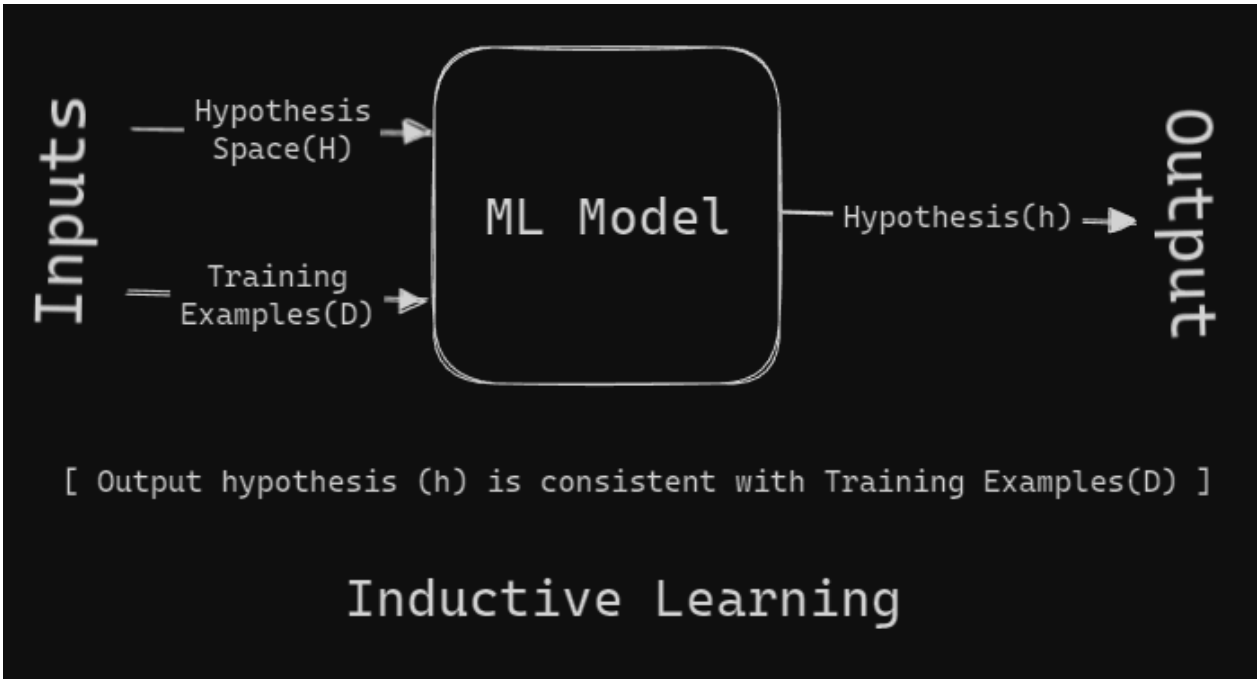
Part-1: Analytical Learning-1

Intro:

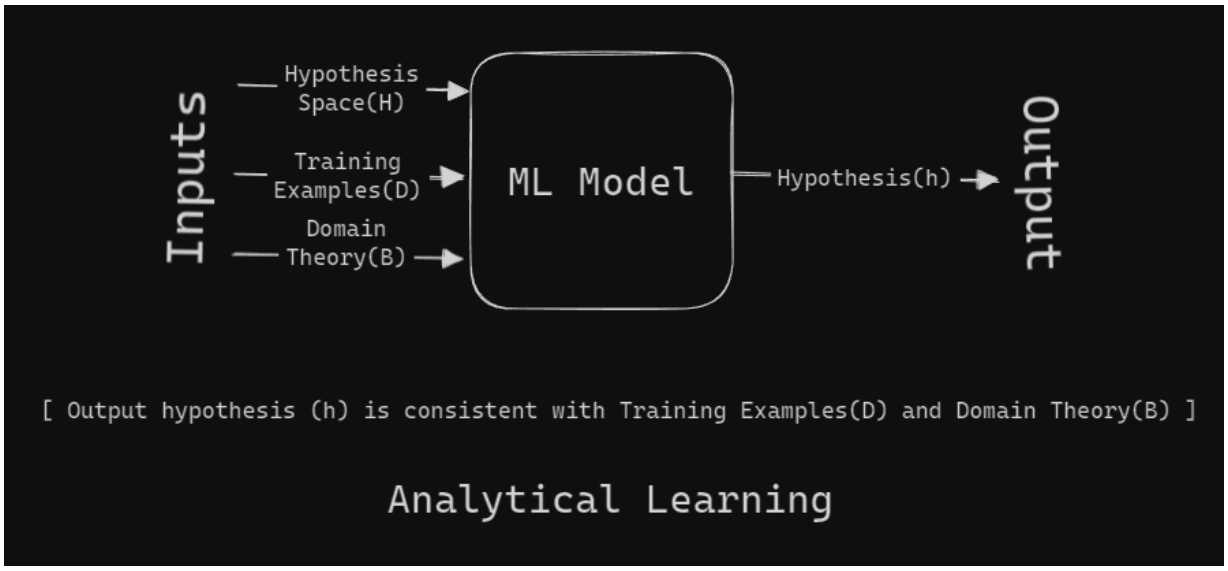
- Analytical Learning is similar to Inductive learning with a minor difference.



- In Inductive Learning, the Hypothesis Space(H) and Training Examples(D) are provided as Inputs to a learner. The Generated output of the learner will be a hypothesis(h) from the Hypothesis Space(H) which is consistent with the training examples(D).

Where, D = { X, f(X) }
X is Instance,
f(X) is the target value (ex: yes/no)

- Inductive Learning follows Statistical Inference, which is a method of making decisions about the parameters of a population, based on random sampling.



- Whereas is Analytical Learning, the Hypothesis Space(H), Training Examples(D) and Domain Theory(B) are provided as the inputs to a learner. The Generated output of the learner will be a hypothesis(h) from Hypothesis Space(H) which is consistent with the training examples(D) and Domain Theory(B).
- Here, Domain Theory(B) is just an additional Info about the training examples.

- Analytical Learning follows Deductive Inference, which is a reasoning process where conclusions are drawn logically from a set of premises or axioms.

Perfect Domain Theory:

- Domain Theory is formal representation of knowledge about a domain, including rules, constraints, and relationships. Domain theory provides the background knowledge necessary for deriving explanations.
- A perfect domain theory is correct and complete.
- A domain theory is **correct** if each of its assertions is a truthful statement about the world.
- A domain theory is **complete** wrt target concept and X, if it covers every positive example in the instance space.
- So, if we have a perfect domain theory, why do we need to learn?
- Chess. Often the theory leads to too many deductions (large breadth) making it impossible to find the optimal strategy. The examples help to focus search.
- Perfect domain theories are often unrealistic, but, learning in them is a first step before learning with imperfect theories.
- Prolog-EBG is an EBL learner. It uses sequential covering.

Prolog:

Prolog (short for **Programming in Logic**) is a high-level programming language primarily used for logic programming and artificial intelligence (AI) applications. It is based on formal logic and provides a powerful framework for solving problems involving symbolic reasoning, pattern matching, and knowledge representation.

Key Features of Prolog

1. **Logic Programming Paradigm**:
   - Prolog is designed around the principles of logic programming, where programs consist of logical statements and the computation is based on logical inference.
   - It uses a form of predicate logic to express facts and rules.
2. **Facts and Rules**:

   Facts: Basic assertions about the world. For example:

   ```
   parent(john, mary).
   ```

   - This fact states that John is a parent of Mary.

   Rules: Define relationships between facts. For example:

   ```
   grandparent(X, Z) :- parent(X, Y), parent(Y, Z).
   ```

   - This rule states that X is a grandparent of Z if X is a parent of Y and Y is a parent of Z.
3. **Queries**:

Prolog allows users to query the database of facts and rules to retrieve information or verify logical statements. For example:

```
?- grandparent(john, mary).
```

- This query checks if John is a grandparent of Mary.

EBG:

Explanation-Based Generalization (EBG) is a machine learning technique that leverages domain knowledge to improve the generalization of learning from specific examples. The core idea is to generalize from individual examples by using explanations based on a domain theory.

How EBG Works

1. **Start with a Specific Example**:
   - Begin with a concrete example that is known to be true or correct. This example is used to derive insights and generalizations.
2. **Generate an Explanation**:
   - Use the domain theory to generate an explanation for why the specific example fits into a particular category or has a particular outcome. This explanation is grounded in the domain knowledge.
3. **Create Generalizations**:
   - Derive general rules or patterns from the explanation of the specific example. These generalizations represent broader knowledge that can be applied to similar instances.
4. **Refine the Generalizations**:
   - Adjust and refine the generalized rules to ensure they are accurate and applicable across a range of examples, consistent with the domain theory.
5. **Apply the Generalizations**:
   - Use the refined generalizations to make predictions or decisions about new, unseen examples. The system applies the learned rules to new data based on the generalizations.

Example of EBG

Consider a domain where the goal is to classify animals as mammals or non-mammals. Suppose you have a specific example:

- **Example**: A dog is classified as a mammal.

The domain theory might include rules such as:

- Mammals are animals that have hair or fur and produce milk.

Process in EBG:

1. **Explanation**: A dog is a mammal because it has hair and produces milk, consistent with the domain theory.
2. **Generalization**: From this example, you can generalize that animals with hair or fur and that produce milk are mammals.
3. **Refinement**: Ensure that this generalization accurately applies to other examples, like cats or humans, and adjust the rule if necessary.

Generalized Rule:

- If an animal has hair or fur and produces milk, then it is a mammal.

Prolog-EBG:

Prolog-EBG (Explanation-Based Generalization in Prolog) is an implementation of the Explanation-Based Generalization (EBG) technique using the Prolog programming language. Prolog, a logic programming language, is well-suited for representing domain theories and performing logical reasoning, making it an effective platform for EBG.

Properties of Prolog-EBG:
- It follows the Sequential Covering Approach.
- It is a Deductive Learning System which assumes that the domain knowledge is correct and complete.
- It produces general hypotheses by using domain knowledge to analyze the individual examples.

Prolog-EBG Algorithm:

```
Prolog-EGB(TargetConcept, TraningExamples, DomainTheory)
    1. LearnedRules = {}
    2. Pos = the positive examples from TraningExamples.
    3. for each PositiveExample in Pos that is not covered by
       LearnedRules do
            1. Explanation = an explanation in terms of DomainTheory that
               Pos satisfies the TargetConcept.
            2. SufficientConditions = the most general set of features of
               PositiveExample sufficient to satisfy the TargetConcept
               according to the Explanation.
            3. LearnedRules = LearnedRules + {TargetConcept ←
               SufficientConditions}.
    4. return LearnedRules
```

Steps(6):

1. Define Domain Theory
2. Provide Specific Example
3. Generate Explanation
4. Derive General Rule
5. Refine General Rule
6. Apply General Rule

Example of Prolog-EBG

Consider a simple domain theory related to animals:

Domain Theory:

```
% Facts
has_hair(dog).
has_hair(cat).
produces_milk(dog).
produces_milk(cat).

% Rule
mammal(X) :- has_hair(X), produces_milk(X).
```

- **Specific Example**:
  - Example: A dog is a mammal.
- **Explanation**:
  - Explanation: The domain theory specifies that an animal is a mammal if it has hair and produces milk. A dog meets these criteria, so it is classified as a mammal.
- **Generalization**:

  The general rule derived from the explanation is:

  ```
  mammal(X) :- has_hair(X), produces_milk(X).
  ```

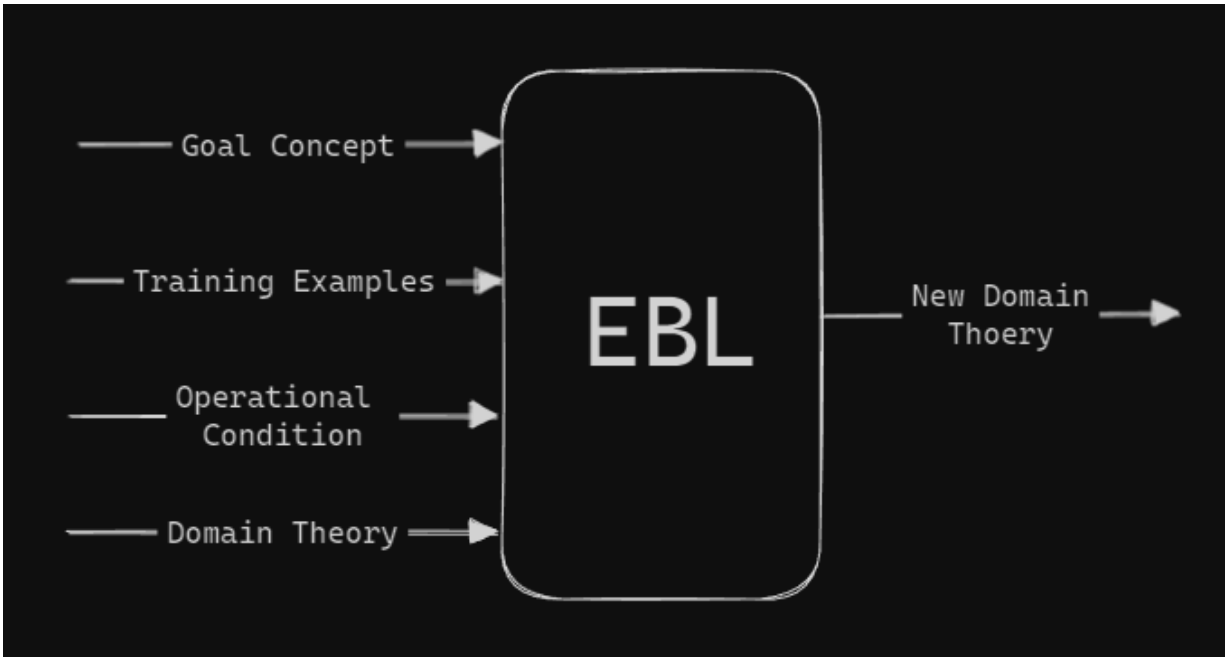
- **Refinement**:
  - Ensure the rule applies to other examples, such as cats, and adjust if necessary.
- **Application**:

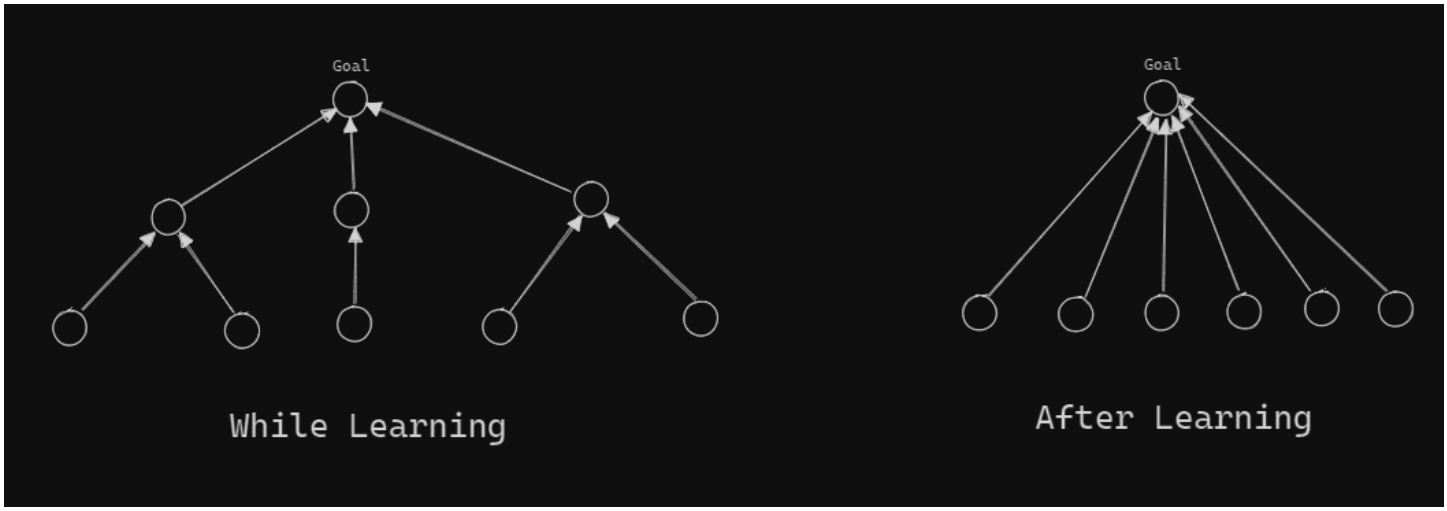
  Query: `?- mammal(cat).`

  - Result: `true`, indicating that a cat is classified as a mammal based on the generalized rule.

Explanation-Based Learning (EBL)

- **Definition**: Explanation-Based Learning is a general learning framework that uses domain knowledge to derive general rules from specific examples. It involves generating explanations for why specific examples fit into particular categories and then using these explanations to form broader generalizations.
- **Purpose**: EBL aims to improve the efficiency of learning by leveraging domain knowledge to generalize from a small number of examples. It helps in deriving more accurate and interpretable rules based on specific instances.
- **Process**:
  1. **Domain Theory**: Represents the background knowledge.
  2. **Specific Examples**: Concrete cases used for learning.
  3. **Explanation Generation**: Understanding why an example fits a certain category.
  4. **Generalization**: Deriving general rules from explanations.
  5. **Refinement and Application**: Validating and applying the learned rules.


Goal Concept
Training Examples
Operational Condition
Domain Theory
EBL
New Domain Thoery


Goal
While Learning
Goal
After Learning