Tensorflow 张量 数据流图

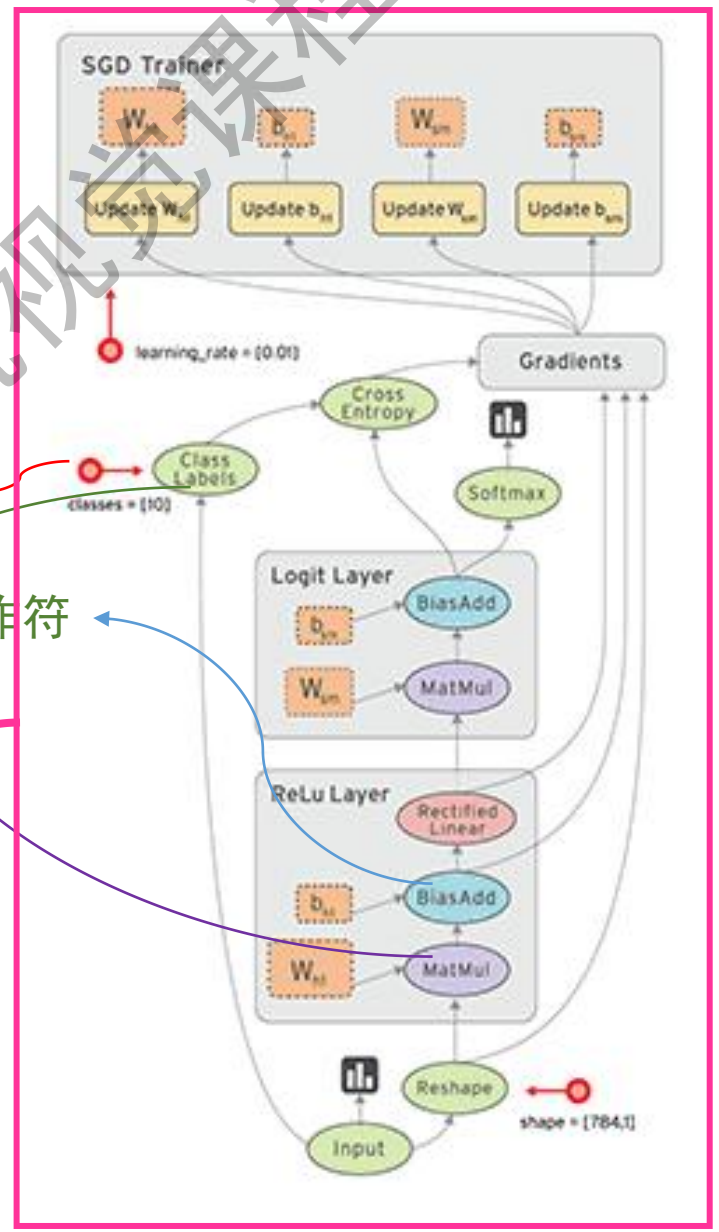
Tensor：在TensorFlow程序中所有的数据都通过张量的形式来表示。从功能的角度看，张量可以被理解为多维数组。其中零阶张量表示标量（scalar）也就是一个数；n阶张量可以理解为一个n维数组。

tf.placeholder()：允许定义一种必须提供值的tensor，也可以随意限定它们的shape。

Operation：计算节点，接受一个或者零个Tensor对象作为输入，然后产生0个或者多个Tensoflow对象作为输出

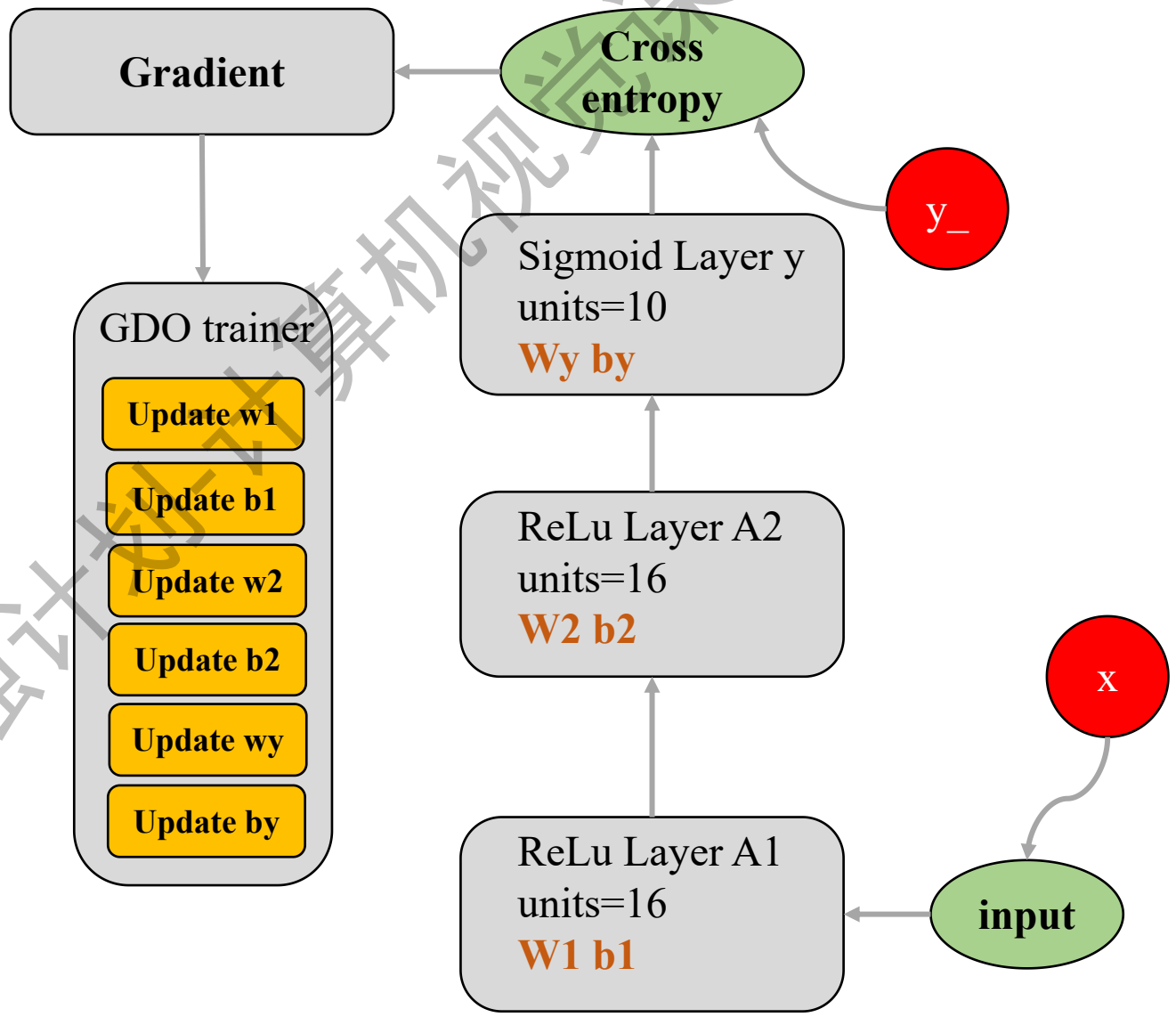
Graph：图包含一些操作对象(operation)，tensor对象是在不同操作之间的数据节点

Session：会话提供了operation执行和Tensor求值的环境，可以拥有一些variable和queue的资源，初始化函数为：
tf.Session.__init__(target='', graph=None, config=None)
如果在创建Session时，没有指定graph,加载默认graph.

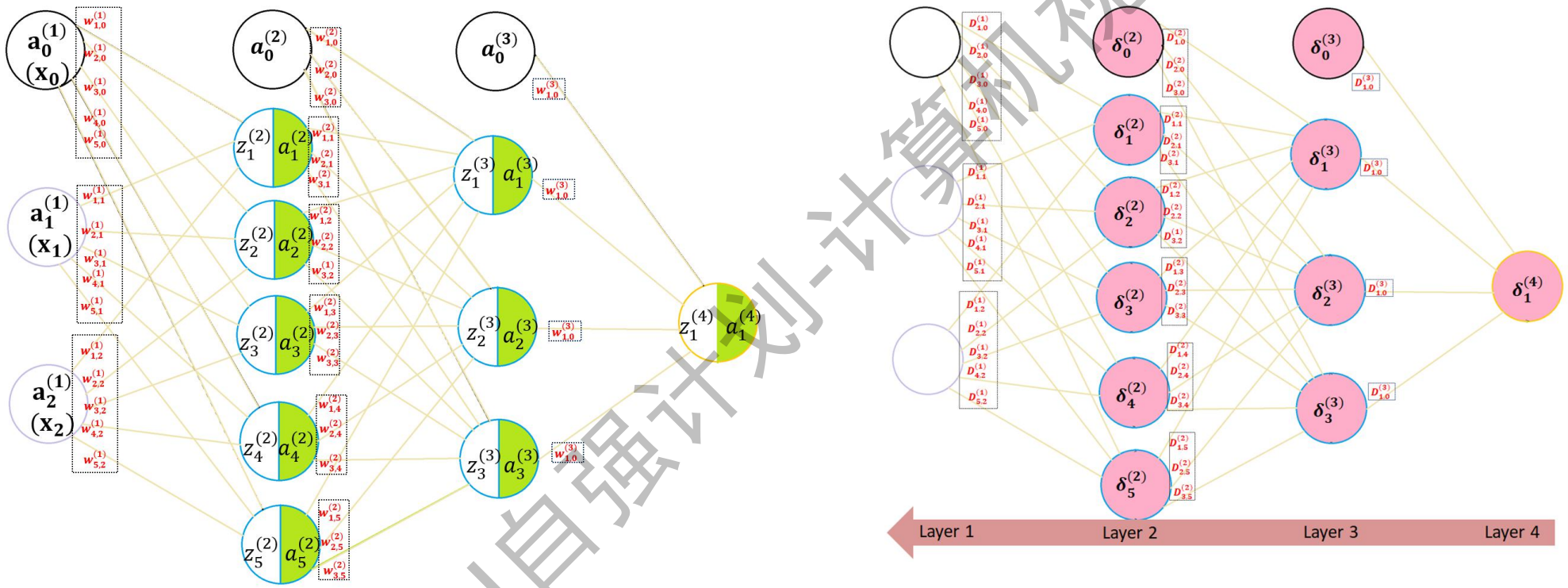


3章 Part3：MINST数据集

1. 添加x、y_占位符，用于输入数据
2. 添加 A1、A2、y 三层全连接层，分别采用 relu、relu 和 sigmoid函数作为激活函数
3. 添加交叉熵计算节点
4. 利用GradientDescentOptimizer方法训练，更新参数



3章 Part2：用numpy的函数写出前馈传播和反向传播算法



反向传播示意图

a2 = np.transpose(np.column_stack((np.ones((a2.shape[1], 1)), a2.T)))

delta3 = np.dot(W3.T[1:,:],delta4) * sigmoidGradient(z3)

3章 Part1：用线性模型拟合MINST模型

交叉熵：是信息论的概念，主要用于度量两个概率分布间的差异性信息，可在神经网络(机器学习)中作为损失函数，p表示真实标记的分布，q则为训练后的模型的预测标记分布，交叉熵损失函数可以衡量p与q的相似性。其公式为：

离散变量：$H(p,q)=\sum_x p(x)\cdot log\left(\frac{1}{q(x)}\right)$

连续变量：$-\int_X P(x)logQ(x)dr(x)=E_p[-logQ]$

正则化：作用是降低模型复杂度，防止过拟合，使用方式是将正则化的结果作为Loss函数的一部分，优化loss函数的w和b，可以优化其绝对值（L1正则）或者平方和（L2正则）

$$L(a)=||a||=\sum_{i=0}^{n}|a_i|$$

L1正则化

$$L(a)=\frac{||a||^2}{2}=\frac{\sum_{i=0}^{n}a_i{}^2}{2}$$

L2正则化

作业需要采用如下增加了正则的交叉熵公式：

cross_entropy = -(1.0/batchSize)*tf.reduce_sum(y_*tf.log(y)+0.1*(1.0-y_)*tf.log(1.0-y)+tf.contrib.layers.l2_regularizer(0.1)(W))

3章 Part1：用线性模型拟合MINST模型

正则之后：

	0	1	2	3	4	5	6	7	8	9	10
0	0	3.30007	2.53574	4.49468	3.94753	4.95892	3.94799	4.49381	3.29901	4.49607	3.94696
1	0	3.94663	3.26999	3.97795	3.93008	3.28825	3.94151	3.31318	4.49952	3.96517	4.41798
2	0	2.43415	3.00792	4.64073	2.29591	3.79572	5.20512	4.36476	3.14085	3.26543	3.63555
3	0	1.4863	-1.72338	0.763146	-2.34519	0.262481	-0.641292	1.17814	1.82144	0.782304	-1.09657
4	0	-37.5968	-64.1277	-50.1456	-87.9657	-36.8166	-45.9315	-41.2627	-39.5851	-32.4876	-75.2419
5	0	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan

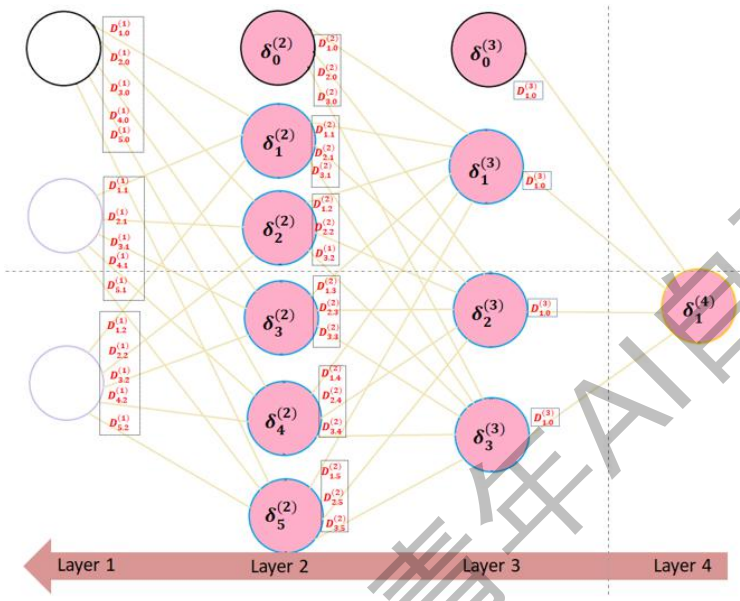
没有正则：

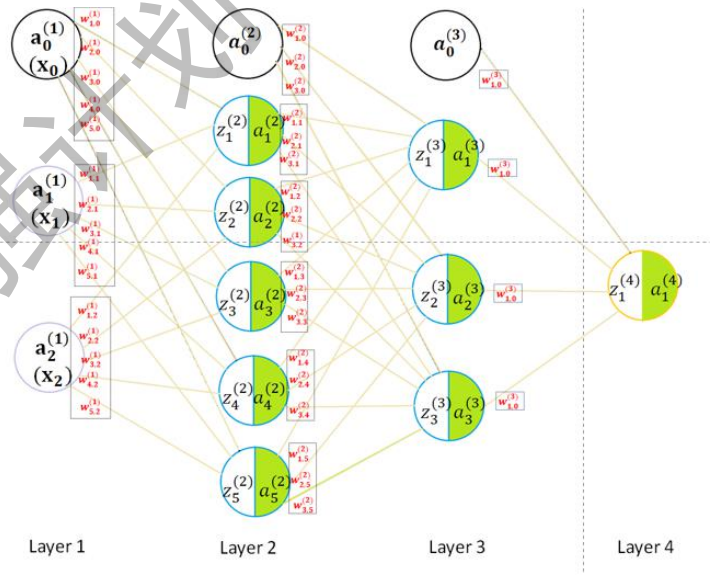
3.94699	4.50018	3.9472	4.95223	2.53723	4.95454	2.5307	5.65276	4.50074	5.81161
4.464	3.98315	4.92368	4.44322	3.32249	1.61288	2.50531	3.94801	4.44183	5.05589
3.29761	1.96505	4.4579	4.92565	4.93477	3.36663	4.05759	2.8982	2.83125	4.44538
3.48942	5.01197	4.45282	3.94805	3.33468	2.56865	4.70169	3.94685	3.26872	2.93039
2.32385	5.18516	1.63427	5.82747	3.26992	5.25067	5.34488	3.96029	3.95605	4.04115
5.12648	5.10888	4.49171	3.33233	3.95056	4.91073	1.83962	3.37033	1.98435	5.03128
4.83426	3.30204	2.6314	5.76654	2.75404	2.61037	4.44378	3.94917	3.96939	3.96209
2.60538	4.45925	3.28107	2.4757	3.32995	3.94628	5.55118	2.54786	3.99344	4.44907
4.51777	3.94926	3.97105	4.49385	4.60584	3.26844	3.28391	2.86081	4.52527	3.3388
2.59618	4.00794	3.96356	3.95275	4.01382	3.29949	5.49043	4.95884	4.01867	5.23891
4.43859	5.03802	2.8849	4.61965	2.64303	4.44391	3.95966	1.60682	4.44262	3.95288
3.28756	4.00727	4.88847	3.26617	2.61479	3.41354	1.85923	3.94544	3.37109	3.9573
4.46077	5.37102	3.29359	3.30571	3.36874	2.55626	3.27099	4.46466	4.93644	3.27757
2.72318	4.45562	3.33613	3.95898	3.97405	3.27641	5.28359	3.31224	5.16644	2.68253
3.96537	2.58958	4.04365	3.952	3.28164	5.69565	5.03527	3.27566	3.30123	2.56204
3.28297	4.86587	5.25633	3.39319	5.12057	4.48939	4.05339	4.48393	5.44376	3.32916
3.27533	4.55582	4.43808	3.34328	1.91384	2.76387	4.45595	4.63157	5.34892	4.43791
2.62105	3.95572	4.44436	4.44901	4.47214	2.86771	3.58671	4.94552	4.05062	1.91372
4.47446	4.43542	4.46627	2.59526	4.02118	4.59022	3.48502	4.43778	2.97088	3.94742

3章 Part2：矩阵相乘的问题

```
# Back propagation
    y_m = np.transpose(np.reshape(y, [-1, 1])) #reshape y_m from (n,)to (1,n)
    delta4 ={0}  # 计算delta4，将预测标签向量a4与y_m做差
    delta3 = {0} # 计算delta3，参数矩阵W3转置后与delta4做矩阵乘法，然后与sigmoidGradient(z3)对应位相乘
    delta2 = {0}  # 计算delta2，参数矩阵W2转置后与delta3做矩阵乘法，然后与sigmoidGradient(z2)对应位相乘
```

delta2 = np.dot(W2.T[1:,:],delta3) * sigmoidGradient(z2)





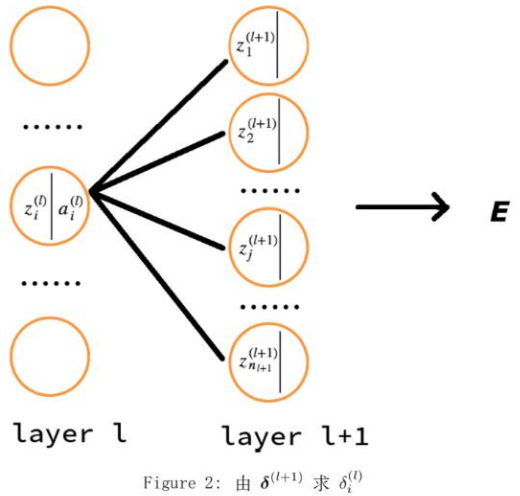


Figure 2: 由 $\delta^{(l+1)}$ 求 $\delta_i^{(l)}$

4章 Part1与Part3

Part1：浅层神经网络	30min	Part1 Part3之间可以互相参考
Part3：采用了DNN	3h左右	Part2 本身之间就可以参考

训练时间差别很大