I am an enthusiast, but not a crank in the sense that I have some pet theories as to the proper construction of a flying machine. I wish to avail myself of all that is already known and then, if possible, add my mite to help on the future worker who will attain final success.

—Wilbur Wright

Genifer 4.0 theoretical notes

YKY (甄景贤)

May 17, 2015

Abstract

Genifer 4 is an attempt to bring logic-based AI (such as OpenCog, NARS, and Genifer 3) into a vector space setting, thus allowing the use of "continuous" techiques such as operators, iterative methods, gradient descent, etc. Such techiques may be more computationally efficient.

Make the correspondence:

logic formulas	$\Leftrightarrow$	algebra over the ring of truth values
facts (propositions)	$\Leftrightarrow$	formal series
rules	$\Leftrightarrow$	non-linear operators
KB	$\Leftrightarrow$	vector space (distributive representation)
pattern matching	$\Leftrightarrow$	filter: KB $\to$ KB
deduction	$\Leftrightarrow$	apply operators
learning	$\Leftrightarrow$	find operator in operator space
memory	$\Leftrightarrow$	vector space $\times t$
actions	$\Leftrightarrow$	some meta-operations (?) between memories

Classically, logic is divided into **propositional logic** and **predicate logic**. Propositional logic concerns with propositions (ie formulas that have truth values) *without internal structure*; it is isomorphic to Boolean algebra, and thus is familiar to mathematicians. Predicate logic, on the other hand, is very difficult to model with abstract algebra. One such attempt is cylindrical algebra (started by Tarski), but it is awkward to use and is little known outside of logicians. I attempt a somewhat alternative route, based on the idea from combinatory logic and relation algebra.

In classical AI the KB (**knowledge base**) contains facts and rules expressed as logic formulas. Rules act on facts to produce new facts; This is called **deduction**. In our new setting, we represent the KB as a very long vector:

$$KB = \mathbf{v} = (v_1, v_2, ..., v_n)$$

and we use **distributive representation** which is well-known in neural networks. This means that concepts are not encoded by individual units (ie, each v_i) but rather by the *combined activations* of all units in a layer.

1 Action

我们有了动作但不知道怎样指向对象。其中一个可能的办法是用 label 标签上次的答案；或者作用於所有答案（但这是不自然的?）

这引申到注意力的问题。或者 Genifer 永远只是 focus 在注意力的前沿?

还有个问题就是：答案并不一定是唯一的。所以需要一个方法去把 KB 的答案捉回到 register 里。

但那又引申到 working memory 和 KB 的区别。或者 attention 只是 derivation 的 time-decaying trace? 换句话说，那些可能的答案只有很少几个。我们应该可以利用某些特徵提取他们。

最简单可能就是 -1 和「没有」的分别。如何区别「有」和「没有」呢? 可以指定答案的 class，例如「广东话字词」。

2 Learning

学习的目的是寻找一组逻辑 formulas 去解释这世界。所谓解释即推导。

学习的方法是 inductive learning，即由事实诱导出法则 (induce rules from facts)。Rules 就是逻辑範式。

Induction 需要的是一个 general-to-specific order。传统逻辑中这个序由两方面达成：

1. 某个 concept 比另一个 concept 更一般，例如：动物 / 狗
2. conjunctions 的增加，例如：戴眼镜 $\wedge$ 长头发

压缩的方法必须是 “semantic distance preserving”，意即：在语义空间中相似的点被压缩到相邻的逻辑範式。

Gradient descent 的原理是我们必须知道 $\frac{\partial \mathcal{E}}{\partial \mathbf{r}}$ 的值，其中 $\mathcal{E}$ 是误差，$\mathbf{r}$ 是法则空间中的座标。

误差 $\mathcal{E}$ 由下式给出：

$$\mathcal{E} = ||R^{\infty}(\mathcal{F}) - \mathcal{F}^{*}||$$

$\mathcal{F}$ 是已知事实，$\mathcal{F}^{*}$ 是新的要学习的事实，R 是所有法则，$r \in R$。

问题似乎是：法则的诱导似乎不能单是基於语法。概念阶层的诱导是基於：Liebniz 和 aRb。

Liebniz extensionality:

$$xZ \rightarrow yZ \Leftrightarrow x \supset y \tag{1}$$
$$Zx \rightarrow Zy \Leftrightarrow x \supset y \tag{2}$$

There are 2 ways to generalize a logic formula:

1. adding **conjunctions**
2. using concepts that admit **substitutions**

The relation of subsumption is *intrinsic* to the logic.