随机梯度下降法的缺点

对于某些特别的函数（非均向，各位置梯度方向几乎不指向最低谷），SGD的优化路径会呈现为之字形，效率很差

SGD的改进算法

SGD with Momentum - 利用积累和抵消效应

$$\boldsymbol{v} \leftarrow \alpha \boldsymbol{v} - \eta \frac{\partial L}{\partial \boldsymbol{W}}$$

$$\boldsymbol{W} \leftarrow \boldsymbol{W} + \boldsymbol{v}$$

AdaGrad - 动态学习率，先迈大步，后迈小步

$$\boldsymbol{h} \leftarrow \boldsymbol{h} + \frac{\partial L}{\partial \boldsymbol{W}} \odot \frac{\partial L}{\partial \boldsymbol{W}}$$

$$\boldsymbol{W} \leftarrow \boldsymbol{W} - \eta \frac{1}{\sqrt{\boldsymbol{h}}} \frac{\partial L}{\partial \boldsymbol{W}}$$

AdaGrad的学习终止问题：可以看到，h是随着学习不断累积的，如果学习轮数很多，后面的梯度更新量几乎是0。解决方法是RMSProp，它会逐渐遗忘过去的梯度。

Adam - 综合了Momentum和AdaGrad，具体原理较复杂，此处略

权重的初始值问题

不允许

不可以将权重的初始值都设为0，也不可以全部设为相同的值 —— 禁止**权重均一化**

$W_0 = 0.01 * randn$

梯度消失问题

称某一层的激活函数输出值为该层的**激活值**，若激活值大量偏向于0和1，Sigmoid等函数在这些位置梯度几乎为0，难以走出，学习没有效果了

如果权重的初始值选取不当，就很可能造成梯度消失

表现力受限问题

如果激活值分布相当集中，则该层大量神经元几乎在做同样的事情，就没有多个神经元的意义了。学习会很慢，甚至完全停滞。

tanh激活函数

是一个更优的激活函数，它严格关于(0, 0)对称（Sigmoid是(0, 0.5)），可以更好地改善各层的激活值形状

Xavier初始值理论

对于在中央几乎是线性的激活函数（Sigmoid、tanh），使用下列权重初始值可以获得很好的效果

与前一层有n个节点连接时，初始值使用标准差为$\frac{1}{\sqrt{n}}$的分布

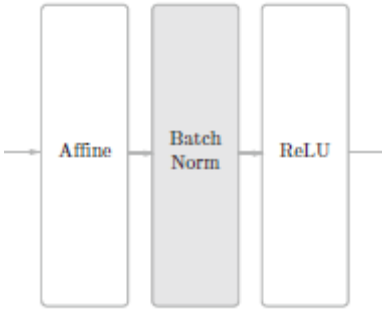
He初始值理论

对于ReLU激活函数

当前一层的节点数为n时，He初始值使用标准差为$\sqrt{\frac{2}{n}}$的高斯分布。当Xavier初始值是$\sqrt{\frac{1}{n}}$时，(直观上)可以解释为，因为ReLU的负值区域的值为0，为了使它更有广度，所以需要2倍的系数。

Batch-Norm 批标准化

原理 - 调整各层的激活值分布



优点 - 加速学习；对初始值有鲁棒性；抑制过拟合

Batch Norm，顾名思义，以进行学习时的mini-batch为单位，按mini-batch进行正规化。具体而言，就是进行使数据分布的均值为0、方差为1的正规化。用数学式表示的话，如下所示。

$$\begin{aligned}\mu_B &\leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \\ \sigma_B^2 &\leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i-\mu_B)^2 \\ \hat{x}_i &\leftarrow \frac{x_i-\mu_B}{\sqrt{\sigma_B^2+\varepsilon}}\end{aligned} \tag{6.7}$$

接着，Batch Norm层会对正规化后的数据进行缩放和平移的变换，用数学式可以如下表示。

$$y_i \leftarrow \gamma\hat{x}_i + \beta \tag{6.8}$$

这里，γ和β是参数。一开始$\gamma=1$，$\beta=0$，然后再通过学习调整到合适的值。

正则化

过拟合问题 - 只能拟合训练数据，但不能很好地拟合不包含在训练数据中的其他数据

过拟合问题的原因

- 模型的参数很多，表现力强
- 训练数据太少

过拟合问题的解决

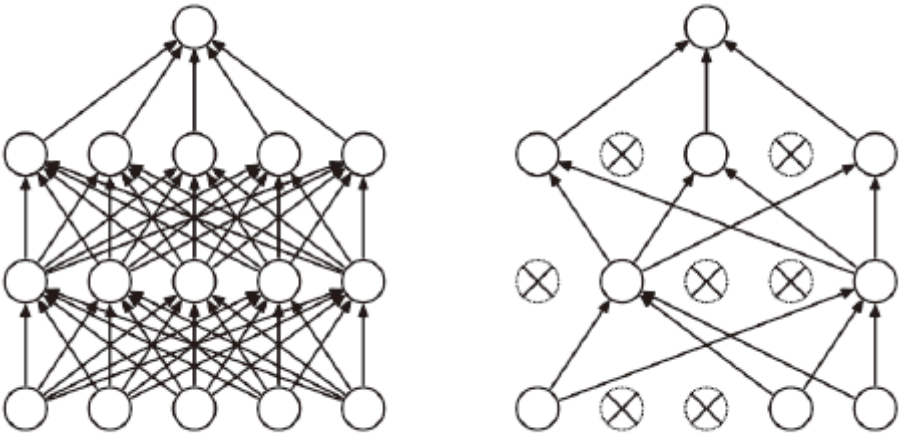
- 方法1 - 权值衰减（不是学习率衰减）

对过大的权重进行惩罚，具体做法是为损失函数加上权重的平方范数（L2范数，各元素的平方和开根号）（也可以用L1范数即绝对值和、L∞范数即最大值）

$Loss = Loss_0 + 0.5\lambda W^2$，λ是一个调节惩罚强度的超参数

- 方法2 - Dropout

在层与层之间的传播过程中随机删除一些神经元（引入dropout_ratio超参数）



(a) Standard Neural Net　　(b) After applying dropout.

集成学习 - 让多个模型单独进行学习，推理时再取多个模型的输出的平均值

集成学习与Dropout的内在联系 - 可以将Dropout理解为，通过在学习过程中随机删除神经元，从而每一次都让不同的模型进行学习

超参数的优化

调整超参数时，必须使用超参数专用的**验证数据**，可以从训练集中提取，不能用测试集中的数据。

优化超参数，就是一个不断试错的过程。先大步，后小步，缩小区间到合适的超参数。