# Version Space Algebras and Category Theory

Bailey Wickham

May 29, 2022

**Abstract**

Version space algebras are are objects primarily used in machine learning to model a domain and restricting that domain to a set of predictions. They are constructed using basic set theory, however many of their definitions translate naturally into a category theoretic context.

We give a category theoretic definition of version space algebras and their operations.

## 1 Introduction

Version spaces were first developed as a framework for machine learning in the 80s [3]. Initially they were used to generate predictions based on a small set of examples. They served as a precursor to modern machine learning methods. In recent years machine learning has mostly taken a different approach, yet there is still some recent work on version spaces.

Similar to current tensor based methods, version spaces provide a space of all possible functions (a domain) and examples which constrain those functions to usable predictions. The set of all possible functions is called the hypothesis space, and a single function is called a hypothesis. These are used to generate predictions, each provide a hypothesis of what the user is asking. The functions are constrained by examples, narrowing the hypothesis space down to better predictions of what the user wants.

The primary example defined in [1] is one of a text editor called SMARTEdit. A user may want to perform complex actions such as delete text between these quotes, or move down five lines then copy the text until the work "Hello". The version space algebra for this example is built by a small set of base version spaces called atomic version spaces, and algebraic operations on them such as union and intersection. For the specific version space that corresponds to moving between rows, the hypothesis space may be all functions which map an integer to another integer. After the user records an action such as "jump from the second row to the fifth", or $f(2) = 5$, the version space collapses to a single function. The complete version space algebra which corresponds the SMARTEdit is built by composing these smaller version spaces.

Due to their development in applied machine learning, the definitions are based entirely in set theory, however these constructs can be translated into a category.

In this paper, we will first outline the original definitions for version spaces, then translate them into a category theoretic context.

## 2 Version Spaces

There are four main components in defining a version space:

- H: The hypothesis space
- D: The examples
- VSA: the version space
- Algebraic operations on VSAs

## 2.1 The hypothesis space

The **hypothesis space** is a set of functions with a common domain $A$ and codomain $B$ labeled $H = \{f | f : A \to B\}$. Such a function $h \in H$ is called a hypothesis. The hypotheses space provides a "domain" to work in, or a set of all hypothesises of functions which could be used to generate predictions. As we train our version space, the set of hypotheses shrinks to match our updated data.

For example, $H = \{f : f : \mathbb{Z} \to \mathbb{Z}\}$, the set of functions which send an integer to an integer.

## 2.2 The examples

Given a domain $A$ and a codomain $B$, an **example** is a pair $(i, o) \in A \times B$. The **examples** are a set of examples $D$ such that $D \subseteq A \times B$.

For instance, $(1, 2) \in \mathbb{Z} \times \mathbb{Z}$.

A hypothesis $f : A \to B$ is **consistent** with an example $(i, o) \in A \times B$ if $f(i) = o$. Using our two examples above, $f \in H$ is consistent with $(1, 2)$ if $f(1) = 2$.

The hypothesis space provides a domain to work in, or a space of functions which could possibly be models for our predictions. The examples serve to constrain which functions we are studying, giving a way to refine the ambient space into a specific model. The examples, $D$, are pairs of elements in the domain and range of these functions for which our model should respect. A version space is the set of the functions in the ambient space that are consistent with the examples.

In practice the ambient space may be something like "words in our document", or "integers representing row and column numbers", however then only need be functions between (small) sets. Version spaces are used to define all possible actions of a specific type, then constrained to the examples observed in the current document. This is used to generate predictions, where the examples we have used in our document are used to predict future actions.

## 2.3 Version space algebras

Let $H$ be a hypothesis space with domain $A$ and codomain $B$ and examples $D \subseteq A \times B$. A **version space** $VS_{H,D} = \{f : f(i) = o \text{ for all } (i, o) \in D\}$, or the hypothesis in $H$ that are consistent with all examples in $D$.

Following our above working example, $VS_{H,D} = \{f : f : \mathbb{Z} \to \mathbb{Z} \text{ and } f(1) = 2\}$

Lau et al.[1] introduces the idea of Version Space Algebras (VSAs) which are version spaces generated by operations on other version spaces. A version space which is defined explicitly, or not defined in terms of operations on other version spaces, is called an **atomic version space**. From a set theory perspective there is no difference between a version space algebra and a version space, there is only a difference in how they are constructed.

There are three main operations defined on a version space.

**Definition 2.1** (Version space union)**.** Let $H_1 = \{f | f : A \to B\}$ and $H_1 = \{f | f : A \to B\}$ be hypothesis spaces with the same domain and range. Let $D$ be a set of examples. Then $VS_{H_1,D} \bigcup VS_{H_2,D} = VS_{H_1 \bigcup H_2,D}$.

**Definition 2.2** (Version space intersection)**.** Let $H_1$ and $H_2$ be two hypothesis spaces such that the domain of functions in $H_1$ equals those of $H_2$. Let $D$ be a sequence of training examples. The version space intersection $VS_{H_1,D} \bigcap VS_{H_2,D} = VS_{H_1 \bigcap H_2,D}$.

**Definition 2.3** (Version space transform)**.** Let $\tau_i$ be a function mapping elements from the domain of $VS_1$ to the domain of $VS_2$, and $\tau_o$ be a one-to-one mapping of elements in the range of $VS_1$ to elements in the range of $VS_2$. Version space $VS_1$ is a transform of $VS_2$ iff $VS_1 = \{g : \exists f \in VS_2, \forall i, g(i) = \tau_o^{-1}(f(\tau_i(i)))\}$

While not provided in [1], a transform can be represented as a commutative diagram. For all $f$ this diagram commutes, where $\tau_o$ is injective and $A_i$ is the domain of $H_i$ and $B_i$ is the range of $H_i$.

$$\begin{array}{ccc} A_1 & \xrightarrow{g} & B_1 \\ \downarrow{\scriptstyle \tau_i} & & \downarrow{\scriptstyle \tau_o} \\ A_2 & \xrightarrow{f} & B_2 \end{array}$$

### 2.4 Training

When training a version space, start with a hypothesis space $H$ and $D = \emptyset$. Training consists of adding pairs to $D$, enlarging $D$ and shrinking $VS_{H,D}$ to be consistent with $D$. This gives us a contravariant chain of inclusion maps which represent training the version space.

$$\begin{array}{ccccc} VS_{H,D_{n-1}} & \xleftarrow{\supseteq} & VS_{H,D_n} & \xleftarrow{\supseteq} & VS_{H,D_{n+1}} \\ \downarrow & & \downarrow & & \downarrow \\ D_{n-1} & \xrightarrow{\subseteq} & D_n & \xrightarrow{\subseteq} & D_{n+1} \end{array}$$

with the downward arrows the projection map onto the examples. Training a version space can be modeled as this sequence of maps.

## 3 Category Theory

The main goal of this paper is to redefine a version space in a categorical framework.

To start, we make a simplification to the definition of a version space. In section 2, we say a the examples are pairs $(i, o) \in A \times B$ where $A$ is the domain and $B$ is the codomain of the hypothesises. If there exists a two pairs $(i, o)$ and $(i, j)$ where $i \neq j$, then the version space is the empty set as no function $f$ can satisfy this condition. If we disallow conflicting examples such as the one above, the the examples can be represented as a partial function: $g_D : A' \to B$ where $A' \subseteq A$. This means our definition of version space is equivalent to $VS_{H,g_D} = \{f \in H : f|_{A'} = g_D\}$ In this section we will use this simplified definition of examples.

Next, we redefine the three major components of a version space:

- H: The hypothesis space
- D: The examples
- VSA: the version space

### 3.1 Working example

In this section, it would be helpful to have a working example to discuss as we update our definitions. If one were trying to generate helpful predictions for a text editor (See SMARTEdit [1]), one would need a set of operations to predict. For example, one may want to predict actions such as delete text starting with a prefix, find the next instance of a word, move to the third row, or insert this text. We will use the example of a function which is used to represent moving between rows as our working example.

We want the version space **Row** to represent jumping between rows in a text document. Initially our version space is going to contain all possible ways to jump between rows, but after training, it will represent only those jumps which have been completed when editing the document.

Let $H = \{f : f : \mathbb{Z} \to \mathbb{Z}\}$. This represents all possible jumps between rows, where the starting row is the domain and the ending row is the codomain. Initially this starts with all possible jumps, but once an example is added, it collapses to functions which contain that jump, i.e. $f(1) = 5$, or a jump from the first line of the document to the fifth. Note that this is just one version space and a complete example would have many version spaces joined together, this allows for more than just one jump.

### 3.2 Hypothesis spaces

We begin by defining the hypothesis space in categorical terms. Recall the previous definition of $H$: $H \subseteq \{f : f : A \to B\}$. By redefining the hypothesis space in categorical terms, we can define a version space transform as an arrow in a category instead of a separate set.

In categorical terms, the set of all maps $f : A \to B$ in a category is called $Hom(A, B)$. In this case the hypothesis space is a subset of $Hom(A, B)$.

The first step to defining the hypothesis space is defining the $hom(\cdot, \cdot)$ functor, which is used to model the functions of a hypothesis space. The *hom* functor is not enough alone though, as it does not

preserve $A$ and $B$. To do this, we will define $H$ first then move the construction of $H$ into a comma category.

**Definition 3.1.** The *hom* functor sends a pair of objects to their $Hom$ set.

Define the *hom* functor $hom(\cdot, \cdot)$:

$$hom(\cdot, \cdot) : Set^{Op} \times Set \to Set$$
$$(A, B) \to Hom(A, B)$$

$$\begin{array}{ccc} (A_1, B_1) & & Hom(A_1, B_1) \\ \downarrow & \Longrightarrow & \downarrow \\ (A_2, B_2) & & Hom(A_2, B_2) \end{array}$$

Then $H$ is a set with a monic arrow into $hom(A, B)$.

This functor sends a pair of sets to their $Hom$ set, which is used to construct the hypothesis space. The functor also sends a pair of arrows in the domain to an arrow between $Hom$ sets through composition which allows maps between the sets to induce maps between hypothesis spaces.

Notice that $hom(\cdot, \cdot)$ is a bifunctor defined in terms of of $hom(\cdot, B)$ and $hom(A, \cdot)$. Given sets $A, B$, $H$ is a monic arrow into $hom(A, B)$, the restriction to monic is necessary to preserve the domain and codomain. The problem with this definition is that it does not preserve the information of $A$ and $B$, we will fix this with a comma category. To strictly match the definition of the hypothesis space given in section 2 where $H$ is a subset of the $Hom$ set, $H$ must map into $hom(A, B)$ via the inclusion map (a monic arrow). Studying arrows other than the inclusion map is worth further work.

The benefit of using the *hom* functor instead of the $Hom$ set is that the *hom* function includes a notion of arrows between objects, these arrows will induce version space transforms.

### 3.2.1 Working example

In our working example, $H = \{f : f : \mathbb{Z} \to \mathbb{Z}\}$. We can redefine $H$ as $H \to hom(\mathbb{Z}, \mathbb{Z})$ where the arrow is the inclusion map. $H$ is an object in $Set$ of the form $Hom(\mathbb{Z}, \mathbb{Z})$.

### 3.2.2 Hypothesis category

Let $(A, B) \in Set^{op} \times Set$, $H \in Set$ where $H$ is a hypothesis space, and $i : H \to hom(A, B)$ the monic inclusion functor.

We can construct the **hypothesis category**, a comma category as such:

$$Set^{Op} \times Set \xrightarrow{hom(\cdot,\cdot)} Set \xleftarrow{I} Set$$

With objects:

$$((A, B), H, i)$$

and arrows:

$$((A_1, B_1), H_1, i) \to ((A_2, B_2), H_2, i)$$

which consist of:

$$u : A_2 \to A_1$$
$$v : B_1 \to B_2$$
$$s : H_1 \to H_2$$

such that this diagram commutes:

$$\begin{array}{ccc} H_1 & \xhookrightarrow{i} & Hom(A_1, B_1) \\ \downarrow{\scriptstyle s} & & \downarrow{\scriptstyle (u,v)^*} \\ H_2 & \xhookrightarrow{i} & Hom(A_2, B_2) \end{array}$$

Now we can define the hypothesis space as an object in the hypothesis category, giving us $Hom(A, B)$ while preserving the information of $A$ and $B$.

### 3.2.3 Working example

We previously defined $H$, now we expand $H$ to sit inside a comma category. The comma category for the **Row** version space has objects: $((\mathbb{Z}, \mathbb{Z}), H, \iota)$ where $H$ is an object in the image of $hom(\mathbb{Z}, \mathbb{Z})$ and $\iota$ is the inclusion map.

While the notation here may be cumbersome, the essential information is the hypothesis, and the domain and codomain of them. In this example, the essential information we care about is that $H \in Hom(\mathbb{Z}, \mathbb{Z})$, and that the domain and codomain of the $Hom$ set are preserved.

## 3.3 The examples

Next is the examples. As we noted in the beginning of this section, we consider the examples to be a partial function $g_D : A' \to B$ where $A' \subseteq A$. Examples are used to check the consistency of a function, or whether for $f \in H$, $f(i) = o$.

One of the difficulties in translating version space algebras to category theory is the notion of "filtering" or checking the consistency of each function in $H$. Filtering is inherently not something category theory handles well as it is designed to study "all" objects of a class.

Using the partial function definition of examples, training can be realized as in one of two ways. Traning can be represted as expanding the example partial $g_D$ to a new partial function $g_{D'}$ where $D \subseteq D'$. This allows us to have only one example partial at a time. The other possibility is that the examples can be realized as a set $D$ of partial functions $g_{D_i} : A_i \to B$ where if $A_i \cap A_j \neq \emptyset$ then $g_i$ and $g_j$ must agree on $A_i \cap A_j$.

### 3.3.1 Working example

In our example of **Row**, examples are supposed to capture the information of jumping from one row to another. If we were to jump from the first row to the fifth, this would be the pair $(1, 5) \in D$. Any function in $H$ that sends 1 to 5 would be consistent with this example, and remain in the version set.

For example, the function $f(x) = x + 4$ would be an element of our version space, and correspond to the action of "move down four lines" in the text editor, a reasonable prediction.

## 3.4 Version spaces

To define a version space, we take the functions in the hypothesis space which are consistent with our examples. We will construct the version space object as a pullback of two morphisms.

The first morphism serves to send the hypothesis space for a partial function to the complete hypothesis space. Let $A$ be the domain and $B$ the codomain of a hypothesis space, and $A' \subseteq A$. Let $j : A' \to A$ and let $res_{A,A'} = (j, 1_B)^*$. Given a function $f \in H$:

$$res_{A,A'}(f) = f|_{A'} : A' \to B$$
$$A' \xrightarrow{j} A \xrightarrow{f} B \to 1_B B$$
so
$$res_{A,A'} : Hom(A, B) \to Hom(A', B)$$

The other morphism we define is $g_D$ (xxx might need to rename this), defined on the one point category to "pick" out the example partial. This morphism sends the single object in the single object category to $g_D$, the example partial.

$$g_D : \{*\} \to Hom(A', B)$$
$$* \to g_D$$

Finally, the pullback is: $VS_{H,D} = H \times_{Hom(A',B)} \{*\} \cong \{f \in H : res_{A,A'}(f) = g_D\}$

# References

[1] Tessa Lao, Pedro Domingos, Daniel S. Weld, Version Space Algebra and its Application to Programming by Demonstration

[2] Saunders Mac Lean, Categories for the Working Mathematician

[3] Tom M. Mitchell, Generalization as search