 2. Harvard University Press, Cambridge (1931)
12. Reisberg, D.: Learning. In: Wilson, R.A., Keil, F.C. (eds.) The MIT Encyclopedia of the Cognitive Sciences, pp. 460–461. MIT Press, Cambridge (1999)
13. Roshtkhari, M.J., Levine, M.D.: An on-line, real-time learning method for detecting anomalies in videos using spatio-temporal compositions. Comput. Vis. Image Underst. **117**(10), 1436–1452 (2013)
14. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach, 3rd edn. Prentice Hall, Upper Saddle River (2010)
15. Settles, B.: Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin-Madison (2010)
16. Shalev-Shwartz, S.: Online learning and online convex optimization. Found. Trends Mach. Learn. **4**(2), 107–194 (2011)
17. Silver, D.: Selective functional transfer: inductive bias from related tasks. In: IASTED International Conference on Artificial Intelligence and Soft Computing, pp. 182–189. ACTA Press (2001)
18. Silver, D., Yang, Q., Li, L.: Lifelong machine learning systems: beyond learning algorithms. In: Technical Report of AAAI Spring Symposium Series (2013)
19. Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., Hassabis, D.: Mastering the game of Go with deep neural networks and tree search. Nature **529**, 484–489 (2016)
20. Taylor, M.E., Kuhlmann, G., Stone, P.: Transfer learning and intelligence: an argument and approach. In: Proceedings of the First Conference on Artificial General Intelligence (2008)
21. Wang, P.: The logic of learning. In: Working Notes of the AAAI workshop on New Research Problems for Machine Learning, pp. 37–40. Austin, Texas (2000)
22. Wang, P.: Artificial general intelligence and classical neural network. In: Proceedings of the IEEE International Conference on Granular Computing. Atlanta, Georgia (2006)
23. Wang, P.: Rigid Flexibility: The Logic of Intelligence. Springer, Dordrecht (2006)
24. Wang, P.: Non-Axiomatic Logic: A Model of Intelligent Reasoning. World Scientific, Singapore (2013)