1 softmax，交叉熵

$$Softmax回归 : softmax(y)_i = e^{yi} / \sum_{j=1}^{n} e^{yi}$$

Softmax 层将神经网络的输出变成一个概率分布，从而可以通过交叉熵来计算预测的概率分布和真实答案的概率分布之间的距离了。

$$H(p,q) = \sum_{x} p(x) log(q(x))$$

交叉熵刻画的是两个概率分布之间的距离，他是不对称的。

$$H(p,q) \neq H(q,p)$$

他刻画的是通过概率分布 q 来表达概率分布 p 的困难程度。因为正确答案是希望得到的结果，所以当交叉熵作为神经网络的损失函数的时候，p 代表的是正确答案，q 代表的师预测值。交叉熵刻画的是两个概率分布的距离，也就是说交叉熵越小，两个概率分布越接近。

2 L1,L2 正则化

L1:Lasso , L2:Ridge

$$L1 : R(w) = ||w||_1 = \sum_i |w_i|$$

$$L2 : R(w) = ||w||_2 = \sum_i |w_i^2|$$

L1: 让参数更稀疏，L2 则不会。因为当参数很小时，比如 0.001，这个参数的平方基本上就可以忽略了，于是模型不会进一步减个这个参数调整为 0。

L1 不可导而 L2 可导。因为在优化的时候需要计算损失函数的偏导数，所以对 L2 正则化的损失函数的优化更加简洁，优化 L1 正则化的损失函数更加复杂。

实践中也可将 L1 和 L2 同时使用:

$$R(w) = \sum_i \alpha|w_i| + (1-\alpha)w_i^2$$

3 神经网络优化过程中可能遇到的问题

1. 通过指数衰减的方式来设置学习率。通过这种方法，既可以加快训练初期的训练速度，同时在训练后期又不会出现损失函数在极小值附近徘徊往返的情况。
2. 通过正则化解决过拟合的问题。
3. 使用滑动平均模型让最后得到的模型在未知数据上更健壮了。