

神经网络中直接注入知识

甄景贤 (King-Yin Yan)

General.Intelligence@Gmail.com

October 24, 2017

Abstract. 在人工智能历史上，迄今为止仍未有一个快速的学习算法，可以同时「像教孩子那样」直接注入知识。经典逻辑 AI 可以直接写入知识，但其学习算法太慢。深度神经网络的学习算法很快，但它是「黑盒」。本文提出一个解决方案：让神经网络直接作用在它自身的 weights 上。

0 Introduction

本文提出的 AGI architecture 揉合了三个结构：

- (1) **reinforcement learning** (RL)
- (2) classical **logic-based AI** (LBAI)
- (3) **deep neural network** (DNN)

其中 (1) 是智能系统根据奖励学习的架构，这架构本身并没有什么内容，单靠它学习人类水平的智能，会太慢，所以要加入 (2) 的结构。而 (3) 是目前为止最有效率的机器学习算法，它本身不是 AGI 的内在结构，我们只是将它应用到 AGI 系统的关键部分。

以下分述这三个成份。

1 Deep neural network

首先讲讲神经网络。它只是一个技术，它不是 AGI 内在的结构，我们只是将它应用到 AGI 上。So let us first get this out of the way.

所谓神经网络就是：

2 Reinforcement learning

3 Logic-based AI

这部分是最复杂的结构，也是一般数学研究者比较不熟悉的。

智能系统需要有能读 / 写它内部的知识。例如说，一个比较蠢的智能系统可以用 sequence-to-sequence 的方式将中文翻译成英文：

$$\boxed{\text{“中文句子”}} \xrightarrow{F} \boxed{\text{“英文句子”}} \quad (1)$$

F 代表系统的函数。但系统并不真的明白句子的意义，句子只是「水过鸭背」地流过系统。一个更聪明的系统是：句子可以进入到 F 里。

4 Applications

DKI (direct knowledge injection) is useful in:

- learning by instructions, or “learn by being told”
(a technique crucial to accelerating the learning of human knowledge)
- belief revision / truth maintenance
(the most challenging and highest-level task in logic-based AI)

举例来说，小孩子的行为是由他内部的知识决定的，「知识决定行为」。

- 当小孩子看到一个成人做的动作，他会模仿那动作。



(2)

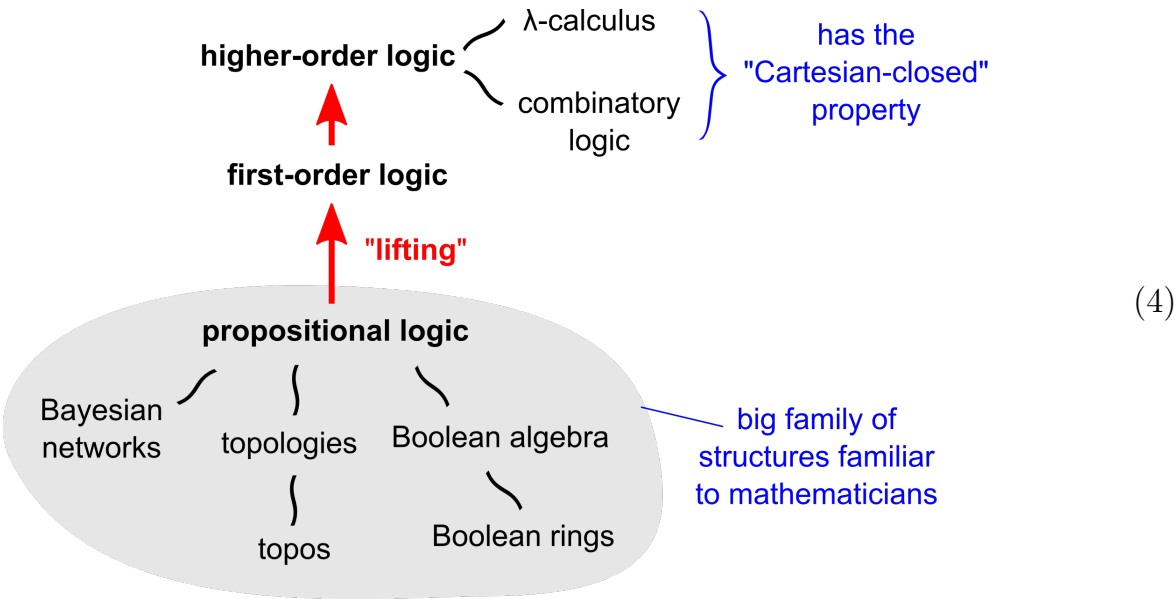
- 或者小孩子听到一句说话：「不要吃污糟食物」，他明白了那句说话的意思而改变行为。
- 或者「今天高考放榜了，这本教科书可以丢进垃圾桶」，这句话应该由 working memory x 进入到 $\boxed{\text{KB}}$ 里面，影响日后的行为（eg，以后不会再见那本书）。

这些例子都涉及到将「感觉资料」放进 F 里面：

$$\boxed{\text{sensory data}} \hookrightarrow F \quad (3)$$

5 Cartesian closure

在数理逻辑中，命题逻辑和一阶逻辑、高阶逻辑之间存在一个 gap。一般来说，数学家比较熟悉命题逻辑，因为它等价於很多在传统数学中常见的结构例如 Boolean algebra 等。在拓模中， \cup 和 \cap 对应於 \vee 和 \wedge ，所以拓扑也可以看成是逻辑的一种形式。另外，我们熟悉的 **Bayesian network** 也是命题逻辑的扩充，亦即在命题上附加了 **概率**。然而，一阶逻辑的结构比较麻烦，在数学上是较偏门的课题。用计算机科学的术语，将命题逻辑的技巧转移到一阶逻辑，这动作叫“lifting”，於是会产生例如 first-order Bayesian network 这类较复杂的东西，而高阶逻辑则更复杂。注意：命题逻辑的复杂性是 NP-complete 的范围，但高阶逻辑的复杂性是 undecidable 的，因为高阶逻辑可以做 Turing machine 的所有工作，而 halting problem 是 undecidable 的。



在人工智能中，我们必须用到一阶逻辑或以上，因此 lifting 是一个很头痛的问题。高阶逻辑的“power”似乎来自於一个特性，即 Cartesian-closure，它是高阶逻辑的本质。



Cartesian-closed 是指在某个範畴内，对任意的 A, B ，都必然可以找到它们的：

$$\boxed{\text{product}} \quad A \times B \quad \text{和} \quad B^A \quad \boxed{\text{exponentiation}} \tag{5}$$

在逻辑中这对应於：

$$A \wedge B \quad \text{和} \quad A \rightarrow B \tag{6}$$

DKI requires the functional closure $\mathbb{X} \simeq \mathbb{X}^{\mathbb{X}}$ which yields a **Cartesian-closed category** (CCC).

举例来说，「吃了污糟的食物会肚痛」是一个句子，它经由  进入 mental state x ，变成 proposition。但我们希望这逻辑命题变成  的一部分。 F is the state-transition function:

$$x_{n+1} = F(x_n) \tag{7}$$

where

$$F = \boxed{\text{KB}} = \text{⌘}$$

x = state

An individual logic rule is a restriction of F to a specific input; Perhaps I could call such elements “micro-functions”.

$F \equiv \boxed{\text{KB}}$ is the “union” of micro-functions:

$$\boxed{\text{KB}} = \bigcup f_i \quad (8)$$

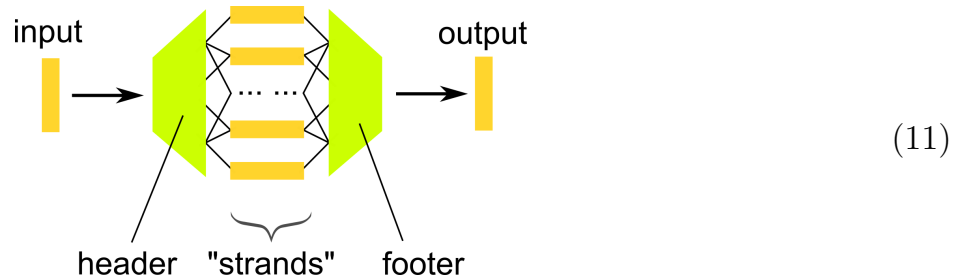
At this point the meaning of \bigcup is unspecified yet. F is the sum total of objects like x :

$$F = \bigcup x_i \quad (9)$$

但 F 是一个神经网络，它的一般形式是：

$$\boxed{\text{output}} \quad x_{n+1} = F(x_n) = \bigcirc \overset{1}{W} \bigcirc \overset{2}{W} \dots \bigcirc \overset{L}{W} x_n \quad (10)$$

L = total number of layers. 由於各层的非线性「纠缠在一起」，表面上无法将神经网络「分解」。直到笔者受了 David Ha *et al* 提出的 PathNet [1] 理论所启发，PathNet 是由一些较小的神经网络 modules 组成，所以或许可以建构如下形式的「丝状神经网络」：



这些「丝条」可以是简单的神经网络，例如每个的宽度或深度很小，因而可以用较短的 weights vector 描述。正是因为这原因，一个本身可以作为神经网络的输入。但整个神经网络 F 无法输入自己，因为根据 Cantor's theorem, $\mathbb{X} = \mathbb{X}^{\mathbb{X}}$ 是不可能的。

Let $\overline{F} = \text{header}$, $\underline{F} = \text{footer}$, $f_i = \text{strands}$, then (abusing the \bigcup notation):

$$F = \overline{F} \circ \bigcup f_i \circ \underline{F} \quad (12)$$

每个 大约对应於逻辑上的一个命题（proposition, 可以是条件命题或普通命题）。

读者或许会质疑，这个「条状」结构为什么一定要设计成这样？其实我也觉得这个设计不够 elegant，甚至不太肯定它会不会 work。在 §4 - §5.2 我们会介绍一个数学上更优美的做法。

6 Overall architecture

For reference, the architecture for **visual recognition** is:

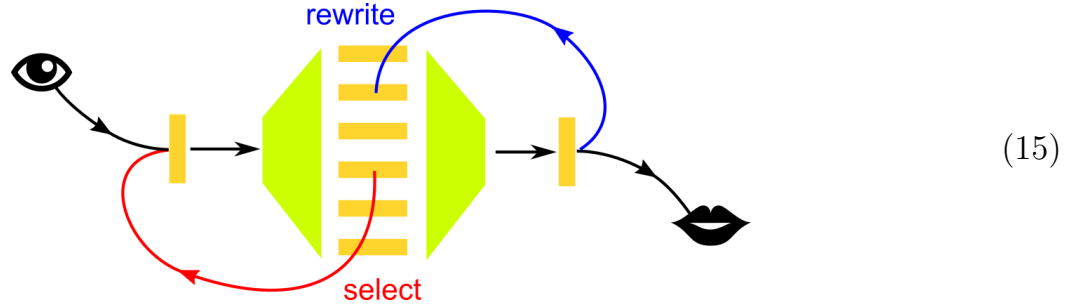






Our basic AGI architecture is:



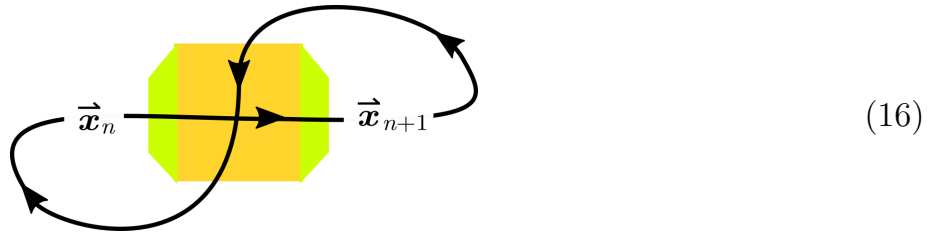
⊗ = [deep] neural network, trained via **reinforcement learning**

The overall **recurrent** setup operates like this:



注意：👁 的原始输入不可以直接写入 ，因为  会变成  = weights，而直接写 weights 的后果当然是灾难性的。换句话说， 的结构是要用 emergent（涌现）的方法 learn 出来，不能被外界的输入干扰。👁 和 👄 的输入 / 输出要透过某些神经网络的 mapping 间接地做。

Viewing the “information flow” in a simplified way, we notice a “second” pass through the network’s internal weights:



这种操作上的结构在经典逻辑 AI 是「免费赠品」，但似乎还未有人提出过神经网络的做法。

对应於经典逻辑 AI:

$$\text{KB} = \text{KB} \quad (17)$$

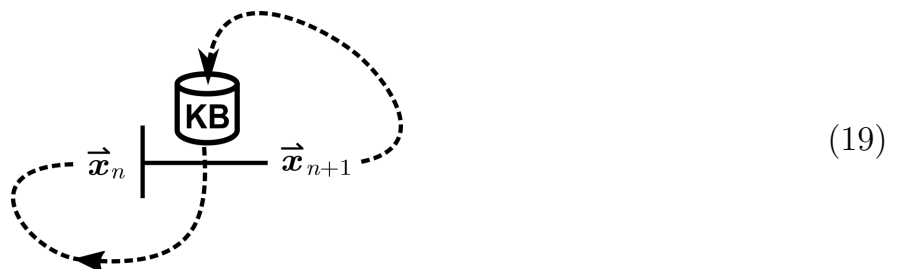
- The **horizontal pass** represents using the KB for logical inference (thinking), ie:

$$x_n \cup \text{KB} \vdash x_{n+1} \quad (18)$$

- The **vertical pass** represents reading/writing information to/from KB :

- **Update:** x 是 KB 的一部分，所以 x_{n+1} 改变了， KB 也要 update。
- **Read:** x = working memory 会因为 注意力 (attention) 而改变，所以 x_{n+1} 并不直接进入下一轮的 iteration，而是先经过 KB 的 **attentional change**。

In logic-based AI this workflow has always been standard (but not made explicit):



7 What is required of F ?

We now try to explain the meaning of

$$F = \bigcup x_i \tag{20}$$

Our goal is to organize the 's into a deep network. What are the most general *desiderata* for such a function F ?

- (1) $F(x; \theta)$ is a function of x parametrized by θ .
- (2) the parameters θ is organized hierarchically with the “deep” property, ie, high-level θ_i 's have higher “degree”.
- (3) $F(x; \theta)$ is capable of universal function approximation.
- (4) x can be put into θ , ie, θ is a collection of x 's.
- (5) F encodes **logical consequence**.

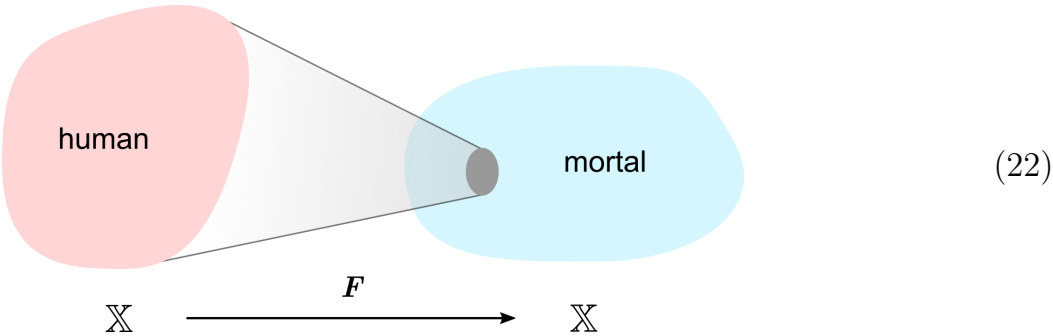
The last condition (F5) is hardest to satisfy, but there is an informal argument that may justify it. For example, suppose:

$x = \text{“all men are mortal”}$
 $x_0 = \text{“Socrates is a man”}$
Then the expected output should be:
 $F(x_0) = \text{“Socrates is mortal”}$

$\text{is put into } F = \boxed{\text{KB}}.$
 is the new input.

(21)

凭什么认为 F 能满足类似上面的要求？可以将一个 conditional proposition 看成是一个 mapping，它将 source domain \mathbb{X} 的某个区域映射到 target domain（也是 \mathbb{X} ）的某个区域，例如：

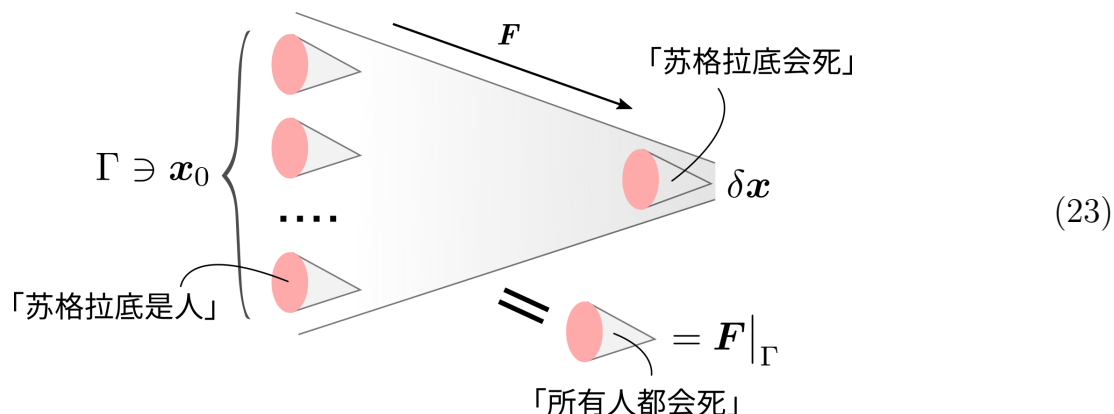



而我们有信心 F 能够表达这个 mapping 的原因，正如在机器视觉中，类似的 F 里面有些神经元可以辨认「眼、耳、口、鼻」等 features，原理是一样的。换句话说，「X 是人 \rightarrow X 会死」这句条件命题，其实和负责辨认「眼」这个 feature 的那些神经元，本质上是沒有分別的。

8 Decomposition of F

回顾一下：经典 AI 中， $\boxed{\text{KB}}$ 装著一堆 logic formulas，我们可以直接写入 / 读出它们。但现在 $\boxed{\text{KB}}$ 变成了一个函数 F ，那些逻辑法则隐藏在 F 这个 mapping 里面，需要某种方法将它「分解」。

换句话说，我们需要的是将 F 分拆成一些小部分，亦即是 F 对於某个输入 x （及其邻域）的 **截面** (section, or restriction)。假设获得了这些截面函数之后，我们再需要一个方法，将一个截面「放进」 F 里面。





x_0 = current state, 它由若干个  = 命题构成。

(要透彻理解上面这幅图，需要熟悉命题逻辑和谓词逻辑的几何化，这在我较早前的论文 [2] 有讲述。那篇论文有些细节需要更新，但基本要点没变。)

δx 的意思是指，将当前状态由 x 改写成 $x + \delta x$ 。换句话说，原本设定的是 $F: x \mapsto x'$ ，但实际上我们 implement 的是 $F: x \mapsto \delta x$ 。因为逻辑中 \vdash 的特性是它 每次只改变一个（或少数的）命题。这也是一种 restriction，即原本 “free” 的函数空间变成某个 subspace。

经过 update: $x'_0 = x_0 + \delta x$ 之后， x'_0 也会「忘记」它里面的一个 / 几个命题，保持 x 的长度不变。

更重要的是： F 的 restriction 也是一个命题。换句话说，图中的大三角形也是一个 。它将 x_0 的邻域 Γ 映射到 δx ，而 x_0 本身也是由其他  组成的。这 restriction 记作 $F|_{\Gamma}$ 。

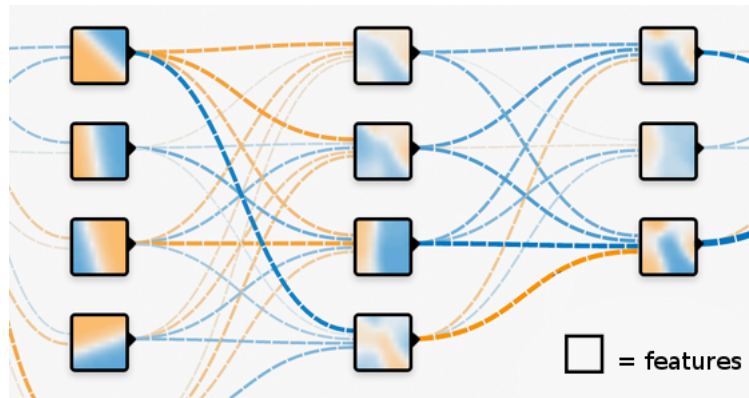
(下一步会将 δx 放进 F 里，但 δx 这个函数的 source 邻域并不是 Γ 。这些细节初读时可以不埋。)

8.1 「责任集」(“responsible set”)

但是，当 F 是一个神经网络时，怎样获得「截面」？

F 这个 mapping 是由它的 parameters 决定，亦即 weights。当输入 = 某个 x_0 时，每个 weights 的贡献不同，只有一个 subset of weights 「负责」这个截面的输出，我将这个 subset 称作「责任集」(responsible set)。

我的想法是受到 TensorFlow Playground 的这图片启发的：



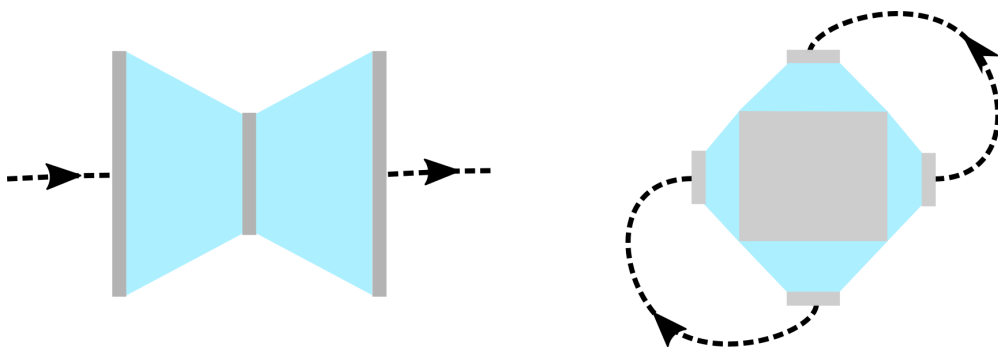
(24)

可以看到 weights 的大小不同，它们对函数 F 在输入 x_0 的贡献也不同。What we need is the set of weights activated by the specific input x_0 , in other words, they are the set of **outputs** (or “activities”) of all neurons. Let’s denote it as $W(x)$.


后来我发现了 [Courbariaux, Bengio, & David 2015] 提出的 “BinaryConnect: training deep neural networks with binary weights” [3]，他们指出，可以将 weights 限制为 $\{-1, 1\}$ 两个值，但 gradients 仍然保持 precision，这种神经网络仍然有 state-of-the-art 的表现。

8.2 Compression

现在问题是怎样压缩那责任集里面的 weights。Joseph Cheng 提议用 auto-encoder，似乎是一个很好的解决办法。左图是典型的 auto-encoder，它的 hidden layer 较小，特徵「被逼」压缩到较小的空间。在我们的 architecture 上加上 auto-encoder，则变成右图的「8 字形 architecture」：



(25)

但有个技术上问题：垂直方向的那个  网络需要连接到中央网络的所有 weights，但这些 weights 数目太多。

Now we need a way to **compress** $W(x)$ to prune out the low-contribution (ie, small) weights. The Fourier (or wavelet) transform is a good candidate because it can compress $W(x)$ to a fixed-length vector, which can then be used as inputs to the neural network F .

The compression (which is a projection, P) is performed via:

$$P F = \sum_{i=1}^N \langle F, \Psi_i \rangle \Psi_i \quad (26)$$

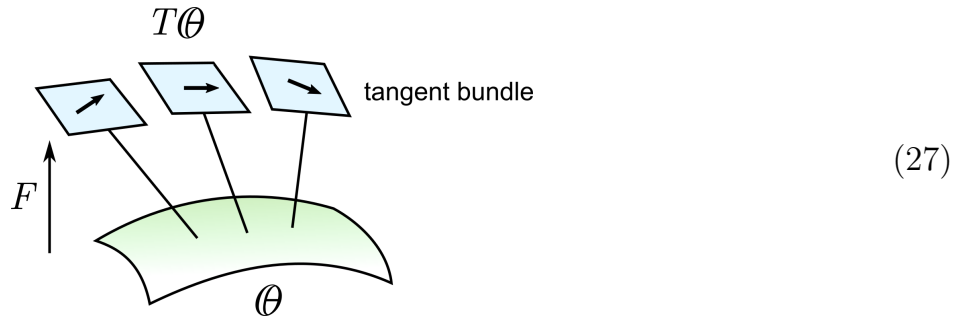
where $\{\Psi_i\}_1^N$ is the basis set.

The Fourier sequence basis set tends to 0: I think the implication is that, for sufficiently smooth functions, the high-frequency contributions of a Fourier sequence would tend to 0. This means that if we chop off the sequence at some number n , we get an approximation, ie, compression. However, this form of compression is weak because the error does not decay fast enough. A better scheme is to choose the first k -largest coefficients; This way, the error decays exponentially. The price we pay is that the compression scheme is no longer **linear**, in the sense that if \hat{f} and \hat{g} approximate 2 signals f and g , then the sum of the signals $(f + g)$ may no longer be approximated by $(\hat{f} + \hat{g})$.

9 几何结构

[此段对熟悉微分几何的人或许有帮助， 否则可以略过。]

首先我们有一个很 standard 的 Hamiltonian 力学系统 / 控制系统的结构：

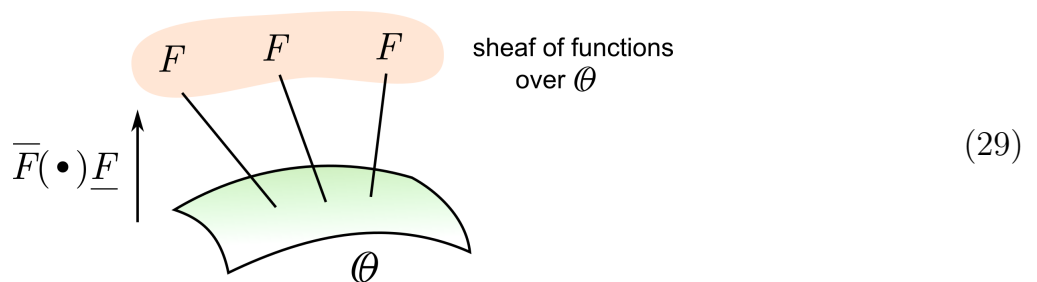


$x = \text{working memory} \subset \theta$, θ 代表整个 \mathbb{KB} 的状态，而 $\theta \in \Theta$ ，后者是所有可能 \mathbb{KB} 的空间。

$$\dot{x} = F(x) \quad (27)$$

是系统的状态方程。换句话说，在思维空间 Θ 中的一个点就是思维状态 $x \subset \theta \in \Theta$ ，而 F 给出的是这个点在思考过程中的「运动速度」= \dot{x} 。换句话说， F 定义了一个 vector field，它是思维空间中思维的“flow”，或者可以叫作「理性流」。每个点的速度属于流形 Θ 上的 tangent space，他们的总和就是 tangent bundle。而 tangent bundle + base manifold (亦即「位置 & 动量」) 构成系统的「相空间」(phase space)。

另外，特别地，有这个 sheaf of functions 的结构：



换句话说，给定 $x \in \Theta$ ，我们可以透过

$$F = \bar{F} \circ (x \subset \theta) \circ \underline{F} \quad (29)$$

$$(30)$$

得出 F ，而这个 F 再给出对应於这点的 \dot{x} 。

注意 (27) 和 (29) 是两个不同的结构，只是它们的 base manifold 相同。

特别之处在於 F 是由参数 $x \subset \theta \in \Theta$ 确定的（因为 x 是 $\theta = \boxed{\text{KB}}$ 的一部分，而所有可能的 $\boxed{\text{KB}}$ 属於思维空间 Θ ），换句话说：

$$F(x) \equiv F_{\theta}(x) \equiv F(x; \theta) \quad (31)$$

这和经典控制并没有抵触，因为经典理论中， F 也是位置 x 的函数。更确切地说，位置空间其实是由 $\theta \in \Theta$ 决定的， x 只是 θ 的一部分。

In general, the differential version seems to be a bad idea, as the increments of x is small, and the $\boxed{\text{KB}}$ may be overwhelmed by recent memories of x , which is a wasteful usage of weights.

10 Conclusion

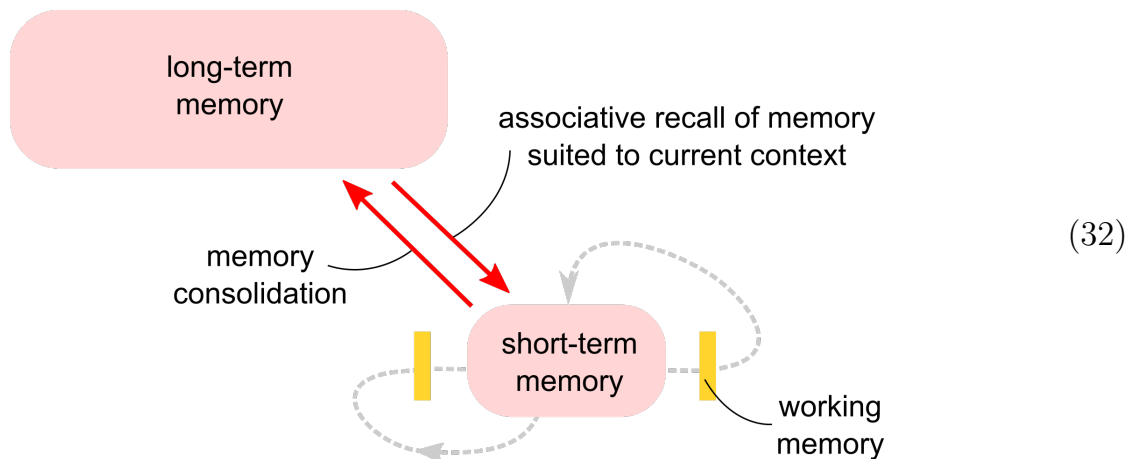
其实我自己也怀疑，这个 architecture 是否太复杂了，究竟值不值得做？而暂时我只能说，它符合了两项非常难达到的要求：

- $\boxed{\text{KB}}$ is represented as a neural network (that admits a fast learning algorithm);
- Symbolic knowledge can be directly **injected** into $\boxed{\text{KB}}$.

11 Future direction

There may exist some variations and improvements under this general architecture.

Another problem is the design of the **memory system**, in particular **episodic memory**. This is the general idea, but details are still undecided:



Acknowledgements

Thanks to David Ha and his co-authors for their PathNet idea.

Bibliography

- [1] Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A. Rusu, Alexander Pritzel, and Daan Wierstra. Pathnet: Evolution channels gradient descent in super neural networks. *CoRR*, abs/1701.08734, 2017. URL <http://arxiv.org/abs/1701.08734>.
- [2] King Yin Yan. A bridge between logic and neural. (to be submitted AGI-2017).
- [3] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. *CoRR*, abs/1511.00363, 2015. URL <http://arxiv.org/abs/1511.00363>.