Version Space Algebras and Category Theory

Bailey Wickham

March 11, 2022

Abstract

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

1 Introduction

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.

Version spaces were first developed as a framework for machine learning in the 80s [3]. In recent years machine learning has taken a differnt approach, yet there is still some recent work on version spaces.

Version space algebras provide a set of hypotheses and a set of examples. The example set is used to produce more relevant hypotheses based on updated information. In an applied context these are ranked by an ordering and then used to generate predictions. 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

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.

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

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 is consistent with Γ_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:

$$\begin{array}{ccc} H \times D & \xrightarrow{\pi_D} & D \\ & \searrow^{\Gamma} & \downarrow{\scriptstyle \iota} \\ & & \Gamma_H \end{array}$$

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:

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

We can transform this into a statement about graphs.

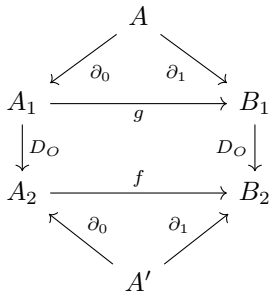
Theorem 3.1. If there is a bijective morphism from the graph of VS_1 to the graph of VS_2, then there exists a version space transform from VS_1 to VS_2

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.

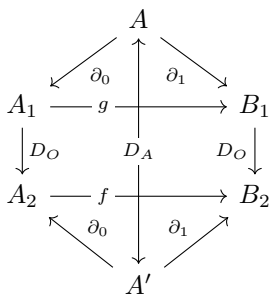
Given a $VS_{H,D}$ consider the graph with objects $O = A \cup B$, and arrows the functions consistent with the examples, $A = VS_{H,D}$. Since $D = A \cup B$ define D_O as such:

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

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 $f \in A'$.



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



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.

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

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

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