**第一章——深度学习的实践领域**

**训练集/开发集/测试集**

划分比例：当数据量较小时6:2:2，当数据量很大(100w)时开发集和测试集的占比减小，可达98:1:1或更少

应确保开发集和测试集中的数据分布相同

**偏差与方差**

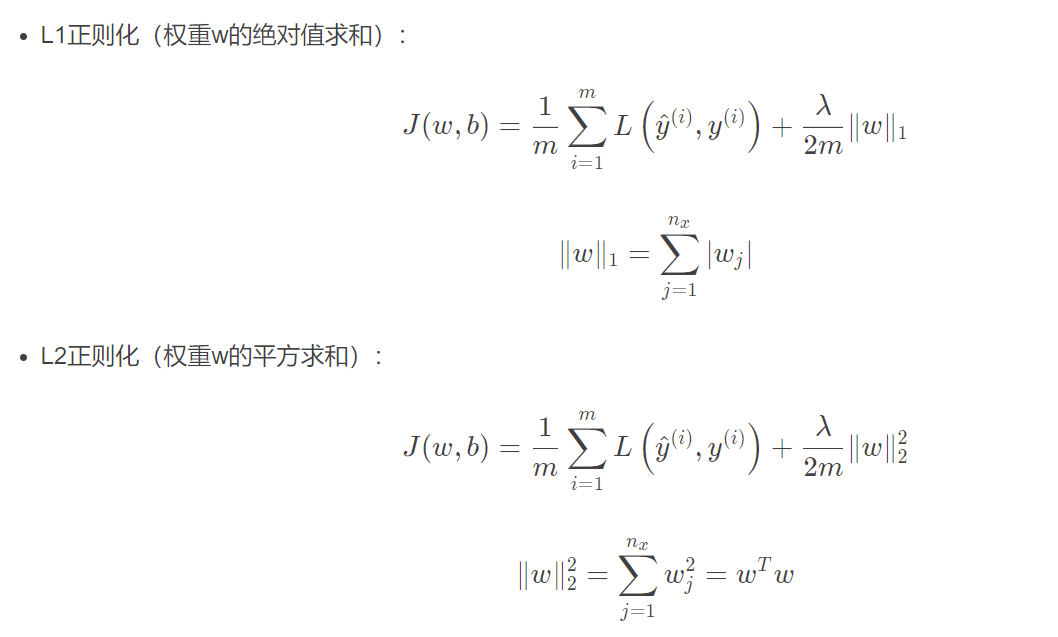
高方差：训练集错误率很低，开发集错误率较高——>获取更多数据/正则化/选用另一种结构

高偏差：训练集和开发集错误率都较高——>挑选更大的神经网络/延长训练时间/选用另一种结构

高方差且高偏差：训练集错误率较高，开发集更糟

**正则化——用以解决过拟合问题和加速训练**

在成本函数中加入一个正则化项，最常见的是L2正则化



：正则化参数（超参数）

**正则化为什么可以解决过拟合问题**：设置一个较大的值会使得权重w变得很小，近乎消除其影响，使神经网络接近高偏差状态。即将复杂的神经网络简单化了。当的值合适时，就能得到理想的中间状态

**Dropout正则化（随机失活）**

**最常用方法：反向随机失活**

为了保证下一层的z值期望不变，要对与dropout矩阵乘积后的权重进行扩大，即除以留存率。

Dropout只在训练阶段使用

**Dropout为什么有用**：简化神经网络，并使神经元对每个输入都给一个比较小的权重而不是依赖于单个输入；越容易出现过拟合的层，其留存率就应当设置得越小

**其他减少过拟合的方法**

增加训练样本数，难以收集新样本时可对现有样本进行处理，如水平翻转图像、随机裁剪等；早终止法：选择合适的迭代次数，较早停止迭代

**归一化输入**

**标准化**：减去均值，除以标准差

对训练集和测试集应使用同样的均值和标准差；如果输入特征本身范围相近，不进行标准化操作也没有太大影响

**梯度消失与梯度爆炸**

网络层数太深而引发的梯度反向传播中的连乘效应

**解决方法之一——更好的权重初始化**

假设神经网络第l层的输入有n[l-1]个

初始化的思路是让w与n[l-1]有关，后者越大，w应当越小

激活函数是tanh时：w[l] = np.random.randn(n[l],n[l-1])\*np.sqrt(1/n[l-1])

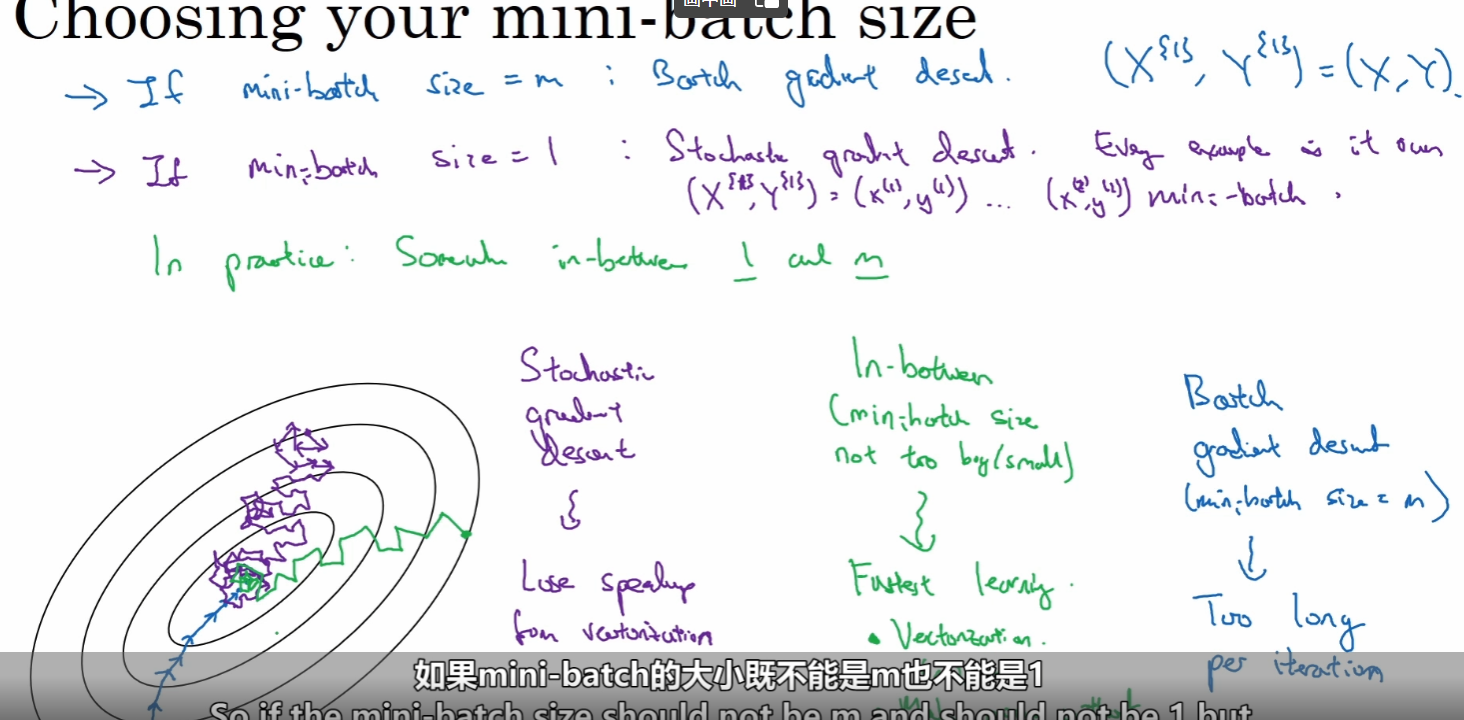
激活函数是ReLU时：w[l] = np.random.randn(n[l],n[l-1])\*np.sqrt(2/n[l-1])

**第二章——优化算法**

**小批量梯度下降**

当m很大时，将其分为若干个子集进行训练，每次训练都包含前向、反向传播过程。当对所有样本进行一轮训练后，称为一个epoch

Batch size大小的选择与影响：



Batch size选用2的幂，应保证cpu/gpu内存足够

**指数加权平均**

优点：计算当前时刻的平均值时只需要前一天的平均值和当前时刻的值。占用计算机资源储存和内存少，只需一行代码，就可近似得到平均值。

常用0.9

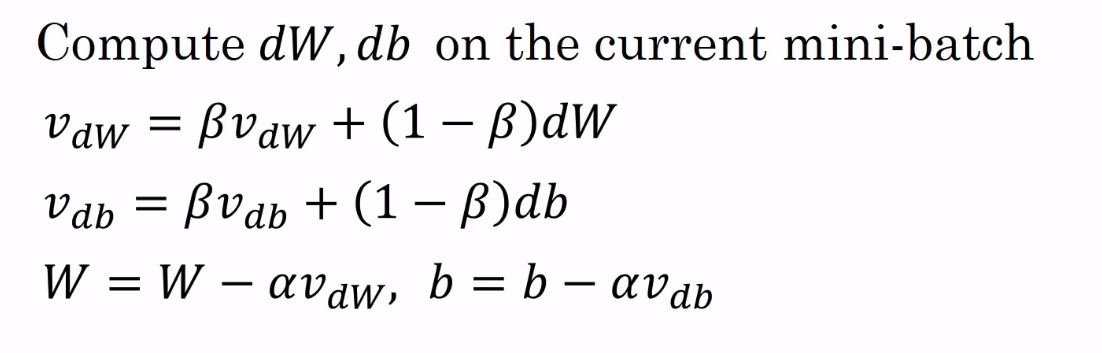
**指数加权平均的偏差修正**

在初始阶段，偏差修正有益于更早做出更好的估计。但在实际机器学习中并不必要，随着迭代次数增加，初始阶段的影响将消失，不再需要偏差修正

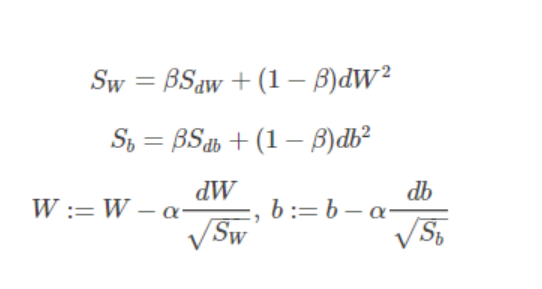
**动量梯度下降**

计算梯度的指数加权平均，然后使用这个梯度来更新权重。比常规梯度下降算法更快

原理：减弱了前往最小值路径上的震荡

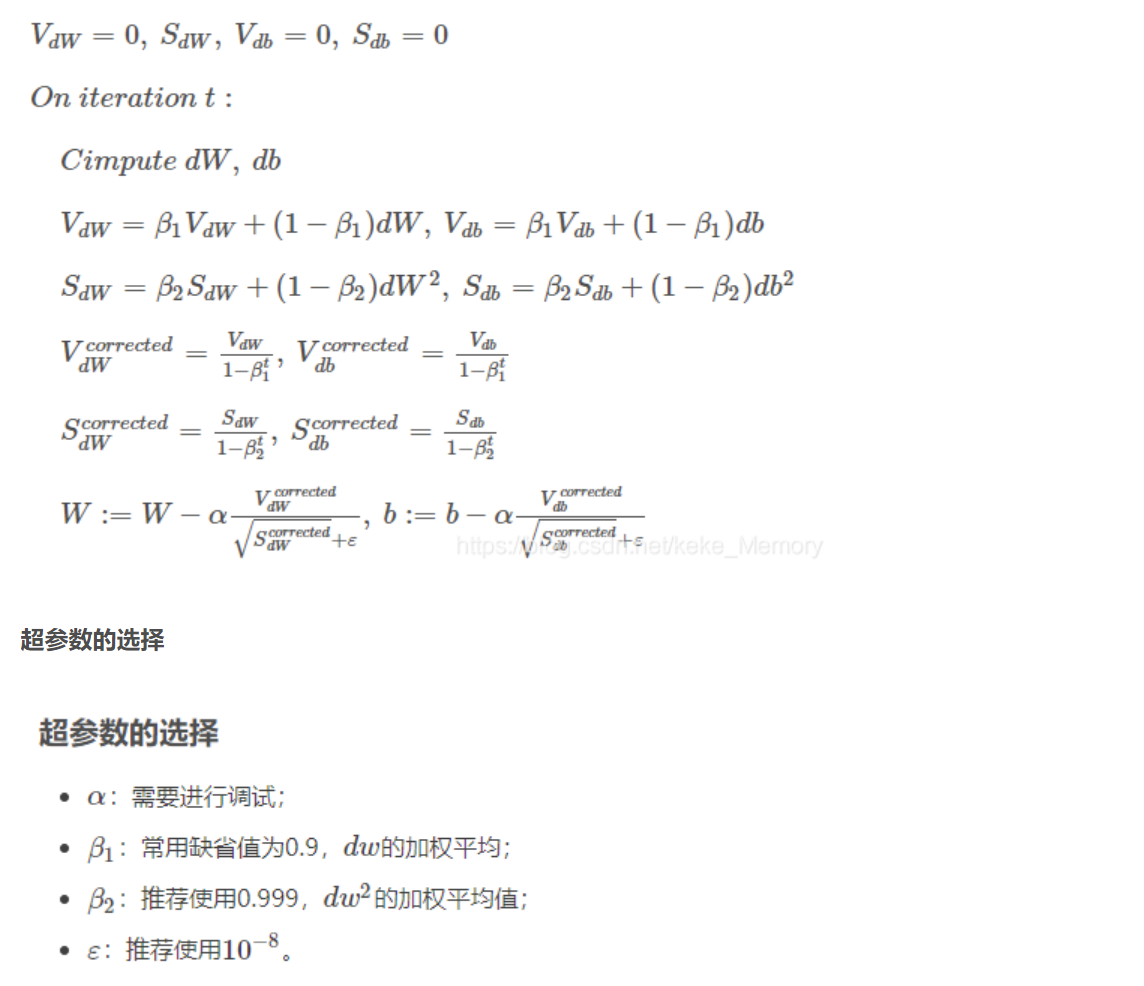


**RMSprop均方根传递——也可以加速梯度下降并减小震荡**



哪个方向的震荡大，就减小该方向的更新速度，从而减小震荡。实际中需要给分母加一个非常小的值以轻微提高数值稳定性

**适应性矩估计优化算法(Adam)——将动量梯度下降与RMSprop结合**



**学习率衰减**

随着迭代次数增加逐渐减小学习率

一般使用：

指数衰减：

**第三章——超参数调整，批量归一化和编程框架**

**超参数调整**

重要的超参数：

其次：

再次：层数，学习率衰减

超参数调整过程：随机抽样+区域定位

使用适当的标准来选择超参数：使用对数尺度代替线性尺度，如将0.0001~1划分为0.0001~0.001，0.001~0.01，0.01~0.1，0.1~1，再在这四个范围内均匀随机取值

实践中的超参数调整：（熊猫模式）精心照料一个模型，不断调整超参数；（鱼子酱模式）或设置多个超参数不同的模型同时训练，最终选取结果最好的那一个

**批量归一化——在隐藏层上归一化**

全为平均值为0方差为1的分布

不希望隐藏层分布都相同，故形成不同的分布，是可以学习更新的参数，通过它们来控制隐藏单元有可控的方差和均值

在实际的神经网络中，在每个隐藏层都用BN将。使用反向传播计算各个参数的梯度然后更新参数

**BN为什么有效？**

第一，BN对隐藏层输入进行了规范化处理，从而加快了学习速度；第二，使得前面w的变化对后面的w造成的影响较小，网络更加稳健；第三，BN层通过为隐藏层的激活函数添加噪声而略微具有正则化的作用，类似于随机失活

**测试时的BN**

在训练过程中，对每个mini-batch使用指数加权平均，最终得到指数加权平均后的均值与方差，用于在测试时计算BN的公式

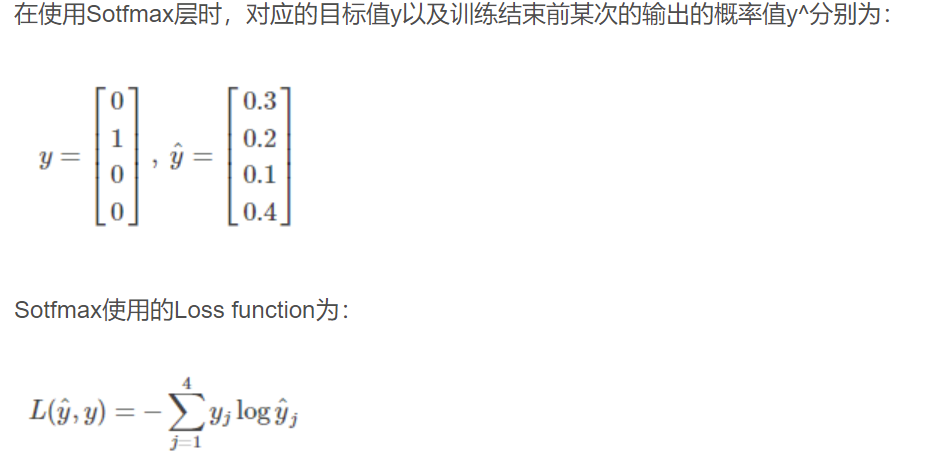
**Softmax**

将多分类任务的输出转换为各个类别可能的概率

输出层激活函数：

C为类别数

**训练softmax分类器**



对整个训练集：

