AN ARCHITECTURE OF LEARNING SYSTEMS

Here we present a general architecture for a machine learning system. In this architecture, the learning system consists of three components: the learning element (LE), the performance element (PE), and the knowledge base (KB). The learning system interacts with an out-side teacher (can be viewed as an oracle) or the out-side environment. The diagram of the system architecture is described in Fig. 1.

The Learning
Element (LE)

The Knowledge
Base (KB)

The Teacher
(The environment)

The Performance
Element (PE)

Fig 1. An Architecture of Learning Systems.

Here we assume that we have a learning system which learns certain kind of rules to perform a specific task in a domain D such as recognizing a concept, performing multiplications, etc. The learning element is the major component of the system. The core of the learning element is a learning algorithm A0 which interacts with the teacher and the performance element to gather examples and then it draws inferences based on the examples and its own knowledge to construct the rules (which we call target rules) which will be subsequently used by the performance element. The performance element can be viewed as a problem solving system which executes domain specific tasks and also provides the learning element with feed-back. The knowledge base (KB) contains both the knowledge about the specific problem domain D (domain knowledge) and the knowledge about learning in the domain D. The domain knowledge is to be used by the performance element while the knowledge about learning will be used by the learning element. For example, assume that we have a system which learns to make a cake from teacher and also through practice. The knowledge about cake-making (such as recipes, steps to follow in making cakes, etc.) in the KB will guide the performance element to make cake, while the knowledge such as "when you observe one successful application of the rules (i.e., you make a very good cake), give credit to the rules used and if possible, generalize the rules to guide future cake-making" will be used by the learning element. The teacher (or oracle) provides the learning element with examples (positive and maybe also negative) of problem solving in the domain D, and it may also answer questions posed by the learning system.

Some learning systems may not have the performance element. In such a case, the learning element interacts only with the teacher (or environment) to get examples of the target rules and proceeds from these examples. We can view such a system as a sub-system of a bigger system where the knowledge learned by the learning system is used by a problem solver. In the extreme case, the learning system may have neither teacher nor performance element, and it does not interacts with its environment either. In this case, the learning element merely does a sort of self-reflection or introspection, which reorganizes the domain knowledge thus (hopefully) makes the subsequent use of such knowledge in problem solving more effective. In some cases, the learning system does not have an out-side teacher available; but the learning element can make observations of its environment and learn useful rules which explain the observations. Also the performance element can perform experiments about the environment which will produce new observations and hence trigger the learning of some new rules. In this case, the learning system is essentially a discovery system.

RESEARCH PARADIGMS IN MACHINE LEARNING

If we look back at the history of the machine learning field, we will find that at different times research efforts in this area placed emphasis on different approaches and goals.

(1) **Neural modeling and decision theoretic approach.** In the late 1950's and early 1960's, researchers in machine learning area focused mostly on this approach. The goal of the approach is to develop general purpose learning systems which start with little knowledge. A learning system, according to this approach, is essentially the neural-network which consists of a set of inter-connected neuron-like elements. Each node in the network can perform simple computations such as logical threshold function, etc, and the links between nodes are attached with weights. The net learns by modifying the weights between the nodes. Such learning system is characterized by low levels of built-in knowledge and the use of continuously changeable parameters to achieve learning. Research in machine learning community in this area was in low ebb in the 1970's. However, the direction was pursued by people in engineering and science applications and resulted in learning systems in control and pattern recognition. In particular, this neural-modeling approach has gone through a resurrection in the 1980's and now is a very hot area of machine learning research.

(2) **Symbolic knowledge learning.** Starting from the late 1960's and early 1970's, researchers in machine learning shifted focus to learning symbolic knowledge by using symbolic inferencing techniques. This approach aims at learning symbolic representations of (domain specific) knowledge starting with sufficient amount of knowledge. This approach is characterized by its symbolic representation of knowledge to be learned and symbolic inference rules used to achieve the learning, as well as the rich amount of built-in knowledge in the learning system.

(3) **Formal, computational approach to inductive learning: PAC-learning.** This was developed since the early 80's. The approach is concerned with learning formal languages, recursive functions, boolean functions by query an external oracle. An important feature of this approach is that it is based upon a computational framework for inductive learning from examples. In this framework (proposed by Valiant in 1984), a class of concepts (or functions, languages) is learnable under certain queries, if there is a procedure, which, after making sufficient queries, will learn the concepts in the class with high probability, and the computation complexity of the procedure is polynomial in the size of the concepts and the parameters specifying the probability. A more recent development of this direction has lead to the research area Inductive Logic Programming (ILP) which focuses on learning (possibly recursive) logic programs from examples (and queries to the Oracle).

(4) **Learning by genetic algorithms.** Genetic algorithms (GA) were developed since the mid 1960's. Genetic algorithms are search algorithms that patterned after natural evolution. The search in GA is based on the mechanics of natural selection and natural genetics. GAs combine "survival of the fittest" among individual strings with a structured, randomized search in looking for good strings. The more broad area of "evolutionary computing" includes GA, genetic programming, and other topics.

(5) **Instance-based learning.** Instance-based learning methods do NOT construct explicit representation of hypotheses as approximations to the target function to be learned. Instead, the learning methods simply store the given training data, and classify a new test example based on the nearest neighbors of the test example in the stored training data. Most well-known instance-based learning algorithm is the K-Nearest Neighbor algorithm.

Instance-based learning is also closely related to case-based reasoning.

(6) **Analytical learning.** Analytical learning was developed in the early 1980's. In analytical learning, prior knowledge and deductive reasoning are used in combination with information from the training examples, in order to achieve more efficient and effective learning. A representative analytical learning method is called *Explanation-Based Learning (EBL).*

(7) **Bayesian learning.** Bayesian learning methods are probabilistic approaches to learning, typically utilizing the Bayes Theorem in someway. Some Bayesian learning algorithms calculate explicit probabilities for hypotheses, whereas other Bayes learning algorithms involve estimation of local probability distributions. A representative Bayesian learning method is the Naive Bayes Classifier. Bayes Belief Networks learning (and related probabilistic relational model learning) has become an important topic of study in recent years.

(8) **Reinforcement learning.** Reinforcement learning focuses on learning control strategies of autonomous agents, which get sensory signals from the world (and thus observe the current world state), form a model of the world, and perform actions to change the world/state. The training information is in the form of a reinforcement signal (real-valued) associated with each state-action transition. The objective of the agent is to learn an optimal control policy that tells the agent which action to take in each state, in order to get maximal reward from any starting state.

Reinforcement learning has been studied actively since the 1980's.

(9) **Integrated approach to machine learning.** The current trend in machine learning research seems toward an integration of various learning techniques (induction, deduction, analogy, adaptive, etc) to perform learning in more sophisticated task domains and more complicated knowledge.

A CLASSIFICATION OF LEARNING ALGORITHMS

One way to classify the learning algorithms is by the type of inference performed by the learning element.

(1) **Rote learning.** This includes learning by memorization of given facts or by being programmed.

(2) **Learning from instruction.** The learning element acquires knowledge from a teacher or a text book. Some transformation of the input from the teacher (or text book) may be needed to absorb the knowledge.

(3) **Learning by deduction.** The learning element draws deductive inference from its knowledge and stores useful conclusions.

(4) **Learning by analogy.** The learning element acquires new facts or rules by transforming and modifying existing knowledge (which bears some similarity to the desired new rules) into a form useful for the new situation.

(5) **Learning by induction.** The learning system acquires new rules by inductive inference from examples. This class of algorithms can be further classified as learning from examples and learning by observation and discovery.

(6) **Adaptive learning.** The learning systems learns by adjusting numerical parameters in a control function or in the neural-network.

Inductive Learning - Learning from Examples

A learning system is said to perform inductive learning if it acquires new knowledge by inductive inference from the facts. There are mainly two types of inductive learning: learning from examples and learning from from observation and discovery, where the former can be viewed as having a teacher to provide it with positive and negative examples of the concepts to be learned, and the latter can be viewed as having no teachers to provide the connection between examples and concepts, and it learns the target concepts by unsupervised learning from observations of the environment. This difference between learning from examples and from observation and discovery is usually minor, and we can in some sense view the observations as positive examples of some concept provided by the environment. Therefore, we will treat the issue of inductive learning the same as the issue of learning from examples.

Learning as information transformation. A system that learns from examples takes as input some specific instances of certain general rules or concepts, and then it performs induction on these instances to obtain an approximation of the general rules or concepts. The general rules are more useful information which can be used to solve a wider range of problems. The process of induction mainly uses generalization. The input to the learning system is low-level information in the form of examples, and the result of learning should be high-level information in the form of general rules or concepts. The process of learning can be viewed as a process of extracting high-level information from examples, which is a kind of information transformation aimed at more useful information.

Learning as search in the hypothesis space and instance space. The problem of learning from examples can be viewed as the problem of using instances from a space of possible instances to guide the search for hypotheses (which may be in the form of rules). The set of all possible examples is called the *instance space* and the set of all possible hypotheses is called the *hypothesis space*. The two-space model of learning from examples is depicted in Fig. 2. The learning system can be viewed as moving back and forth between these two spaces in the process of learning. For example, assume that the system is trying to learn the concept of a bird. The instance space may consist of all instance of animals, some of which are birds and some are not, the hypothesis space may consist of all recognition rules which assert that the object is a bird when certain features of the object are observed. The learning system gets (or actively selects) some examples from the instance space and then uses the information in the instances to search the hypothesis space, excluding some of the candidate hypotheses from the hypothesis space. To further reduce

the range of candidate hypotheses, the system may go back to search the instance space for some suitable examples to eliminate the irrelevant rules.

Experiment Planning
Instance Selection

Instance Space Hypothesis Space

Interpretation

Fig. 2. The two space model of learning from examples.

Instance space. The two issues with the instance space are (1) The quality of the instances, i.e., whether the instances contain noise or false information, whether the instances are representative or not, and whether the order in which the instances are presented to the learner is suitable; (2) The efficient algorithms to search the instance space.

Hypothesis space. There are two main issues concerning the hypothesis space: (1) What is the hypothesis space? In other words, what are the vocabularies (terms and operators) in the language with which we represent all the possible hypotheses? (2) How can the space be searched efficiently? In designing a learning system, we need to choose a hypothesis space which contains the desired hypotheses and is easy to search. In terms of representation language of the hypothesis space, we need to choose a language which is rich enough to represent our target hypothesis/rules and yet it is constrained enough to make the search easier.

Syntactic rules of inference. Both the expressiveness of a representation and the ease of searching the hypothesis space depend on the kind and complexity of the inference rules supported by the representation. An inference rule maps one hypothesis into another hypothesis in the hypothesis space, and the search of the hypothesis space is guided by the inference rules. In general, a language with many features (operators) allows more inference rules. Such a language is expressive but the search in the hypothesis space may be difficult, due to large size of the hypothesis space. On the other hand, a more restrictive language allows fewer inference rules thus makes the search easier, but the gain in ease of search is obtained at the price of less representation power. Obviously, there is a trade-off between the expressiveness of the language and the ease of searching the hypothesis space, which we must resolve when we choose the representation and the inference rules.

In learning from examples, the most important inference is generalization. Therefore, one should choose a representation for the hypothesis space in which generalization can be performed by inexpensive syntactical operations. Predicate calculus is pretty suitable for certain kind of syntactical generalizations. The most often used syntactical rules of generalization are given below, note that these rules are not truth-preserving:

(1) Turning constants to variables. For example, from human(Socrates) $\rightarrow$ mortal(Socrates), we can get $\forall$x[human(x) $\rightarrow$ mortal(x)].

(2) Dropping conditions. For example, from important(x) $\wedge$ interesting(x) $\rightarrow$ worth_doing(x), we can get interesting(x) $\rightarrow$ worth_doing(x).

(3) Adding options. For example, from intelligent(x) $\rightarrow$ happy(x), we may get intelligent(x) $\vee$ pretty(x) $\vee$ rich(x) $\rightarrow$ happy(x).

The single representation trick. As we already see, if the representations of the hypothesis space and the instance space are very different, the learning system has to go through the process of

transformations back and forth between the two representations, which complicates the learning process. One way out of this difficulty is to use the single representation trick: choose the same representation for both spaces. Therefore, training instances are viewed as specific knowledge acquired directly from the environment. The advantage of this single representation approach is to allow searching the two spaces to be performed by simple syntactical operations.

Methods to search the hypothesis space. There are several ways to search the hypothesis space. In general, those methods maintain a set of hypotheses H and reduce the set H to find the desired target rule(s).

Data-driven methods. In data-driven search, the presentation of instances drives the searching process. Two types of data-driven search methods are: the version space approach, and the rule refinement approach.

Model-driven methods. In model-driven search, the models given a priori guide the search. Two types model-driven search methods are: generate and test, and schema instantiation.