Version Space Algebras and Category Theory

Bailey Wickham

June 10, 2022

Abstract

Version space algebras are objects primarily used in machine learning to model a domain and then restrictions of that domain to a set of predictions. They are constructed using basic set theory, but many of their definitions translate naturally into category theory. 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 1980s [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, with each providing a hypothesis of what the user is asking. The functions available are constrained by the examples, which narrow the hypothesis space down to the best predictions of what the user wants.

The primary example defined in [1] is a text editor called SMARTEdit. In this example, 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 word "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 might be the space of 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 only those functions for which $f(2) = 5$. The complete version space algebra which corresponds to SMARTEdit is built by combining these atomic version spaces.

Due to their development in applied machine learning, all of the constructions 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*
- $VS_{H,D}$: the *version space*
- The algebraic operations on version spaces

2.1 The hypothesis space

A **hypothesis space** is a set of functions with common domain A and codomain B, labeled $H = \{f \mid f : A \to B\}$. Each function $h \in H$ is called a **hypothesis**. The hypothesis space provides a "domain" to work in, or a set of all hypotheses 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, perhaps $H = \{f \mid f : \mathbb{Z} \to \mathbb{Z}\}$, the set of functions which send an integer to an integer. Some elements of this space are $f(x) = x + 3$ and $f(x) = 3$. The only requirement for f is that it is a function. For a more applicable example, let $A = \{\text{hi}, \text{hello}, \text{world}\}$, $H = \{f \mid f : A \to \mathbb{Z}\}$, and $f \in H$ the length of the word in A.

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** is a set of examples D i.e., $D \subseteq A \times B$. For instance, $(1, 2) \in \mathbb{Z} \times \mathbb{Z}$, or using the previous set, $A = \{\text{hi}, \text{hello}, \text{world}\}$, an example is $(hi, 2) \in A \times \mathbb{Z}$

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 which our model should respect.

2.3 Version space algebras

Let H be a hypothesis space with domain A and codomain B and let $D \subseteq A \times B$ be a set of examples. The **version space** $VS_{H,D}$ is defined to be the hypotheses in H that are consistent with all examples in D. Explicitly, $VS_{H,D} = \{f \mid f(a) = b \text{ for all } (a, b) \in D\}$

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

Lau et al.[1] introduced the idea of version space algebras (VSAs), which are version spaces generated by algebraic 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.

2.4 Algebraic operations on version spaces

There are three main operations defined on a version space.

Definition 2.1. Let $H_1 \subseteq \{f \mid f : A \to B\}$ and $H_2 \subseteq \{f \mid f : A \to B\}$ be hypothesis spaces with the same domain and codomain. Let D be a set of examples. Then the **version space union** of $VS_{H_1,D}$ and $VS_{H_2,D}$ is $VS_{H_1,D} \bigcup VS_{H_2,D} = VS_{H_1 \bigcup H_2,D}$. The **version space intersection** of $VS_{H_1,D}$ and $VS_{H_2,D}$ is $VS_{H_1,D} \bigcap VS_{H_2,D} = VS_{H_1 \bigcap H_2,D}$.

Definition 2.2. Let A_1, A_2, B_1 and B_2 be sets, $H_1 \subseteq \{f \mid f : A_1 \to B_1\}$ and $H_2 \subseteq \{g \mid g : A_2 \to B_2\}$. Let $\tau_i : A_1 \to A_2$ be a function between the domains and $\tau_o : B_1 \to B_2$ be an invertible function between the codomains then version space VS_{H_1,D_1} is a **transform** of VS_{H_2,D_2} if $VS_1 = \{g \mid \exists f \in VS_2, \forall i, g(i) = \tau_o^{-1}(f(\tau_i(i)))\}$

While not provided in [1], a transform can be represented as the set of all $f : A_1 \to B_1$ for which there exists $g \in VS_2$ such that the diagram below commutes.

$$
\begin{tikzcd}
A_1 \arrow[r, "f"'] \arrow[d, "\tau_i"] & B_1 \arrow[d, "\tau_o"] \\
A_2 \arrow[r, "g"] & B_2
\end{tikzcd}
$$

The original definition of a version space transform is part of what originally motivated our study in category theory. The definition can be represented as a commutative diagram which simplifies presentation. We are interested in studying what other aspects of version spaces can be simplified using category theory.

2.5 Training

When *training* a version space, we start with a hypothesis space H and no examples. Training consists of adding pairs to D, which causes $VS_{H,D}$ to shrink. This gives us a contravariant chain of inclusion maps which represent training the version space,

$$VS_{H,D_{n-1}} \hookleftarrow VS_{H,D_n} \hookleftarrow VS_{H,D_{n+1}}$$

$$D_{n-1} \hookrightarrow D_n \hookrightarrow D_{n+1}$$

3 Category Theory

We next redefine version spaces in a categorical framework. To start, we make a simplification to the definition of examples. In Section 2, we defined the examples as pairs $(i, o) \in A \times B$ where A is the domain and B is the codomain of the hypothesis space. If there exists two pairs (i, o_1) and (i, o_2) where $o_1 \neq o_2$, then the version space is empty as no function f can satisfy both conditions. If we disallow conflicting examples such as this, then a set of examples can be represented as a function $g_D : A' \to B$ where $A' \subseteq A$. Then our definition of version space can be restated as $VS_{H,g_D} = \{f \in H \mid f|_{A'} = g_D\}$ In this section we will use this simplified definition of examples.

3.1 Working example

It will 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 deleting text starting with a prefix, finding the next instance of a word, moving to the third row, or inserting text. We will use the example of 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 while editing the document.

Let $H = \{f \mid f : \mathbb{Z} \to \mathbb{Z}\}$. This represents all possible jumps between rows, where the domain contains the starting row and the codomain contains the ending row. Initially this starts with all possible jumps, but once an example is added, it collapses to functions which contain that jump, e.g. $f(1) = 5$, or a jump from the first row 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, to allow for more than just one jump.

3.2 Hypothesis spaces

We begin by defining the hypothesis space in categorical terms. 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 collection of all set maps $f : A \to B$ is denoted $\mathrm{Hom}(A, B)$. Here a hypothesis space is a subset of $\mathrm{Hom}(A, B)$.

The first step to defining the hypothesis space is defining the $\mathrm{Hom}(\cdot, \cdot)$ 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. Define a functor $\mathrm{Hom}(\cdot,\cdot) : \mathrm{Set}^{\mathrm{op}} \times \mathrm{Set} \to \mathrm{Set}$ as follows. For a pair of sets (A, B), let $\mathrm{Hom}(A, B)$ denote the collection of set maps $f : A \to B$. As for arrows, first recall that an arrow $(u, v) : (A_1, B_1) \to (A_2, B_2)$ in corresponds to a pair of set maps $u : A_2 \to A_1, v : B_1 \to B_2$. Let $(u, v)^* : \mathrm{Hom}(A_1, B_1) \to \mathrm{Hom}(A_2, B_2)$ be the map that sends the function $f : A_1 \to B_1$ to the function $g : A_2 \to B_2$ defined by $g = v \circ f \circ u$ This can be visualized by the commutative diagram below:

$$\begin{tikzcd}
A_1 \arrow[r, "f"] & B_1 \arrow[d, "v"] \\
A_2 \arrow[u, "u"] \arrow[r, "g"'] & B_2
\end{tikzcd}$$

A **hypothesis space** is then a monic arrow $i : H \to \mathrm{Hom}(A, B)$

Remark. 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, H must map into $\mathrm{Hom}(A, B)$ via the identity 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 functor includes a notion of arrows between objects. These arrows will induce version space transforms.

In our working example, we have $H = \{f \mid f : \mathbb{Z} \to \mathbb{Z}\}$. In this case the identity map gives us a monic arrow $H \hookrightarrow \mathrm{Hom}(\mathbb{Z}, \mathbb{Z})$.

3.2.1 Hypothesis category

To preserve the data of A and B, we can construct a category of hypothesis spaces, borrowing the notion of a comma category. The objects of the hypothesis category **Hyp**, consist of the triple $((A, B), H, i)$ where A and B are sets and $i : H \to \mathrm{Hom}(A, B)$ is a hypothesis space. An arrow from a tuple $((A_1, B_1), H_1, i_1)$ to a triple $((A_2, B_2), H_2, i_2)$ consists of arrows $(u, v) : (A_1, B_1) \to (A_2, B_2)$, $s : H_1 \to H_2$ such that the diagram below commutes.

$$\begin{tikzcd}
H_1 \arrow[r, hook, "i_1"] \arrow[d, "s"] & \mathrm{Hom}(A_1, B_1) \arrow[d, "(u,v)^*"'] \\
H_2 \arrow[r, hook, "i_2"] & \mathrm{Hom}(A_2, B_2)
\end{tikzcd}$$

Now we can define a hypothesis space $i : H \to \mathrm{Hom}(A, B)$ as an object in the **Hyp** category, giving us $\mathrm{Hom}(A, B)$ while preserving the information of A and B.

While the notation here may be cumbersome, the essential information is the set of hypothesises, and the domain and codomain of them.

In our working example, the **Row** version space corresponds to the objects in **Hyp** of the form: $((\mathbb{Z}, \mathbb{Z}), H, i)$.

3.3 The examples

Next we consider the examples. As we noted in the beginning of this section, we consider the set of examples to be equivalent to a function $g_D : A' \to B$ where $A' \subseteq A$. Here there is a one-to-one correspondence between a set of examples D and a function g_D defined on a subset of the domain of the hypothesis space. 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.

In **Row**, an example might be a function $f : \{1\} \to \mathbb{Z}$. 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 only the functions in the hypothesis space that are consistent with our examples. Filtering is inherently not something category theory handles well, so we will use a pullback to attempt to filter the hypothesis space. We will construct the version space object as a pullback of two morphisms in **Set**.

We first recall the notion of a pullback in **Set**. Suppose $f : X \to Z$ and $g : Y \to Z$ are two set maps. The set of all pairs $(x, y) \in X \times Y$ such that $f(x) = g(y)$ is called the **pullback** of f and g (or in **Set**, the **fibered product** of f and g). We denote this set $X \times_Z Y$. Note that we have a commutative diagram:

$$\begin{tikzcd} X \times_Z Y \arrow[r, "\pi_2"] \arrow[d, "\pi_1"'] & Y \arrow[d, "g"] \\ X \arrow[r, "f"'] & Z \end{tikzcd}$$

Now suppose $i : H \to \mathrm{Hom}(A, B)$ is a hypothesis space, and $g_D : A \to B$ corresponds to a set of examples. Note that the subset inclusions $A' \subseteq A$ and $B \subseteq B$ induce a map $\mathrm{Hom}(A, B) \to \mathrm{Hom}(A', B)$, which we denote $\mathrm{res}_{A,A'}$ and call the **restriction map**. A hypothesis $h \in H$ is consistent with a set of examples D exactly when $res_{A,A'}(h) = g_D$. Note that $res_{A,A'}(h)$ is a notational simplification of $res_{A,A'}(i(h))$. Again abusing notation, let g_D also denote the map $g_D : \{*\} \to \mathrm{Hom}(A', B)$ that sends the single object $*$ to $g_D \in \mathrm{Hom}(A', B)$. Then, the version space $VS_{H,D}$ can be viewed as the pullback diagram below.

$$\begin{tikzcd} VS_{H,D} \arrow[r] \arrow[dd] & H \arrow[d, "i"] \\ & \mathrm{Hom}(A, B) \arrow[d, "res_{A,A'}"] \\ \{*\} \arrow[r, "g_D"'] & \mathrm{Hom}(A', B) \end{tikzcd}$$

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

4 Conclusion

In this paper we translated the definitions of a version space into category theory. We translated the hypothesis space into a category **Hyp** using the idea of a comma category. We gave a commutative diagram for the definition of a version space transform and gave an equivalent function definition for the examples. Finally we redefined a version space as a pullback.

The original definition for a version space transform is what motivated the translation into category theory. The set theoretic definition was unwieldy and could be represented better as a commutative diagram. Once version spaces are translated into category theory, we can apply the category theoretic machinery to study version spaces. This is the subject of future work.

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