Analytical Learning:

Inductive learning and analytical learning are the two categories into which machine learning methods can be classified, according to Thrun (1995). **Inductive learning,** or discovery learning, is the process in **which learners find patterns and rules only by processing examples.**

One problem faced by inductive learning is a limited sample size. For example, if someone is trying to identify different types of plants in a forest by only observing a small area, they may miss certain species or make incorrect assumptions about the characteristics of those plants **based on limited data.**

Ex: Decision tree learning and neural networks etc.

Analytical learning comes to solve the problem of limited

data. How Does Analytical Learning Work?

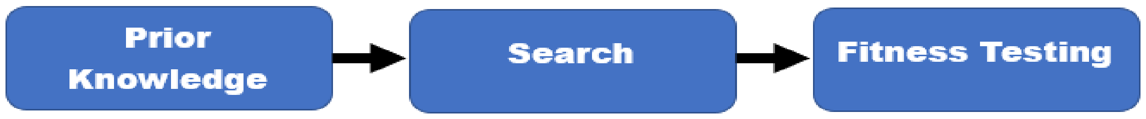
Analytical learning uses prior knowledge to guide the learning process

It uses prior knowledge to guide the machine learning process rather than relying solely on the data to determine the best model.

EX: Chess game

The Process of Analytical Learning:

Analytical learning requires models to fit both prior knowledge and data



1. We use prior knowledge to select the model family and features or make a problem-specific cost function.
2. The search process uses the available data to find a model.
3. When testing a model, we require that it fits both data and theory.

Once a satisfactory solution is found, it can be further refined and improved using additional data or feedback.

Analytical learning is particularly useful in situations where the available data is limited or noisy. It's also useful when there are complex relationships between features, and a purely inductive approach might not have enough power to reveal them without the help of prior knowledge.

Benefits of Analytical Learning:

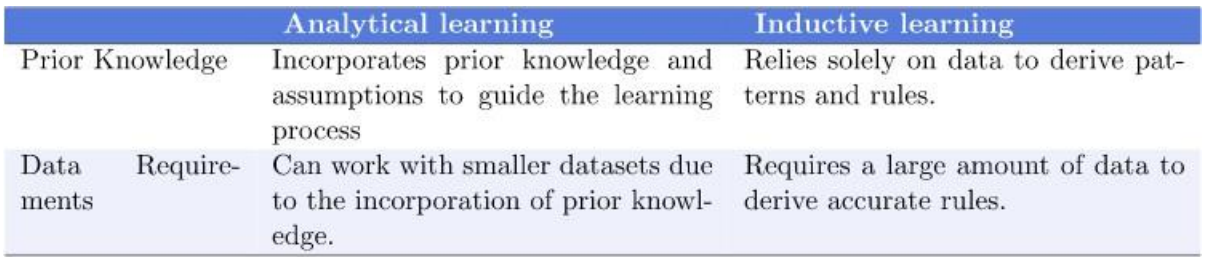
- Improved accuracy
- Faster learning:
- Better generalization
- Increased interpretability

Drawbacks of Analytical Learning:

- Assumption of prior knowledge
- Difficulty in handling complex relationships
- Requires expertise

Analytical vs. Inductive Learning:

The two main differences between analytical and inductive learning methods are the use of prior knowledge and data requirements:



	Analytical learning	Inductive learning
Prior Knowledge	Incorporates prior knowledge and assumptions to guide the learning process	Relies solely on data to derive patterns and rules.
Data Requirements	Can work with smaller datasets due to the incorporation of prior knowledge.	Requires a large amount of data to derive accurate rules.

Learning With Perfect Domain Theories: PROLOG-EBG:

Or

The explanation-based learning algorithm PROLOG-EBG:

Explanation-based learning from domain theories that are perfect, that is, domain theories that are correct and complete. A domain theory is said to be ***correct*** if each of its assertions is a truthful statement about the world. A domain theory is said to be ***complete*** with respect to a given target concept and instance space, if the domain theory covers every positive example in the instance space.

Put another way, it is complete if every instance that satisfies the target concept can be proven by the domain theory to satisfy it. Notice our definition of completeness does not require that the domain theory be able to prove that negative examples do not satisfy the target concept. However, if we follow the usual PROLOG convention that unprovable assertions are assumed to be false, then this definition of completeness includes full coverage of both positive and negative examples by the domain theory.

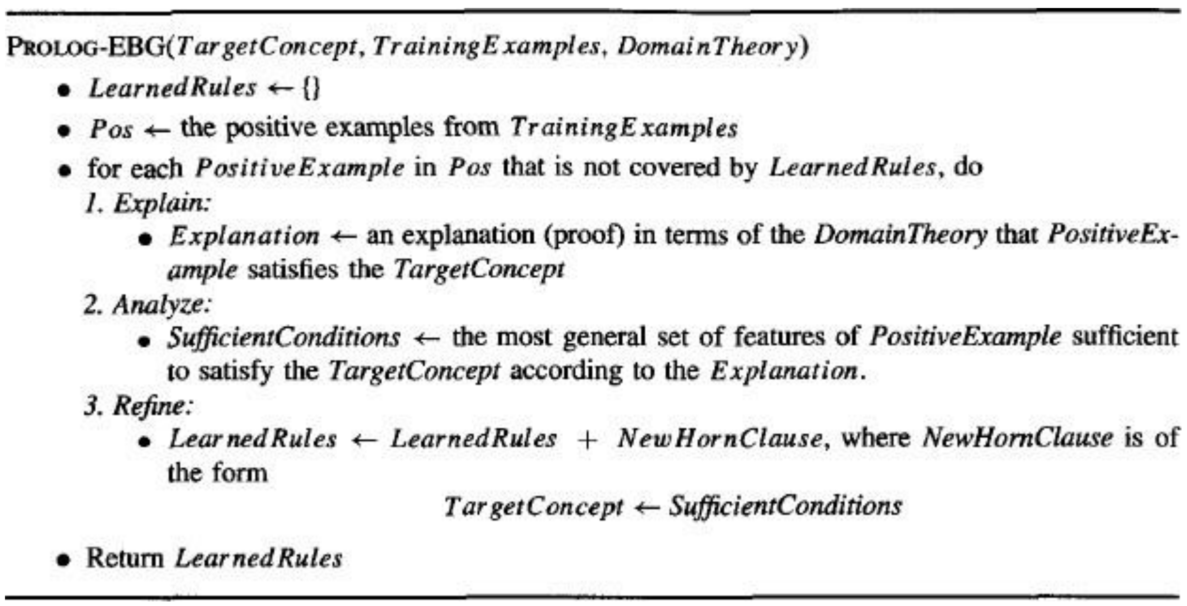
The reader may well ask at this point whether it is reasonable to assume that such perfect domain theories are available to the learner. After all, if the learner had a perfect domain theory, why would it need to learn? There are two responses to this question.

1. First, there are cases in which it is feasible to provide a perfect domain theory.

2. Second, in many other cases it is unreasonable to assume that a perfect domain theory is available.

PROLOG-EBG is representative of several explanation-based learning algorithms. PROLOG-EBG is a sequential covering algorithm . In other words, it operates by learning a single Horn clause rule, removing the positive training examples covered by this rule, then iterating this process on the remaining positive examples until no further positive examples remain uncovered. When given a complete and correct domain theory, PROLOG-EBG is guaranteed to output a hypothesis (set of rules) that is itself correct and that covers the observed positive training examples. For any set of training examples, the hypothesis output by PROLOG-EBG constitutes a set of logically sufficient conditions for the target concept, according to the domain theory. PROLOG-EBG is a refinement of the EBG algorithm introduced by Mitchell et al. (1986) and is similar to the EGGS algorithm described by DeJong and Mooney (1986).

The PROLOG-EBG algorithm is :



```
PROLOG-EBG(TargetConcept, TrainingExamples, DomainTheory)
    • LearnedRules ← {}
    • Pos ← the positive examples from TrainingExamples
    • for each PositiveExample in Pos that is not covered by LearnedRules, do
      1. Explain:
          • Explanation ← an explanation (proof) in terms of the DomainTheory that PositiveExample satisfies the TargetConcept
      2. Analyze:
          • SufficientConditions ← the most general set of features of PositiveExample sufficient to satisfy the TargetConcept according to the Explanation.
      3. Refine:
          • LearnedRules ← LearnedRules + NewHornClause, where NewHornClause is of the form
                TargetConcept ← SufficientConditions
    • Return LearnedRules
```

The explanation-based learning algorithm PROLOG-EBG. For each positive example that is not yet covered by the set of learned Horn clauses ***(LearnedRules),*** a new Horn clause is created.

This new Horn clause is created by
(1) explaining the training example in terms of the domain theory,
(2) analyzing this explanation to determine the relevant features of the example, then
(3) constructing a new Horn clause that concludes the target concept when this set of features is satisfied.

Remarks On Explanation-Based Learning:

PROLOG-EBG conducts a detailed analysis of individual training examples to determine how best to generalize from the specific example to a general Horn clause hypothesis. The following are the key properties of this algorithm.

- Unlike inductive methods, PROLOG-EBG produces **justified** general hypotheses by using prior knowledge to analyze individual examples
- The explanation of how the example satisfies the target concept determines which example attributes are relevant: those mentioned by the explanation.
- The further analysis of the explanation, regressing the target concept to determine its weakest preimage with respect to the explanation, allows deriving more general constraints on the values of the relevant features.
- Each learned Horn clause corresponds to a sufficient condition for satisfying the target concept. The set of learned Horn clauses covers the positive training examples encountered by the learner, as well as other instances that share the same explanations.
- The generality of the learned Horn clauses will depend on the formulation of the domain theory and on the sequence in which training examples are considered.
- PROLOG-EBG implicitly assumes that the domain theory is correct and complete.If the domain theory is incorrect or incomplete, the resulting learned concept may also be incorrect.

There are several related perspectives on explanation-based learning that help to understand its capabilities and limitations.

- EBL as theory-guided generalization of examples.
- EBL as example-guided reformulation of theories.
- EBL as "just" restating what the learner already "knows”.

Explanation-Based Learning Of Search Control Knowledge:

Practical applicability of the PROLOG-EBG algorithm is restricted by its requirement that the domain theory be correct and complete. This type id learning focused on the speeding of complicated search programs.In fact, the largest scale attempts to apply explanation-based learning have addressed the problem of learning to control search, or what is sometimes called "speedup" learning. For example, playing games such as chess involves searching through a vast space of possible moves and board positions to find the best move. Many practical scheduling and optimization problems are easily formulated as large search problems, in which the task is to find some move toward the goal state. In such problems the definitions of the legal search operators, together with the definition of the search objective, provide a complete and correct domain theory for learning search control knowledge.

Consider a general search problem where S is the set of possible search states, 0 is a set of legal search operators that transform one search state into another, and G is a predicate defined over S that indicates which states are goal states. The problem in general is to find a sequence of operators that will transform an arbitrary initial state si to some final state sf that satisfies the goal predicate G. One way to formulate the learning problem is to have our system learn a separate target concept for each of the operators in 0. In particular, for each operator o in 0 it might attempt to learn the target concept "the set of states for which o leads toward a goal state." Of course the exact choice of which target concepts to learn depends on the internal structure of problem solver that must use this learned knowledge. For example, if the problem solver is a means-ends planning system that works by establishing and solving sub goals, then we might instead wish to learn target concepts such as "the set of planning states in which sub goals of type A should be solved before sub goals of type B."

One system that employs explanation-based learning to improve its search is PRODIGY. PRODIGY is a domain-independent planning system that accepts the definition of a problem domain in terms of the state space S and operators 0.

The use of explanation-based learning to acquire control knowledge for PRODIGY has been demonstrated in a variety of problem domains including the simple block-stacking problem.

A second example of a general problem-solving architecture that incorporates a form of explanation-based learning is the SOAR system.

PRODIGY and SOAR demonstrate that explanation-based learning methods can be successfully applied to acquire search control knowledge in a variety of problem domains.

USING PRIOR KNOWLEDGE TO ALTER THE SEARCH OBJECTIVE:

Combining Inductive And Analytical Learning:

The main two paradigms for machine learning:

1. Inductive learning and
2. Analytical learning.

1. Inductive learning:

Inductive methods, such as decision tree induction and neural network BACKPROPAGATION and seek general hypotheses that fit the observed training data. Analytical methods, such as PROLOG-EBG, seek general hypotheses that fit prior knowledge while covering the observed data. These two learning paradigms are based on fundamentally different justifications for learned hypotheses and offer complementary advantages and disadvantages. Combining them offers the possibility of more powerful learning methods.

Purely analytical learning methods offer the advantage of generalizing more accurately from less data by using prior knowledge to guide learning. However, they can be misled when given incorrect or insufficient prior knowledge. Purely inductive methods offer the advantage that they require no explicit prior knowledge and learn regularities based solely on the training data. However, they can fail when given insufficient training data, and can be misled by the implicit inductive bias they must adopt in order to generalize beyond the observed data.

2. Analytical learning:

It is known that analytical methods provide logically justified hypotheses and inductive methods provide statistically justified hypotheses, it is easy to see why combining them would be useful: Logical justifications are only as compelling as the assumptions, or prior knowledge, on which they are built. They are suspect or powerless if prior knowledge is incorrect or unavailable. Statistical justifications are only as compelling as the data and statistical assumptions on which they rest. They are suspect or powerless when assumptions about the underlying distributions cannot be trusted or when data is scarce. In short, the two approaches work well for different types of problems. By combining them we can hope to devise a more general learning approach that covers a more broad range of learning tasks.

The difference between inductive and analytical learning methods can be seen in the nature of the ***justiJications*** that can be given for their learned hypotheses. Hypotheses output by purely analytical learning methods such as **PROLOGEBG** carry a ***logical*** justification; the output hypothesis follows deductively from the domain theory and training examples. Hypotheses output by purely inductive learning methods such as **BACKPROPAGAT**c**I**a**O**r**N**ry a ***statistical*** justification; the output hypothesis follows from statistical arguments that the training sample is sufficiently large that it is probably representative of the underlying distribution of examples.

Table summarizes a spectrum of learning problems that varies by the availability of prior knowledge and training data.

	Inductive learning	Analytical learning
Goal:	Hypothesis fits data	Hypothesis fits domain theory
Justification:	Statistical inference	Deductive inference
Advantages:	Requires little prior knowledge	Learns from scarce data
Pitfalls:	Scarce data, incorrect bias	Imperfect domain theory

Table:Comparison of purely analytical and purely inductive learning.

At one extreme, a large volume of training data is available, but no prior knowledge. At the other extreme, strong prior knowledge is available, but little training data. Most practical learning problems lie somewhere between these two extremes of the spectrum.

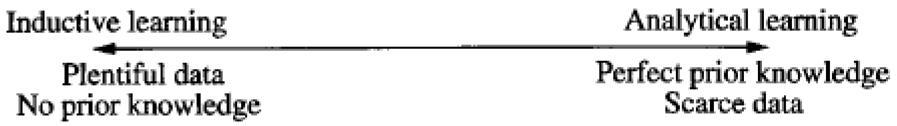


Fig:A spectrum of learning tasks

Some specific properties we would like from such a learning method include:

- Given no domain theory, it should learn at least as effectively as purely inductive methods.
- Given a perfect domain theory, it should learn at least as effectively as purely analytical methods.
- Given an imperfect domain theory and imperfect training data, it should combine the two to outperform either purely inductive or purely analytical methods.
- It should accommodate an unknown level of error in the training data.
- It should accommodate an unknown level of error in the domain theory.

Inductive-Analytical Approaches To Learning:

There are two types of approaches for inductive-analytical learning.They are

1. **The Learning Problem**
2. **Hypothesis Space Search**

1. The Learning Problem:

The learning problem consists of the following.

Given:

- A set of training examples D, possibly containing errors
- A domain theory B, possibly containing errors
- A space of candidate hypotheses H

Determine:

- A hypothesis that best fits the training examples and domain theory

Let us define the error *error_B (h) of h* with respect to a domain theory B to be the probability that *h* will disagree with B on the classification of a randomly drawn instance. We can attempt to characterize the desired output hypothesis in terms of these errors. For example, we could require the hypothesis that minimizes some combined measure of these errors, such as

$$\underset{h \in H}{\operatorname{argmin}} \; k_D error_D(h) + k_B error_B(h)$$

While this appears reasonable at first glance, it is not clear what values to assign to k_D and k_B to specify the relative importance of fitting the data versus fitting the theory. If we have a very poor theory and a great deal of reliable data, it will be best to weight $error_D(h)$ *m*ore heavily.

An alternative perspective on the question of how to weight prior knowledge and data is the Bayesian perspective. Bayes theorem computes this posterior probability based on the observed data ***D,*** together with prior knowledge in the form of ***P(h), P(D),*** and ***P(Dlh).*** Thus we can think of ***P(h), P(D),*** and ***P(Dlh)*** as a form of background knowledge or domain theory, and we can think of Bayes theorem as a method for weighting this domain theory, together with the observed data ***D,*** to assign a posterior probability ***P(hlD)*** to ***h.*** The Bayesian view is that one should simply choose the hypothesis whose posterior probability is greatest, and that Bayes theorem provides the proper method for weighting the contribution of this prior knowledge and observed data.

2. Hypothesis Space Search:

One way to understand the range of possible approaches is to return to our view of learning as a task of searching through the space of alternative hypotheses. We can characterize most learning methods as search algorithms by describing the hypothesis space H they search, the initial hypothesis ***ho*** at which they begin their search, the set of search operators ***0*** that define individual search steps, and the goal criterion G that specifies the search objective.

1. Use prior knowledge to derive an initial hypothesis from which to begin the search:

In this approach the domain theory B is used to construct an initial hypothesis ***ho*** that is consistent with
B. A standard inductive method is then applied, starting with the initial hypothesis ***ho.*** For example, the **KBANN** system described below learns artificial neural networks in this way. It uses prior knowledge to design the interconnections and weights for an initial network, so that this initial network is perfectly consistent with the given domain theory. This initial network hypothesis is then refined inductively using the BACKPROPAGATaIlOgNor ithm and available data. Beginning the search at a hypothesis consistent with the domain theory makes it more likely that the final output hypothesis will better fit this theory.

2. Use prior knowledge to alter the objective of the hypothesis space search:

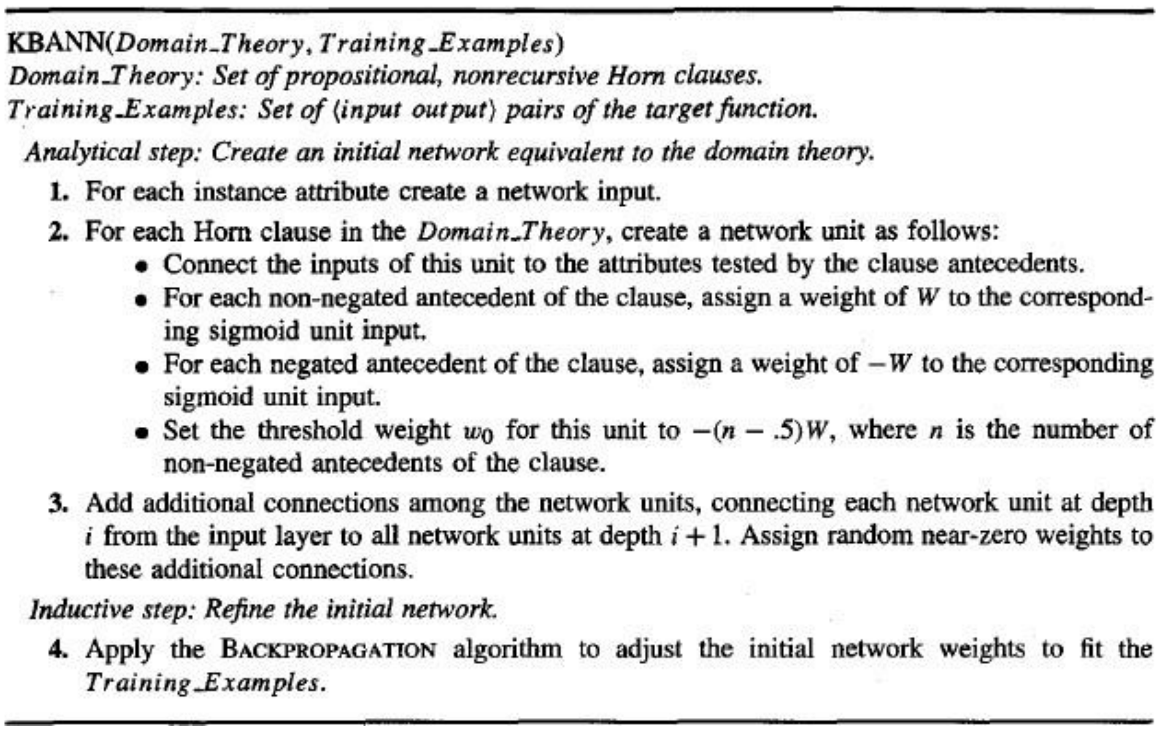
In this approach, the goal criterion G is modified to require that the output hypothesis fits the domain theory as well as the training examples. For example, the EBNN system described below learns neural networks in this way. Whereas inductive learning of neural networks performs gradient descent search to minimize the squared error of the network over the training data, EBNN performs gradient descent to optimize a different criterion. This modified criterion includes an additional term that measures the error of the learned network relative to the domain theory.

3. Use prior knowledge to alter the available search steps:

In this approach, the set of search operators ***0*** is altered by the domain theory. For example, the FOCL system described below learns sets of Horn clauses in this way. It is based on the inductive system FOIL, which conducts a greedy search through the space of possible Horn clauses, at each step revising its current hypothesis by adding a single new literal. FOCL uses the domain theory to expand the set of alternatives available when revising the hypothesis, allowing the addition of multiple literals in a single search step when warranted by the domain theory.

Using Prior Knowledge To Initialize The Hypothesis(KBANN Algorithm):
One approach to using prior knowledge is to initialize the hypothesis to perfectly fit the domain theory, then inductively refine this initial hypothesis as needed to fit the training data. This approach is used by the KBANN (Knowledge-Based Artificial Neural Network) algorithm to learn artificial neural networks. In KBANN an initial network is first constructed so that for every possible instance, the classification assigned by the network is identical to that assigned by the domain theory.

The KBANN algorithm exemplifies the initialize-the-hypothesis approach to using domain theories. It assumes a domain theory represented by a set of propositional, nonrecursive Horn clauses. **A** Horn clause is propositional if it contains no variables. The input and output of KBANN are as follows:



```
KBANN(Domain_Theory, Training_Examples)
Domain_Theory: Set of propositional, nonrecursive Horn clauses.
Training_Examples: Set of ⟨input output⟩ pairs of the target function.
  Analytical step: Create an initial network equivalent to the domain theory.
    1. For each instance attribute create a network input.
    2. For each Horn clause in the Domain_Theory, create a network unit as follows:
         • Connect the inputs of this unit to the attributes tested by the clause antecedents.
         • For each non-negated antecedent of the clause, assign a weight of $W$ to the corresponding sigmoid unit input.
         • For each negated antecedent of the clause, assign a weight of $-W$ to the corresponding sigmoid unit input.
         • Set the threshold weight $w_0$ for this unit to $-(n - .5)W$, where $n$ is the number of non-negated antecedents of the clause.
    3. Add additional connections among the network units, connecting each network unit at depth $i$ from the input layer to all network units at depth $i + 1$. Assign random near-zero weights to these additional connections.
  Inductive step: Refine the initial network.
    4. Apply the BACKPROPAGATION algorithm to adjust the initial network weights to fit the Training_Examples.
```

Fig: The KBANN algorithm.

Given:
- A set of training examples
- A domain theory consisting of nonrecursive, propositional Horn clauses

Determine:

- An artificial neural network that fits the training examples, biased by the domain theory.

The two stages of the KBANN algorithm are first to create an artificial neural network that perfectly fits the domain theory and second to use the BACKPROPACATION algorithm to refine this initial network to fit the training examples.

Remarks of KBANN Algorithm:
KBANN analytically creates a network equivalent to the given domain theory, then inductively refines this initial hypothesis to better fit the training data. In doing so, it modifies the network weights as needed to overcome inconsistencies between the domain theory and observed data.

The chief benefit of KBANN over purely inductive BACKPROPAGATION is that it typically generalizes more accurately than BACKPROPAGATION given an approximately correct domain theory, especially when training data is scarce. KBANN and other initialize-the-hypothesis approaches have been demonstrated to outperform purely inductive systems in several practical problems.

Horn clauses extracted from the final trained network provided a refined domain theory that better fit the observed data. Although it is sometimes possible to map from the learned network weights back to a refined set of Horn clauses, in the general case thisis problematic because some weight settings have no direct Horn clause analog. Craven and Shavlik (1994) and Craven (1996) describe alternative methods forextracting symbolic rules from learned networks.

Limitations of KBANN include the fact that it can accommodate only propositional domain theories; that is, collections of variable-free Horn clauses. It is also possible for KBANN to be misled when given highly inaccurate domain theories, so that its generalization accuracy can deteriorate below the level of BACKPROPAGATION. Nevertheless, it and related algorithms have been shown to be useful for several practical problems.

KBANN illustrates the initialize-the-hypothesis approach to combining analytical and inductive learning. Other examples of this approach include Fu (1993); Gallant (1988); Bradshaw et al. (1989); Yang and Bhargava (1990); Lacher et al. (1991). These approaches vary in the exact technique for constructing the initial network, the application of **BACKPROPAGAT**ION weight tuning, and in methods for extracting symbolic descriptions from the refined network. Pratt (1993a, 1993b) describes an initialize-the- hypothesis approach in which the prior knowledge is provided by a previously learned neural network for a related task, rather than a manually provided symbolic domain theory.