Deep Learning and Shallow Learning

http://freemind.pluskid.org/machine-learning/deep-learning-and-shallow-learning

posted on Free Mind on August 26, 2013
generated with pandoc on December 3, 2015
category: Machine Learning

tags: Deep Learning, Survey, Learning Models, Reading

由于 Deep Learning 现在如火如荼的势头，在各种领域逐渐占据 state-of-the-art 的地位，上个学期在一门课的 project 中见识过了 deep learning 的效果，最近在做一个东西的时候模型上遇到一点瓶颈于是终于决定也来了解一下这个魔幻的领域。

据说 Deep Learning 的 break through 大概可以从 Hinton 在 2006 年提出的用于训练 Deep Belief Network (DBN) 的算法开始，打破了持续了数十年的多层神经网络效果很差的尴尬局面，其之后各种其他的算法和模型也相继被提出来并在各个应用领域中大展神威。而近来 Google 聘请 Hinton、纽约时报的封面报道等公众媒体的宣传，更是使得 deep learning 变得大红大紫。记得之前在老板的某个 draft 中看到过这样一段话：

> *I do not expect that this paper will ever be published in the usual journals. "Success" for a paper published in this way would consist, I believe, of making an impact – measured in terms of citations for instance – and perhaps of being eventually "reviewed" in sites such as Wired or Slashdot or Facebook or even in a News and Views-type article in traditional journals like Science or Nature.*

在学术界也是，比如在各个应用领域里，Automatic Speech Recognition (ASR) 中不仅 deep learning 超越了传统的 state-of-the-art 算法，而且超越程度之大使得 ASR 领域本身迎来了一次新的 break through [Hinton et al., 2012a]；Collaborative Filtering 里，Deep Learning 在 Netflix 最后获奖算法中占据重要地位；Computer Vision (CV) 里除了在各种大型 benchmark 数据库上得到超越 state-of-the-art 结果（例如 [Krizhevsky et al., 2012]）之外，据说 Google 也在它的图像搜索中开始使用 Deep Learning；NLP 领域我不是很了解，不过从这个 Deep Learning for NLP (without Magic) 的 Tutorial 来看，Deep Learning 在 NLP 里也取得了相当的成功。甚至连纯机器学习理论的会 COLT 也开始凑这趟热闹了。deeplearning.net 上有一个 reading list，里面列举了一些各个领域关于 deep learning 的代表性文章。从 2013 年开始，deep learning 甚至有了自己专门的会：International Conference on Learning Representations (ICLR)。

从会议的名字也可以看出，deep learning 其实很重要的一点就是得到好的 representation，各种实验表明，通过 deep learning 的出来的网络，即使把最上层的分类/回归模型丢掉，直接把网络当做一个 feature extractor，把抽出来的特征丢到普通的 SVM 之类的分类器里，也经常会得到性能提高。虽然从信息论的角度来说，由于 Data Processing Inequality 导致 feature extraction 并不会在 "信息量" 上带来什么改善，但是从 practical 的角度来说，一个好的 representation 无疑是非常重要的。关于这一点，我最近听说了一个非常形象的例子：有人在抱怨乘法比加

法难算好多，比如 9480208 和 302842 的和，只要各位对齐，一位一位地加并处理好进位就好了，即使连我这样的渣口算能力估计都没问题；但是如果是乘法的话……但是其实这里的难易程度是由于我们常用的数字的十进制表达偏向于加法计算的缘故。如果我们换一种表达：每一个数字可以等价地表达为它的素数因子的集合，例如

$$9480208 \triangleq \{2, 2, 2, 2, 131, 4523\}$$
$$302842 \triangleq \{2, 53, 2857\}$$

那么两个数相乘就再简单不过了：

$$9480208 \times 302842 \triangleq \{2, 2, 2, 2, 2, 53, 131, 2857, 4523\}$$

反过来在这种 representation 下做加法就很困难了。基于同样的原因，representation 的问题在机器学习以及相关领域中一直是一个非常重要的研究课题。因为不同的问题、不同的数据和不同的模型，合适的 representation 可能会很不一样，而找到正确的 representation 之后往往就可以事半功倍。

在特定的问题中，一般采集到数据之后会进行一些特征提取的处理，例如 Vision 里的 SIFT + Bag of Words，或者 Speech 里的 MFCC 之类的特征，这些特征提取的算法往往都是人们根据该问题数据的特征人工设计出来的，并且一直以来设计更好的 feature 实际上在各个领域里也是非常重要的研究问题。而现在 deep learning 的结果展示比较喜欢做的一件事情就是从“原始数据”（比如 Vision 里的像素 bitmap）出发自动学习 representation，并给出比之前精心设计的人工 feature 的效果还要好。不过我觉得这也并不代表说 deep learning 就在这里是万能的，因为一方面能够有效地结合已知的领域内的 domain knowledge 实际上是非常重要的一个特性，另一方面，deep network 也并不是像一个 black box 一样直接把 raw data 丢过去它就能 magically 给出像样的特征来。deep model 训练困难似乎算是得到公认的了；并且比如像 convolutional neural network (CNN) 这样的模型其网络结构本身就是根据 underlying data 本身所要求的 invariance 特性而人工设计的；再比如像在 speech 里目前效果最好的做法似乎也还是在基于 speech data 的各种经典处理工序之后得到的 Mel Frequency Filter Bank 数据上而不是最原始的声音波形上做 deep learning。

除了手工特征提取之外，deep learning 之前也有许多其他所谓“shallow” 的 data-driven 的特征提取的算法。最经典的 PCA 降维可以从去除噪音等等各个方面来进行解释。像生物的 microarray 之类的数据上，每个样本点的维度非常高，同时由于采集样本的成本高昂，导致样本的数量有非常低，所以各种各样的降维或者特征选择的方法涌现出来，以限制模型的复杂度，避免在小样本数据上出现严重的过拟合问题。

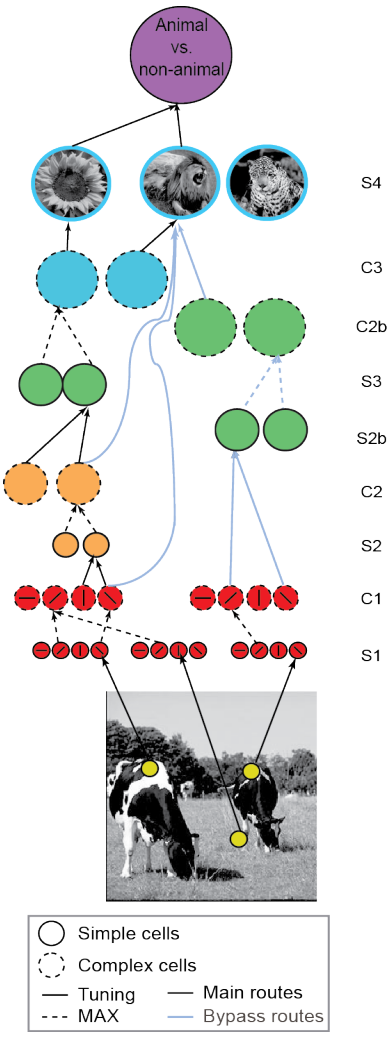

Figure 1: **HMAX。** Convolutional feedforward object recognition models inspired by the primate ventral visual pathway.

世界的另一头，是数据如白菜一样便宜的 “big data” 时代，样本的充裕（和计算机性能的提升）使得训练更加复杂的模型成为可能，因此反过来又 “升维” 以获得更丰富的数据表达，这里 Kernel Method 是一个经典的工具。核方法的基本思路是通过一个正定核 K 诱导出来的线性映射 $\Phi_K : \mathcal{X} \to \mathcal{H}_K$ 将数据映射到一个 Reproducing Kernel Hilbert Space (RKHS) 中，然后使用 RKHS 中的线性模型来处理数据。

这里的 Φ_K 就相当于是一个非线性的特征抽取的过程。一方面，通过核函数可以有效地在原始的数据空间维度的复杂度下面对映射过后的特征空间中的点进行（内积）计算；另一方面，诸如像高斯核之类的核函数对应的特征空间实际上是无限维空间，可以说有相当大的自由度。此外，核方法还是 non-parametric 的，也就是说，并不需要假设 target hypothesis h 是某种形式的函数，而是可以在整个 RKHS $\mathcal{H}_K$ 中寻找最接近 h 的函数，然后通过 Representer Theorem 可以把这些优化函数转换为有限维空间上的优化问题，最终的近似 target $\hat{h}$ 也变成了由核函数在训练数据上所 “Interpolate” 成的函数：

$$\hat{h}(\cdot) = \sum_{i=1}^{N} \alpha_i K(x_i, \cdot)$$

所以说从某种意义上来说，核方法似乎并不是在使用整个 $\mathcal{H}_K$，而只是在用一个由训练数据映射后的 $\{K(x_i, \cdot)\}_{i=1}^N$ 张成的子空间在做近似。由于 Representer Theorem 实际上是保证在整个 $\mathcal{H}_K$ 中优化和在这个子空间中优化的最优解是一样的，所以这里的局限性其实并不是来自于 Kernel Method，而是来自于使用有限的训练数据通过 Empirical Risk Minimization 去近似 Risk Minimization 的时候造成的问题。另外，核方法在 learning theory 方面也有非常多的研究和结论。虽然到目前为止 Machine Learning Theory 里的 Theoretical Bounds 很少有可以用来实际直接指导具体问题中的诸如参数选择之类的事情，但是理论上的研究工作仍然是不能忽视的。

核方法的工作原理有一个比较粗糙的直观解释，考虑最常用的高斯核 $K(x, y) = e^{-\lambda \|x-y\|^2}$，其中 $\lambda > 0$ 是核的参数。对于一个特定的点 x 来说，取决于实现给定的 λ 的大小，在一定半径范围之外的数据点 x_i，$K(x_i, x)$ 的值基本上就可以小到可以忽略不计了。所以 $\sum_{i=1}^N \alpha_i K(x_i, \cdot)$ 的线性组合其实只是在 x 点的周围一个 local neighborhood 里求和。

也就是说，可以近似地看成是在每一个 local neighborhood 里进行局部的线性回归，同时又全局地限制重叠的那些 local neighborhood 所对应的线性回归的重叠的系数必须要相等，可能还有一些全局的 regularization 之类的。如果一个函数比较光滑，或者我们的数据点足够密集，而核函数的 λ 又选得比较好使得 local neighborhood 大小比较合适的话，函数通常在每个点的局部领域里都能很好地通过线性函数来进行近似。

不过，看似优良的性质同时也受到了质疑 [Bengio et al., 2005]，因为如果在训练数据 cover 比较少的区域的话，似乎这样的差值的准确性就

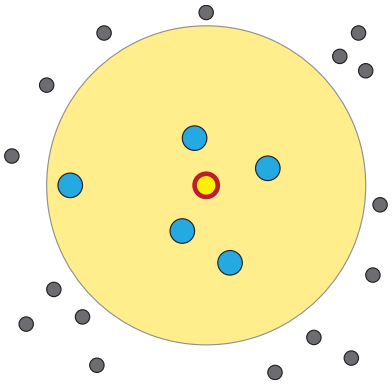

Figure 2: **核方法通过 local neighbors 进行插值计算。**

有待商榷了，而且这样的方法看起来更像是在做 “memorizing”，而不是 “learning”。Yoshua Bengio 把这种称作是 local representation，这类基于 local smoothness 假设的模型严重地(通常指数增长地)依赖于数据的维度(或者数据流形的本征维度)，从而造成维度灾难。在 [Bengio et al., 2005] 一文中他还进一步指出这类方法同样不适合学习在局部有很多变化的函数（例如像高频的 sin 函数之类的)。实际应用中的那些问题对应的函数究竟是局部平滑的还是高频变化的呢？高频变化的函数如果是毫无规律的，那当然这个问题本身从信息论的角度来说就是困难的，但是如果局部变化但是全局却体现出规律性的话，即使局部算法无法处理，从全局出发考虑的算法也许还是可以处理的。最简单的例子就是单纯的高频 sin 函数本身。理想情况下，如果我们有这个函数的全局模式的先验知识的话，那么只要在合适的 representation 下，问题通常是可以转化为 “简单” 的形式，但是这样的全局模式是否可以让算法自动地通过数据学习出来呢？

而对为什么 deep learning 会有更好的效果的一个尝试的解释就是 deep learning 得到的是所谓的 “distributed representation” [Bengio, 2009, Bengio et al., 2013]。

核方法还有另外一个问题就是在实际使用中基本上也没有太多的选择，比如 LIBSVM 的帮助信息里可供选择的核函数为：

```
-t kernel_type : set type of kernel function (default 2)
    0 -- linear: u'*v
    1 -- polynomial: (gamma*u'*v + coef0)^degree
    2 -- radial basis function: exp(-gamma*|u-v|^2)
    3 -- sigmoid: tanh(gamma*u'*v + coef0)
```

虽然在一些特殊的领域可能会有诸如文本核之类的 domain specific 的核函数，但是要构造一个核函数也并不是一件 trivial 的事情，因为你必须得保证它是正定的。由于这方面的限制，也有一些工作研究如何将类似于 kernel 的那一套 framework 推广到普通的不需要正定性质的 similarity function 上 [Balcan et al., 2008]。

另一个问题是在计算复杂性方面的：核方法的计算中牵涉到的核矩阵是 $N \times N$ 大小的，这里 N 表示训练数据点的个数，在大量数据的应用中，核矩阵不论从计算上还是存储上都变得无比困难，虽然也有很多通过采样子集的方法去对核矩阵进行近似的研究 [Williams and Seeger, 2000]，但是很多时候还是不得不 fall back 到 linear kernel 上，使用另一套 formulation，可以允许计算复杂度随着数据的维度增长而不是随着数据点的个数而增长。但是这样一来 kernel 所带来的非线性特征映射的功能就没有了，因为所谓 linear kernel 实际上就是等价于不使用任何 kernel 嘛。

抛开计算性能方面的考虑，就拿普通的核函数来说的话，和 deep learning 中 representation learning 还有一个重要的区别就是像高斯核之类的

核函数，其对应的表达都是事先设定好的，而不是通过数据得出来的。当然，data-driven 的 kernel 方面的研究也是有不少的，比如之前有人指出 Isomap、LE 和 LLE 之类的经典的流形学习算法其实是等价于构造一个特殊的 data driven 的 kernel 然后做 kernel PCA [Ham et al., 2004]，更 explicit 的 data-driven 的 Kernel 是直接将 Kernel Matrix 当做一个变量（正定矩阵）通过 Semi-Definite Programming (SDP) 来进行优化 [Lanckriet et al., 2004]，不过 SDP 虽然是凸优化，但是基本上数据规模稍微大一点就慢到不行了。

除此之外，还有一支相关的工作是 Multiple Kernel Learning (MKL)，将多个 kernel 组合起来，因为 kernel 组合时候的系数是根据 training data 优化而得的，所以这实际上也是 data-driven 的 representation learning 的一种特殊情况，并且，由于在 kernel 的基础上在做一层组合，所以看起来已经比普通的 shallow architecture 要多一层了。Kernel 组合的系数有点类似于多层神经网络中的 hidden layer。一般把以前最常用的只有一层（或者没有）hidden layer 的神经网络（或者其他）模型叫做 shallow 的，而超过一层以上的 hidden layer 的称为 deep 模型。

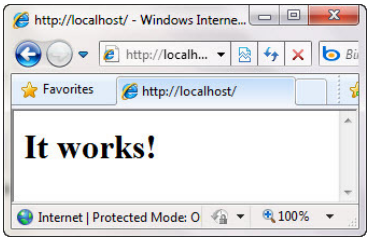

Figure 3: **The classical Apache "It works" page**. Image from the Internet.

得到优良的 representation 是至关重要的问题，而仅仅基于 Kernel 的那种 local representation 又在 AI 相关的复杂问题前面碰到了各种瓶颈。但是为什么一定要 deep 呢？理由有各种各样的，但是我觉得最重要的一个理由，开源的 Web Server Apache 在若干年来就一直在悄悄强调了：每次刚装好 Apache 打开主页面时显示的那句：

> *It Works!*

正如同一开始提到的那样，虽然 deep model 据说很多 tricky 各种难以训练，但是人们还是在各个应用领域里成功地用 deep learning 的方法击败甚至是完败了以前的各种 state-of-the-art。从实际应用的角度来说这已经足够有说服力了，但是求知欲旺盛的人类当然还想知道究竟为什么会 work。对这方面进行诠释和探索的工作也挺多，下面列举个别的。

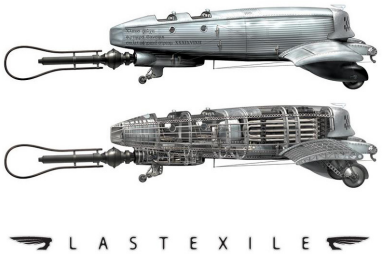

Figure 4: **Vanship from 《Last Exile》**。

其中一个解释是从生物或者神经科学角度：因为就目前对于人类的智能系统，特别是视觉系统的研究方面表面，人脑对于这方面的信息处理机制就是一个逐层抽象的 hierarichical architecture [Serre et al., 2007]。虽然听起来很有说服力，但是其实也并没有说明为什么多层结构更好，而只是说人类这样我们就跟着学了，所以这听起来多少有点让人想提高警惕不要被蒙混过关，Yann LeCun 在某个 tutorial 中举过一个比较形象的例子：人类制造飞机并不是简单地跟着动物学了在手上贴两个翅膀就能飞的，而是在了解了为什么那样的结构能飞的本质原因，也就是背后的空气动力学之类的理论之后，才真正掌握了天空飞行的 “技能”。

另一方面是关于刚才讨论过的 Kernel 之类的方法无法很好地处理的所谓 Highly Variable Functions [Bengio et al., 2005]，而 deep architecture 则可以比较有效地表达这样的映射。更 general 地，虽然我们刚才提到只有一层 hidden layer 的神经网络就已经具有一定的 universal 性质，但是却不一定是 efficient 的：存在**某些**函数可以简洁地通过 k 层

逻辑门网络计算出来，但是如果限制为 $k-1$ 层的话，就需要指数级别的逻辑门才行 [Bengio, 2009]。当然还有很多问题没有回答，比如说这些逻辑门构成的 bool 电路和机器学习问题中所遇到的函数之间的联系是怎么样的 [Orponen, 1994]？机器学习中所碰到的问题是否是这样的 highly variable 的、必须要用 deep architecture 才能有效表达的？这样的函数空间纠结是不是 learnable 的？在优化和求解方面有哪些困难 [Glorot and Bengio, 2010]？是否/如何能保证学习的 generalization performance？等等等等。

实际上人们从上个世纪就各种尝试训练像人脑一样的多层复杂神经网络，但是通常在神经网络的层数变大以后都无法训练出理想的模型，特别是在证明只需要一个 hidden layer 就能保证神经网络能够表达任意的 bool 函数 [Mendelson, 2009]，更是变得不太有动力了。所以除了 Convolutional Network 这类结构经过专门设计的神经网络之外，general 的 deep architecture，直到 2006 年 Hinton 他们引入 greedy layer wise pre-training [Hinton et al., 2006] 之后人们才第一次体会到了 deep 模型的威力。在基于 Restricted Boltzmann Machine (RBM) [Hinton et al., 2006] 的 pre-training 之后，又冒出了基于各种 Auto Encoder (AE) 变种 [Vincent et al., 2010, Rifai et al., 2011] 以及甚至是 supervised 的 layer-wise pre-training [Bengio et al., 2006]。

所以肯定会有人要问：为什么 pre-training 会 work？是不是一定要 pre-training 才能 work？等等。一般来说，训练神经网络的目标函数是非常不好优化的，比如说，有非常非常多的局部最优值等等。通常认为使用 pre-training 的结果作为 back-propagation 的初始化有助于将 (stochastic) gradient descent 的初始搜索点放在一个比较好的地方，从而收敛到比较好的（局部最优）解。另外，pre-training 还被认为是起到 regularization 的作用，能够增强 generalization performance。关于这方面的详细讨论，可以参考 [Erhan et al., 2010]。

至于是否一定要做 pre-training，从实验结果方面，我们已经知道，当训练数据足够多的情况下，选择好合适的（随机）初始值和神经元之间的 non-linearity 的话，不使用 pre-training 而直接进行 supervised training 也是可以得到很好的效果的 [Ciresan et al., 2010, Glorot et al., 2011, Sutskever et al., 2013]。不过这些结果通常都是在大量数据的情况下，结合各种 trick [Montavon et al., 2012]，再加上高性能的 GPU 设备和特别优化的并行算法，在训练了 “足够长” 的时间之后得到的结果。所以为什么在 “大数据” 时代和 “GPU 并行” 时代之前没有能很成功地训练出 deep neural network 模型似乎也并不难解释。

而更深入的分析和 justification 方面，则通常从 “deep architecture 的训练为什么困难” 这个问题出发去探讨 [Glorot and Bengio, 2010]。一般认为，训练 deep neural network 的时候，目标函数本身有非常多的 local minima 和 plateaus，一阶的 gradient descent 方法很容易陷入局部最优而无法自拔，因此人们自然地会想要去尝试二阶方法。不过由于神经网络的参数非常多，Hessian 矩阵不仅计算上有困难，即使是用各种近似

的方法，光是要存储整个 Hessian 矩阵都比较麻烦。因此其中一个叫做 Hessian Free (HF) 的二阶优化算法 [Martens, 2010] 显得特别有意思，它利用 R-operator [Pearlmutter, 1994] 直接计算 Hessian 矩阵与一个向量的乘积，而不是先把 Hessian 矩阵整个算出来再用普通矩阵运算去乘以该向量。实验结果表明使用 HF 二阶优化，可以在不使用任何 pre-training 的情况下取得非常好的效果。

这里中途插一句：有一个叫做 Theano 的 Python 库，提供了 deep learning 优化相关的各种 building block，比如提供了符号运算自动推算 gradient 的功能，所以就不用自己去手算 gradient 写 back-propagation 了，并且也集成了用于二阶优化的 R-operator。最终计算用代码会自动编译为本地代码以实现快速执行，并且在 GPU 设备存在的情况下还可以无缝地编译为 GPU 并行代码来加速计算（虽然目前好像还只支持 CUDA 的样子）。有一个 Deep Learning Tutorial 就是使用 Theano 来介绍和实现了几个主流的 deep learning 算法。

回到刚才的问题，HF 优化取得的成功，可以说是打开了一扇门：直接从 general 的优化算法入手，也会是一个非常值得探索的方向。不过 deep architecture 的训练除了 local minima 和 plateaus 之外，还有一个问题就是网络的最高两层还非常容易 overfit，所以光看目标函数的优化有时候也并不能太说明问题：由于基本上都被最上面两层 overfitting 去了，流回下面的层的信息很少很少，所以下面层的 weights 几乎没有得到什么 training，还停留在原始的 random initialization 阶段，结果这样的训练结果几乎完全没有 generalization 能力。进来关于 rectifier non-linearity [Glorot et al., 2011, Krizhevsky et al., 2012] 相关的研究中的一个叫做 maxout [Goodfellow et al., 2013] 被发现能够使得底层的权重得到更多的 training。另外，诸如 dropout [Hinton et al., 2012b, Wang and Manning, 2013] 一类的添加 noise 也在实践中被用作强大的 regularizer 来避免 overfitting。

虽然提到 neural network 首先想到的肯定是 overfitting，大家的着眼点也差不多都是试图解决 overfitting 的问题，但是最近的一些实验 [Dauphin and Bengio, 2013] 表明，在数据和神经网络的规模达到一定程度之后，似乎由于优化问题的困难，导致 under fitting 的问题也出现了。还有其他各方面的一些困难，可以参考 Yoshua Bengio 在最近的一篇文章 [Bengio, 2013] 中总结了一下目前在 deep learning 中碰到的各种问题和挑战，以及可能的解决思路等等。

最后提一句关于应用方面，我倒是并没有专门去做全面的 survey，但是目前满天飞的 Deep Learning 相关的应用似乎大都集中在 AI 相关的经典问题（例如 Objection Recognition、Speech Recognition、NLP 之类的）方面，或者更 general 一点，很多工作集中在 classification 方面。所以说让我觉得挺感兴趣的一点是不知道这类 deep 模型是否是对于 AI 相关的问题有一些特殊的结构优势（类比人类智能系统的层级抽象机制），或者说这类模型是否在其他非传统 AI 领域也能取得远超其他普通的 shallow 模型的效果呢？另外就是层级抽象或者是像 convolutional network 那样

逐层提高 invariablility 的机制对于 classification 问题来说似乎是比较自然的，但是对于 regression 呢？似乎比较少看到有用 deep neural network 去解决具体的 multi-output regression 的问题的例子的样子。

至于具体的 deep learning 的模型以及相关的 training 的算法的细节之类的，原本想有时间的话也详细整理一下，但是好像暑假即将结束，我自己也挖了好多坑都还没有填，所以一时半会似乎不太能写更详细的东西了。Deep Learning 将会如何发展？究竟是否是 AI 的圣杯？就拭目以待了。:)

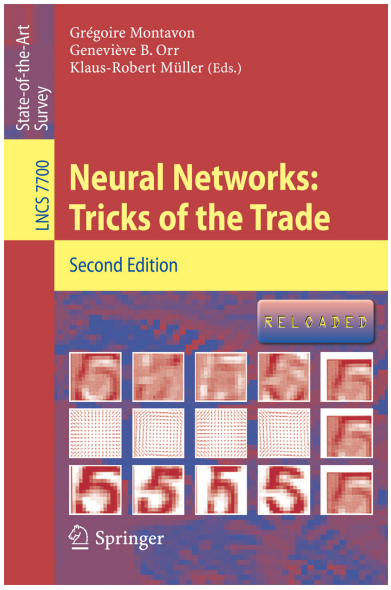

Figure 5: **Neural Networks: Tricks of the Trade (2nd Edition)**.

References

[Balcan et al., 2008] Balcan, M.-F., Blum, A., and Srebro, N. (2008). A theory of learning with similarity functions. *Machine Learning*, 72(1-2):89–112.

[Bengio, 2009] Bengio, Y. (2009). Learning deep architectures for ai. *Foundations and Trends in Machine Learning*, 2(1):1–127.

[Bengio, 2013] Bengio, Y. (2013). Deep learning of representations: Looking forward. In *SLSP*, pages 1–37.

[Bengio et al., 2013] Bengio, Y., Courville, A. C., and Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(8):1798–1828.

[Bengio et al., 2005] Bengio, Y., Delalleau, O., and Roux, N. L. (2005). The curse of highly variable functions for local kernel machines. In *NIPS*.

[Bengio et al., 2006] Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. (2006). Greedy layer-wise training of deep networks. In *NIPS*, pages 153–160.

[Ciresan et al., 2010] Ciresan, D. C., Meier, U., Gambardella, L. M., and Schmidhuber, J. (2010). Deep, big, simple neural nets for handwritten digit recognition. *Neural Computation*, 22(12):3207–3220.

[Dauphin and Bengio, 2013] Dauphin, Y. and Bengio, Y. (2013). Big neural networks waste capacity. *CoRR*, abs/1301.3583.

[Erhan et al., 2010] Erhan, D., Courville, A. C., Bengio, Y., and Vincent, P. (2010). Why does unsupervised pre-training help deep learning? *AISTATS*, 9:201–208.

[Glorot and Bengio, 2010] Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. *AISTATS*, 9:249–256.

[Glorot et al., 2011] Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep sparse rectifier neural networks. *AISTATS*, 15:315–323.

[Goodfellow et al., 2013] Goodfellow, I. J., Warde-Farley, D., Mirza, M., Courville, A. C., and Bengio, Y. (2013). Maxout networks. In *ICML*.

[Ham et al., 2004] Ham, J., Lee, D. D., Mika, S., and Scholkopf, B. (2004). A kernel view of the dimensionality reduction of manifolds. In *ICML*.

[Hinton et al., 2012a] Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., and Kingsbury, B. (2012a). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97.

[Hinton et al., 2006] Hinton, G. E., Osindero, S., and Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554.

[Hinton et al., 2012b] Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2012b). Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1106–1114.

[Lanckriet et al., 2004] Lanckriet, G. R. G., Cristianini, N., Bartlett, P. L., Ghaoui, L. E., and Jordan, M. I. (2004). Learning the kernel matrix with semidefinite programming. *JMLR*, 5:27–72.

[Martens, 2010] Martens, J. (2010). Deep learning via hessian-free optimization. In *ICML*, pages 735–742.

[Mendelson, 2009] Mendelson, E. (2009). *Introduction to Mathematical Logic*. Chapman and Hall/CRC, 5th edition.

[Montavon et al., 2012] Montavon, G., Orr, G., and Muller, K.-R. (2012). *Neural Networks: Tricks of the Trade*. Springer, 2nd edition.

[Orponen, 1994] Orponen, P. (1994). Computational complexity of neural networks: A survey. *Nordic Journal of Computing*.

[Pearlmutter, 1994] Pearlmutter, B. A. (1994). Fast exact multiplication by the hessian. *Neural Computation*, 6(1):147–160.

[Rifai et al., 2011] Rifai, S., Vincent, P., Muller, X., Glorot, X., and Bengio, Y. (2011). Contractive auto-encoders: Explicit invariance during feature extraction. In *ICML*, pages 833–840.

[Serre et al., 2007] Serre, T., Kreiman, G., Kouh, M., Cadieu, C., Knoblich, U., and Poggio, T. (2007). A quantitative theory of immediate visual recognition. *Progress in brain research*, 165:33–56.

[Sutskever et al., 2013] Sutskever, I., Martens, J., Dahl, G., and Hinton, G. (2013). On the importance of initialization and momentum in deep learning. In *ICML*.

[Vincent et al., 2010] Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., and Manzagol, P.-A. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *JMLR*, 11:3371–3408.

[Wang and Manning, 2013] Wang, S. and Manning, C. (2013). Fast dropout training. In *ICML*.

[Williams and Seeger, 2000] Williams, C. K. I. and Seeger, M. (2000). Using the nystrom method to speed up kernel machines. In *NIPS*, pages 682–688.