一、参数初始化

下面几种方式随便选一个，结果基本差不多，但是一定要做。否则可能会减慢收敛速度，影响收敛结果，甚至造成nan等一系列问题。

下面的n_in为网络的输入大小，n_out为网络的输出大小，n为n_in或(n_in+n_out)*0.5

Xavier初始法论文：

http://jmlr.org/proceedings/papers/v9/glorot10a/glorot10a.pdf

He初始化论文：https://arxiv.org/abs/1502.01852

uniform均匀分布初始化：w = np.random.uniform(low=-scale, high=scale, size=[n_in,n_out])

Xavier初始法，适用于普通激活函数(tanh,sigmoid)：scale = np.sqrt(3/n)

He初始化，适用于ReLU：scale = np.sqrt(6/n)

normal高斯分布初始化：w = np.random.randn(n_in,n_out) * stdev # stdev为高斯分布的标准差，均值设为0

Xavier初始法，适用于普通激活函数 (tanh,sigmoid)：stdev = np.sqrt(n)

He初始化，适用于ReLU：stdev = np.sqrt(2/n)

svd初始化：对RNN有比较好的效果。参考论文：

https://arxiv.org/abs/1312.6120

二、数据预处理方式

zero-center，这个挺常用的。

```
X -= np.mean(X, axis = 0)
X /= np.std(X, axis = 0)    # normalize
```

三、训练技巧

要做梯度归一化，即算出来的梯度除以minibatch size。

clip c(梯度裁剪）：限制最大梯度，其实是value=sqrt(w1^2+w2^2….),如果value超过了阈值，就算一个衰减系数，让value的值等于5,10,15

dropout对小数据防止过拟合有很好的效果，值一般设为0.5，小数据上dropout+sgd在我的大部分实验中，效果提升都非常明显。

adam,adadelta等,在小数据上,我这里实验的效果不如sgd, sgd收敛速度会慢一些，但是最终收敛后的结果，一般都比较好。如果使用sgd的话,可以选择从1.0或者0.1的学习率开始,隔一段时间,在验证集上检查一下,如果cost没有下降,就对学习率减半. 我看过很多论文都这么搞,我自己实验的结果也很好. 当然,也可以先用ada系列先跑,最后快收敛的时候,更换成sgd继续训练.同样也会有提升.据说adadelta一般在分类问题上效果比较好，adam在生成问题上效果比较好。

四、尽量对数据做shuffle

Batch Normalization据说可以提升效果，不过我没有尝试过，建议作为最后提升模型的手段，参考论文：Accelerating Deep Network Training by Reducing Internal Covariate Shift

如果你的模型包含全连接层（MLP），并且输入和输出大小一样，可以考虑将MLP替换成Highway Network,我尝试对结果有一点提升，建议作为最后提升模型的手段，原理很简单，就是给输出加了一个gate来控制信息的流动，详细介绍请参考论文:

http://arxiv.org/abs/1505.00387

五、ensemble

Ensemble是论文刷结果的终极核武器，深度学习中一般有以下几种方式：

同样的参数，不同的初始化方式

不同的参数，通过cross-validation选取最好的几组

同样的参数，模型训练的不同阶段，即不同迭代次数的模型
不同的模型，进行线性融合，例如RNN和传统模型。