Version Space Learning

Lecture Notes

Nadjib Lazaar
Paris-Saclay University
lazaar@lisn.fr

Introduction

Version Space Learning is a supervised learning approach proposed by *Tom Mitchell*. It is based on the principle of filtering hypotheses according to training examples to restrict the search space to a set of compatible solutions. This method is particularly useful in contexts where prior knowledge is limited and where the goal is to learn a precise conceptual representation.

1 Context and History

Machine Learning (*ML*) is currently dominated by **connectionist** approaches, particularly deep neural networks. These models, inspired by the functioning of the human brain, adjust thousands of parameters to capture regularities in data. However, this approach is not the only existing one in machine learning.

Before the rise of neural networks, learning was often seen as a **symbolic process**, based on logical rules and the manipulation of explicit concepts. Among these approaches, **Version Space Learning** (*VSL*), proposed by **Tom Mitchell in 1978**, holds an important place in **supervised learning**. This model is based on the idea of refining a set of candidate hypotheses over successive observations, maintaining a **boundary between the most general and the most specific hypotheses**.

VSL belongs to **concept learning** methods, where the objective is to learn a rule or category from positive and negative examples. Unlike neural networks, which learn implicit representations in the form of weights, *VSL* directly manipulates concepts expressible in propositional logic.

2 A Symbolic Learning Approach

Symbolic learning relies on the manipulation of **logical structures** and the deduction of **explicit rules**. Unlike connectionist methods, which are based on numerical optimization, symbolic models seek to **extract interpretable concepts** from data.

Version Space Learning belongs to this **symbolic** tradition, where the goal is to manipulate a **set of hypotheses** and refine them based on the observed data. This approach allows for **understandable and interpretable** models, a major advantage in fields where explainability is crucial (healthcare, law, finance).

3 Concept Learning

Concept learning is a fundamental approach in machine learning, where the goal is to identify a decision rule that distinguishes positive instances from negative instances. It is a specific case of binary classification.

3.1 Problem Definition

A binary classification problem is also called a **concept learning problem**. The goal is to learn a target function :

$$f : X \to \{0, 1\} \tag{1}$$

where :
- — X represents the instance space, i.e., the set of objects or examples under consideration.
- — $f(x) = 1$ means that the instance $x \in X$ belongs to the positive class.
- — $f(x) = 0$ means that x belongs to the negative class.

The set of positive instances, denoted C, is defined as :

$$C = f^{-1}(1) = \{x \in X \mid f(x) = 1\} \tag{2}$$

In other words, a **concept** is simply a subset $C \subseteq X$ containing all the positive instances. Its complement, $X \setminus C$, represents the negative instances.

Characteristic	Connectionist Learning	Symbolic Learning
Representation	Neural networks, synaptic weights	Logical rules, symbolic structures
Interpretability	Black box, hard to explain	Explainable, clear concepts
Required Data	Large volumes, need for annotations	Can work with few examples
Approach	Approximate, probabilistic	Deductive, rule-based
Abstraction Capability	Learns statistical patterns	Can generalize with few examples
Need for Background Knowledge	No	Yes, often

Table 1 – Comparison between connectionist and symbolic learning

3.2 Hypotheses and Constraints

Learning concepts is generally based on certain hypotheses :

- **Noise Hypothesis** : It is assumed that the training examples are **accurate**, meaning there are no errors or ambiguities in the labeling of the data.
- **Well-defined Concept Hypothesis** : It is assumed that the target concept is well-defined and can be learned from a finite set of examples.
- **Closed World Hypothesis** : Anything not specifically mentioned as positive is assumed to be negative.

However, these hypotheses are restrictive in real-world contexts where data may contain noise (mis-labeled examples, inaccurate sensors, etc.), and where the target concept may be more complex than expected.

3.3 Illustration and Example

Example 1 : Shape Recognition

Suppose that the instance space X is a set of geometric shapes, and the target concept is "circles." In this case :

- The concept C is defined as :

$$C = \{x \in X \mid x \text{ is a circle}\} \tag{3}$$

- All other shapes (squares, triangles, etc.) belong to the negative set $X \setminus C$.

Example 2 : Email Classification

Consider another example where the goal is to classify emails as *spam* or *non-spam* (ham). Here :

- The instance space X represents all the received emails.
- The concept C corresponds to the emails considered **spam**.
- The complement $X \setminus C$ contains all the non-spam emails.

In an ideal case, a concept learning algorithm would learn a function f that correctly predicts whether an email is spam or not, based on characteristics such as the presence of certain keywords, the structure of the message, or the sender.

3.4 Connection with Learning Hypotheses

Concept learning relies on fundamental hypotheses that guide generalization :

- **Stationarity Hypothesis** : The distributions of positive and negative instances remain stable over time.
- **Representativity Hypothesis** : The training examples are representative of the overall problem.
- **No Ambiguity Hypothesis** : Each instance belongs either to the positive class or the negative class, but never to both at the same time.

4 Version Space Lattice

In concept learning, the hypothesis space typically forms an ordered structure that can be represented as a lattice. This organization helps better understand how an algorithm explores the hypothesis space to identify the target concept.

4.1 Definition of the Version Space

The **version space** is the set of hypotheses that are consistent with the set of training examples observed so far.

$$VS = \{h \in H \mid h(x) = f(x) \text{ for all observed examples } x\} \tag{4}$$

where :
— H is the set of all possible hypotheses.
— $h(x)$ is the prediction of the hypothesis h for an instance x.
— $f(x)$ is the true label of the instance x.
Thus, the version space consists of all hypotheses that are consistent with the training examples.

4.2 General-Specific Order and Lattice

Hypotheses in H can be ordered according to a **general-to-specific ordering**. We say that a hypothesis h_1 is more general than a hypothesis h_2, denoted :

$$h_1 \geq h_2 \tag{5}$$

if and only if h_1 covers at least all the instances covered by h_2 :

$$\forall x \in X, \quad h_2(x) = 10, \Rightarrow h_1(x) = 1 \tag{6}$$

This order induces a **lattice** structure, where :
— The most general hypothesis (denoted $\top$) accepts all instances.
— The most specific hypothesis (denoted $\bot$) rejects all instances.
— Each hypothesis is positioned in the lattice according to its generality relative to others.

4.3 Lattice Bounds : Maximum and Minimum Hypotheses

In the version space elimination algorithm (*Version Space Learning*), two sets of hypotheses are manipulated :
— **G-set (General)** : the set of the most general hypotheses still consistent with the examples.
— **S-set (Specific)** : the set of the most specific hypotheses consistent with the examples.

The version space can be viewed as the set of hypotheses located between these two bounds :

$$S \subseteq VS \subseteq G \tag{7}$$

5 Formal Foundations of Version Space Learning

Learning by *VSL* (Version Space Learning) is a symbolic learning approach based on maintaining a set of hypotheses compatible with the observed data. This method allows progressively restricting the hypothesis space until a single model (or a restricted subset of models) is identified.

5.1 Definitions and Notations

— Let X be the example space and H be the hypothesis space.

— A hypothesis $h \in H$ is a classification function $h : X \to \{0, 1\}$, where $h(x) = 1$ means that x is classified as positive.

— A set of training examples is given by $D = \{(x_i, y_i)\}_{i=1}^{N}$, where $x_i \in X$ and $y_i \in \{0, 1\}$ is the associated label.

— The set of hypotheses compatible with D is called the **Version Space** and is denoted $VS(D)$.

Before introducing the boundaries of the version space, it is essential to understand the inclusion relationship between hypotheses. A hypothesis h is a subset of another hypothesis h' (denoted $h \subset h'$) if and only if h imposes stricter conditions on the classification of examples than h'. In other words, any instance accepted by h is also accepted by h', but the reverse is not necessarily true. This inclusion relationship is fundamental to defining the boundaries of the version space.

Définition 1 (Boundaries of the Version Space) *The version space is characterized by two sets called **boundaries** :*

— *The **specific boundary** S contains the **most specific** hypotheses that correctly classify all the observed positive examples :*

$$S = \{h \in H \mid h(x) = 1, \quad \forall (x, 1) \in D \textit{ and } \nexists h' \in H, h' \subset h\}.$$

— *The **general boundary** G contains the **most general** hypotheses that do not incorrectly classify any negative example :*

$$G = \{h \in H \mid h(x) = 0, \quad \forall (x, 0) \in D \textit{ and } \nexists h' \in H, h \subset h'\}.$$

6 Example of Hypothesis Space

Consider a classification task where each example is described by three attributes :

— **Color** : Red (`R`) or Blue (`B`)
— **Shape** : Circle (`C`) or Square (`S`)
— **Size** : Large (`G`) or Small (`P`)

The hypothesis space H includes all possible combinations of these attribute values, including both general and specific hypotheses.

Hypothesis	**Color**	**Shape**	**Size**
h_{gen} (General hypothesis)	?	?	?
h_1	R	?	?
h_2	?	C	?
h_3	R	C	?
h_4	R	C	G
h_{spec} (Specific hypothesis)	R	C	G

Table 2 – Example of the hypothesis space H

The hypothesis h_{gen} represents the most general case where all examples are accepted, while h_{spec} is the most restrictive, accepting only one specific configuration of attributes.

6.1 Version Space Learning Algorithm

The algorithm follows an iterative process where it updates the sets S and G based on the observed examples.

Algorithme 1 : Version Space Learning Algorithm

```
Input : A training set $D$ with attributes and class labels
Output : The sets $S$ and $G$ defining the version space
Initialization :
$S \leftarrow$ most specific hypothesis (e.g., an empty set or a specific instance)
$G \leftarrow$ most general hypothesis (e.g., $(?, ?, ..., ?)$)
foreach example $(X, y) \in D$ do
    if $y$ is positive (accepted example) then
        foreach $g \in G$ do
            if $g$ does not cover $X$ then
                Remove $g$ from $G$
            end
        end
        foreach $s \in S$ do
            if $s$ does not cover $X$ then
                Generalize $s$ minimally to cover $X$
            end
        end
        Remove from $S$ any hypothesis that is more general than another
    else if $y$ is negative (rejected example) then
        foreach $s \in S$ do
            if $s$ covers $X$ then
                Remove $s$ from $S$
            end
        end
        foreach $g \in G$ do
            if $g$ covers $X$ then
                Specialize $g$ to exclude $X$
            end
        end
        Remove from $G$ any hypothesis that is more specific than
          another
    end
end
return $S, G$
```

6.2 Convergence and Properties

— The algorithm guarantees convergence to a unique hypothesis h^* if and only if $S = G$.

— If S and G do not converge to a unique hypothesis, it means the available data is insufficient to identify a single model.

— The version space can be used to generate new queries or ask strategic questions to distinguish between candidate hypotheses.

7 Why Study Version Space Learning ?

Although neural networks dominate machine learning today, understanding **symbolic** approaches is crucial for several reasons :

1. **Explainability and Interpretability** : Unlike neural models, methods like *VSL and ILP* allow easy explanation of decisions.
2. **Learning with few data** : Symbolic methods require less data to learn general concepts.
3. **Utilizing background knowledge** : They can be integrated with existing knowledge bases.
4. **Complementarity with modern AI** : Today, there is a resurgence of **neuro-symbolic** approaches, combining **neural networks and symbolic models**.

8 Applications and Challenges

Version Space Learning is particularly useful in domains with limited data and where generalization must be carefully controlled. Its applications include :

— Learning concepts in supervised classification.
— Extracting explainable rules from data.
— Planning and robotics, where generalization constraints are crucial.

However, the method suffers from limitations, particularly its sensitivity to errors and inefficiency when the hypothesis space becomes too large. This is where ILP (Inductive Logic Programming) and other more robust approaches can be considered.