Logic in Hilbert space

YKY

January 28, 2021

Summary

- It seems possible to construct a model of the **untyped** λ-calculus in Hilbert space, with function application $f(g)$ implemented as $[\![g]\!] \circ [\![f]\!]$.
- Doing so allows **self-application** of logic predicates (Curry's paradox can be avoided by the fuzzy truth value "don't know")
- The notion of **continuous maps** may be advantageous in machine-learning because **generalization** seems to work best with "continuous" domains (as opposed to maps acting on symbolic logic representations which may be discontinuous).
- Elements in the infinite-dimensional $\mathcal{H}$ can be realized on a computer as **neural networks** (which can be seen as functions $\mathbb{R}^n \to \mathbb{R}^n$ **finitely** generated from sets of weights).

0 Background

In the 1960's Dana Scott constructed a model for untyped λ-calculus, using a domain D_∞ with the property $D_\infty^{D_\infty} \cong D_\infty$. This started off the field known as **domain theory**.

Scott initially believed that such models cannot exist, but later discovered that they can be constructed. In retrospect, this is not surprising because the Church-Rosser theorem demonstrated that the untyped λ-calculus is consistent.

Scott's idea is to begin with an initial domain D_0 and define D_{n+1} to be the function space $D_n \to D_n$.

Thus it is guaranteed, for any domain $d \in D_\infty$, one can always find a function space $d \to d$. Therefore the space D_∞ is isomorphic to $D_\infty \to D_\infty$.

The detailed definition of D_∞ involves building a cumulative hierarchy of infinite sequences, with pairs of operators ψ_n, Ψ_n going up and down levels. For a detailed exposition one may refer to [**Stenlund1972**], Ch.1 §6.

1 Elements in $\mathcal{H}$

The structure of D_∞ is reminescent of Cantor's ordinal number ε_0:

$$\varepsilon_0 = \omega^{\omega^{\omega^{\cdot^{\cdot^{\cdot}}}}} = \sup\{\omega, \omega^\omega, \omega^{\omega^\omega}, \omega^{\omega^{\omega^\omega}}, \dots\} \tag{1.0.1}$$

and this number is "smaller" than the **continuum**, ie. the real line $\mathbb{R}$. This led me to think that models of λ-calculus might be found in the Hilbert space of continuous functions.

But such a Hilbert space would be ∞-dimensional. Next I consider the **neural network** as a function f:

$$\begin{array}{ccc} \mathbb{R}^n & \xrightarrow{f} & \mathbb{R}^n \\ x & \mapsto & y \end{array} \tag{1.0.2}$$

and notice that f and x, y are "unequal" because f can **act on** x, y but not the other way round. This is partly because f is ∞-dimensional whereas x, y are finite-dimensional. So, what if we increase n to ∞, then perhaps x, y would become the same kind of objects as f ? In an informal sense $\mathcal{H}$ can be regarded as $\mathbb{R}^\infty$.

Now we lack the notion of a function **applying** to another function, such as $f(g)$. Since we only need the functions as elements of $\mathcal{H}$, the domains $\mathbb{R}^n$ is somewhat "immaterial". We might as well assume $\mathbb{R}^n$ to be common among all functions, so the function **composition** such as $f \circ g$ always exists. So we define:

$$[\![f(g)]\!] = [\![g]\!] \circ [\![f]\!] \tag{1.0.3}$$

where $[\![\bullet]\!]$ denotes "model of".

2 Logical operations in $\mathcal{H}$

记得在我的 AGI 架构中，$\vdash$ 是用 神经网络 F 实现的。

一般来说，$\Delta \Rightarrow \Gamma$ 就是 $\vdash$ 这个映射 对於 Δ 的一个 **截面** (a restriction of the $\vdash$ map to the domain Δ). 这一点很重要：一个 map 作用在某些元素上，但这些元素 和那个 map 是「同类」的。这其实是逻辑结构的一个 defining characteristic.

The map $\vdash$ 存在於某 infinite-dimensional Hilbert space $\mathcal{H}$. 所以 逻辑命题 也应该存在於同一空间内，而且 空间内的 元素可以 act on 自身。一般来说这是没有可能的，因为 Cantor's theorem 说 $A \neq A^A$. 但如果符合某些 domain theory 的条件则可以有 $A \cong A^A$.

这个做法有两个问题：

- 如何定义 $f(e)$ where $f, e \in \mathcal{H}$
- given an element $e \in \mathcal{H}$, translate it to a (syntactic) logic formula

Need:

- family of maps that is dense
- self-application: maps can act on maps

$$f(g) \quad \rightsquigarrow \quad [\![g]\!] \circ [\![f]\!] \tag{2.0.1}$$

In order to handle tuples like (x, y), we need to expand our domain to allow functions such as $\mathbb{R}^{2n} \to \mathbb{R}^n$ with input dimensions $2n, 3n, 4n, \ldots$ etc. Then an n-ary function can be implemented via (shown here in the binary case):

$$f(g, h) \quad \rightsquigarrow \quad \begin{bmatrix} \llbracket g \rrbracket \\ \llbracket h \rrbracket \end{bmatrix} \circ \llbracket f \rrbracket \tag{2.0.2}$$

where f is of type $\mathbb{R}^{2n} \to \mathbb{R}^n$. With this, we can implement the combinators $\mathbf{S}, \mathbf{K}, \mathbf{I}$ in combinatory logic. The treatment for λ-calculus would be analogous, but I'm too busy to work it out at this time.

Suppose $a \wedge b \Rightarrow c$ and $d \Rightarrow c$, we would like to make the following two definitions of c be consistent with f:

$$\begin{tikzcd} a \arrow[dr, dash] & & \\ & f \arrow[r] & c \\ b \arrow[ur, dash] & & \end{tikzcd} \qquad f(a, b) = c \tag{2.0.3}$$

$$d \;—\; f \longrightarrow c \qquad f(d) = c$$

but the arity of f appears different in the two equations. My solution is to adjoin a zero input to the second definition, that is:

$$\begin{tikzcd} d \arrow[dr, dash] & & \\ & f \arrow[r] & c \\ \varnothing \arrow[ur, dash] & & \end{tikzcd} \qquad f(d, \varnothing) = c \tag{2.0.4}$$

3 Associative attention / recommendation of inference

The word "attention" is used here alternatively, not the same as Attention in BERT or Transformers.

It may be advantageous to use a graph neural network (GNN) as the **state** of our AI system and such that the transition function F maps the current-state GNN to the next-state GNN.

The size of the GNN is the "working memory" size and may be moderately large. So we need an algorithnm to select a subset of nodes in the GNN as **candidates** for applying deduction:

$$A_1 \wedge A_2 \wedge ...A_n \Rightarrow B. \tag{3.0.1}$$

There are $\binom{M}{N}$ ways of choosing a cluster of N nodes from a total of M nodes. Finding such subsets is akin to what **recommendation engines** do, where our problem can be regarded as the recommendation of candidates for logic rules application.

Perhaps an efficient algorithm is to calculate scores of something....

References

欢迎提问和讨论 ☺