

No problem can withstand the assault of sustained thinking.

— Voltaire

Genifer 4.1 理论笔记

YKY (甄景贤)

30 June 2015

现在来 recap 一下，我想做的是什麼...

首先，我們有一個原子概念的集合 $A = \{\text{男人, 女人, 爱, 恨, 足球, ...}\}$ ，在抽象代數中這個集通常叫作 alphabet，它的元素叫字母 (letters)。

2013 年，Tomáš Mikolov 等人在 Google 研究出 Word2Vec，引起了廣泛影響。基本上它將離散的詞汇集合映射到連續的向量空間（需要某種近似），這個近似是基於所謂的分布性原理 (Distributional Hypothesis)，即是說：兩個詞如果分布相似，即經常出現在相似的句子中，它們的語義也會相近。

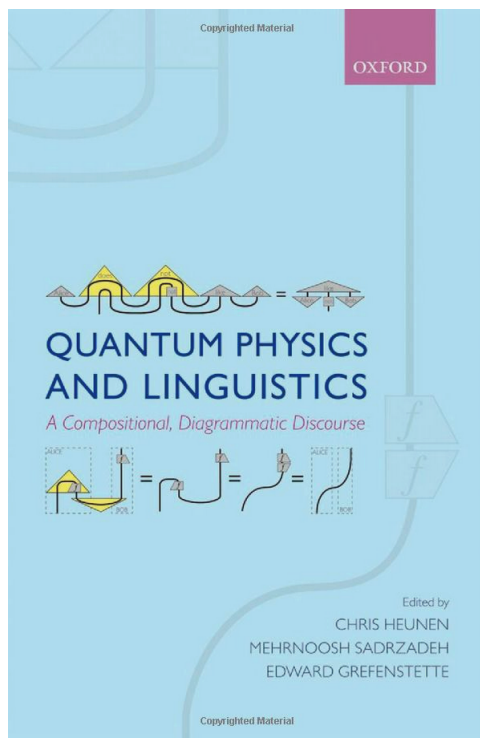
我們的目的不是探討如何計算詞匯的 distributive representation，而是假設在原始概念的層次上，這份工作已經做好，不管是用 Word2Vec 或其他的方法。

我們要處理的問題是：由詞語過渡到句子，進而從事推理和學習。

表達句子的方法：

- 最簡單可以只是 3 個元素構成的 $a R b$ 。
- Smolensky 提出的 tensor 方法，可以表達任何樹狀結構。

- 最新的研究方向，人们发现句子的结构和抽象代数的结构是一样的，所以注意力转移到抽象的範畴的研究，例如这本书：



其中最一般的範畴是 monoidal categories，或者叫 tensor categories。在向量空间里的 tensor product 是这个範畴中的一个例子。换句话说，我们基本上是用 tensor product 来表达句子；範畴论只是将这一点说得更一般化和抽象化而已。

1 命题空间的 dimension

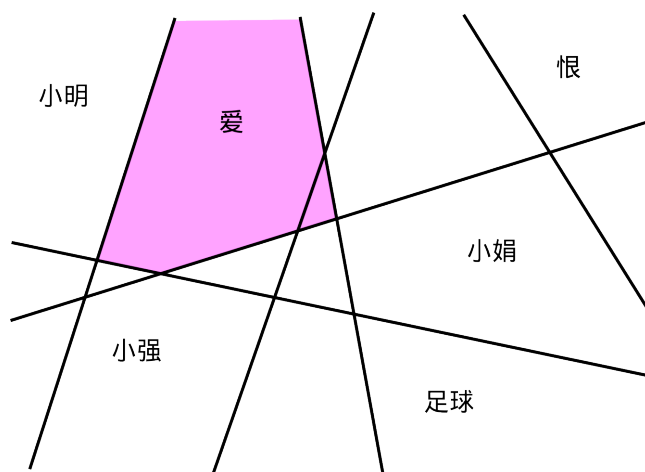
在知识空间中，如果 v_1, v_2, v_3 是三个互不相关的命题，那么它们的线性相关 $a_1v_1 + a_2v_2 = a_3v_3$ 是不容许的，否则会出现谬误。结论是：要么知识空间的 dimension 等於命题的总数（那会很大，可能无限大），要么放弃 a_i 是真假值的做法。

人的词汇 (vocabulary) 的个数大约在 3,000 至 10,000's 之间。暂时假设我们可以用 dimensionality reduction 把原始概念的个数缩小到 3000, 那么概念空间 C 的维数就是 3000。

如果只考虑像 aRb 那样的关系, 则命题空间是 $C \times C \times C$, 维数是 $3000 \times 3 = 9K$ 。但如果是 $C \otimes C \otimes C$ 则维数是 $3000^3 = 9G$ 。现时的电脑记忆 (例如 GPU) 似乎可以处理后者。

2 Distributive representation (分布式表示法)

如果每个概念是向量空间中的一个独立分量, 似乎很浪费空间。在神经网络中通常用分布式的表示法:



一粒神经元的公式是: $y = \sigma(w^T x \leq b)$

(但如果只有一层神经元) 那非线性函数可以不理 (它不影响 decision boundary)。

$(w^T x \leq b)$ 表示一个 hyperplane 的切割。

一个概念就是一组 hyperplanes 切割而成的多面体 (polyhedron),

$$c : (Wx \leq B).$$

矩阵 W 对应於一层神经网络中的 weights。

问题：在这个表示法中，「线性无关」代表什么？

3 「连续性」

第二个问题是「连续」指的是什么。例如命题 $P_1 =$ 「小明爱小娟」, $P_2 =$ 「小强爱踢足球」, 那么由 P_1 变化到 P_2 必然会有 discrete jump, 那是无可置疑的。除非我们重新定义命题是像 $k_1 P_1 + k_2 P_2$ 那样的东西, 才可以有「连续变化」; 这一点需要再加以精确化。

我发觉「连续空间」不是一个严谨的术语, 因为在 discrete 空间中也可以定义 "continuous" 这概念, 例如在 computer science 的 functional programming 里有 domain theory、denotational semantics 等, 常常谈到离散空间中的连续函数。而且「可微分」的概念也有一些 generalizations, 例如 Fréchet derivatives (在泛函空间中)。

有个方法可以表示我想说的「连续空间」的性质: 在两点 $x \neq y$ 之间, 总可以找到第三点 z , 位置介乎 x 与 y 之间, 换句话说:

$$d(x, z) < d(x, y) \wedge d(y, z) < d(x, y).$$

似乎 Hausdorff 性质就是我想说的「连续空间」的性质: A Hausdorff space is a topological space in which distinct points have disjoint neighbourhoods.

4 近似

那些算子或许可以合写成一个:

$$T_1 v + T_2 v + \dots + T_n v = \mathcal{T} v$$

然后合写后用 function approximation 来近似。

但这种近似必须有代价, 否则违反了两项原则:

1. Turing 已经很有远见地意识到，逻辑推导的普适算法是不存在的，因为它违反了停机原理 (halting problem)。后者是用 Cantor 的集合论中的 diagonal argument 证明的。如果我们的优化算法必然会给出答案，而答案又可以转换回逻辑，那是不可能的。所以在近似过程中必然会有某些误差。
2. 如果 $P \neq NP$ ，我们的算法也不可能总是在 polynomial time 之内回覆正确答案。

5 “Second-order”

还有一个“second-order”的想法。在 first-order 我们的算子是这个形式：

$$T : V \rightarrow V$$

但 second-order 的做法是将所有物体都看成是算子，算子可以作用在算子之上，这似乎是一种 duality。

如果我们简单地将 \mathcal{T} 近似，那些逻辑推导就会有错误。或者应用某个 duality 或二次形式后，近似的做法会有较好结果？

从另一个角度看，详细一点分析那推导和学习的过程。推导的过程是：

$$K_0 \xrightarrow{R} K_1 \xrightarrow{R} \dots \xrightarrow{R} K_\infty$$

可以将它简写，并加上反向的 L map：

$$K_0 \xrightarrow[\infty]{R} K_\infty \quad \text{with a curved arrow } L \text{ from } K_\infty \text{ back to } R$$

L 是由结论 K_∞ 到法则 R 的映射，可以叫 L 做 learning map。这个 map 比较复杂，它相当於学习的过程，通常要用 iteration 逼近。而且，有无限多个 R

可以符合条件，我们通常选取长度短的 R ，这是 minimum description length 原理，又或者选取资讯压缩比最高者。

上面的 map 可以再简化，因为 K_0 是常项，可以省略。写成垂直形式：

$$\begin{array}{c} K_{\infty} \\ \updownarrow L \\ R \end{array}$$

现在如果我们改变推导的结果 K^* ，亦即改变 K_{∞} ，那么 R 亦要改变成 R' ，但 L 未必要改变。学习这个 L 是一种二次形式的学习。

$$\begin{array}{ccc} K_{\infty} & \longrightarrow & K'_{\infty} \\ \updownarrow L & & \updownarrow L \\ R & \longrightarrow & R' \end{array}$$

问题是 R 的空间太大... 但 R 是所有有意识的知识... 不想直接近似 R ，但如何间接地近似它？

6 Metric 从何而来？

现在概括一下逻辑系统是什么。 $\mathcal{L} = \{C, P, +, \rightarrow, d(\cdot, \cdot), \subseteq\}$ 包含以下元素：

1. 原子概念集 $C \ni c_1, c_2, \dots$
2. 命题集 $P \ni p_1, p_2, \dots$ （可以限制在简单关系 $a R b$ ）
3. 连结词 (conjunction) $p_1 \wedge p_2$ 或 $p_1 + p_2$
4. 箭头 $p_1 + p_2 + \dots \rightarrow p_0$
5. 概念之间的距离 $d(c_1, c_2)$

6. 概念之间的偏序 (partial order) $c_1 \subseteq c_2$

但其实 $d(\cdot, \cdot)$ 和 \subseteq 并不是逻辑内在的，它们是从资料中 induce 出来的。其原理是基於 Liebniz extensionality:

$$f = g \Leftrightarrow \forall x. f(x) = g(x).$$

它原本是描述两个函数的相等，但我们可以将它普及到任何概念的乘积：

$$c_1 \approx c_2 \Leftrightarrow \forall_{\#} P, P'. P[c_1], P'[c_2].$$

这是说：概念 c_1 和 c_2 相似，如果它们经常出现在相似的命题之中。 $\forall_{\#}$ 是 probabilistic quantification，它计算所有命题中相似者的比例。 $d(\cdot, \cdot)$ 是 \approx 的量度。

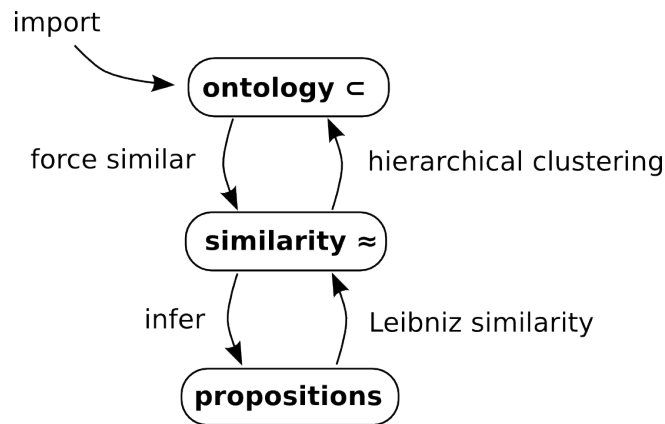
有了 \approx 关系，可以用 hierarchical clustering 建立 \subseteq 关系（注意这里的 \subseteq 包含我们平时所说的 \subseteq 和 \in 关系）。

其实 \subseteq 和 \approx 基本上是等价的。例如，学习过程中我们观察到猫和狗有相似的句子：

$$\forall_{\#} P : P[\text{猫}], P[\text{狗}]$$

於是我们可以归纳出「猫 \approx 狗」，但我们也可以新增一个类别：「动物 \supseteq 猫，狗」。换句话说，我们说某些物体相似，也可以说它们是同类。

\approx 和 \subseteq 之间的关系是：



我们想把这个结构映射到 Banach 空间。

虽然这个 metric 很难计算，但它可能是我们的学习问题里唯一比较合理的结构，除了用它之外可能没有其他选择。

这个 metric 其实不是良好定义的，因为它可能不会 converge。举例来说，Reimann hypothesis 原来和质数的分布有关，但这关系在 1859 年以前人们还不知道。在逻辑上我们可以推导出很深的关系，这似乎是永无止境的。但如果我们假设可以得到近似的 metric，这个 heuristic 也许还不错。

7 Neural Tensor Network

Andrew Ng 是香港人，Coursera 的 machine learning 教授，他在 Stanford，最近加入了百度。他和合作者提出了 NTN 模型 [5]:

$$\top(a R b) = u^T \boxed{\text{ReLU}}(a^T U b + W \begin{pmatrix} a \\ b \end{pmatrix} + c)$$

$\top(a R b)$ 表示 $a R b$ 这个关系的强度。

U 是一个 tensor，取两个向量 a, b 给出第三个向量 $a^T U b$ （它服从张量的双线性性质）。

W 是一个矩阵， $W \begin{pmatrix} a \\ b \end{pmatrix}$ 是一层传统的神经网络， $\boxed{\text{ReLU}}$ 是一个非线性函数。

u^T 是一粒神经元，作用是把各个分量加起来，得出一个数。

8 Paul Smolensky's tensor representation

Paul Smolensky 的书是《The harmonic mind》(2006) [4]，第一卷是 AI 理论，第二卷是语言学理论。

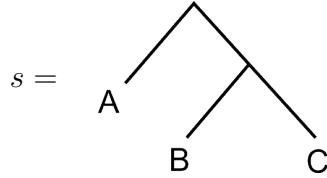
他提出用张量来表示关系，一般而论：

$$\begin{aligned} \text{father}(\text{john}, \text{pete}) &\Leftrightarrow \text{father} \otimes \text{john} \otimes \text{pete} \\ \text{Var}_1 : \text{val}_1, \text{Var}_2 : \text{val}_2 &\Leftrightarrow \text{Var}_1 \otimes \text{val}_1 + \text{Var}_2 \otimes \text{val}_2 \end{aligned}$$

第二句的意思是， $variable_1$ 的值是 $value_1$ ， $variable_2$ 的值是 $value_2$ 。

他引入了“role”和“filler”的概念： $variable_i$ 是 role， $value_i$ 是 filler。用这个方法来做 variable binding，好处是可以避免 $A \cdot B = B \cdot A$ 的问题。

举例来说，在自然语言中可以有这个句子结构：



r_0 和 r_1 分别是树的左右孩子的 roles。用张量积表示：

$$\begin{aligned}
 s &= A \otimes r_0 + q \otimes r_1 \\
 q &= B \otimes r_0 + C \otimes r_1 \\
 \therefore s &= A \otimes r_0 + B \otimes r_0 \otimes r_1 + C \otimes r_1 \otimes r_1
 \end{aligned}$$

张量积的好处是， r_0 和 $r_0 \otimes r_1$ 是不同的元素，它们彼此不会「相撞」。坏处是当乘积的长度增加时，dimension 也增加得很快。

下表是 Smolensky 表示法的总结：

	Example	Vector representation
Set element	$\{c_1, c_2\}$	$c_1 + c_2$
Filler-role binding	$AB = \{A \setminus r_1, B \setminus r_2\}$	$A \otimes r_0 + B \otimes r_1$
Recursive structure		$A \otimes r_0 + [BC] \otimes r_1$

我在博客¹上解释过，tensor 是所有 bi-linear forms 的 universal form。如果有向量空间 U 和 V ，

$$\begin{aligned}
 T &: U \otimes V \rightarrow W \\
 t &: u \otimes v \mapsto w
 \end{aligned}$$

¹blog post

那么在 U, V 中线性无关的 $\{u_1, \dots, u_m\}$ 和 $\{v_1, \dots, v_n\}$, 它们的乘积 $\{u_i \otimes v_i\}$ 在 W 中仍然是线性无关的。这是 tensor product 的一个 defining property, 换句话说:

$$\text{线性无关集} \otimes \text{线性无关集} \mapsto \text{线性无关集}$$

而这正是我们需要的, 因为根据「scalar = 逻辑命题真假值」的诠释, 这正是命题之间不「相撞」的条件。

Antony Browne & Ron Sun 的较早的论文《Connectionist inference models, 2001》[1] 有讲述更多 neural-symbolic integration 的做法。

9 Coecke-Sadrzadeh-Clark representation

这是 Coecke, Sadrzadeh, Clark (2010) [2] 提出的表示法。

它首先是基於 categorial grammar (类型语法), 那是一种有「左右除法」的文法, 例如在 “*John likes Mary*” 这句话中, “*likes*” 的类型是:

$$likes := (S \backslash NP) / NP$$

意思是当 “*likes*” 这个字, 在左边乘一个 noun phrase, 右边乘另一个 noun phrase, 之后便可得到一个句子 S 。这种文法由 Lambek 1958 年开创。

换成 tensor product, “*likes*” 表示成一支 vector:

$$\overrightarrow{likes} \in N \otimes S \otimes N$$

这是很合理的, 因为这个 tensor product space, 它的元素就是那些 multi-linear functions:

$$f : N \times N \rightarrow S.$$

10 Metric embedding

最近有一本新书: [Mikhail Ostrovskii 2013] *Metric embeddings: bi-Lipschitz and coarse embeddings into Banach spaces* [3], 它似乎正正讲述了我们将 logic 嵌入到「连续空间」的问题。

What is the bi-Lipschitz condition? A map $f : X \rightarrow Y$ is called a C -bi-Lipschitz embedding if there exists $r > 0$ such that

$$\forall u, v \in X, \quad r d_X(u, v) \leq d_Y(f(u), f(v)) \leq rC d_X(u, v)$$

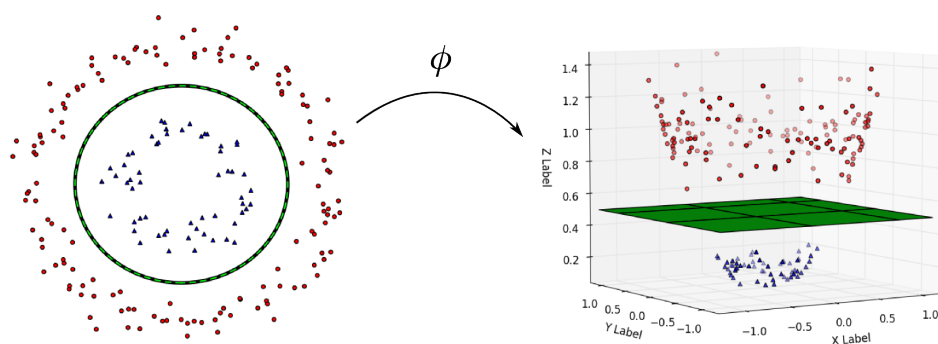
for some $C < \infty$. The smallest constant C for which there exists $r > 0$ such that the above is satisfied is called the *distortion* of the map f . 这似乎定义了「远的保持远、近的保持近」的意思。

11 SVMs and the kernel trick

印象中 **SVM** (support vector machine) 似乎可以将非凸的问题转化成凸优化的问题；可不可以将这个想法应用到我们的问题上呢？

SVM 的算法，首先用 kernel trick 将 data points 转换到高维空间，然后在高维空间中进行线性的 hyperplane 分割。

所谓 **kernel trick** 是指：给定一个 inner product，它暗含了一个到高维空间的转换 ϕ （但这个转换不需要真的计出来）。



而，在高维空间中用线性分割，那 error term 是一些正交距离，所以是 **quadratic form**，所以这问题是凸优化。

我们的问题是性质不同的，但两者似乎有些相似...

Acknowledgments

最近在香港认识了两个朋友，Dr 陈启良 (CUHK) 和 Dr 譚志斌 (HSMC)，和他们谈过我的 AI 理论之后获益良多。

References

- [1] Browne and Sun. Connectionist inference models. *Neural networks 14*, 2001.
- [2] Coecke, Sadrzadeh, and Clark. Mathematical foundations for distributed compositional model of meaning. *Linguistic Analysis 36*, 345-384, 2010.
- [3] Ostrovskii. *Metric embeddings: bi-Lipschitz and coarse embeddings into Banach spaces*. De Gruyter, 2013.
- [4] Smolensky and Legendre. *The harmonic mind, vol 1: cognitive architecture*. MIT Press, 2006.
- [5] Socher, Chen, Manning, and Ng. Reasoning with neural tensor networks for knowledge base completion. *NIPS*, 2013.