Week 4 Lecture Notes

机器学习：神经网络：表示 - ML:Neural Networks: Representation

非线性假设 - Non-linear Hypotheses

用具有许多特征的复杂数据集进行线性回归是非常笨拙的。假设你想从包含所有二次项的3个特征中创建一个假设：

$$g(\theta_0 + \theta_1 x_1^2 + \theta_2 x_1 x_2 + \theta_3 x_1 x_3 \\ + \theta_4 x_2^2 + \theta_5 x_2 x_3 \\ + \theta_6 x_3^2)$$

这就有了6个特征。计算所有多项式项有多少特征的精确方法是使用阶乘：http://www.mathsisfun.com/combinatorics/combinations-permutations.html> $\frac{(n+r-1)!}{r!(n-1)!}$。这样，我们将三个特征，两两组合：$\frac{(3+2-1)!}{(2!\cdot(3-1)!)} = \frac{4!}{4} = 6$。（注：你不必知道这些公式，我只是发现它有助于理解）。

对于100个特征，如果我们想用它们做一个二次假设函数，我们将得到$\frac{(100+2-1)!}{(2\cdot(100-1)!)}$=5050个新特征。

我们可以用$\mathcal{O}(n^2/2)$来近似表示所有的二次项的增长速度。如果你想在你的假设中包含所有的三次项，特征数量将以 $\mathcal{O}(n^3)$近似的速度增长。这些增长非常快速，因此随着特征数量的增加，二次或三次特征的数量增长非常迅速，并且很快变得不切实际。

示例：训练集是50x 50像素的黑白照片的集合，我们的目标是对汽车照片进行分类。如果我们比较每个像素，那么我们的特征集大小是n＝2500。

现在我们需要做一个二次假设函数。具有二次特征项，增长速度$\mathcal{O}(n^2/2)$）。所以我们的总特征数将是大约$2500^2/2 = 3125000$，这是非常不切实际的。

当有很复杂的假设函数，或许多特征时，神经网络提供了一种新的方式来进行机器学习。

神经元与大脑 - Neurons and the Brain

神经网络是对我们大脑运作方式的有限模仿。由于计算机硬件的进步，他们最近有了很大的发展。

有证据表明，大脑只使用一种“学习算法”来实现其不同的功能。科学家们试图切断（在动物大脑中）耳朵和听觉皮层之间的连接，并将视神经与听觉皮层重新连接，然后发现听觉皮层实际上也学会了看东西。

这一原理被称为“神经可塑性”，并有许多实例和实验证据。

模型表示 I - Model Representation I

让我们来看看我们如何用神经网络来表示假设函数。

在一个非常简单的层面上，神经元基本上是计算单元，将输入（**树突**）作为电信号输入（称为“峰”），引导信号到输出（**轴突**）。

在我们的模型中，我们的树突像输入特征$x_1 \cdots x_n$，输出是我们假设函数的结果：

在这个模型中，我们的X0输入节点有时被称为“偏置单元”，它总是等于1。

在神经网络中，我们使用与分类中相同的逻辑函数：$\frac{1}{1+e^{-\theta^T x}}$。有时，在神经网络中，把它称为sigmoid（Logistic）激活函数。

在神经网络模型中，“θ”参数有时被称为“权重”。

从视觉上看，一个简单化的表示：

$$\begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} \rightarrow \begin{bmatrix} \ \ \ \end{bmatrix} \rightarrow h_\theta(x)$$

我们的输入节点（第1层）进入另一个节点（2层），也就是假设函数的输出。

第一层称为“输入层”，最后一层称为“输出层”，它给出根据假设函数计算最终值。

我们可以在输入层和输出层之间添加中间层的节点，它们称为“隐藏层”。

我们将这些中间层或“隐藏”层节点标记为$a_0^2 \cdots a_n^2$，并称之为“激活单元”。

$$\begin{aligned} & a_i^{(j)} = \text{j层的第i个激活单元} \\ & \Theta^{(j)} = \text{控制j到j+1层的权重矩阵} \end{aligned}$$

如果我们有一个隐藏层，它看起来像这样：

$$\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \rightarrow \begin{bmatrix} a_1^{(2)} \\ a_2^{(2)} \\ a_3^{(2)} \end{bmatrix} \rightarrow h_\theta(x)$$

每个“激活”节点的值如下：

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

这就是说，我们通过使用3×4矩阵来计算我们的激活节点。我们将参数的每一行应用到我们的输入，以获得一个激活节点的值。我们的假设输出应用于第二层节点值之和的逻辑函数，这些值被乘以另一个参数矩阵$\Theta^{(2)}$，该矩阵包含第二层节点的权重。

每一层都有它自己的权重矩阵，$\Theta^{(j)}$。

这些权重矩阵的维数应这样确定：

$\text{如果网络在j层有}s_j\text{个节点，在j+1层有}s_{j+1}\text{个节点，那么}\Theta^{(j)}\text{的维数就是}s_{j+1} \times (s_j + 1).$

+1来自于“偏置节点”。换句话说，输出节点不包括偏置节点，但输入会包含。

示例：层1具有2个输入节点，层2具有4个激活节点。$\Theta^{(1)}$的维数将是4×3，其中$s_j = 2$，$s_{j+1} = 4$，因此$s_{j+1} \times (s_j + 1) = 4 \times 3$。

模型表示 II - Model Representation II

在本节中，我们将对上述功能进行矢量化的实现。我们将定义一个新的变量$z_k^{(j)}$，它包含了g函数中的参数。在前面的例子中，如果我们用变量z替换参数：

$$a_1^{(2)} = g(z_1^{(2)})$$
$$a_2^{(2)} = g(z_2^{(2)})$$
$$a_3^{(2)} = g(z_3^{(2)})$$

换言之，对于层j＝2和节点k，变量z将是：

$$z_k^{(2)} = \Theta_{k,0}^{(1)} x_0 + \Theta_{k,1}^{(1)} x_1 + \cdots + \Theta_{k,n}^{(1)} x_n$$

x和z^j的向量表示为：

$$x = \begin{bmatrix} x_0 \\ x_1 \\ \cdots \\ x_n \end{bmatrix} z^{(j)} = \begin{bmatrix} z_1^{(j)} \\ z_2^{(j)} \\ \cdots \\ z_n^{(j)} \end{bmatrix}$$

设$x = a^{(1)}$，我们可以将方程重写为：

$z^{(j)} = \Theta^{(j-1)} a^{(j-1)}$

我们将维数$s_j \times (n+1)$(其中s_j是激活节点的数量)的矩阵 $\Theta^{(j-1)}$乘以高度(n+1)的向量$a^{(j-1)}$。得到高度s_j的向量$z^{(j)}$。

现在得到了层j的向量$a^{(j)}$：

$a^{(j)} = g(z^{(j)})$

g函数应用到了向量$z^{(j)}上。

然后，添加一个偏置单元（等于1）到层j，也就是添加到我们计算过的$a^{(j)}$。此时元素$a_0^{(j)}$等于1。

然后计算另一个z向量：

$z^{(j+1)} = \Theta^{(j)} a^{(j)}$

通过将$\Theta^{(j-1)}$之后的一个Θ矩阵乘以我们刚算出来的所有激活节点的值，就得到这个最终的z向量。

最后的Θ矩阵$\Theta^{(j)}$将只有**一行**，因此我们的结果就是一个数。

得到最终结果：

$h_\Theta(x) = a^{(j+1)} = g(z^{(j+1)})$

注意，在**最后一步**中，即在层j和层j+1之间，我们做的事与在逻辑回归中做的事**完全相同**。

在神经网络中添加这些中间层，可以让我们更优雅地生成更有趣和更复杂的非线性假设。

示例和直觉 I - Examples and Intuitions I

关于神经网络的应用，一个简单示例是预测x_1 AND x_2，这是逻辑“与”运算符，并且仅当x_1和x_2都是1时才为真。

现在函数的图形看起来像这样：

$$\begin{bmatrix} x0 \\ x1 \\ x2 \end{bmatrix} \rightarrow [g(z^{(2)})] \rightarrow h\Theta(x)$$

x_0是偏置单元变量，总为1。

把第一个θ矩阵设为：

$\Theta^{(1)} = [-30\ 20\ 20]$

仅仅当x_1和x_2都是1的时候，假设函数的输出才是正的。换言之：

$$\begin{aligned} & h\Theta(x) = g(-30 + 20x_1 + 20x_2) \\ & x_1 = 0 \quad and \quad x_2 = 0 \quad then \quad g(-30) \approx 0 \\ & x_1 = 0 \quad and \quad x_2 = 1 \quad then \quad g(-10) \approx 0 \\ & x_1 = 1 \quad and \quad x_2 = 0 \quad then \quad g(-10) \approx 0 \\ & x_1 = 1 \quad and \quad x_2 = 1 \quad then \quad g(10) \approx 1 \end{aligned}$$

现在我们使用了一个小的神经网络构造了计算机的基本操作之一，而不是使用实际的与门。神经网络也可以用来模拟所有其他逻辑门。

示例和直觉 II - Examples and Intuitions II

与，或，非门对应的 $\Theta^{(1)}$ 矩阵:

$$\begin{aligned} & AND: \\ & \Theta^{(1)} = \begin{bmatrix} -30 & 20 & 20 \end{bmatrix} \\ & NOR: \\ & \Theta^{(1)} = \begin{bmatrix} 10 & -20 & -20 \end{bmatrix} \\ & OR: \\ & \Theta^{(1)} = \begin{bmatrix} -10 & 20 & 20 \end{bmatrix} \end{aligned}$$

我们可以组合这些运算，来获得同或逻辑运算符（如果x_1和x_2都是0或两者都是1，则输出1）。

$$\begin{bmatrix} x0 \\ x1 \\ x2 \end{bmatrix} \rightarrow \begin{bmatrix} a_1^{(2)} \\ a_2^{(2)} \end{bmatrix} \rightarrow \begin{bmatrix} a^{(3)} \end{bmatrix} \rightarrow h\Theta(x)$$

对于第一层和第二层之间的过渡，$\Theta^{(1)}$矩阵包含AND和NOR的运算：

$$\Theta^{(1)} = \begin{bmatrix} -30 & 20 & 20 \\ 10 & -20 & -20 \end{bmatrix}$$

对于第二层和第三层之间的过渡，$\Theta^{(2)}$矩阵使用OR运算：

$$\Theta^{(1)} = \begin{bmatrix} -10 & 20 & 20 \end{bmatrix}$$

写出所有节点的值：

$$\begin{aligned} a^2 &= g(\Theta^{(1)} \cdot x) \\ a^3 &= g(\Theta^{(2)} \cdot a^{(2)}) \\ h_\Theta(x) &= a^{(3)} \end{aligned}$$

在这里我们用两个隐藏层实现了XNOR操作符。

多分类问题 - Multiclass Classification

为了把数据分成多个类，可以让假设函数最终返回一个向量，而不是一个数。假设我们想把数据归类为四个类：

$$\begin{bmatrix} x0 \\ x1 \\ x2 \\ \cdots \\ x_n \end{bmatrix} \rightarrow \begin{bmatrix} a_0^{(2)} \\ a_1^{(2)} \\ a_2^{(2)} \\ \cdots \end{bmatrix} \rightarrow \begin{bmatrix} a_0^{(3)} \\ a_1^{(3)} \\ a_2^{(3)} \\ \cdots \end{bmatrix} \rightarrow \cdots \rightarrow \begin{bmatrix} h_\Theta(x)_1 \\ h_\Theta(x)_2 \\ h_\Theta(x)_3 \\ h_\Theta(x)_4 \end{bmatrix} \rightarrow$$

在计算最后一层节点，就是假设值的向量。

最终输出的假设值看起来就像这样：

$$h_\Theta(x) = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

在这种情况下，我们得到的类别是第三个类，或者表示为$h_\Theta(x)_3$。

将所有结果的集合定义为y：

$$y^{(i)} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix},$$

一个假设值一定是y中元素之一。