Knowledge in Learning:

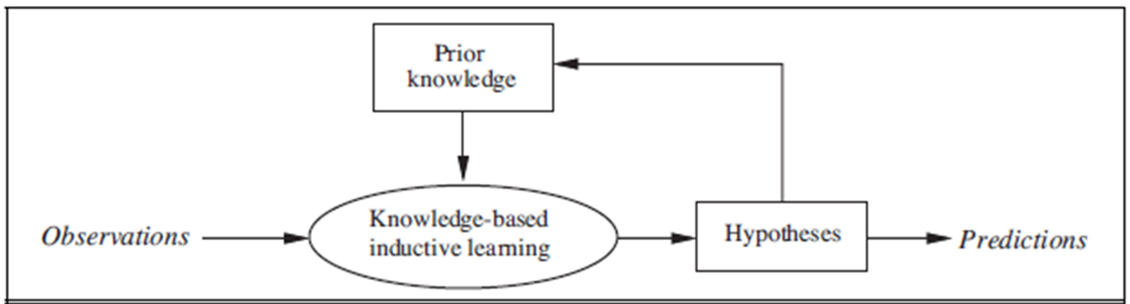


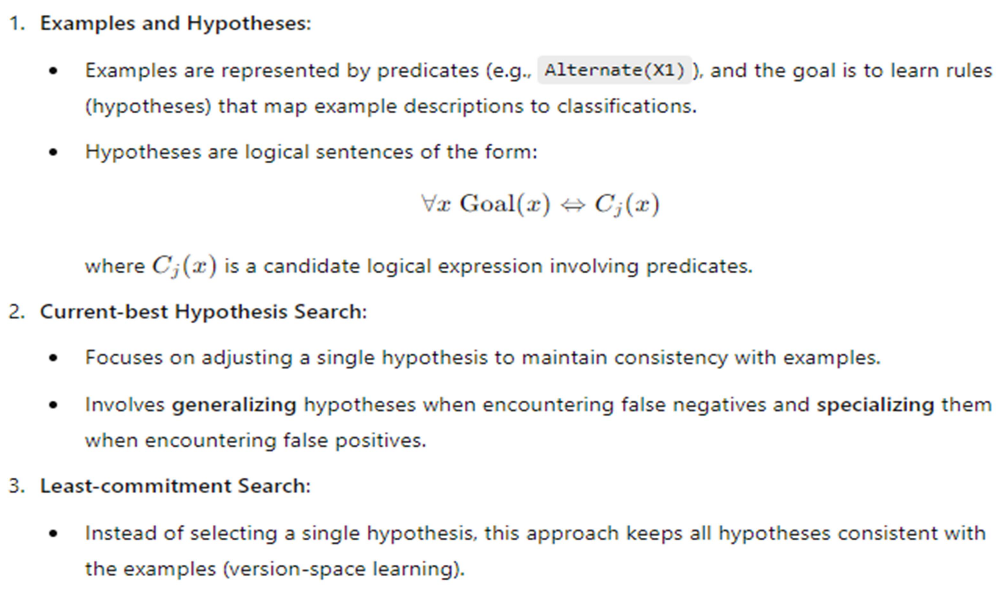
Figure 19.6 A cumulative learning process uses, and adds to, its stock of background knowledge over time.

19 Knowledge in Learning

This chapter explores how prior knowledge influences the process of learning and how leveraging that knowledge can reduce the complexity of learning tasks.

Key Subtopics:

- **Inductive Learning**: Searching a hypothesis space to find a function that maps inputs to outputs.
- **Role of Prior Knowledge**: The importance of background knowledge to improve the efficiency of learning, reducing the size of the hypothesis space.



1. **Examples and Hypotheses:**
   - Examples are represented by predicates (e.g., `Alternate(X1)`), and the goal is to learn rules (hypotheses) that map example descriptions to classifications.
   - Hypotheses are logical sentences of the form:

     $$\forall x \ \mathrm{Goal}(x) \Leftrightarrow C_j(x)$$

     where $C_j(x)$ is a candidate logical expression involving predicates.
2. **Current-best Hypothesis Search:**
   - Focuses on adjusting a single hypothesis to maintain consistency with examples.
   - Involves **generalizing** hypotheses when encountering false negatives and **specializing** them when encountering false positives.
3. **Least-commitment Search:**
   - Instead of selecting a single hypothesis, this approach keeps all hypotheses consistent with the examples (version-space learning).

Current-Best Hypothesis Search Algorithm

The **current-best hypothesis search** algorithm is an iterative approach that maintains a single hypothesis and updates it as new examples are introduced. The aim is to make the hypothesis consistent with the observed data by either generalizing or specializing the current hypothesis.

Steps:

1. **Start with a hypothesis** h (which can be any reasonable guess).
2. **For each example** e:
   - If e is **consistent** with h, continue.
   - If e is a **false positive**, specialize h (narrow the hypothesis).
   - If e is a **false negative**, generalize h (broaden the hypothesis).
3. **Repeat** this process until all examples are consistent with h.

- Suppose you have a hypothesis:

$$\text{WillWait}(r) \Leftrightarrow \text{Patrons}(r, \mathit{Some})$$

  If a new example arrives where patrons are full but they still wait, you might generalize it to:

$$\text{WillWait}(r) \Leftrightarrow \text{Patrons}(r, \mathit{Some}) \lor \text{Patrons}(r, \mathit{Full})$$

1. Current-best Hypothesis Search Algorithm

Example: **Restaurant Waiting Problem**

Suppose you are trying to learn when people will wait for a table at a restaurant based on attributes like:

- `Patrons` : Full, Some, or None.
- `WaitEstimate` : Short, Medium, or Long.
- `Hungry` : True or False.
- `Type` : French, Italian, Thai, or Burger.

You start with an initial hypothesis based on the first example:

- Example 1: `Patrons = Some` , `WaitEstimate = Medium` , `Hungry = True` , `Type = Thai` , and `WillWait = True` .

Your initial hypothesis might be:

$$\text{WillWait}(r) \Leftrightarrow \text{Patrons}(r, \text{Some})$$

Step 1: **A new example arrives** (False Positive)

- Example 2: `Patrons = Some` , `WaitEstimate = Medium` , `Hungry = False` , `Type = Burger` , and `WillWait = False` .

The current hypothesis predicts `WillWait = True` because `Patrons = Some` , but the example says `WillWait = False` . This is a **false positive**, so you specialize the hypothesis by adding the `Hungry` condition:

$$\text{WillWait}(r) \Leftrightarrow \text{Patrons}(r, \text{Some}) \land \text{Hungry}(r)$$

Step 2: **Another new example arrives** (False Negative)

- Example 3: `Patrons = Full`, `WaitEstimate = Short`, `Hungry = True`, `Type = French`, and `WillWait = True`.

The hypothesis predicts `WillWait = False` because `Patrons \neq Some`, but the example says `WillWait = True`. This is a **false negative**, so you generalize the hypothesis to include cases where `Patrons = Full` and `Hungry = True`:

$$\text{WillWait}(r) \Leftrightarrow (\text{Patrons}(r, \text{Some}) \wedge \text{Hungry}(r)) \vee (\text{Patrons}(r, \text{Full}) \wedge \text{Hungry}(r))$$

Version-Space Learning Algorithm

Version-space learning maintains **all hypotheses** consistent with the observed examples. The idea is to have a space (bounded by the **S-set** and **G-set**) that includes all consistent hypotheses. The algorithm incrementally refines this version space as more examples are introduced.

Steps:

1. **Initialize the version space:**
   - **S** (specific boundary): The most specific hypothesis consistent with the data.
   - **G** (general boundary): The most general hypothesis consistent with the data.
2. **For each new example:**
   - If the example is **positive**, generalize the **S-set.**
   - If the example is **negative**, specialize the **G-set.**
3. **Update the version space** by refining the boundaries until you find the correct hypothesis or the space collapses.

Example: **Learning Fruit Classification**

You are trying to learn how to classify fruits based on attributes like:

- `Color`: Red, Green, Yellow.
- `Shape`: Round, Oval.
- `Size`: Small, Large.

Step 1: **Initialize Version Space**

- **S-set** (most specific hypothesis): `False` (no fruit is classified as positive).
- **G-set** (most general hypothesis): `True` (every fruit is classified as positive).

Step 2: **Positive Example**

- Example 1: A red, round, small fruit (an apple), and the classification is `Fruit = Yes`.
- **G-set update:** The most general hypothesis still holds, so no change to G-set.
- **S-set update:** The most specific hypothesis is generalized to include this example:

$$\text{Fruit}(x) \Leftrightarrow \text{Color}(x, \text{Red}) \wedge \text{Shape}(x, \text{Round}) \wedge \text{Size}(x, \text{Small})$$

Step 3: **Negative Example**

- Example 2: A green, oval, large fruit (not a fruit), and the classification is `Fruit = No`.
- **G-set update**: Specialize the most general hypothesis to exclude this example. The G-set is updated to:

$$\text{Fruit}(x) \Leftrightarrow \text{Size}(x, \text{Small})$$

- **S-set remains unchanged** since this negative example does not contradict the current specific hypothesis.

Step 4: **Positive Example**

- Example 3: A green, round, small fruit (a lime), and the classification is `Fruit = Yes`.
- **S-set update:** The most specific hypothesis is generalized to cover both apples and limes. It becomes:

$$\text{Fruit}(x) \Leftrightarrow (\text{Color}(x, \text{Red}) \wedge \text{Shape}(x, \text{Round}) \wedge \text{Size}(x, \text{Small})) \vee (\text{Color}(x, \text{Green}) \wedge \text{Shape}(x, \text{Round}) \wedge \text{Size}(x, \text{Small}))$$

- G-set remains unchanged.

19.2 **Knowledge in Learning**

This section emphasizes the interaction between prior knowledge and learning. A key point is that knowledge-based learning agents use prior knowledge to guide hypothesis generation, thus reducing complexity.

Key Subtopics:

1. **Entailment Constraints:**
    - The agent's hypothesis and the example descriptions must logically entail the observed classifications:

$$\text{Hypothesis} \wedge \text{Descriptions} \models \text{Classifications}$$

2. **Knowledge-based Inductive Learning (KBIL):**
    - Algorithms that leverage prior knowledge to constrain the hypothesis space are known as **KBIL algorithms.**

Steps in a KBIL Algorithm:

1. **Input:** The algorithm receives two types of inputs:
    - **Background Knowledge:** This could be a set of rules or facts that are already known about the problem domain. For example, in a medical diagnosis task, background knowledge might include rules about the relationship between symptoms and diseases.
    - **Examples:** These are the new data points from which the algorithm will learn. Each example consists of a set of attributes and a classification.
2. **Initialize Hypothesis Space:**
    - Begin with a **hypothesis space** that includes all possible hypotheses that are consistent with the background knowledge.
    - For example, if the background knowledge specifies that certain attributes are irrelevant to the problem (such as ignoring a patient's age in diagnosing a specific disease), the algorithm excludes hypotheses that rely on those attributes.

3. **Process Examples:**
    - For each new example, the algorithm checks the consistency of the background knowledge with the example.
    - Based on the data and the knowledge, the algorithm proposes new hypotheses or refines existing ones.
4. **Refinement of Hypotheses:**
    - If an example is inconsistent with the current hypothesis, the algorithm refines the hypothesis. The refinement process ensures that the new hypothesis is still consistent with the background knowledge.
    - This step involves adjusting the hypothesis so that it correctly classifies both the current and prior examples.
5. **Generate Final Hypothesis:**
    - After processing all examples, the final hypothesis is generated. This hypothesis explains all the examples and respects the constraints provided by the background knowledge.
6. **Output:** The final hypothesis is output, which is consistent with both the background knowledge and the observed examples.

Example of KBIL in Action:

Task: Learning the **grandparent** relationship in a family tree, using prior knowledge about **parent-child relationships.**

Step 1: **Background Knowledge**:

- The algorithm is given the following prior knowledge:
    - A **parent** is either a **mother** or a **father**:

$$\text{Parent}(x, y) \Leftrightarrow \text{Mother}(x, y) \vee \text{Father}(x, y)$$

    - A **grandparent** is someone who is a parent of a parent:

$$\text{Grandparent}(x, y) \Leftrightarrow \exists z(\text{Parent}(x, z) \wedge \text{Parent}(z, y))$$

Step 2: **Examples**:

- The algorithm receives data (examples) such as:
    - Example 1: **Elizabeth** is the parent of **Charles**, and **Philip** is also the parent of **Charles.**
    - Example 2: **Charles** is the parent of **William.**

Step 3: **Hypothesis Space**:

- The initial hypothesis space includes all possible relationships between individuals in the family tree. However, using the **background knowledge** about parents and grandparents, the hypothesis space is reduced. For example, the algorithm excludes hypotheses that involve relationships irrelevant to the grandparent relationship, such as "spouse" or "sibling."

Step 4: **Refinement**:

- After processing Example 1 and Example 2, the algorithm generates the following hypothesis:
$$\text{Grandparent}(x, y) \Leftrightarrow \exists z(\text{Parent}(x, z) \wedge \text{Parent}(z, y))$$
This hypothesis states that for x to be a grandparent of y, there must be some z such that x is a parent of z, and z is a parent of y.

Step 5: **Output**:

- The final hypothesis is:
$$\text{Grandparent}(x, y) \Leftrightarrow \exists z(\text{Parent}(x, z) \wedge \text{Parent}(z, y))$$
This rule now allows the system to infer the grandparent relationship for new individuals in the family tree using both the learned examples and the background knowledge.

Advantages of KBIL:

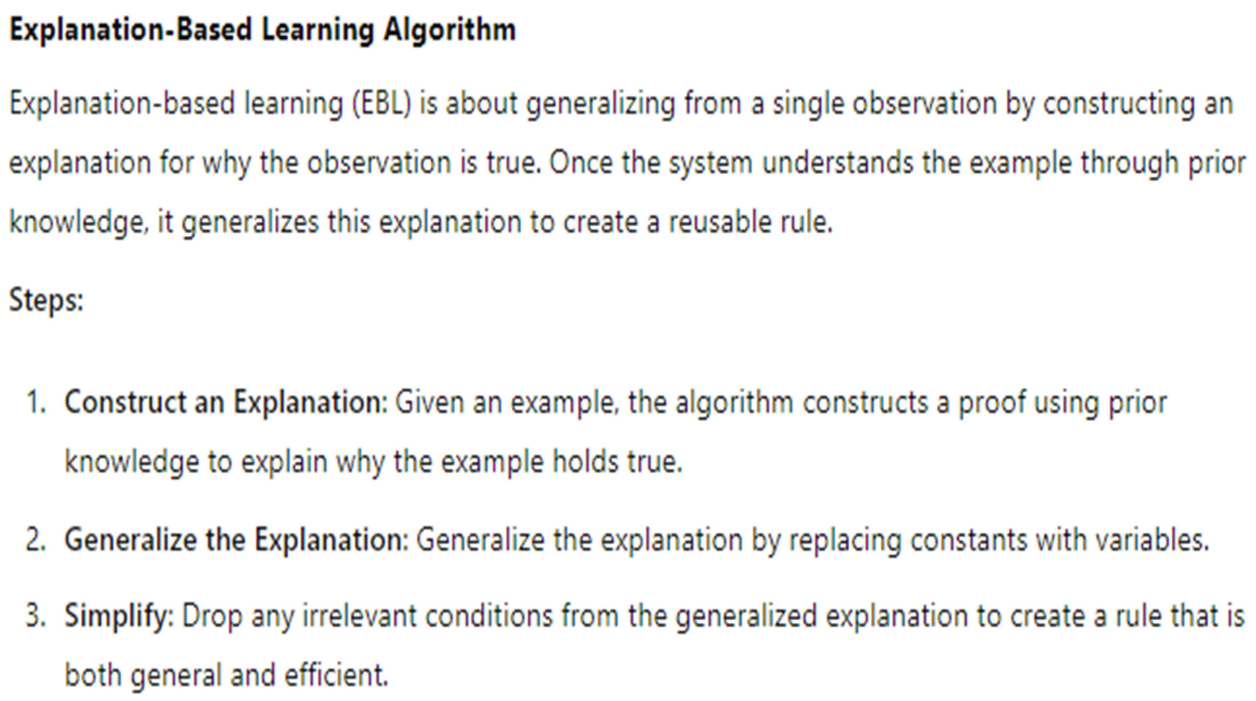
1. **Reduced Hypothesis Space:** By using background knowledge, KBIL dramatically reduces the number of hypotheses the system needs to consider, leading to faster and more accurate learning.
2. **Consistency with Prior Knowledge:** KBIL ensures that the final hypothesis is consistent with both the observed examples and the existing knowledge about the domain.
3. **Ability to Handle Complex Domains:** Since it can leverage rich, domain-specific knowledge, KBIL is particularly effective in complex tasks such as scientific discovery, medical diagnosis, and natural language understanding.

19.3 **Explanation-based Learning (EBL)**

Explanation-based learning (EBL) extracts general rules from individual observations by constructing an explanation for each observation based on prior knowledge.

Key Subtopics:

1. **Constructing Explanations:**
   - The process involves constructing a logical proof for the observed data using prior knowledge. For example, the rule for differentiating a polynomial like X^2 can be generalized from a single observation.
2. **Generalization of Rules:**
   - From specific observations, more general rules are derived. For instance, from $\text{Derivative}(X^2, X) = 2X$, the general rule $\text{Derivative}(u^n, u) = n \cdot u^{n-1}$ can be derived.
3. **Memoization:**
   - Instead of recalculating solutions, previously solved problems and their solutions are stored and reused, which is similar to the technique of **memoization.**

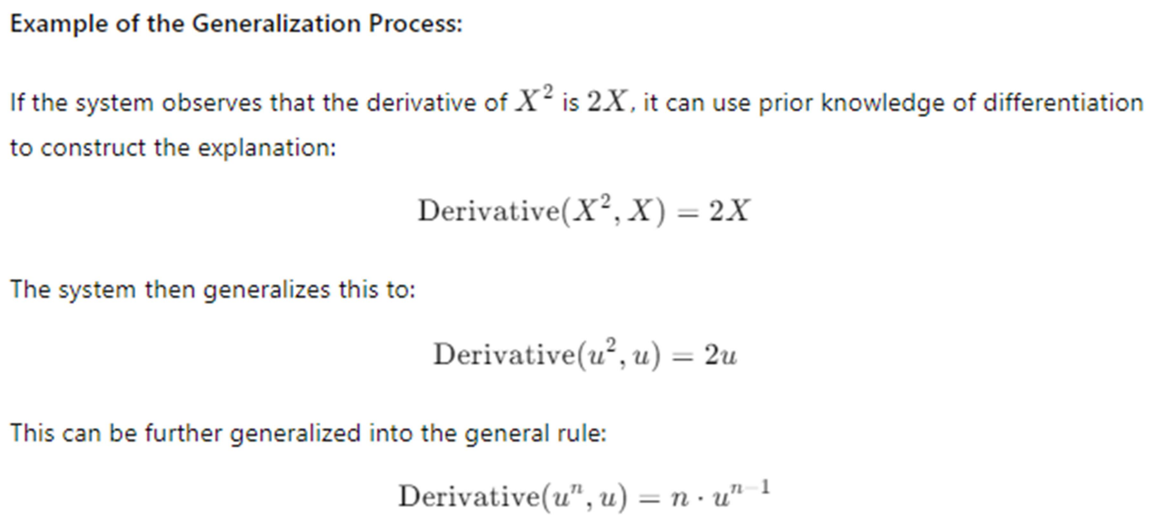


Explanation-Based Learning Algorithm

Explanation-based learning (EBL) is about generalizing from a single observation by constructing an explanation for why the observation is true. Once the system understands the example through prior knowledge, it generalizes this explanation to create a reusable rule.

Steps:

1. **Construct an Explanation:** Given an example, the algorithm constructs a proof using prior knowledge to explain why the example holds true.
2. **Generalize the Explanation:** Generalize the explanation by replacing constants with variables.
3. **Simplify:** Drop any irrelevant conditions from the generalized explanation to create a rule that is both general and efficient.



Example of the Generalization Process:

If the system observes that the derivative of X^2 is $2X$, it can use prior knowledge of differentiation to construct the explanation:

$$\text{Derivative}(X^2, X) = 2X$$

The system then generalizes this to:

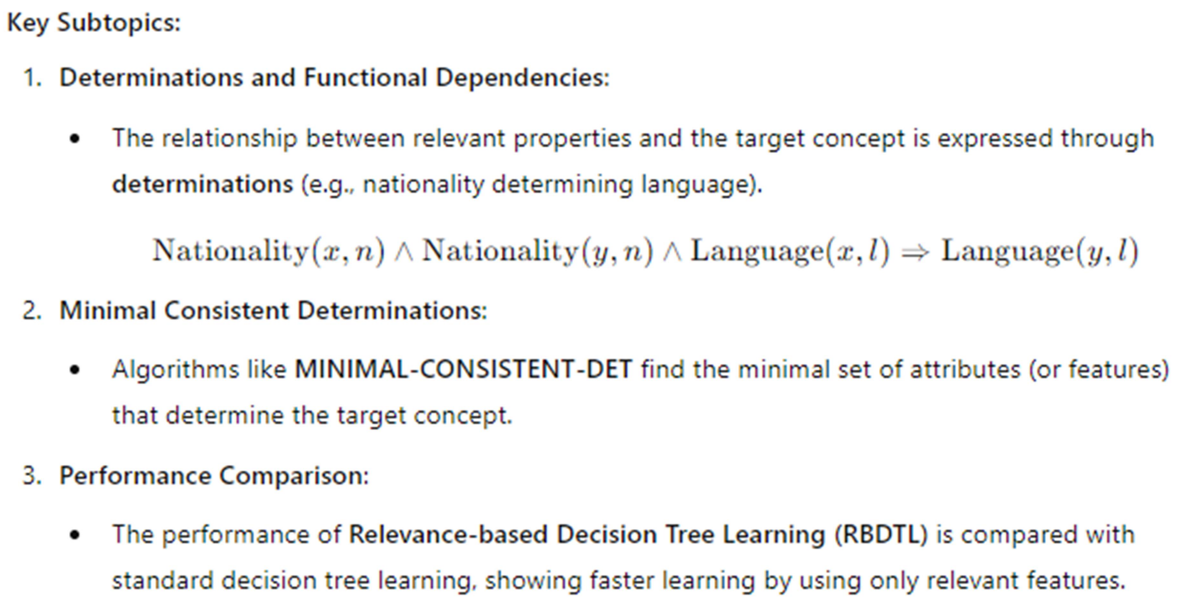
$$\text{Derivative}(u^2, u) = 2u$$

This can be further generalized into the general rule:

$$\text{Derivative}(u^n, u) = n \cdot u^{n-1}$$

19.4 Learning Using Relevance Information

This section covers how learning can focus on **relevant attributes** using background knowledge.



Key Subtopics:

1. **Determinations and Functional Dependencies:**
   - The relationship between relevant properties and the target concept is expressed through **determinations** (e.g., nationality determining language).

$$\text{Nationality}(x, n) \land \text{Nationality}(y, n) \land \text{Language}(x, l) \Rightarrow \text{Language}(y, l)$$

2. **Minimal Consistent Determinations:**
   - Algorithms like **MINIMAL-CONSISTENT-DET** find the minimal set of attributes (or features) that determine the target concept.
3. **Performance Comparison:**
   - The performance of **Relevance-based Decision Tree Learning (RBDTL)** is compared with standard decision tree learning, showing faster learning by using only relevant features.

Minimal Consistent Determination Algorithm (MINIMAL-CONSISTENT-DET)

This algorithm searches for the minimal set of attributes (called **determinations**) that are relevant for predicting the target concept. It helps reduce the complexity of the hypothesis space by focusing only on relevant attributes.

Steps:

1. **Initialize with all attributes.**
2. **Check subsets of attributes** to find the smallest set that is consistent with the observed examples.
3. **Return the minimal subset** of attributes that consistently predicts the target concept.

19.5 Inductive Logic Programming (ILP)

Inductive Logic Programming (ILP) combines inductive learning with the power of first-order logic. It is particularly useful in domains that require learning complex relational structures, such as protein folding or family relationships.

Key Subtopics:

1. **Logic Programming:**
   - Hypotheses are represented as logic programs, which allows for more expressive representations than attribute-based methods.
   - **Example:** Learning family relations like `Grandparent(Mum,Charles)` using logic programming.
2. **Knowledge-based Induction:**
   - ILP systems utilize prior knowledge to reduce hypothesis space and generate interpretable rules. For instance, hypotheses are learned using both background knowledge (e.g., family relations) and new examples (e.g., family trees).
3. **Scientific Applications:**
   - ILP has been applied to scientific problems like discovering rules for protein folding or mutagenicity. These rules are interpretable by experts, making ILP a valuable tool for scientific discovery.