

# Genifer 5.3 theoretical notes

YKY (甄景贤)

August 10, 2015

## 1 Top-level architecture

在最高层面上，RL 控制 RNN：



可以想像，RNN 是 RL 的 “mental model”；换句话说，我们的 RL 比普通 RL 有更复杂的内部模型。

余下的工作是，定义 RL 的四个元素：states, actions, rewards, policy。

## 2 Unification of RL and RNN

在统一 RL 和 RNN 的时候，我发现它们之间的关系非常简单：

从强化学习的角度来看，我们需要学习的是这个 policy 函数：

$$\text{policy} : \text{state} \xrightarrow{\text{action}} \text{state}'$$

而  $K$  可以看成是我们的 **mental state**，那么 RL 的 **action** 就是将  $K$  变成  $K'$  的作用，而那正是  $D$ ！

$$K \xrightarrow{D} K'$$

换句话说， $D$  就是 **policy**。

从强化学习的角度看，某些推导过程的结果，可以给予奖励：

$$K_0 \xrightarrow{D^n} K_{\perp} \quad \updownarrow \star$$

$\updownarrow \star$  的意思是「给予正或负奖励」。我们要学习的是  $D$  也就是 **policy**。学习算法的基础是著名的 **Bellman optimality condition** (见 box)。

但  $D$  本身是 **RNN**，它还可以透过 **back-prop** 进行学习，两者似乎是不同的。**Back-prop** 是透过  $\frac{\partial}{\partial D}$  error 的梯度来学习。

下面我们分别讲述 4 种学习模式：

- 学习 听 / 讲
- RL-based learning
- inductive learning

### 3 Bellman equation

回顾一下 Bellman equation 是:

$$U(S) = \max_a \{R(S, a) + \beta U(S')\}$$

其中  $U$  = utility,  $R$  = reward,  $\beta$  = discount factor,  $a$  = action,  $S \xrightarrow{a} S'$ 。它是一条 recursive equation, 表示本状态的最佳效用, 是下一状态的效用, 加上本状态的 reward, 再取最优化。

这个方程有微分版本, 叫 Hamilton-Jacobi equation。假设:

$$U(x, t) = \min_u \left\{ \int_t^{t'} C(x, u) dt + U(x', t') \right\}$$

其中  $t$  是时间,  $u$  是 control parameters,  $C$  是 cost-rate function:

$$\int C dt = R = \text{reward}$$

这个积分表示「路径中的 cost」,  $U(x', t')$  是「终点的 cost」。

这条方程写成微分形式就是 Hamilton-Jacobi equation:

$$\frac{d}{dt} U(x, t) = \min_u \{C(x, u) + \nabla U(x, t) \cdot f(x, u)\}$$

而  $x$  必需遵从:

$$\dot{x}(t) = f(x(t), u(t))$$

这个方程的形式很接近量子力学中的 Schrödinger equation:

$$i\hbar \frac{\partial}{\partial t} \Psi(x, t) = \left[ V(x, t) + \frac{-\hbar^2}{2\mu} \nabla^2 \right] \Psi(x, t)$$

$\Psi$  类似於我们的  $U$  (可能它是一个自然界想 minimize 的东西?)

在思维过程中，除了结果以外，其他步骤是没有奖励的，所以 **reward** 只可以是一个微小的负常数（代表时间损耗）。在这情况下 **Bellman** 算法可能变到和 **back-prop** 差不多甚至一样，但我还未详细验算。

## 4 学习 听 / 讲

我发现 听 / 讲 是可以靠 **RL** 学习的。

Learning modes:

1. Listening:  $K$  contains understanding of linguistic input, test  $K$
2. Speaking:  $K$  is known, output  $\approx_L K$

$$K_0 \xrightarrow{D^n} \text{输出句子} \quad \updownarrow \star$$

## 5 RL-based learning

## 6 Inductive learning

Actions:

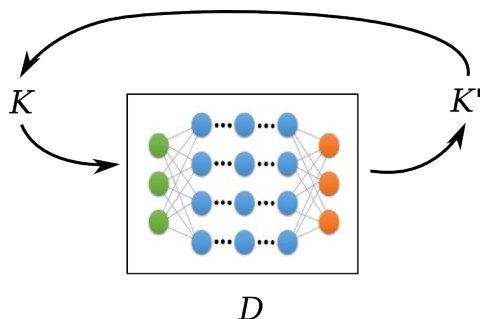
(A) deduction (apply  $D$ )

(B) write / add to  $K$  partially:  $K := K'$  or  $K+ = K'$

(C) learn = change  $D$

(D) learn = neurogenesis

上次的模型：



$K$  可以直接输出一些 words, words 的序列构成句子。

这做法太简单，因为我假设自然语言是一些 sequences，但自然语言的结构不是序列那么简单，例如「未吃过饭」包含「吃过饭」的序列，但语义是相反的。所以自然语言必须要在语义 (semantic) 层面上处理，而不是语法 (syntactic) 层面上。自然语言的复杂性是不会轻易消失的。

## 7 Language map

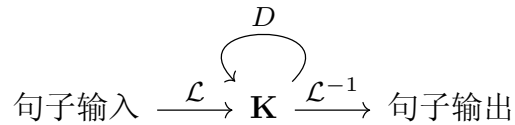
在 Memory Networks [1] 里，他们用了个 language map，将自然语言句子转换成 internal representation of knowledge in  $\mathbf{K}$ ,  $\mathbf{K} \ni K$  是知识状态的空间。可以叫这个 map 做 language map:

$$\mathcal{L}: \mathbf{S} \rightarrow \mathbf{K}$$

$$\mathcal{L}: \text{句子} \mapsto K$$

(但其实句子只是知识状态  $K$  的一部分。)

於是我们的 RNN model 变成这样：



有了这个 language map 是很方便的，但它有几个问题：

- 需要用到 NLP parsing，很麻烦，如果能省略比较好
- 这个 map 基本上 fix 了 internal knowledge representation 的形式。但我的直观（也可能错误）觉得 knowledge representation 应该是「不知道的」比较好，它应该是由学习  $D$  的过程「诱导」出来的，换句话说：the knowledge representation format should be *induced* from the learning of  $D$ . 因为以前在经典逻辑系统中， $K = \text{KB (knowledge base)}$  是一些命题的集合，换句话说  $K = \bigcup S_i$ ， $S_i$  是句子或「命题」。当时的做法是将 KB 组织成 hierarchical 结构，方便搜寻。但我觉得如果  $\mathbf{K}$  的结构是这样的话，跟原问题的情况太相似，一切都太「有秩序」，可能不是神经网络最有效的用法。
- 自然语言是需要慢慢「吸收」或「理解」(comprehend) 的，但这过程在 Memory Networks 的模型里忽略了。将自然句子转换成 logical 形式（即 internal representation），几乎是不需时间的 transliteration 过程。如果输入一本《世界历史》的原文，不消一秒便可以转换成 internal representation，但不能说 Genifer 已经「理解」了全书的内容吧？

## 8 理解 / 慢吸收

所以  $\mathcal{L}$  不是一个普通 map 而是一个很复杂的过程。

对新输入知识的「慢吸收」包括这些运作：

- **consequences** (找出新命题的所有可能推论结果, 至少在  $n$  步之内)
- **consistency** (新信念不和旧信念抵触)
- **explanations** (新信念可以被已有知识解释)

「理解」的过程可以看成是: 经过  $n$  次的推导  $\mathcal{D}$ ,  $K$  变成  $K'$ :

$$K \xrightarrow{D^n} K'$$

而  $K'$  有我们想要的特性 (即「后果、和谐、解释」)。问题就是怎样测试  $K'$  有这些特性? 特别困难的原因是  $K' \in \mathbf{K}$  的 **representation** 是不透明的。

$K'$  的特性可以怎样测试?

似乎唯一的方法是透过这样的查询:

$$K \xrightarrow{D^n} Q?$$

但  $Q$  也是  $\in \mathbf{K}$ , 而  $\mathbf{K}$  的结构是不透明的。

如果有  $\mathcal{L}$  的话, 直接将问题用自然语言问, 变成  $Q = \mathcal{L}(\text{句子})$ , 就可以查询  $K$ 。换句话说,  $\mathcal{L}$  的好处是容许 **direct access to  $K$** , 将知识状态的内容, 直接用自然语言读出来 (或写入去)。但现实中不存在一个简单的  $\mathcal{L}$  map。

$\mathcal{L}$  不存在, 找不到  $Q$  用来查询  $K$ 。

## 9 Speech generation (发言)

如果 Genifer 学会「说话」，就等於查看  $K$  的内容。但说话不等於将所有知识状态的内容「和盘托出」，而是有两种模式：

- 查询某句子  $S_{query}$  是不是真的
- 查询关于某题目  $S_{interest}$  的内容，eg: "Tell me about your mother"

但这两个动作都要求有某种办法将  $query$  或  $interest$  映射到  $\mathbf{K}$  的某个「位置」，而这正是我们没有的。

「说话」本身是一个复杂的算法，它不是一个简单的映射，而是需要用  $D$  思考的运作。

无论是「理解」或「说话」，都是复杂算法，而且  $\mathbf{K}$  是不透明的。表面上似乎没有希望，但神经网络的好处是：可以同时学习两个过程，即使这两个过程互相依赖。

## 10 学习

为了方便思考，现假设有  $\mathcal{L}$ ，容易做到以下两种学习方式：

### 10.1 Inductive learning

举例：

小明是香港人  $\wedge$  小明戴眼镜



小强是香港人  $\wedge$  小强戴眼镜  
小雄是香港人  $\wedge$  小雄戴眼镜  
小娟是香港人  $\wedge$  小娟戴眼镜  
结论：所有香港人都戴眼镜  
用逻辑表示这法则就是：

$$\forall X. \text{Hong-Kong}(X) \Rightarrow \text{wear-glasses}(X)$$

但在 RNN 里，这法则是暗含在  $D$  之中。例如如果  $k$  = 「大强是香港人」，那么  $D$  作用在  $K$  上最终会得出  $K' =$  「大强戴眼镜」。

换句话说，已知  $K_0$ ， $K^*$ ，学习  $D$ ：

$$K_0 \xrightarrow{D} \dots K^*$$

因为我们可以用  $K = \mathcal{L}(\text{自然语言句子})$  来计算  $K_0$  和  $K^*$ ，所以这个算法是可行的。

## References

- [1] Weston, Chopra, and Bordes. Memory networks. *ICLR (also arXiv)*, 2015.