人工智能的知识表述

甄景贤

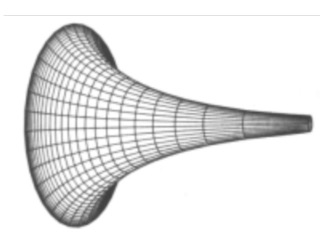
August 12, 2018

Abstract

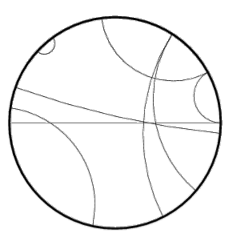
目前（2018 年 8 月）强人工智能的发展，问题已经不再是「能不能做到」，而是到了「哪个方案比较好」的地步。本文介绍知识表述的理论，顺带提出两个方案：A）使用基因算法，上层经典逻辑 + 底层视觉神经网络；B）整个系统用神经网络作用在 graph 记忆体上。

1　什么是 model theory？

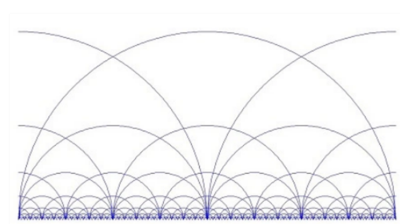
举例来说，hyperbolic geometry（双曲几何）可以「实现」为某些 **模型**：



pseudo-sphere



Poincaré disc

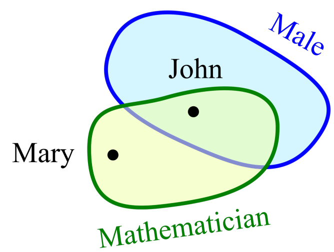


Poincaré half-plane

模型不是唯一的，可以有很多种。

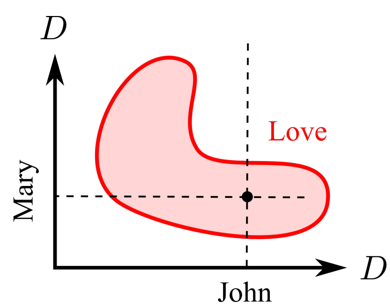
在数理逻辑中，**模型论** 研究的是 syntax / **theory** 和 **model** 之间的 **对偶**。

First-order logic 的 模型 可以用一些 **集合** 及其 **元素** 组成。例如，
John $\in$ Male, John, Mary $\in$ Mathematician：



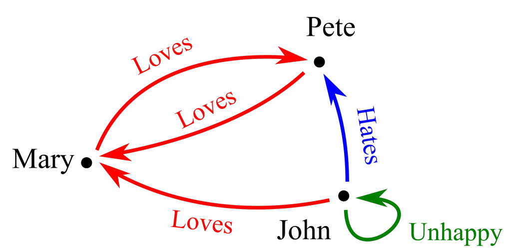
(1)

而 first-order objects（**个体**）之间的 **关系** 是 domain D 的 Cartesian product $D \times D$ 内的一些 **子集**，例如：



(2)

对计算系的人来说，更熟识的 model 是以下这种 relation graph 或 **knowledge graph**：



(3)

这是一个 **directed multi-graph**，或者叫 **quiver**。Quivers 是 代数表示论 (representation theory) 中的重要结构。Quivers 的范畴 $\mathcal{Q}$ 是一个 **topos**（这在会 §6 介绍），基本意思是 <u>它有条件做 first-order logic 的**模型**范畴</u>。[1]

[1]严格来说，hyper-graph 不是 topos，multi-graph 也不是 topos，但当它们变成 directed 则可以。

以上的 knowledge graph 可以简单地转换成 **逻辑式子** 的集合：

```
Loves(John, Mary)
Loves(Pete, Mary)
Loves(Mary, Pete)
Hates(John, Pete)
Unhappy(John)
```
(4)

所以说，逻辑与 graph 基本上是**等价**的。

> 如果 graph 的每条 边 可以包含任意个 顶点，则有 **hyper-graph**。换句话说，hypergraph 的每条 边 $\in \wp(V)$，V 是 顶点集。也可以说，hypergraph 就是 V 的 **子集系统** (set system)。对逻辑来说，这好处是：关系之上可以有关系。
> Hypergraph 可以一一对应於拓扑学上的 **simplicial complex**，可以研究它的 homology 和 cohomology。Simplicial complex 也可以和 **square-free monomial ideals** 一一对应。Square-free 的意思是 x_i 的指数只可以是 0 或 1。后者是 **组合交换代数** (combinatorial commutative algebra) 的研究范围。暂时我不知道这些关联有没有用，详细可参看 [Brown 2013], [Miller and Sturmfels 2005]。

逻辑的 syntactic **theory** 方面，例如可以有以下这个式子（"失恋则不开心"）：

$$\forall x, y.\ \text{Loves}(x, y) \wedge \neg\text{Loves}(y, x) \rightarrow \text{Unhappy}(x) \tag{5}$$

这个式子含有 universal quantification，所以不是 model 的一部分。逻辑上来说，只有 **ground sentences** （没有变量的式子）的集合才可以组成 model，例如 (4) 。

所以，logic **theory** 中的一个式子 可以导致 model 中出现很多 **新的** 顶点和连接。这是 model theory 研究的问题。

例如，每一个自然数 $n \in \mathbb{N}$ 都有 它的 successor $S(n)$。这个函数的存在，导致 model 空间里有一系列 **无穷** 的顶点：

$$\bullet \quad \bullet \quad \bullet \quad \bullet \quad \bullet \quad \bullet \quad \bullet \quad \bullet \quad \bullet \ ..... \tag{6}$$

如果加入这条 **法则**：

$$\forall n \in \mathbb{N}. \quad S(n) \geq n \tag{7}$$

则立即产生无穷多个关系：

$$\bullet \xleftarrow{\geq} \bullet \xleftarrow{\geq} \bullet \xleftarrow{\geq} \bullet \xleftarrow{\geq} \bullet \xleftarrow{\geq} \bullet \xleftarrow{\geq} \bullet \xleftarrow{\geq} \bullet \;..... \tag{8}$$

虽然，在 **日常智能** (common-sense intelligence) 中，似乎比较少出现这种无穷的结构，而更多是 “shallow” 的结构。

值得注意的是，经典逻辑人工智能 (classical logic-based AI) 的知识表述 是分拆成 **rules** 和 **facts** 两部分。前者是带有 $\forall$ **变量** 的式子，后者是 ground sentences。Rules 储存在 KB 内，facts 储存在 **working memory** 内。前者是一个 **theory**，后者可以看成是一些 **“partial” models**。说 partial 的原因是因为它不代表整个 model。事实上 model 是非常庞大的东西，不可能储存在物理系统中。人工智能或大脑只能储存 某些 theories 和部分的 models。人工智能的关键问题是如何找一种良好的 syntax 结构，令 theory 的学习更快、更有效率。

2 Distributive representations

Distributive representation 当然是针对神经网络而言的，因为神经网络是现时最强的机器学习方法（除了我最近开始提倡使用的 **基因算法**）。

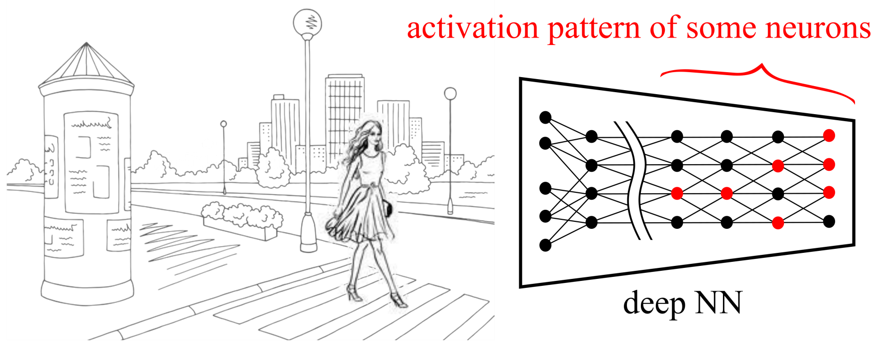
Distributive 的意思是：假设有一个 vector 表示神经网络的输出端有 $n = 10$ 粒神经元：

$$\vec{x} = (x_1, x_2, ..., x_{10}) \tag{9}$$

用 2 进制，每个 $x_i \in \{0, 1\}$，则 $\vec{x}$ 可以分别表示 10 个 “**one-hot**” 的概念。但如果用 distributive representation，这 10 个 bits 最多可以表达 $2^n = 1024$ 个不同的状态 / 概念。所以 distributiveness 可以非常有效地善用神经元。

举例来说，「遇见美女过马路」。这个图像经过譬如 CNN 的处理后，可以得

到一个 **分布式知识表述**：



(10)

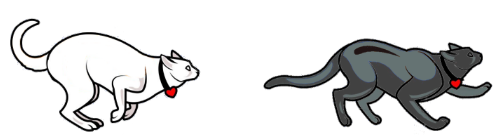
注意那些红点不一定是**最后**那层的输出。这是我心目中的图像，比较 general，不是指某个特定的 implementation。重点是：「美女过马路」是一个 "**neat**" 命题，但在感知过程中，我们会认得很多细节，例如「裙子、高跟鞋、金发、斑马线、路灯」等。这些特徵 (features) 构成整个 representation，至少我是这样理解 分布式表述 的。

经典逻辑表述是由 命题 构成的，其实 features 也可以看成是命题，例如「高跟鞋」可以看成是「有一只高跟鞋在这位置」的命题。逻辑上来说：

$$\boxed{\text{neat proposition}} \quad p \Leftrightarrow \bigwedge q_i \quad \boxed{\text{distributive features}} \tag{11}$$

有时（例如纯文字输入时），知道的只是一个 neat 命题，例如「美女过马路」，并不知道其他细节（例如「金发」），这时仍然可以有分布式表述，但那些特徵会是比较抽象的。

考虑「白猫追黑猫」这个图像：



(12)

「猫」的概念需要出现 **两次**，但神经网络内对应於「猫」的特徵只有 **一组**（除非有两个重复的可以表示任何概念的 modules，但很浪费）。换句话说，现时的 CNN 没有「巡迴 (traverse)」视野域的能力；它不能辨别和描述物体之间的**关系**。我很难想像一个 "monolithic" neural module 怎样可以做到这功能。似乎必须将命题表述成一连串 概念 的 **时间序列** (time sequence)，某种 **短期记忆** (short-term memory)。

3 为什么 神经网络 做不到 substitutions?

考虑上节讲过的 逻辑 rule（“失恋则不开心”）：

$$\forall x, y.\ x \heartsuit y \wedge \neg y \heartsuit x \rightarrow \frownie x \tag{13}$$

这个 rule 的 **前件** (antecedent) 要成立，必须 两次出现的a **相等**、两次出现的b **相等**：

$$a \heartsuit b \wedge \neg b \heartsuit a \tag{14}$$

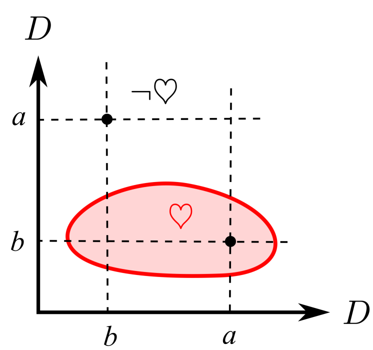
而且，要产生正确的 **后件** (consequent)，需要从前件中将 a **copy** 过来：

$$a \heartsuit b \wedge \neg b \heartsuit a \rightarrow \frownie a \tag{15}$$

这两个动作（**compare** 和 **copy**）都是用神经网络很难做到的。但它们是 variable substitution 的本质，也是 谓词逻辑 麻烦之处。换句话说，很难用一个 monolithic 的 end-to-end 神经网络 一口气完成这两个动作：

$$a \heartsuit b \wedge \neg b \heartsuit a \xrightarrow{\qquad\qquad} \frownie a \tag{16}$$

基於集合论的**几何图像**是这样的：



$$\Rightarrow \frownie a \tag{17}$$

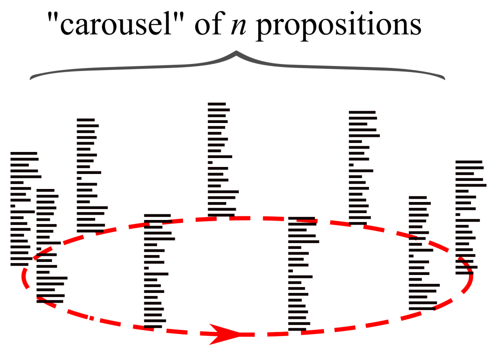
$\in$ 很容易用神经网络解决，例如可以定义 $\heartsuit$ 为一个神经网络：

$$\heartsuit(\vec{x}) = \begin{cases} 0 & \vec{x} \notin \text{region} \\ 1 & \vec{x} \in \text{region} \end{cases} \tag{18}$$

但即使这样，仍然馀下一个 pattern matching 问题：

$$\begin{aligned} \vec{p}_1 = (\vec{a}, \vec{b}) \quad &\in \quad \heartsuit \\ \vec{p}_2 = (\vec{b}, \vec{a}) \quad &\in \quad \neg\heartsuit \end{aligned} \tag{19}$$

事实上，要做到以上功能，似乎必需有一个 **动态的记忆体**，它接收新来的元素时，会对记忆体中其他元素逐一 **比较**，而且具备 **复制** 功能。例如我考虑过一个像「迴转木马」的时间序列机制（每个 代表一支 distributive vector）：



(20)

但仍未解决 compare 和 copy 的问题。总之，明显地很麻烦。

4 Genetic algorithm

这是 plan A。

Substitution 的麻烦令我想到 放弃用 neural network 直接处理逻辑，而是用 hybrid 的神经 / 逻辑混合：视觉神经用 deep neural network，到高层次转用符号逻辑表述，后者用 genetic algorithm 做学习。

首先需要一个 logic-based **rule engine**，它负责 forward-chaining（正向逻辑推导），这完全是经典 AI 的范围。例如 经典的 Soar architecture [Carnegie-Mellon 大学] 就是一个 rule-base 引擎。还有我这一代的 OpenCog, OpenNARS, 和我写的 Genifer[2]，或许还有更新一代的 AGI 系统（?）

经典逻辑 AI 的樽颈问题是 rules 的学习算法太慢，

[2]Genifer 仍未到达 production stage.

- plan A：继续用经典逻辑引擎，用 genetic algorithm 做学习算法
- plan B：将逻辑记忆映射到向量空间，再用神经网络学习逻辑 rules

基因算法的 population 是由个别的 rules 组成，但 winner 并不是单一条 rule，而是一整套 rules（最高分的 N 个）。这叫 **cooperative co-evolution**（COCO）。

输入和输出是 logic formulas，其实更易处理。

整个系统仍然是基於 reinforcement learning 的，但不需要直接做 RL，因为那些 rules 其实就是 **actions**，每条 rule 的 probabilistic strength 就像 Q-learning 中 Q 值的作用。

5 Graph neural networks

这是 plan B。

6 Categorical semantics

Categorical semantics 是用 category theory 表达的 model theory。以下内容主要来自 [Caramello 2018] 这本新书的第一章。更经典的参考书是 [Goldblatt 1984, 2006].

不同的 logics 可以透过 **proof theory**（它研究的是 *syntactic* rules of deduction）定义：

algebraic logic	no additional rules	
Horn logic	finite $\wedge$	
regular logic	finite $\wedge$, $\exists$, Frobenius axiom	
coherent logic	finite $\wedge$ and $\vee$, $\exists$, distributive axiom, Frobenius axiom	
geometric logic	finite $\wedge$, infinitary $\vee$, $\exists$, infinitary distribution axiom, Frobenius axiom	(21)
first-order intuitionistic logic	all finitary rules except law of excluded middle	
first-order classical logic	all finitary rules	

举例来说，**algebraic theory** 的意思是：它只有一个 relation $=$，而所有 axioms 都是 $s = t$ 这种形式。

还有这些 deduction rules 的例子：

$$\boxed{\wedge \text{ rule}} \quad \frac{\Phi \vdash \Psi,\ \Phi \vdash \chi}{\Phi \vdash (\Psi \wedge \chi)} \tag{22}$$

$$\boxed{\exists \text{ double rule}} \quad \frac{\Phi \vdash_{\vec{x},y} \Psi}{\exists y\, \Phi \vdash_{\vec{x}} \Psi} \tag{23}$$

$$\boxed{\text{Frobenius axiom}} \quad \Phi \wedge \exists y\, \Psi \vdash_{\vec{x}} \exists y\, (\Phi \wedge \Psi) \tag{24}$$

Tarski 的模型论 将 first-order syntax「对应」到 集合论 的 **结构** (structures) 上。由此推广，不同的 逻辑 syntax 对应於不同的 结构范畴：

categories with finite products	algebraic logic	
Cartesian categories	Cartesian logic	
regular categories	regular logic	
coherent categories	coherent logic	(25)
geometric categories	geometric logic	
Heyting categories	first-order intuitionistic logic	
Boolean coherent categories	first-order classical logic	

"Geometric" logic 的意思来自 **geometric morphisms**，它可以粗略地理解为两个 topoi 之间的映射，类似於 **continuous maps** between topological spaces。在 1963 年左右，topos 的概念独立地来自几个不同的发源地：Alexander Grothendieck 在代数几何方面发展的 sheaf theory，和 F William Lawvere 用范畴论重新表述集合论，还有 Paul Cohen 的 forcing 理论（后者用来解决 **连续统假设**）。Sheaf 的意思是：在一些 open sets V_i 上定义的物体，它们在 overlap $V_i \cap V_j$ 上是吻合的，即可以 "collate"。二战后，Leray，接著 Cartan，用 open sets 的方法定义了 sheaf。其后 Lazard 用 étale 定义 sheaf（基本上是一个 **functor**），后者是 topos 理论的主要动机。Grothendieck 将 sheaf 应用在 topology (cohomology) 上，而后 Jean-Pierre Serre 发现它也可以用在代数几何上，他们和其他合作者 写了 1623 页的巨著《SGA IV》，重大影响了代数几何的发展，导致 1974 年 Deligne 解决了 Weyl 猜想。但我暂时不熟悉代数几何，所以不太清楚 Grothendieck 他们做了什么.... 详细可参看 [MacLane and Moerdijk 1992] 一书。

在拓樸空间上，一个 open set U 的 complement 是 closed 而且未必 open，所以如果局限在 open sets 之内，则 U 的 "negation" 应该定义为 "the interior of its complement". 这导致 U 的「双重否定」不一定等於 U，换句话说，<u>the algebra of open sets follows **intuitionistic logic**</u>, such an algebra is called a **Heyting algebra**. （参考书同上）

Topoi 之间有两种 morphisms：**geometric morphisms** 和 **logical functors**. 前者保持「几何结构」，后者保持逻辑中的 元素 (elementary) 和 type theory，所以有 **elementary topos** 的定义。后者的特点是它有 **sub-object classifier** Ω。Ω 是一个特殊的 object，例如 $\{\top, \bot\}$，代表 **真 假** 二值。用 Ω 可以定义 sub-objects 亦即集合论中的 **子集** 概念。举例来说，"Love" 是一个在 $D \times D$ 内的 **关系**，D 是所有「人」的集合。可以将 $D \times D$ 看成是 full relation，则 Love $\subset D \times D$ 是它的**子集**。换句话说，这是 elementary topos 可以用来做 relation algebra 或 first-order logic 的 **模型** 的原因。（参见 [Goldblatt 1984, 2006]）

以下是根据 [Caramello 2018] Ch.1 整理出来的一张关系图，但我暂时还不

太熟悉范畴论的概念，所以也不完全理解：

$$
\begin{tikzcd}
 & \textbf{categories} \arrow[d, "\text{has all finite limits}"'] & \\
 & \textbf{Cartesian categories} \arrow[d, "\text{stable under pullbacks}"'] & \\
 & \textbf{regular categories} \arrow[d, "\text{every subobject lattice has finite unions}"'] \arrow[dddr, dashed, "\substack{\text{well-powered,} \\ \text{subobject lattice has arbitrary unions}}"] & \\
 & \textbf{coherent categories} \arrow[d, "\text{pullback functor has a right adjoint}"'] & \\
 & \textbf{Heyting categories} \arrow[dl, "\text{every subobject is complemented}"'] \arrow[dd] \arrow[dr] & \\
\textbf{Boolean categories} & & \textbf{geometric categories} \arrow[d] \\
 & \textbf{elementary topos} \arrow[r] & \textbf{Grothendieck topos}
\end{tikzcd}
\tag{26}
$$

7 Domain theory

λ-calculus 和 combinatory logic 都是可以表达任意 **函数** 的形式。如果全体函数的 domain 是 D，而 由 $D \to D$ 的函数的个数是 $|D^D|$，则根据集合论的 Cantor's theorem，$|D^D|$ 必定大於 $|D|$，即使 D 是无穷依然成立。换句话说，λ-calculus 和 combinatory logic 不可能有 models。这结论是非常令人不安的。但在 1971 年，这个问题被 Dana Scott 和 C Strachey 解决了，开创了 **domain theory**。

以下内容主要来自 [Vickers 1989]，是一本很易懂的书，还有更新和更详尽的 [Goubault-Larrecq 2013].

8 Group theory

References

Brown (2013). *Discrete structures and their interactions*. CRC Press.

Caramello (2018). *Theories, sites, toposes — relating and studying mathematical theories through topos-theoretic 'bridges'*.

Goldblatt (1984, 2006). *Topoi — the categorical analysis of logic*.

Goubault-Larrecq (2013). *Non-Hausdorff topology and domain theory — selected topics in point-set topology*. Cambridge new mathematical monographs 22.

MacLane, Saunders and Ieke Moerdijk (1992). *Sheaves in geometry and logic – a first introduction to topos theory*. Springer.

Miller and Sturmfels (2005). *Combinatorial commutative algebra*. GTM 227.

Vickers (1989). *Topology via logic*.