Learning algorithms like neural networks, decision trees, inductive logic programming, etc. all require a good number of examples to be able to do good predictions.

Dataset must be sufficiently large
We don’t need a large dataset if besides taking examples as input, the learning algorithm can take prior knowledge.

Dataset does not need to be large
Prior knowledge is used to reduce the size of the hypothesis space.

It analyzes each example to infer which features are relevant and which ones are irrelevant.
Suppose we want to learn a concept like “what is a board position in which black will lose the queen in x moves?”.

Chess is a complex game. Each piece can occupy many positions. We would need many examples to learn this concept.

But humans can learn these type of concepts with very few examples. Why?
Humans can analyze an example and use prior knowledge related to legal moves.

From there it can generalize with only few examples.

“Because the white king is attacking both king and queen; black must avoid check, letting white capture the queen”
What is the prior knowledge involved in playing chess?

It is knowledge about the rules of chess:

- Legal moves for the pieces.
- Players alternate moves in games.
- To win you must capture the opponent’s king.
Inductive Learning

Input: HS, D,
Output: hypothesis h
h is consistent with D

Analytical Learning

Input: HS, D, B
Output: hypothesis h
h is consistent with D and B
( B ⊬ ¬h )

HS: Hypothesis Space
D: Training Set
B: Background knowledge (domain theory)
Example, Analytical Learning

Given:

- Dataset where each instance is a pair of objects represented by the following predicates: Color, Volume, Owner, Material, Density, On.
- SafeToStack(x,y) is the target concept
Example, Analytical Learning

- **Hypothesis space**: set of Horn clause rules.
  - The head of each rule has the predicate SafeToStack.
  - The body of each rule is based on the instances and the predicates LessThan, Equal, Greater and plus, minus and times.
  - Example:
    \[
    \text{SafeToStack}(x,y) \iff \text{Volume}(x,vx) \land \text{Volume}(y,vy) \land \text{LessThan}(vx,vy)
    \]

- **Training examples**:
  - On(Obj1,Obj2) \quad \text{Owner(Obj1,Fred)}
  - Type(Obj1,Box) \quad \text{Owner(Obj2,Louise)}
  - Type(Obj2,Endtable) \quad \text{Density(Obj1,0.3)}
  - Color(Obj1,Red) \quad \text{Material(Obj1,Cardboard)}
  - Color(Obj2,Blue) \quad \text{Material(Obj2,Wood)}
  - Volume(Obj1,2)
Example, Analytical Learning

Domain theory

SafeToStack(x,y) \iff \neg\text{Fragile}(y)

SafeToStack(x,y) \iff \text{Lighter}(x,y)

\text{Lighter}(x,y) \iff \text{Weight}(x,wx) \land \text{Weight}(y,wy) \land \text{LessThan}(wx,wy)

... 

\text{Fragile}(x) \iff \text{Material}(x,\text{Glass})

Determine:

Hypothesis consistent with both the training examples and the domain theory

Note:

The domain theory refers to predicates not contained in the examples.
The domain theory is sufficient to prove the example is true.
A domain theory is **correct** if each statement is true.

A domain theory is **complete** if it covers every positive example of the instance space (w.r.t a target concept and instance space).

A **perfect** domain theory is correct and complete.
Examples of where to find perfect domain theories:

*Rules of chess*

Examples of where not to find perfect domain theories:

*SafetoStack problem*

In this chapter, only learning problems with perfect domain theories is considered.
We consider an algorithm that has the following properties:

✓ It is a sequential covering algorithm considering the data incrementally

✓ For each positive example not covered by the current rules it forms a new rule by:
  
  - Explain how training example satisfies target concept, in terms of domain theory.
  - Analyze the explanation to find a the most general conditions under which this explanation holds.
  - Refine the current hypothesis by adding a new Horn Clause rule to cover the example.
**PROLOG-EBG**

- Learning a single Horn clause rule
- Remove the positive training examples covered by this rule

Until no further positive examples remain uncovered

The explanation is a proof that the example belongs to the target (if the theory is perfect)
Explanation:

Safe-to-Stack(OBJ1,OBJ2) → Lighter(OBJ1,OBJ2) →

Weight(OBJ1, 0.6) → Volume(OBJ1, 2) → Density(OBJ1, 0.3) → Equal(0.5, 2*0.3) → Less-Than(0.6, 5) → Type(OBJ2, ENDTABLE)

Weight(OBJ2, 5) →

Training Example:

OBJ1

Volume 0.3
Density 0.3
Material
Color
Owner
Box

OBJ2

Wood
Material
Fragile
Type
Color
Owner
Box

EndTable
There might be more than one explanation to the example. In that case one or all explanations may be used.

An explanation is obtained using a backward chaining search as is done by PROLOG. PROLOG-EBG stops when it finds the first proof.
Many features appear in an example. Of them, how many are truly relevant?

We consider as relevant those features that show in the explanation.

Example:
- Relevant feature: Density
- Irrelevant feature: Owner
Taking the leaf nodes of the explanation and substituting variables x and y for Obj1, and Obj2:

\[ \text{SafeToStack}(x,y) \leftarrow \text{Volume}(x,2) \land \text{Density}(x,0.3) \land \text{Type}(y,\text{Endtable}) \]

- Remove features that are independent of x and y such as Equal(0.6,times(2,0.3)) and LessThan(0.6,5).
- The rule is now more general and can serve to explain other instances matching the rule.
Computing the weakest preimage of explanation
The current hypothesis is the set of Horn clauses that have been learned to this point.

By using sequential covering more rules are added, thus refining our hypothesis.

A new instance is negative if it is not covered by any rule.
Discovering New Features

The PROLOG-EBG system described can formulate new features that are not in the training examples.
Example: Volume * Density > 5
(derived from the domain theory)

Similar to features represented by hidden neurons in ANN.
Difference:

ANN is a statistical method that requires many training examples to obtain the hidden neuron features

PROLOG-EBG use an analytical process to obtain new features from analysis of single training examples
What is the inductive bias of explanation based learning?

The hypothesis h follows deductively from D and B

D: database  B: Background knowledge

Bias: Prefer small sets of maximally general Horn Clauses
Problem: learning to speed up search programs ("speedup learning")
Examples include: playing chess scheduling and optimization problems.
Problem formulation:

S: set of possible search states
O: set of legal operators (transform one state into another state)
G: predicate over S indicating the goal states
Prodigy is a planning system.
Input: state space $S$ and operators $O$.
Output: A sequence of operators that lead from the initial state to the final state.

Prodigy uses a **means-end planner**: we decompose goals into subgoals:
Example:

Goal: accommodate blocks to read “UNIVERSAL”


Question: what subgoal should be attacked first?

Answer is given by search control knowledge
Prodigy defines a set of target concepts to learn, e.g., which operator given the current state takes you to the goal state?

An example of a rule learned by Prodigy in the block-stacking problem is:

IF One subgoal to be solved is On(x,y) AND One subgoal to be solved is On(y,z) THEN Solve the subgoal On(y,z) before On(x,y)
The rationale behind the rule is that it would avoid a conflict when stacking blocks.

Prodigy learns by first encountering a conflict, then explaining the reason for the conflict and creating a rule like the one above.

Experiments show an improvement in efficiency by a factor of two to four.
The number of control rules that must be learned is very large.

If the control rules are many, much time will be spent looking for the best rule. Utility analysis is used to determine what rules to keep and what rules to forget.
Another problem with EBL is that it is sometimes difficult to create an explanation for the target concept.

For example, in chess, learning a concept like: “states for which operator A leads to a solution” The search here grows exponentially.
Different from inductive learning, analytical learning looks for a hypothesis that fit the background knowledge and covers the training examples.

Explanation based learning is one kind of analytical learning that divides into three steps:
- Explain the target value for the current example
- Analyze the explanation (generalize)
- Refine the hypothesis
Summary

- PROLOG-EBG constructs intermediate features after analyzing examples.
- Explanation based learning can be used to find search control rules.
- Depend on a perfect domain theory.