## 反向传播的框架

* 神经网络有m层，第n层的神经元数目为N(n)
* 第n层的输入为：偏置项（1）+第n-1层的输出，是一个N(n-1)+1维的列向量Xn
* 第n层的权值矩阵为Wn，每个元素wnij（j>=1）表示第n层的i号单元和第n-i层j号单元的关联权值，wni0是n层i单元的偏置
* Wn是 N(n)\*(N(n-1)+1)维的矩阵
* 第n层的输出Yn =f(Wn \* Xn)是个N(n)维的行向量，yni是n层第i个神经元的输出，f([x])是激活函数，作用于矩阵[x]的每一个元素
* 在前向传播过程中，第n层会缓存Yn对Wn的导数矩阵DYWn，i行j列元素为，DYWn是 N(n)\*(N(n-1)+1)维的矩阵
* 在前像传播过程中，第n层会缓存Yn对Y(n-1)的导数矩阵DYXn，i行j列元素为 ，DYXn是 N(n)\*N(n-1)维的矩阵

下面的表格表示了一个神经网络，各层之间全连接：

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 1 | 1 | 1 |
| 2 | 2 | 2 | 2 |
| 3 |  | 3 |  |
| L0（输入） | L1（2个神经元） | L2（3个神经元） | L3（2个神经元） |

**L1层：**

权值矩阵W1：

输入列向量X1：

输出列向量Y1：

Y1对于W1的导数矩阵DYW1：

Y1对于X1的导数矩阵DYX1：

**L2层：**

权值矩阵W2：

输入列向量X2：

输出列向量Y2：

Y2对于W2的导数矩阵DYW2：

Y2对于X2的导数矩阵DYX2：

**L3层：**

权值矩阵W3：

输入列向量X3：

输出列向量Y3：

Y3对于W3的导数矩阵DYW3：

Y3对于X3的导数矩阵DYX3：

**反向传播示例：**

最终输出结果的损失函数F(Y3)，（不含正则项）

F(Y3)关于L3层的各个神经元的偏导向量DFY：

F关于第3层的权值的偏导矩阵DFW3：

DFW3=【DFY】\*DYW3

F关于第2层的权值的偏导矩阵DFW2：

DFW2=【DFY\*DYX3】\*DYW2

由于输入向量X中去除第一项1后就是前面一层的输出向量，所以上面的矩阵可以写为：

F关于第1层的权值的偏导矩阵DFW1：

DFW1=【DFY\* DYX3 \* DYX2】\*DYW1

(注：【】符号表示将向量扩展为以该向量为对角线的方阵)

**反向传播的流程：**

循环n次：

对于有m个样本的样本集中的每个样本i：

1 正向传播计算出该样本的损失函数Fi

2 对Fi使用梯度下降法迭代一步，计算出每一层对于样本i而言的权值修正矩阵dWi

对每一层的权值矩阵W进行修正 W=W-(dWi/m)\*a，a是学习速率

能够使用以上反向传播流程的原因是：所有样本的损失函数F

由于

所以，F对各层权值的偏导向量 = sum(Fi对各层权值的偏导向量) / m

如果F中加入了正则项（二阶正则），正则项会让W朝着每个分量减少的方向移动

修改方式就是在反向传播的最后一行，改为：

对每一层的权值矩阵W进行修正 W=W-(dWi/m +λW)\*a，a是学习速率