Version Space Algebras and Category Theory

Bailey Wickham

May 13, 2022

Abstract

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]. In recent years machine learning has taken a different approach, yet there is still some recent work on version spaces.

Version Space Algebras are useful as a framework for machine learning in Computer Science, however their associated definitions are limited by this applied context. Many of the constructions can be generalized to be category theoretic.

Similar to current tensor based methods, version spaces provide a space of all possible functions and examples which constrain those functions use 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.

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

The **hypothesis space** H is a set of functions: $H = \{f | f : A \to B\}$. An element of the hypothesis space is a **hypothesis**.

Lau et al.[1] define a Version Space $VS_{H,D}$ as such:

Definition 2.1 (Version Space)**.** A function f is said to be consistent with an example (a, b) if $f(a) = b$.

Let the hypothesis space $H = \{f|f : A \to B\}$, where A and B are sets. Let the examples $D = \{(i, o)|i \in A \text{ and } o \in B\}$ Then our version space $VS_{H,D} = \{f|f(i) = o \text{ for all } (i, o) \in D\}$

The hypothesis space provides an "ambient space" to work in, or a space of functions which could possibly be models. 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.

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 2.2 (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 2.3 (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 2.4 (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 2.5 (Version space transform)**.** Let τ_i be a function mapping elements from the domain of VS_1 to the domain of VS_2, and τ_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 τ_o is injective and A_i is the domain of H_i and B_i is the range of H_i.

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

3 Category Theory

In the previous section we defined a version space and a set of operations on it. In this section we will redefine version spaces in categorical terms. First we define our objects which are version spaces.

Definition 3.1 (Version Space (Category))**.** Let A and B be small sets. Then a hypothesis space $H \hookrightarrow Hom_{Set}(A, B)$. Let the examples, D, be a set of pairs $\{(a, b) : a \in A, b \in B\}$ (a, b) that project via maps ∂_0, ∂_1 into A and B respectively.

Let Γ_f be the graph functor: $\Gamma_f = \{(a, b)|b = f(a)\} \subseteq A \times B$. We can define a version space as the set $VS_{H,D} = \{f : D \to \Gamma_f \text{ is injective}\}$. We say D is consistent with f if $D \to \Gamma_f$ is injective. A version space is all of the functions which are consistent with all of the examples: $VS_{H,D} = \{f : D \text{ is consistent with } \Gamma_f \}$

Define the projection $\pi_D : VS_{H,D} \to D$, and $\Gamma_H = \{\Gamma_f : f \in H\}$ We can phrase this as a diagram:

$$\begin{tikzcd}
H \times D \arrow[r, "\pi_D"] \arrow[dr, "\Gamma"] & D \arrow[d, dashed, "\iota"'] \\
& \Gamma_H
\end{tikzcd}$$

Note that since Γ_H is a set of sets, ι must be defined as such: $\iota : D \to \{D\}$. Therefore a $VS_{H,D}$ is the set of D such that ι is injective and the diagram commutes.

This construction mirrors the set theoretic construction given by Lau et al. [1] Union and Intersection are defined almost identically to the set theory version.

Definition 3.2 (Category of VSAs)**.** Let **VSA** be the category of version spaces. We define the category of all version spaces.

Objects in our category are of the form $VS_{H,D}$, over any H and any D. A morphism assigns $VS_{H_1,D_1} \to VS_{H_2,D_2}$ if and only if $H_1 \subseteq H_2$.

Proof. Associativity of morphisms follows from associativity of the inclusion map. $H \subseteq H$, so there exists an identity morphism for every object. Therefore **VSA** forms a category. □

Definition 3.3. The union and intersection definitions are the same as before:

$$VS_{H_1,D} \bigcup VS_{H_2,D} = VS_{H_1 \bigcup H_2,D}$$

and

$$VS_{H_1,D} \bigcap VS_{H_2,D} = VS_{H_1 \bigcap H_2,D}$$

We can use the language of graphs from Mac Lean[2] to describe transformations. Recall that a graph G is a set O of objects and a set A of arrows, and a pair of functions, $A \rightrightarrows B$:

$$A \underset{\partial_1}{\overset{\partial_0}{\rightrightarrows}}, B \quad \partial_0 f = \text{domain} f, \quad \partial_1 f = \text{codomain} f \tag{1}$$

A morphism $D : G \to G'$ of graphs is a pair of functions $D_O : O \to D'$ and $D_A : A \to A'$ such that:

$$D_O \partial_0 f = \partial_0 D_A f \quad \text{and} \quad D_O \partial_1 f = \partial_1 D_A f \tag{2}$$

for every arrow $f \in A$. Every category C determines a graph UC with the same objects and arrows, forgetting the composite arrows.

Here is the definition of a version space transform from earlier:

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

We can transform this into a statement about graphs.

Theorem 3.1. If there is a bijective morphism from the graph of VS_1 to the graph of VS_2, then there exists a version space transform from VS_1 to VS_2

Proof. Let $H_1 : A_1 \to B_1$ and $H_2 : A_2 \to B_2$ and let VS_1 have hypothesis space H_1 and VS_2 have hypothesis space H_2. Assume there exists a bijective graph morphism from the graph of VS_1 to the graph of VS_2.

Given a $VS_{H,D}$ consider the graph with objects $O = A \cup B$, and arrows the functions consistent with the examples, $A = VS_{H,D}$. Since $D = A \cup B$ define D_O as such:

$$D_O = \begin{cases} \tau_i(a) & a \in A \\ \tau_o(b) & b \in B \end{cases} \tag{3}$$

We can replace our diagram of version space transforms with the language of graphs. These two definitions are equivalent, A is a version space transform of A' if and only if this diagram commutes for every $f \in A'$.

$$\begin{tikzcd}
 & A \arrow[dl, "\partial_0"] \arrow[dr, "\partial_1"] & \\
A_1 \arrow[rr, "g"'] \arrow[d, "D_O"] & & B_1 \arrow[d, "D_O"'] \\
A_2 \arrow[rr, "f"] & & B_2 \\
 & A' \arrow[ul, "\partial_0"] \arrow[ur, "\partial_1"'] &
\end{tikzcd}$$

The graph morphism gives us $D_A : A \to A'$ and we have the completed diagram:

$$\begin{tikzcd}
 & A \arrow[dl, "\partial_0"] \arrow[dr, "\partial_1"] & \\
A_1 \arrow[rr, "g"] \arrow[d, "D_O"] & & B_1 \arrow[d, "D_O"'] \\
A_2 \arrow[rr, "f"] & D_A & B_2 \\
 & A' \arrow[ul, "\partial_0"] \arrow[ur, "\partial_1"'] \arrow[uuu, <->] &
\end{tikzcd}$$

Take an $f \in A'$. We can chase this f around the diagram to produce show that VS_1 is a transform of VS_2.

$$D_O^{-1} f D_O \partial_0 D_A^{-1} f = o \quad \text{for some } o \in B_1$$

Call $D_A^{-1} f = g$, and $i \in \partial_0 g$. Then we have

$$\partial_0 D_A^{-1} f = D_O f D_O^{-1}$$
$$g(i) = D_O^{-1} f D_O \quad \forall i \in A_1$$

With D_O playing the role of τ, we conclude:

$$g(i) = \tau^{-1} f \tau(i) \quad \forall i \in A_1$$

and VS_1 is a transform of VS_2. □

3.1 Training

In machine learning applications, we train the version space by adding pairs to the example set. This gives us a contravariant pair of inclusion maps which represent training the version space.

$$\begin{tikzcd}
VS_{H,D_{n-1}} \arrow[d] & VS_{H,D_n} \arrow[l, "\supseteq"'] & VS_{H,D_{n+1}} \arrow[l, "\supseteq"'] \arrow[d] \\
D_{n-1} \arrow[r, "\subseteq"] & D_n \arrow[r, "\subseteq"] & D_{n+1}
\end{tikzcd}$$

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

4 New

We need to redefine xxx major definitions in category theoretic terms:

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

4.1 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, 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.

4.2 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.

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

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

$$\begin{tikzcd} (A_1, B_1) \arrow[d] \arrow[dr, phantom, "\Longrightarrow"] & Hom(A_1, B_1) \arrow[d] \\ (A_2, B_2) & Hom(A_2, B_2) \end{tikzcd}$$

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, 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 come to represent version space transforms.

4.2.1 Working example

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

4.2.2 Comma Category

To define the hypothesis space we need the *hom* functor as well as a specific comma category. Using I as the inclusion functor, we can construct the **hypothesis category**, a comma category as such:

$$\begin{tikzcd} Set^{Op} \times Set \arrow[r, "hom(\cdot,\cdot)"] & Set & Set \arrow[l, "I"'] \end{tikzcd}$$

With objects: $((A, B), H, i)$ where $i : I(H) \to hom(A, B)$ monic
and arrows:

$$\begin{tikzcd} ((A_1, B_1), H_1, i) \arrow[d] & & (A_1, B_1) \arrow[d] & H_1 \arrow[d, "\psi"] \\ ((A_2, B_2), H_2, i) \arrow[r, phantom, "\Longrightarrow"] & & (A_2, B_2) & H_2 \end{tikzcd}$$

where ψ is the induced arrow from one hypothesis space to another. Therefore 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.

4.2.3 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 ι 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.

4.3 The examples

Next is the examples. The examples are pairs of the form $(i, o) \in D$. 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.

We can define our examples as such: Given sets A, B and a set $A' \subseteq A$, an example is a partial function $d : A' \to B$. Our examples are $D = \{d : d : A' \to B\}$. An example d is consistent with a hypothesis $((A, B), H, i)$ if for all $f \in H$, $f|_{A'} = d$. A version set consists of all hypothesis consistent with the examples.

4.3.1 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.

4.4 Version spaces

A version space is the set of hypothesis which are consistent with the current set of examples. Prof Easton, not sure how to define the VSA here? We could go with ”the set of hyp. consistent with the examples, but that isn't very categorical. I'm thinking we could use the right side of the comma category we defined since it's currently the identity map (pretty much unused), but I'm not sure what that functor would look like? It would have to map into objects of the form $Hom(A, B)$... I think there is something there but I'm not sure yet. I think the set on the right would probably be D, the examples, so all that's left is finding a functor to map into Set which assigns D to $Hom(A, B)$

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