deep learning 学习笔记

计算机科学与技术系 52 班杨定澄　　学号：2015011274
E-mail:892431401@qq.com

1　前言

虽然神经网络算法早在上个世纪 80 年代就提出了沿用至今的反向传播算法，但是 deep learning 却是在 2006 年才得以提出。这次作业要求阅读的“Reducing the Dimensionality of Data with Neural Networks”这篇论文，就是使深度学习算法火起来的第一步。

这篇论文涉及到许多我们之前没有学习过的概念、算法，比如 PCA（主成分分析），RBM（限制性玻尔兹曼机）、autoencoder（自动编码器）等等，相关知识的缺乏给我的理解带来了一定的难度（比如以为课上讲的玻尔兹曼机就是 RBM）。再查阅了一些和该论文有关的相关资料后，才稍微明白其来龙去脉。

2　论文的主要内容

为了便于理解，我们首先来概括一下论文的内容。

神经网络虽然早已被提出来，**但是要想得到优秀的效果其实需要初始值足够接近一个非常好的值**，因此这篇论文做的第一件事，就是用一种“pretraining”的方法，对数据进行“预训练”，得到神经网络参数的一个较好的初值。这是论文中的核心部分。

接着，这篇论文究竟做了什么呢？他所做的是将数据进行降维。

在许多问题中，问题的维度是相当高的（比如图像处理问题的维度就是像素点个数那么多），直接求解会有非常多的问题。因此我们有很多数据降维的算法，比如 PCA。而这篇论文最终得到的结果，就是一个新的数据降维的方法，并且通过实验表明其效果是优于 PCA 的。这也就是这篇论文最终的成果。

所以论文大概讲了这么几件事：

1. 通过“预训练”来得到一个比较好的初值（这部分直接写到了论文的最后一页）。我们称之为“pretraining”。
2. 做完 pretraining 后，就会输出一个“code layer”，也就是降维的结果。我们通过一个“unrolling”操作，将降维结果进行解码，得到一个与原始输入较为相似的结果。
3. 为了让解码结果与原始输入尽可能的相似，最后使用神经网络的反向传播算法，对参数进行微调。我们称之为“fine-tuning”操作。

3 pretraining

这是论文最主要的内容，他主要包含两个部分。

3.1 autoencoder

autoencoder（自动编码器）是一个多层神经网络，但是他不是用来做分类问题的，而是学习一个输入什么就输出什么的东西。当然如果真的输入什么就输出什么，那就毫无意义了，事实上他输出的是一个用来“代表”输入数据的东西，或者说输出的是输入数据的“特征”。

首先来考虑第一个隐含层，他的维度比输入数据要小，可以看做是一个编码器。数据输入进来之后，他就会进行编码，输出一些东西来代表输入进来的数据。

但是怎么说明输出的东西代表了输入的数据呢，我们就再弄一个解码器，他通过将编码器的输出进行解码，然后要尽可能的让解码器的输出结果和最开始的输入数据一样。

通过学习神经网络的参数，我们可以让差异尽可能小。我们再让第一层编码器的输出，作为第二层编码器的输入，如此做下去。

这样子做了几层，就代表了几个原始数据的特征。每层的维度都会变小，就可以看作是将原始的输入数据进行降维了。最后一个编码器的输出，就是原始数据降维的最终结果。

autoencoder 完成了 pretraining 所要做的事情，而让 autoencoder 的效果更好，可以引入 RBM。

3.2 RBM

RBM（限制性玻尔兹曼机）不同于玻尔兹曼机里隐含层之间能互相连边，他要求可见层和隐含层构成一个二分图。即可见层内部和隐含层内部之间是没有边的。

和玻尔兹曼机相同的是，他也是假设节点是随机二值变量，概率分布满足玻尔兹曼分布。

假设可见节点集合是 v，隐藏节点集合是 h，一旦定好了 v，则 h 之间就是条件独立的了；同理一旦定好了 h，v 之间也是条件独立的。这是由于二分图的性质。

我们仍然定义能量函数 $E(v,h;\theta) = -\sum_{ij} w_{ij}v_ih_j - \sum_i b_iv_i - \sum_j a_jh_j$，$\theta$ 代表所有参数（即 w_{ij}, a_i, b_j）。

而 $P_\theta(v,h) = \frac{1}{Z}e^{-E(v,h;\theta)}$，其中 Z 是归一化常数。

我们根据隐藏节点是条件独立这一性质，可以推导出

$$P(h_j = 1|v) = \frac{1}{1 + exp(-\sum_i w_{ij}v_i - a_j)}$$

同理可以推导出

$$P(v_i = 1|h) = \frac{1}{1 + exp(-\sum_j w_{ij}h_j - b_i)}$$

如果定下了 v，就能定下 h 的概率分布，就能通过 h 的概率分布得到 v 的概率分布并进行比较。通过学习参数，可以让概率最大。

在这篇论文中，RBM 和 autoencoder 结合了起来，他是用 RBM 来对 autoencoder 的每一层进行学习，让初值更好。

4 unrolling

再 pretraining 的时候既学习编码器的参数，同时也学了解码器的参数。

由于所有的解码器参数已知，解码也就非常容易了，直接推回去就好。

5 fine-tuning

最后再算一次梯度下降来让差异函数最小。

他这里的差异函数不是用距离平方和进行度量，而是用了一种叫交叉熵损失函数的度量方法，最小化 $-\sum_i p_i \log \hat{p}_i - \sum_i (1-p_i)\log(1-\hat{p}_i)$