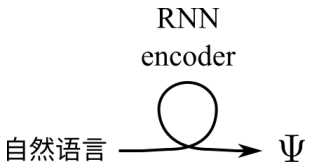
Foundation of AGI

甄景贤 (King-Yin Yan)

May 14, 2018

0 Basic architecture

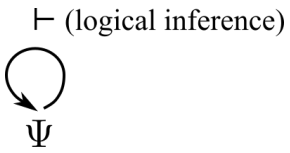
首先，用一个 RNN encoder 将自然语言转换成 knowledge representation Ψ：



(1)

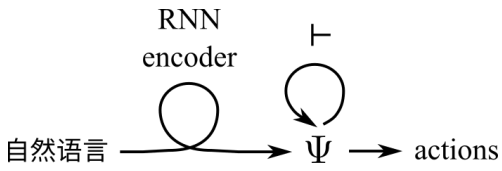
这 **知识表述**是人工智能中最重要的概念，它和数学上的 representation 其实是同一概念，但前者较为广义。

Ψ 的特点是它可以用来做 **逻辑推导**，用 ⊢ 符号表示：



(2)

两方面合起来就是这个基本的 architecture：



(3)

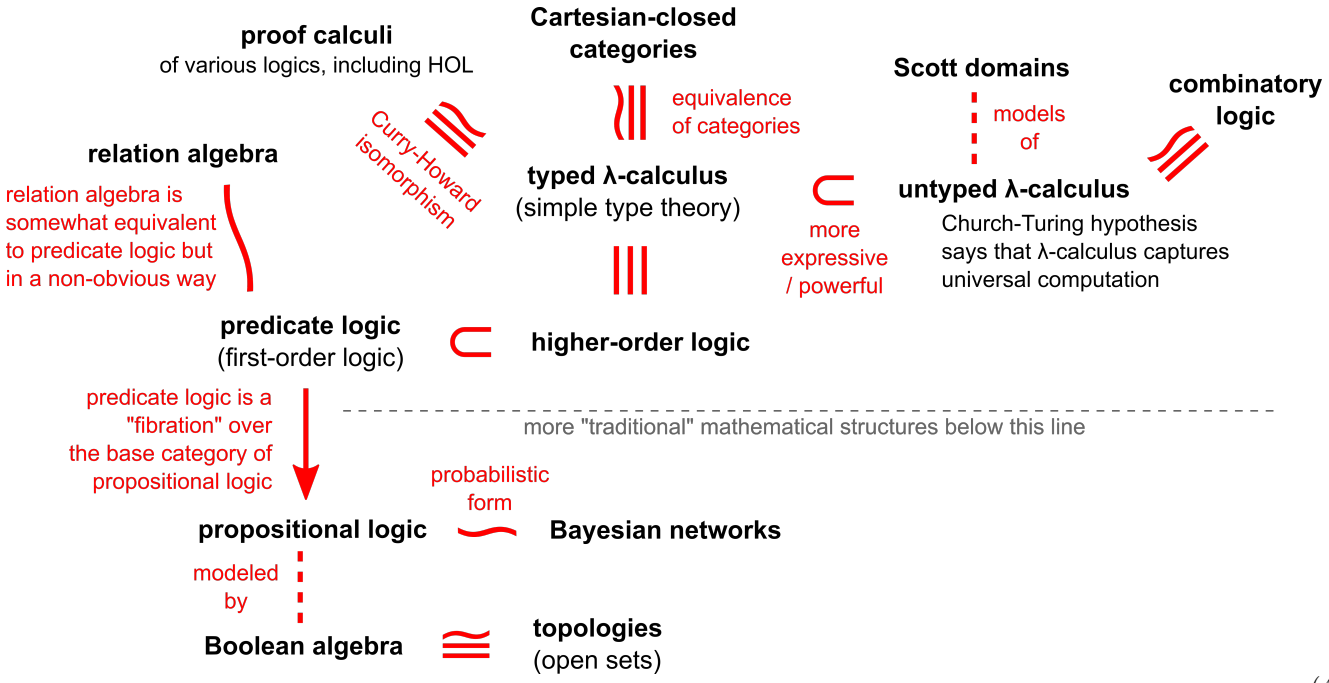
（不排除其他读者可能有截然不同的 architectures，但这是作者最熟悉的）

整个系统是靠 deep reinforcement learning 的 Bellman update 来进行 **学习**。很明显，如果系统的结构复杂，会有 vanishing gradient 的危险。系统的神经网络是庞大的（它包含所有知识），所以，有效率的学习算法，是樽颈问题。

1 Logic

有很多 **逻辑形式**可以用，基本上这是个开放的问题。在较早的 classical AI 时期，人们手动设计某些 knowledge representation schemes，但无论怎样仔细，这些 KR schemes 总会有不尽人意之处。进入「**后结构主义**」时期，我们只是 specify abstract structure，其他细节留给机器学习。而这 abstract structure 的选取，似乎有很多种可能。

以下是各种主要逻辑结构之间的关系：



(4)

在我正在写的书中有更详细介绍。

1.1 A simple logic based on sequence-rewriting

在本文中，我著重介绍一种最简单的逻辑结构。

「我爱你」≠「你爱我」（造成这世界很多痛苦）：

$$\boxed{\text{世界很多痛苦}} \quad a \cdot b \neq b \cdot a \tag{5}$$

换句话说，在 sub-propositional level（由概念组成命题的层次），其结构是一个 **non-commutative** 的东西，这东西或许是 group / semi-group / monoid.

「我爱你 ∧ 你爱我」=「你爱我 ∧ 我爱你」（为方便记忆，找个口号）：

$$\boxed{\text{人生不是绝望}} \quad A \wedge B = B \wedge A \tag{6}$$

换句话说，在 propositional level（命题层次），结构是某个 **commutative** group / semi-group / monoid.

将这两个层次合起来，得到的是一个 group algebra（或者可以是 non-commutative ring, ... etc），这些是 **抽象代数**中最为熟悉的结构，在数学上十分方便。（暂时我还未考虑 distributive law 是否成立这个问题）

补充一点：在机器学习中，为了加快学习速度，会引入一些 **约束**，缩小搜寻空间的大小，此谓之 **inductive bias**. 数学上，搜寻空间通常是一些代数结构，例如 lattice，我们的目的是求原空间的 quotient（商空间），即 mod 掉某 equivalence class，而这 equivalence 可以是某个 symmetry。

再补充一点：加速学习的方法，还有 **transfer learning** 和 **meta-learning**. 其实这二者都是 inductive bias，基本上它们 start with a relatively "free" structure，在某些 **狭窄**的 domain 上学习，由此学习出来的 bias 当成是 **长期的** bias。而我现时的做法是（在抽象代数方面）用人手设计 inductive bias。

TO-DO: 证明这种逻辑的 expressive power is equivalent to first-order logic (or higher-order logic?)（我个人推测它至少等价於 FOL，这证明或许在某些书里已有）

Proof outline:

- FOL terms can be translated as a sum-of-product terms
- how to handle ∀ and ∃ formulas?
- every FOL inference step can be realized as sequence-rewriting of such terms

将逻辑分拆成概念（命题）和知识（推导）

在我设计的这个 representation scheme 里：

$$\begin{aligned} \text{逻辑项 (terms)} \quad &\rightsquigarrow \quad \text{代数结构中的元素 (elements)} \\ \text{逻辑蕴涵 (logical implication)} \quad &\rightsquigarrow \quad \text{代数结构之间的 mapping: element} \mapsto \text{element} \end{aligned} \tag{7}$$

换句话说，将逻辑结构分拆成两部分：元素，和元素之间的 mappings。

代数元素 = **逻辑概念**及其组成的 **命题**。Mapping = 由某些命题 $\to$ 另一些命题。换句话说，the set of mappings 代表系统的**知识**的全部，它是人工智能赖以思考的「引擎」，对应於经典 AI 的 "rules base"。这些 mappings 交给 deep neural network 学习，因为 deep learning 是现时最强的机器学习方法。

换句话说，整体策略是：

- 将逻辑 **概念**和 **命题** embed 到 vector space 上
- 用 deep learning 学习这些 vectors 之间的 mapping

Implementation with symmetric neural networks

Architecture 的细节有待 spell out 出来，但其中最关键的似乎是那 commutative 部分：如何用神经网络处理 commutative（或者可以叫 symmetric）结构？将神经网络 ⧆ 记作 $F(\boldsymbol{x})$，则我们希望有：

$$F(x, y) = F(y, x) \tag{8}$$

x, y 是输入 vector 的两部分。

如果将第一层 weight matrix 造成 **左右对称**，则 symmetry 很容易满足，但这忽略了更深层的 layers，不算是 "deep symmetric neural network"。然而，深层由於有非线性的 sigmoid functions，很难求出满足对称性的 weights 的条件。换句话说，单靠设定神经网络内部的 weights，很难做到 symmetric NN。

最近我间接读到，Abel 在 1826 年提出了一个 symmetric functional equation 的问题，那篇可能是最早的 semi-group 论文，受此启发：

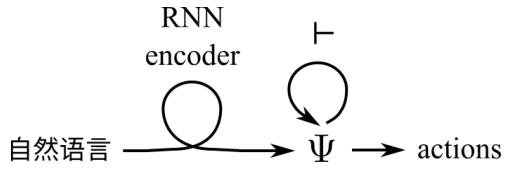
$$F(x, y) + F(y, x) \quad \text{或} \quad F(x, y) \cdot F(y, x) \tag{9}$$

等形式都可以是 symmetric functions，其中 F 是任意神经网络。这解决了关键的一步。

1.2 Higher-order logic

如上所述，逻辑形式的选择有很多可能，包括 λ-calculus 和 combinatory logic。和它们比较，我提出的 simple re-writing logic 可能不够 powerful，这一点有待证明。

重温一下基本 architecture：



(10)

这里 Ψ 的逻辑形式基本上可以是任何结构。接下来我们考虑怎样用 λ-calculus 或 combinatory logic 做 Ψ 的结构....

λ-calculus

基本上，λ-calculus 是一种关於 **函数**的演算法。它可以表示成一种 monoid，其乘法是 function composition. 但 λ-calculus 还包含 α-, β-, η- reduction 的定义，这些法则决定了 **variable substitutions** 的意义，同时也很麻烦，它们似乎不能被纳入上述 monoid 的定义之中。

依据我先前 simple logic 的思路，问题是如何基於 λ-calculus 做到类似 (7) 的分拆，但这似乎不是最好的途径，因为由λ-calculus 通往 logic 的最「自然」途径是 Curry-Howard isomorphism. 根据后者的理论：

$$\begin{array}{rcl} \lambda\text{-terms} & \leftrightsquigarrow & \text{proofs} \\ \text{types} & \leftrightsquigarrow & \text{propositions} \end{array} \tag{11}$$

按照 Curry-Howard 的思想，问题是如何将 **神经网络**引入到这框架中？例如，将所有 λ-terms embed 到 vector space 上，然后用神经网络学习 λ-terms 的演算法，亦即 λ-calculus. 问题是，为什么觉得神经网络能胜任这个任务？

更重要的是：釐清在 Curry-Howard 思想下，λ-calculus 是如何做 **逻辑推导**的？经典 AI 的 knowledge rules 对应於 λ-calculus 或 Curry-Howard correspondence 里的什么？

Combinatory logic

Combinatory logic 等价於 λ-calculus，其逻辑化方法应该根据以上的理论。另方面，或者可以参考 **illative combinatory logic** 的做法。（待续）

肖达 *et al*

Acknowledgements