深层学习为何要“Deep”（下）

为何深层学习

深层学习开启了人工智能的新时代。不论任何行业都害怕错过这一时代浪潮，因而大批资金和人才争相涌入。但深层学习却以“黑箱”而闻名，不仅调参难，训练难，“新型”网络结构的论文又如雨后春笋般地涌现，使得对所有结构的掌握变成了不现实。我们缺少一个对深层学习合理的认识。

> 神经网络并不缺少新结构，但缺少一个该领域的 $E = mc^2$

很多人在做神经网络的实验时会发现调节某些方式和结构会产生意想不到的结果。但就我个人而言，这些发现并不会让我感到满足。我更关心这些新发现到底告诉我们了什么，造成这些现象的背后原因是什么。我会更想要将新的网络结构归纳到已有的体系当中。这也是我更多思考“为何深层学习有效”的原因。下面便是目前YJango关于这方面的见解。

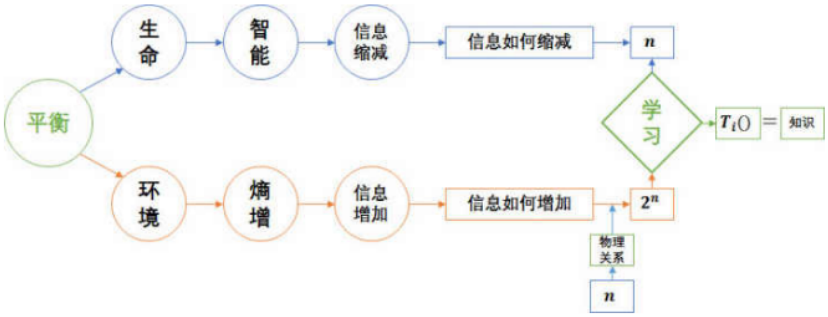
深层神经网络相比一般的统计学习拥有从数学的严谨中不会得出的关于物理世界的先验知识（非贝叶斯先验）。该内容也在Bengio大神的论文和演讲中多次强调。大神也在Bay Area Deep Learning School 2016的Founda’ ons and Challenges of Deep Learning pdf（这里也有视频，需翻墙）中提到的distributed representations和compositionality两点就是神经网络和深层神经网络高效的原因（若有时间，强烈建议看完演讲再看该文）。虽然与大神的思考起点可能不同，但结论完全一致（看到Bengio大神的视频时特别兴奋）。下面就是结合例子分析：

1. 为什么神经网络高效
2. 学习的本质是什么
3. 为什么深层神经网络比浅层神经网络更高效
4. 神经网络在什么问题上不具备优势

其他推荐读物

- Bengio Y. Learning deep architectures for AI[J]. Foundations and trends® in Machine Learning, 2009, 2(1): 1-127.
- Brahma P P, Wu D, She Y. Why Deep Learning Works: A Manifold Disentanglement Perspective[J]. 2015.
- Lin H W, Tegmark M. Why does deep and cheap learning work so well?[J]. arXiv preprint arXiv:1608.08225, 2016.
- Bengio Y, Courville A, Vincent P. Representation learning: A review and new perspectives[J]. IEEE transactions on pattern analysis and machine intelligence, 2013, 35(8): 1798-1828.
- Deep Learning textbook by Ian Goodfellow and Yoshua Bengio and Aaron Courville

YJango的整个思考流程都围绕**减熵**二字进行。之前在《熵与生命》和《生物学习》中讨论过，生物要做的是降低环境的熵，将不确定状态变为确定状态。通常机器学习是优化损失函数，并用概率来衡量模型优劣。然而概率正是由于无法确定状态才不得不用的衡量手段。生物真正想要的是没有丝毫不确定性。



深层神经网络在自然问题上更具优势，因为它和生物学习一样，是找回使熵增加的“物理关系”（知识，并非完全一样），将变体（2^n）转化回因素（n）附带物理关系的形式，从源头消除熵（假设每个因素只有两种可能状态）。这样所有状态间的关系可以被确定，要么肯定发生，要么绝不发生，也就无需用概率来衡量。因此下面定义的学习目标并非单纯降低损失函数，而从**确定关系**的角度考虑。一个完美训练好的模型就是两个状态空间内所有可能取值间的关系都被确定的模型。

> 学习目标：是确定（determine）两个状态空间内所有可能取值之间的关系，使得熵尽可能最低。

注：对熵不了解的朋友可以简单记住，事件的状态越确定，熵越小。如绝不发生（概率0）或肯定发生（概率为1）的事件熵小。而50%可能性事件的熵反而大。

为举例说明，下面就一起考虑用神经网络学习以下两个集合的不同关联（OR gate 和 XOR gate）。看看随着网络结构和关联的改变，会产生什么不同情况。尤其是最后网络变深时与浅层神经网络的区别。

注：选择这种XOR这种简单关联的初衷是输入和输出空间状态的个数**有限**，易于分析变体个数和熵增的关系。

注：用“变体（variation）”是指同一类事物的不同形态，比如10张狗的图片，虽然外貌各异，但都是狗。

问题描述：集合A有4个状态，集合B有2个状态。0和1只是用于表示不同状态的符号，也可以用0,1,2,3表示。状态也并不一定表示数字，可以表示任何物理意义。

$A = \{00, 01, 10, 11\}$ $B = \{0, 1\}$

方式1：记忆

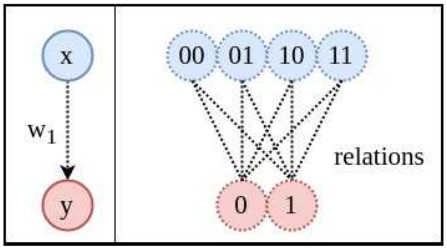
- **随机变量**X：可能取值是 $\{00, 01, 10, 11\}$
- **随机变量**Y：可能取值是 $\{0, 1\}$

  注：随机变量（大写X）是将事件投射到实数的函数。用对应的实数表示事件。而小写字母x表示对应该实数的事件发生了，是一个具体实例。

- **网络结构**：暂且不规定要学习的关联是OR还是XOR，先建立一个没有隐藏层，仅有一个输入节点，一个输出节点的神经网络。
- **表达式**： $y = M(x) = \phi(w_1 \cdot x + b)$ ， ϕ 表示sigmoid函数。（只要是非线性即可，relu是目前的主流）
- **说明**：下图右侧中的虚线表示的既不是神经网络的链接，也不是函数中的映射，而是两个空间中，所有可能值之间的关系（relation）。学习的目的是**确定**这些状态的关系。比如当输入00时，模型要尝试告诉我们00到1的概率为0，00到0的概率为1，这样熵 $H(X) = -\sum_i p_i(x) log p_i(x)$ 才会为零。

- **关系图**：左侧是网络结构，右侧是状态关系图。输入和输出空间之间共有8个关系(非箭头虚线表示关系)。除非这8个关系对模型来说都是相同的，否则用 w_{h1} 表示 $f:X\to Y$ 时的熵 $H(M(X),X)$ 就会增加。（ w_{h1} 无法照顾到8个关系，若完美拟合一个关系，其余的关系就不准确）

  注：这里YJango是 w_{h1} 用表示 $\phi(w_{h1}\cdot x+b)$ 的缩写。



- **数据量**：极端假设，若用查找表来表示关系：需要用8个不同的 (x,y) 数据来记住想要拟合的 $f:X\to Y$ 。

方式2：手工特征

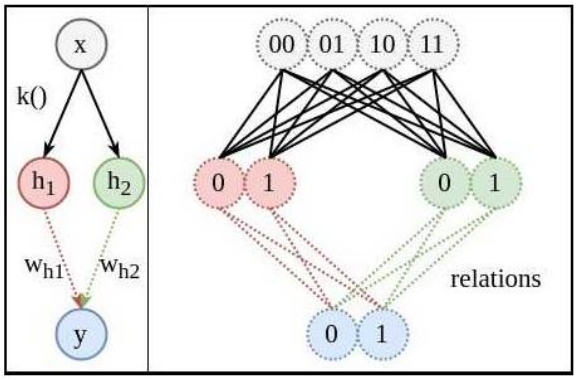
- **特征**：空间A的4个状态是由两个0或1的状态共同组成。我们可以观察出来（计算机并不能），我们利用这种知识 $k()$ 把A中的状态分解开（disentangle）。分解成两个**独立的**子随机变量 $H_1=\{0,1\}$ 和 $H_2=\{0,1\}$ 。也就是用二维向量表示输入。
- **网络结构**：由于分成了二维特征，这次网络结构的输入需改成两个节点。下图中的上半部分是，利用人工知识 $k()$ 将随机变量 X **无损**转变为 H_1 和 H_2 的共同表达（representation）。这时 h_1 和 h_1 一起形成网络输入。

  注：k()旁边的黑线（实线表示确定关系）并非是真正的神经网络结构，只是方便理解，可以简单想象成神经网络转变的。

- **表达式**： $y=M(h)=\phi(W_h\cdot h+b)$
  注：方便起见， $w_{h1}\cdot h_1+w_{h2}\cdot h_2$ 写成了矩阵的表达形式 $W_h\cdot h$ ，其中 b 是标量，而 $W_h=[w_{h1}\quad w_{h2}]$ ， $\vec{h}=\begin{bmatrix}h_1\\h_2\end{bmatrix}$

- **关系图**：由于 $k()$ 固定，只考虑下半部分的关系。因为这时用了两条线 w_{h1} 和 w_{h2} 来共同对应关系。原本需要拟合的8个关系，现在变成了4个（两个节点平摊）。同样，除非右图的4条红色关系线对 w_{h1} 来说相同，并且4条绿色关系线对 w_{h2} 来说也相同，否则用 w_{h1} 和 w_{h2}
  表示 $f:X\to Y$ 时，一定会造成熵 $H(M(X),X)$ 增加。

  注：下图中左侧是网络结构图。右侧关系图中，接触的圆圈表示同一个节点的不同变体。分开的、并标注为不同颜色的圆圈表示不同节点，左右两图连线的颜色相互对应，如红色的 w_{h1} 需要表示右侧的4条红色关系线。



- **关联1**：下面和YJango确定想要学习的关联（函数）。如果想学习的关联是只取 H_1 或者 H_2 的值，那么该结构可以轻松用两条线 w_{h1} 和 w_{h2} 来表达这4个关系(让其中一条线的权重为0，另一条为1)。

- **关联2**：如果想学习的关联是或门，真值表和实际训练完的预测值对照如下。很接近，但有误差。不过若要是分类的话，可以找到0.5这个超平面来分割。大于0.5的就是1，小于0.5的就是0，可以完美分开。

H_1 and H_2	Y	predict
[0,0]	0	0.25
[0,1]	1	0.75
[1,0]	1	0.75
[1,1]	1	1.25

  注:第一列是输入，第二列是真实想要学习的输出，第三列是训练后的网络预测值。

- **关联3**：如果想学习的关联是异或门（XOR），真值表和实际训练完的预测值对照如下。由于4条关系线无法用单个 w 表达，该网络结构连XOR关联的分类都无法分开。

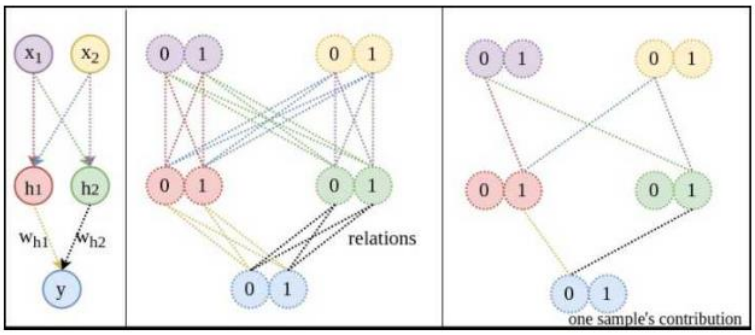
H_1 and H_2	Y	predict
[0,0]	0	0.5
[0,1]	1	0.5
[1,0]	1	0.5
[1,1]	0	0.5

- **数据量**：学习这种关联至少需4个不同的 $([h_1,h_2],y)$ 来拟合 $f_{hy}:H\rightarrow Y$ 。其中每个数据可以用于确定2条关系线。

方式3：加入隐藏层

- **网络结构1**：现在直接把 h_1 和 h_2 作为输入（用 x_1，x_2 表示），不考虑 $k()$。并且在网络结构中加入一个拥有2个节点（node）隐藏层（用 h_1 和 h_2 表示）。
- **表达式**： $y=M(x)=\phi(W_h\cdot\phi(W_x\cdot x+b_x)+b_h)$
- **关系图1**：乍一看感觉关系更难确定了。原来还是只有8条关系线，现在又多了16条。但实际上所需要的数据量丝毫没有改变。因为以下**两条先验知识的成立**。

  注：下图最右侧是表示：当一个样本进入网络后，能对学习哪些关系线起到作用。



- **1. 并行**： $f_{xh_1}:X_1,X_2\rightarrow H_1$ 和 $f_{xh_2}:X_1,X_2\rightarrow H_2$ 的学习完全是独立并行。这就是神经网络的两条固有先验知识中的：**并行**：网络可以**同时**确定 f_{xh_1} 和 f_{xh_2} 的关联。也是Bengio大神所指的distributed representation。

> **注**：有效的**前提**是所要学习的状态空间的关联是可以被拆分成并行的因素（factor）。
>
> **注**：$f_{hy}:H_1,H_2\rightarrow Y$ 就没有并行一说，因为 Y 是一个节点拥有两个变体，而不是两个独立的因素。但是也可以把 Y 拆开表示为one-hot-vector。这就是为什么分类时并非用一维向量表示状态。更验证了YJango在机器学习中对学习定义：**学习是将变体拆成因素附带关系的过程。**

- **迭代**：第二个先验知识是：在学习 $f_{hy}:H_1,H_2\rightarrow Y$ 的**同时**，$f_{xh_1}:X_1,X_2\rightarrow H_1$ 和 $f_{xh_2}:X_1,X_2\rightarrow H_2$ 也可以被学习。这就是神经网络的两条固有先验知识中的：**迭代**：网络可以在确定上一级的同时确定下一级的**所有内容**。也是Bengio大神所指的compositionality。

  > **注**：有效的**前提**是所要学习的空间的关联是由上一级迭代形成的。所幸的是，我们所生活的世界几乎都符合这个前提（有特殊反例）。

- **关联**：如果想学习的关联是异或门（XOR），真值表和实际训练完的预测值对照如下。

  f_{xh_1} 和 f_{xh_2} ：期初若用两条网络连接表示 $X_1,X_2\rightarrow H_1,H_2$ 的16个关系可能，那么熵会很高。但用两条线表示 $X_1,X_2\rightarrow H_1$ 的8个关系，模型的熵可以降到很低。下图中 f_{xh_1} 的输出值对应红色数字。 f_{xh_2} 对应输出值是由蓝色数字表达。

  f_{hy} :这时再看 $H_1,H_2\rightarrow Y$ 的关系，完全就是线性的。光靠观察就能得出 $f(h_1,h_2)$ 的一个表达。

X_1 and X_2	Y	H_1 and H_2	predict
[0,0]	0	[1,1]	0
[0,1]	1	[1,0]	1
[1,0]	1	[0,1]	1
[1,1]	0	[1,1]	0

$$f(h_1,h_2)=-h_1-h_2+2$$

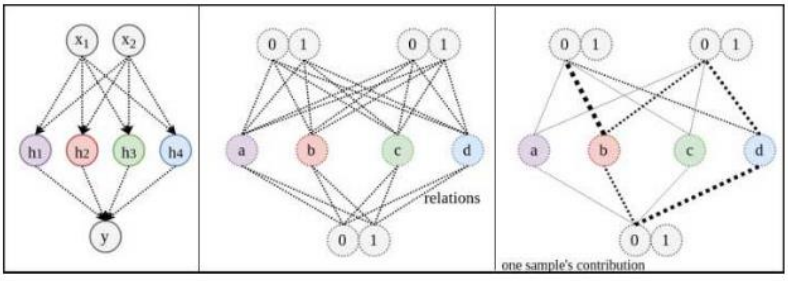
- **数据量**：如关系图1中最右侧图所示，一个输入[0,0]会被关联到0。而这个数据可用于学习2+4个关系。也就是说**网络变深并不需要更多数据**。

模型的熵 $H(M(X),X)$ 与用一条 $\phi(w_1\cdot x+b)$ 所要拟合的关系数量有关。也就是说，

> 变体（variation）越少，拟合难度越低，熵越低。

- **网络结构2**：既然这样，X 有4个变体，这次把节点增加到4。
- **关系图2**：与网络结构1不同，增加到4个节点后，每个节点都**可以**完全没有变体，只取一个值。想法很合理，但实际训练时，模型不按照该方式工作。

  > **注**：太多颜色容易眼花。这里不再用颜色标注不同线之间的对应关系，但对应关系依然存在。



- **问题**：因为会出现右图的情况：只有两个节点在工作（线的粗细表示权重大小）。a和c的节点在**滥竽充数**。这就跟只有两个节点时没有太大别。原因是神经网络的权重的初始化是随机的，数据的输入顺序也是随机的。这些随机性使得权重更新的方向无法确定。

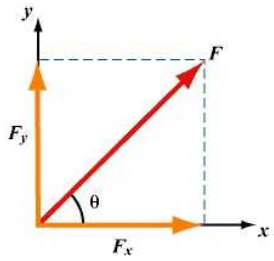
讨论：网络既然选择这种方式来更新权重，是否一定程度上说明，用较少的节点就可以表示该关联？并不是，原因在于日常生活中的关联，我们无法获得所有变体的数据。所得数据往往是很小的一部分。较少的节点可以表示这一小部分数据的关联，但却无法涵盖所有变体情况。造成实际应用时效果不理想。

- **缓解**：缓解的方式有L2正则化（L2 regularization）：将每层权重矩阵的平方和作为惩罚。
- **表达式**：$\lambda/2 \cdot \sum_w w^2$，$\lambda$是惩罚的强弱，可以调节。除以2是为了求导方便（与后边的平方抵消）。
- **意义**：当同一层的权重有个别值过高时，平方和就会增加。而模型在训练中会降低该惩罚。产生的作用是使所有权重平摊任务，让a, b, c, d都有值。以这样的方式来使每个节点都工作，从而消除变体，可以缓解过拟合（overfitting）。
- **例如**：$L2(W_{hy}) = \lambda/2 \cdot (w_{hy1}^2 + w_{hy2}^2 + w_{hy3}^2 + w_{hy4}^2)$

> 具有一个隐藏层的神经网络可以模拟**任何函数**，最糟的情况需要k^n个节点。

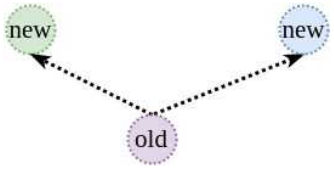
也叫Universal Approximation Theorem。但最糟的情况是输入空间有多少个变体，就需要多少个节点。k表示独立因素的变体个数，n表示独立因素的个数。上述的例子中最糟的情况需要2^2个隐藏节点。而代价也是需要k^n个不同数据才可以完美拟合。

- **使用条件**：浅层神经网络可以分开几乎所有自然界的关联。因为神经网络最初就是由于可移动的生物需要预测未来事件所进化而来的。所学习的**关联是过去状态到未来状态**。如下图，物理中的力也可以分离成两个独立的分力来研究。

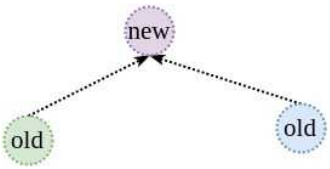


但有一种不适用的情况：非函数。

- **实例**：函数的定义是：每个输入值对应唯一输出值的对应关系。为什么这么定义函数？对应两个以上的输出值在数学上完全可以。但是在宏观的世界发展中却不常见。如下图：
    - **时间顺流**：不管从哪个起点开始，相同的一**个**状态（不是维度）可以独立发展到多个不同状态（如氧原子可演变成氧分子和臭氧分子）。也就热力学第二定律的自发性熵增：原始状态通过物理关系，形成更多变体。



- **时间倒流**：宏观自然界中难以找到**两个以上的不同状态共同**收回到一个状态的例子（如氧分子和臭氧分子无法合并成氧原子）。如果这个可以实现，熵就会自发性减少。也就不需要生物来消耗能量减熵。我们的房间会向整齐的状态发展。但这违背热力学第二定律。至少在我们的宏观世界中，这种现象不常见。所以进化而来的神经网络可以拟合任何函数，但在非函数上就没有什么优势。毕竟生物的预测是从过去状态到未来状态。也说明神经网络并不违背无免费午餐定理。



  - **实例**：XOR门的输入空间和输出空间若互换位置，则神经网络拟合出来的可能性就非常低（并非绝对无法拟合）。

方式4：继续加深

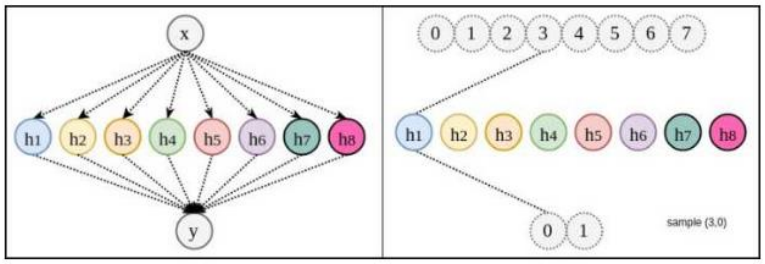
浅层神经网络可以模拟任何函数，但数据量的代价是无法接受的。深层解决了这个问题。相比浅层神经网络，深层神经网络可以用**更少的数据量**来学到更好的拟合。上面的例子很特殊。因为 $2 \cdot 2 = 4$ ， $2^2 = 4$ ，比较不出来。下面YJango就换一个例子，并比较深层神经网络和浅层神经网络的区别。

- **问题描述**：空间 A 有8个可能状态，空间 B 有2个可能状态：

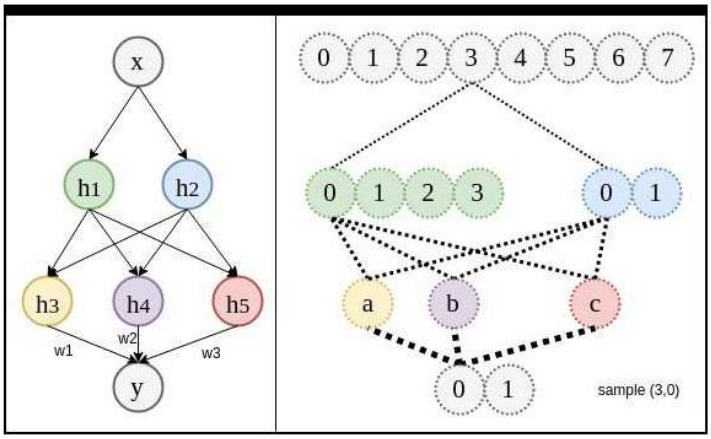
  $A = \{0, 1, 2, 3, 4, 5, 6, 7\}$ $B = \{0, 1\}$

- **网络结构**：
  - 浅层神经网络：8节点隐藏层
  - 深层神经网络：2节点隐藏层 + 3节点隐藏层
- **深浅对比**：

  浅层神经网络：假如说输入3和输出0对应。数据 $(3, 0)$ 只能用于学习一个节点（如 h_1 ）前后的两条关系线。完美学习该关联需要所有8个变体。然而当变体数为 10^{10} 时，我们不可能获得 10^{10} 个**不同变体**的数据，也失去了学习的意义。毕竟我们是要预测没见过的数据。所以与其说这是学习，不如说这是强行记忆。好比一个学生做了100册练习题，做过的题会解，遇到新题仍然不会。他不是学会了做题，而是记住了怎么特定的题。



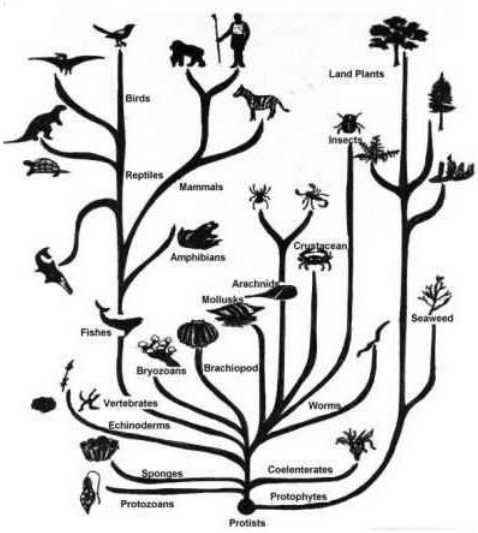
  深层神经网络：如果只观察输入和输出，看起来同样需要8个不同的 (x, y) 训练数据。但**不同** (x, y) 之间有**共用部分**。比如说，在确定3和0关系时，也同时对所有**共用** $w1, w2, w3$ 连接的其他变体进行确定。这样就使得原本需要8个不同数据才能训练好的关联变成只需要3,4不同数据个就可以训练好。（下图关系线的粗细并非表示权重绝对值，而是**共用度**）



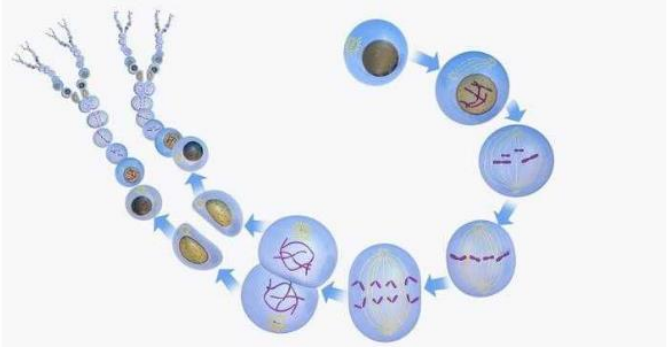
- **深层的前提**：使用浅层神经网络好比是用 $y=ax+b$ 解 a,b ，需要2个不同数据。而深层神经网络好比用 $y=ax$ 解 a ，只需要一个数据。无免费午餐定理告诉我们，优化算法在一个任务上优秀，就一定会在其他任务上有缺陷，深层同样满足该定理。如果用 $y=ax$ 去解实际上有 b 的 $y=ax+b$ ，或者去解实际为 $y=ax2+bx+c$ 的关联时，搜索效果只会更差。所以深层的前提是： X 空间中的元素可以由 Y 迭代发展而来的。换句话说 X 中的所有变体，都有共用根源 Y （root）。

- **我们的物理世界**：幸运的是，我们的物理世界几乎都满足**迭代**的先验知识。

  - **实例**：比如进化。不同动物都是变体，虽然大家现在的状态各异，但在过去都有共同的祖先。



  - **实例**：又如细胞分裂。八卦中的8个变体是由四象中4个变体的基础上发展而来，而四象又是由太极的2个变体演变而来。很难不回想起“无极生太极，太极生两仪，两仪生四象，四象生八卦”。（向中国古人致敬，虽然不知道他们的原意）



> 学习的过程是因素间的关系的拆分，关系的拆分是信息的回卷，信息的回卷是变体的消除，变体的消除是不确定性的缩减。
>
> 自然界两个固有的先验知识：
>
> **并行**：新状态是由若干旧状态并行组合形成。
>
> **迭代**：新状态由已形成的状态再次迭代形成。
>
> 为何深层学习：深层学习比浅层学习在解决**结构问题**上可用更少的数据学习到更好的关联。

随后的三篇文章正是用tensorflow实现上述讨论的内容，以此作为零基础实现深层学习的起点。

- TensorFlow基本用法
- 代码演示LV1
- 代码演示LV2

最后总结一下开篇的问题：
1. **为什么神经网络高效**：并行的先验知识使得模型可用线性级数量的样本学习指数级数量的变体
2. **学习的本质是什么**：将变体拆分成因素和知识（Disentangle Factors of Variation）

- 稀疏表达：一个矩阵被拆分成了稀疏表达和字典。
- one hot vector：将因素用不同维度分开，避免彼此纠缠。

3. **为什么深层神经网络比浅层神经网络更高效**：迭代组成的先验知识使得样本可用于帮助训练其他共用同样底层结构的样本。
4. **神经网络在什么问题上不具备优势**：不满足并行与迭代先验的任务

- 非函数：需要想办法将问题转化。
- 非迭代（非结构）：该层状态不是由上层状态构成的任务（如：很深的CNN因为有max pooling，信息会逐渐丢失。而residual network再次使得迭代的先验满足）

到此我们讨论完了神经网络最基础的，也是最重要的知识。在实际应用中仍会遇到很多问题（尤其是神经网络对noise的克服更加巧妙）。随后YJango会再和大家一起分析过深后会产生什么效果，并一步步引出**设计神经网络的本质**。