《数据库应用》心得与体会

BY 蒋方朔

13120031
2014.11.25

目录

1 课程心得 3

2 神经网络简介 3

3 神经网络的基本结构 3

4 神经网络的训练 5

5 手写识别 6

6 手写识别结果 7

7 改进方案 8

8 总结 8

1 课程心得

第一次上覃老师的《 数据库应用》这门课程时，本以为会主要是以IBM DB2数据库应用为主，但是覃老师却提出给我们分配多一点时间来给我们讲数据挖掘相关的知识，十分高兴。 因为在之前学校的课程中学习的大多是非常基础的知识，而像数据挖掘这种处在前沿的技术没有办法从学校获得，在上完《 数据库应用》这门课之后，我也是获益匪浅。

在上这门课之前，我就了解过一些关于机器学习方面的知识，并且在Coursera上学习了斯坦福大学吴恩达教授的机器学习课程，并对于课上学到的很多算法十分感兴趣，尤其是神经网络。 而由于机器学习与数据挖掘在许多方面有着共同点，所以我的这篇心得体会主要叙述一下关于神经网络的内容，并且有一个手写识别的样例作为神经网络的应用。

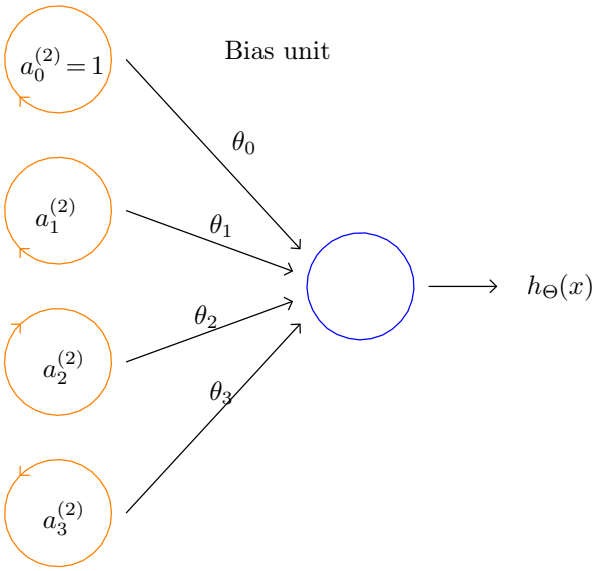
2 神经网络简介

人工神经网络（Artificial Neural Network, ANN），简称神经网络,是一种模仿生物神经网络的结构和功能的数学模型或计算模型。神经网络由大量的人工神经元联结进行计算。大多数情况下人工神经网络能在外界信息的基础上改变内部结构，是一种自适应系统。 现代神经网络是一种非线性统计性数据建模工具，常用来对输入和输出间复杂的关系进行建模，或用来探索数据的模式。

简单来说，神经网络就是通过大量数据的训练来改变自身的结构来对未知数据进行预测或者识别。

3 神经网络的基本结构

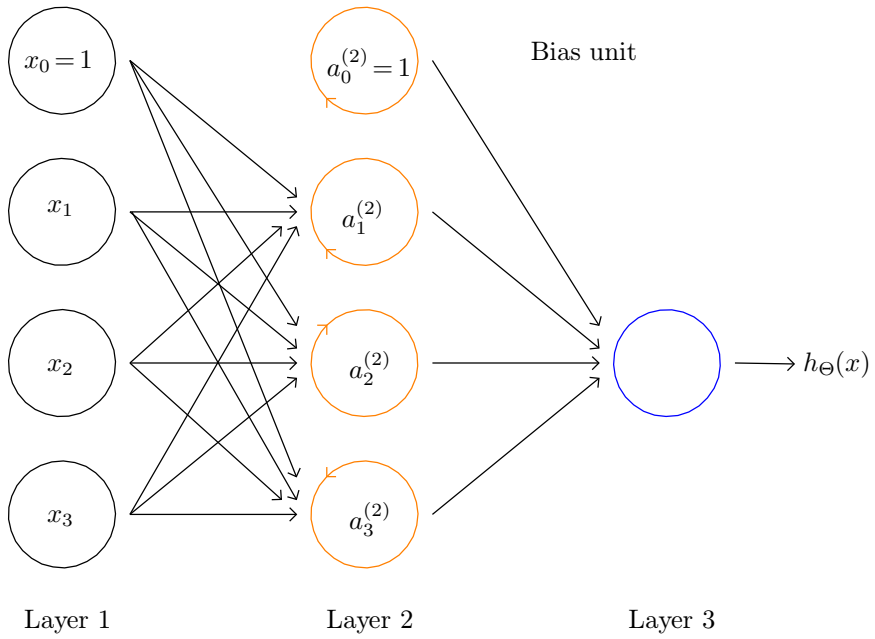
神经网络的由基本的单位神经元构成，神经元的基本结构如下图所示。



- 向量a为输入向量
- 向量θ为该神经元的权值
- $h_\theta(x)$为输出，一般使用符号函数

偏置单元作为a0=1，偏置单元作为计算时的常量，这样可以使神经元的输出表示为向量的乘积，然后经过传递函数计算之后作为输出，便于矩阵运算。

神经网络由输入单元，输出单元以及多个隐藏层组成，每个隐藏层由多个神经元构成，一个只有一层隐藏层的神经网络表示如下。



上图中Layer 1表示输入向量，Layer2为隐藏层，Layer3为输出。 将每个隐藏层节点的权值均放入矩阵θ，以方便矩阵运算，则对于上述神经网络的计算过程如下。

- $a_i^{(j)}$ = 第j层的第i个单元的输出
- $\Theta^{(j)}$ = 权值映射矩阵，从 j 层到 $j+1$层

$$\begin{aligned} a_1^{(2)} &= g\Big(\Theta_{10}^{(1)}x_0+\Theta_{11}^{(1)}x_1+\Theta_{12}^{(1)}x_2+\Theta_{13}^{(1)}x_3\Big) \\ a_2^{(2)} &= g\Big(\Theta_{20}^{(1)}x_0+\Theta_{21}^{(1)}x_1+\Theta_{22}^{(1)}x_2+\Theta_{23}^{(1)}x_3\Big) \\ a_1^{(2)} &= g\Big(\Theta_{30}^{(1)}x_0+\Theta_{31}^{(1)}x_1+\Theta_{32}^{(1)}x_2+\Theta_{33}^{(1)}x_3\Big) \\ h_\Theta(x)=a_1^{(3)} &= g\Big(\Theta_{10}^{(2)}a_0^{(2)}+\Theta_{11}^{(2)}a_1^{(2)}+\Theta_{12}^{(2)}a_2^{(2)}+\Theta_{13}^{(2)}a_3^{(2)}\Big) \end{aligned}$$

这就是前向传播算法，如果用矩阵运算表示前向传播算法，那么对于一个有两层隐藏层的神经网络，他的矩阵算法如下。

算法 前向传播算法

给出输入向量 x:

$$\begin{aligned} a^{(1)} &= x \\ z^{(2)} &= \Theta^{(1)}a^{(1)} \\ a^{(2)} &= g\big(z^{(2)}\big) \quad \Big(\text{add}\, a_0^{(2)}\Big) \\ z^{(3)} &= \Theta^{(2)}a^{(2)} \\ a^{(3)} &= g\big(z^{(3)}\big) \quad \Big(\text{add}\, a_0^{(3)}\Big) \\ z^{(4)} &= \Theta^{(3)}a^{(3)} \\ a^{(4)} &= h_\Theta(x)=g\big(z^{(4)}\big) \end{aligned}$$

4 神经网络的训练

训练神经网络采用后向传播算法,在介绍后向传播算法之前,先来看一下神经网络的花费函数。

花费函数:

$$J(\Theta)=-\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K}y_k^{(i)}\log\big(h_\Theta\big(x^{(i)}\big)\big)_k+\Big(1-y_k^{(i)}\Big)\log\big(1-h_\Theta\big(x^{(i)}\big)\big)_k\right]+\frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_l+1}\Big(\Theta_{\text{ji}}^{(l)}\Big)^2$$

该函数的右边项为Regularization项，目的是避免神经网络陷入过拟合的情况，这里相当于对参数进行惩罚,注意不对参数项中的偏置项进行惩罚。

该函数的左边项为m个输入k维向量的情况下,根据符号函数来计算的。

符号函数的定义如下

$$g(z) = \frac{1}{1-e^{-z}}(\text{Logistic}/\text{Sigmoid Function})$$

那么对于一个符号函数,他的花费函数为$J(\theta)$,然后根据符号函数的性质,将$J(\theta)$化简

$$\begin{aligned} J(\theta) &= \frac{1}{m}\sum_{i=1}^{m}\mathrm{Cost}(h_\theta(x),y) \\ \mathrm{Cost}(h_\theta(x),y) &= \begin{cases} -\log(h_\theta(x)) & \text{if } y=1 \\ -\log(1-h_\theta(x)) & \text{if } y=0 \end{cases} \\ J(\theta) &= \frac{1}{m}\sum\left[-y\log(h_\theta(x))-(1-y)\log(1-h_\theta(x))\right] \end{aligned}$$

有了花费函数，我们可以采用如下算法来快速计算$J(\theta)$的导数，其中δ表示对应层的每个神经元的误差。首先是根据前向传播算法求出输出值，然后计算输出值与标注数据的误差，然后根据后一层的误差去求前一层的误差,这就是后向传播算法,具体算法如下。

算法 后向传播算法

Training set $\left\{\left(x^{(1)},y^{1}\right),...,\left(x^{(m)},y^{(m)}\right)\right\}$
Set $\triangle_{\mathrm{ij}}^{(l)}=0$ (for all l,i,j). (used to compute $J(\Theta)$)
For $i=1$ to m
Set $a^{(1)}=x^{(i)}$
Perform forword propagation to compute $a^{(l)}$ for $l=2,3,...,L$
Using $y^{(i)}$, compute $\delta^{(L)}=a^{(L)}-y^{(i)}$
Compute $\delta^{(L-1)},\delta^{(L-2)},...,\delta^{(2)}$
$\Delta_{\mathrm{ij}}^{(l)}=\Delta_{\mathrm{ij}}^{(l)}+a_j^{(l)}\delta_i^{(l+1)}$

$$\begin{aligned} D_{\mathrm{ij}}^{(l)} &= \frac{1}{m}\Delta_{\mathrm{ij}}^{(l)}+\lambda\Theta_{\mathrm{ij}}^{(l)} \text{ if } j\neq 0 \\ D_{\mathrm{ij}}^{(l)} &= \frac{1}{m}\Delta_{\mathrm{ij}}^{(l)} \qquad \text{if } j\neq 0 \\ \frac{\partial}{\partial\Theta_{\mathrm{ij}}^{(l)}}J(\Theta) &= D_{\mathrm{ij}}^{(l)} \end{aligned}$$

上面算法中的δ的计算方法如下。

算法 后向传播算法中δ的计算

对第四层

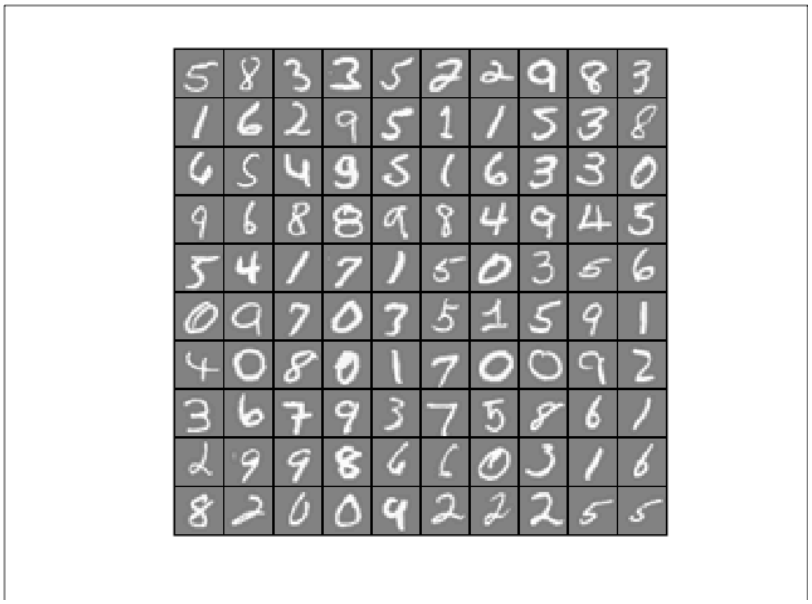
$$\begin{aligned} \delta^{(4)} &= a^{(4)}-y \\ \delta^{(3)} &= \left(\Theta^{(3)}\right)^T\delta^{(4)}.*g'\left(z^{(3)}\right) \\ \delta^{(2)} &= \left(\Theta^{(2)}\right)^T\delta^{(3)}.*g'\left(z^{(2)}\right) \end{aligned}$$

5 手写识别

在斯坦福的机器学习课程中，有关于神经网络的练习，主要内容为进行手写识别。下面就将这个练习作为例子来演示神经网络。相关代码由Matlab编写。

输入数据为5000组20*20像素的灰度图像,每组数据都有标注。

从数据中随机选取100个元素，显示如下。



在Matlab脚本的运行过程中，我们可以对优化算法的迭代次数进行限制，以达到最优的效果。

该程序使用有一层隐藏层，25个隐藏节点的神经网络进行训练，训练后重新在该训练集上测试准确度。

以下是几个主要文件的说明：

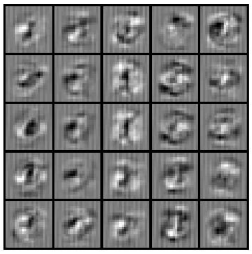
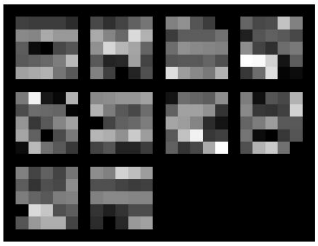
- main.m 主程序，读取数据并对神经网络进行训练，评价训练后的神经网络，并保存结果，可以设置优化函数的迭代次数
- display_theta.m 可视化保存的神经网络参数
- displayData.m 数据可视化函数
- fmincg.m 最优化函数
- nnCostFunction.m 通过正向传播算法以及反响传播算法，计算神经网络的花费的函数
- predict.m 根据计算得出的神经网络在数据集上计算准确率函数
- randInitializeWeigts.m 随机化初始参数函数
- sigmod.m 符号函数
- sigmodGradient.m 符号函数的导函数
- trained_theta_X_Y.mat 记录了在迭代X次之后的到的神经网络的参数，其准确率为Y%

6 手写识别结果

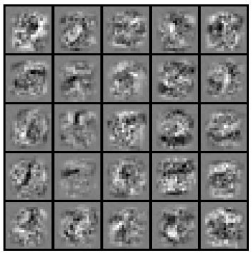
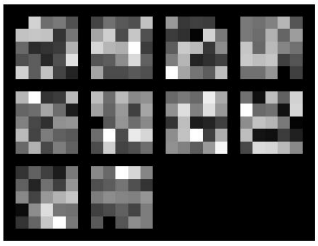
对神经网络进行训练,得到的最终结果如下:

迭代次数	花费时间	花费函数值	训练集上的准确率
40	1min	6.063369e-01	93.34%
1000	12min	3.118762e-01	99.64%
3000	35min	3.070351e-01	99.72%

经过40次迭代之后,第一个参数矩阵(左)以及第二个参数矩阵(右)的参数可视化效果如下:

经过3000次迭代之后,第一个参数矩阵(左)以及第二个参数矩阵(右)的参数可视化效果如下:

7 改进方案

I. 由于数据集数据量并不是很大,所以需要更多的数据来避免过拟合情况的发生。

II. 可以将现有测试集分为样例集和交叉验证集,在样例集上进行训练,在交叉验证集上进行验证。

III. 增加隐藏层的个数或者增加隐藏层内的节点数量。

8 总结

虽然说本学期已经进入尾声,《数据库应用》也结课了。 在这门课上学到了很多知识,即使有些知识点在上课时并没有了解的十分清楚,有些题目也有一定的难度,上机的时间也不足以解决问题,但我会在课下重新去看书或者网上查资料来学习。

通过一个学期的学习,不论是在课堂上的学习,还是课下自己看书,在Coursera上学习,都给我带来不小的收获,很多算法真的十分神奇,能在简单的计算以及控制下,就取得很好的效果,这也让我有更多地兴趣去深入学习。

[1]

1. neveralso@gmail.com