# 一 调试处理

## 1.1 有哪些超参数?

按照重要程度如下划分:

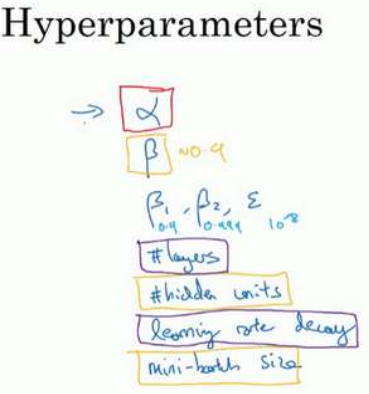
1. 学习率 alpha
2. 优化算法(adam)中的beta.

网络的隐藏层单元数(hidden units)

Minibatch size(可以影响收敛)

1. 神经元层数(深度)

学习率衰减因子(lr decay ratio)

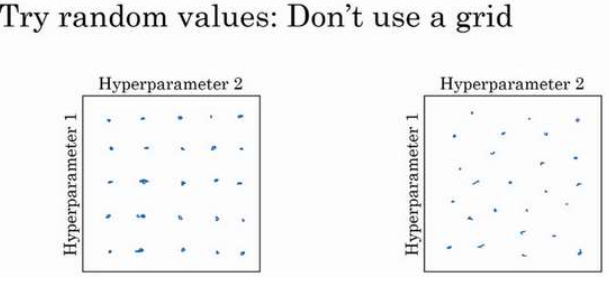


## 1.2 传统参数搜索方法

传统参数搜索方法(机器学习)采用网格搜索.

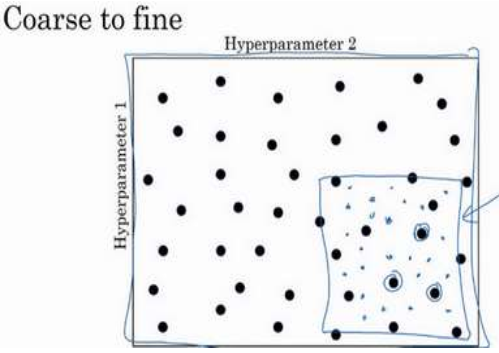
1. 如下左图, 假设需要测试(尝试)25次, 对于有两个超参数的系统. 每个超参数上选5个(5等分,随机). 共有5x5的表现.
2. 如果对于参数1,2 其重要性不是对等的. 比如参数2的重要性更大, 参数1及时选择5个随机值也对情况改善作用很小的情况下. 我们可以给参数2选择25个随机值. 这样表现会更好.

深度学习建议使用随机参数搜索方法. 如图2.



另外,可以采用由粗到细的策略(方法).

1. 找到一组表现好的参数后(下图的右下角).
2. 对这个区域(放大这个区域)做精细的尝试(更密集取值,随机取值).



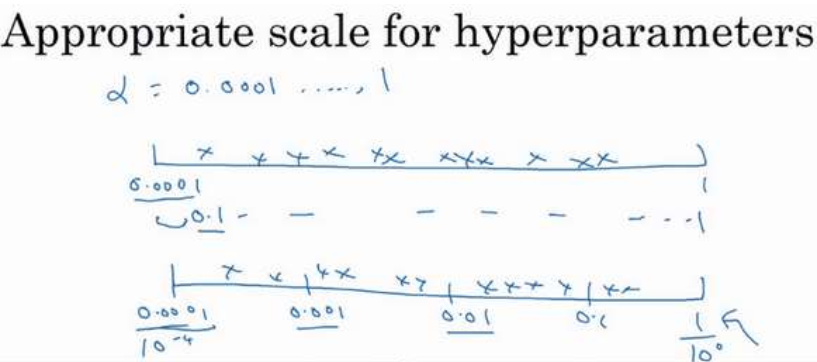
# 二 为超参数选择合适范围

Using an approprate scale to pick hyperparameters

## 2.1 何为超参数合理范围?

种类一) 学习率alpha和adam的beta.

1. 学习率alpha在0.0001~0.1间有很好的表现, [0.0001,0.1]是其**合理**的范围.但是其实际可以取到[0.0001,1]的范围.
2. 如果按照线性区分的话. 参数搜索能落在0.0001~0.1的概率大约只有10%. 落在0.1~1间概率是90%.
3. 引入指数区分. 这样参数搜索落在0.0001~0.1的概率是80%.落在0.1~1间概率是20%.
4. 10为底指数, 随机[-4,0]的整数r. 带入10r.



种类二) adam的beta参数.

1. Beta在0.9~0.999中表现很好.
2. 还是考虑指数随机(10为底的指数).
3. 引入1-beta, 其范围为[0.1,0.001], 翻转为[0.001, 0.1], r的范围是[-3,-1].

# 三 超参数调试实践 Pandas Vs Caviar

调试技巧

1. 不同应用场景的深度学习模型的超参数有借鉴性. 可以相互借鉴, 引发思路.
2. 硬件变化后(比如服务器变了), 数据分布变化后, 都需要重新回归(检验)之前的超参数配置.

主流的调参:

1. Pandas.

对于硬件有限制(GPU有限), 数据量很大. 只能一次尝试一种参数.

可以逐渐改良超参数. 主要观察损失函数的表现. 逐渐添加momentum,或者逐渐减小变量,然后观察其loss表现.

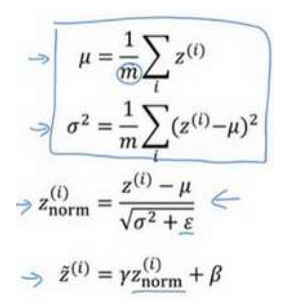
1. Caviar.

当有足够的GPU, 可以一次(并行)尝试多种超参数. 绘制出不同超参数的loss曲线.找到一组最合适的超参数.

# 四 归一化网络激活函数

何为归一化激活函数?

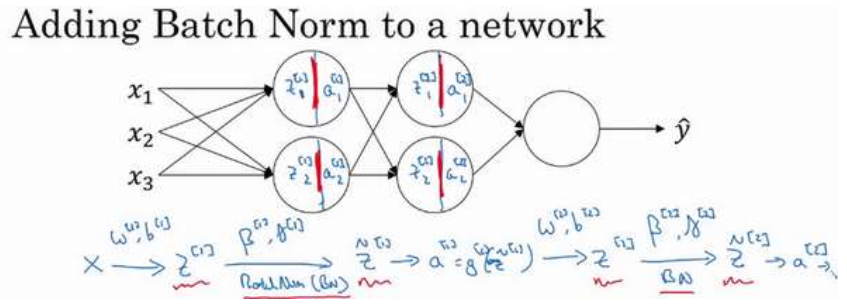
是Batch Norm.



1. 作用域是隐藏层.在激活函数之前.
2. 均值μ和方差σ， 归一化的不是l层的输出a，而是中间变量的z。
3. Z[l]是第l层的中间结果。
4. Znorm是归一化之后的中间结果，znorm是均值为0，方差为1的分布。 这样就损失了不同样本的差异性了。
5. 为了解决不同样本的差异性， 引入γ和β。

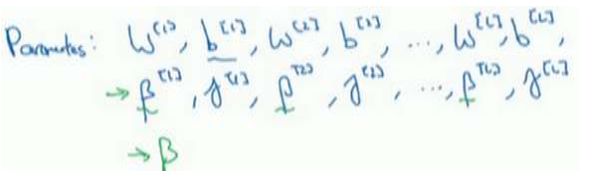
# 五 讲BatchNorm拟合进神经网络

## 5.1 Batchnorm在网络中的用法



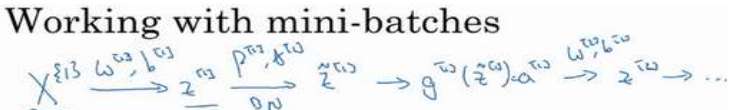
1. Z[l]是第l层的中间输出.
2. a[l]是第l层的输出.
3. g()是激活函数.

## 5.2 引入Batchnorm后的参数更新



1. 网络中需要学习的参数,由之前的W[l],b[l].在引入γ和β后,需要对γ[l],β[l]做更新.
2. 注意BatchNorm的β和adam的β不是一个. (名字相同, 但完全没关系)

## 5.3 minibatch下的BatchNorm



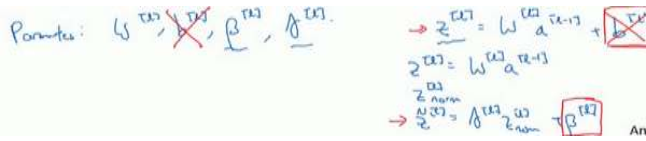
1. X{1}作为第一批的minibatch的样本组.
2. X{1}经W[1],b[1]得到中间变量Z[1]
3. Z[1]经batchnorm得到Z~[1](aka, Znorm[1])
4. Z[1]含有一组值, 计算这组值均值μ.和方差σ.
5. 将z[1]归一化成均值为0, 方差为1数据分布.
6. 然后引入γ和β,对归一化得分布做差异性”映射”.
7. 对z~[l]做激活函数,产生输出α.
8. 对于下一批minibatch同样处理.
9. 对于b[1]而言, 在生成z[1]时,每个样本都会加上b[1].

这些z[1]的均值就都含有b[1]

等到做归一化时候,这个共有的b[1]就会被消除掉.

在引入Bn后网络中的b[1]参数实际上是不需要的.

1. BatchNorm的β就能保证(体现)b[1]的作用.



# 六 为何BatchNorm生效

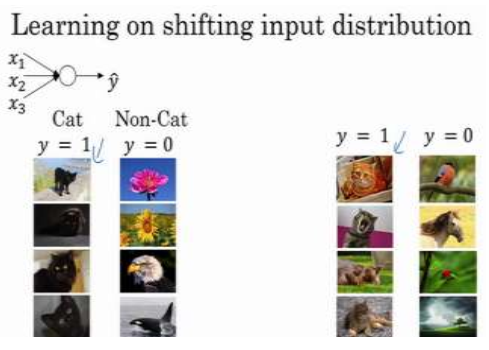
## 原因一) 加速训练

1. Batch norm将featuremap(隐藏层)归一化到均值0,方差1.
2. 能够一定程度上,消除不同特征量程之间的偏差.
3. 能够加速训练.

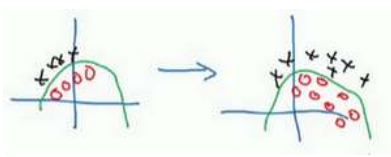
## 原因二) 保持网络输出的稳定

BatchNorm使权重比你的 网络更滞后,更深层.

1. 在左下图上训练好黑猫的分类. 当应用到彩色猫分类问题时. 表现不好.

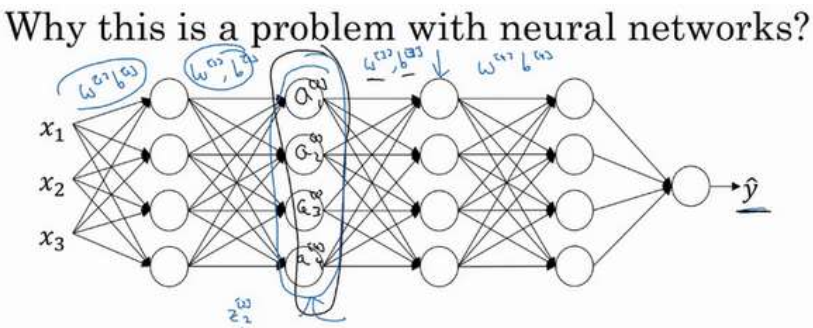


1. 其表现如下.
2. 左下图时黑猫的收敛范围.
3. 右下图时彩色猫的样本空间表现.
4. **应该无法期待在左下图训练好的网络能够在右下图有好的表现.**

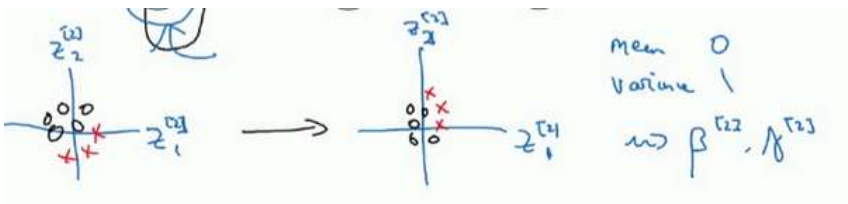


这种体现出一个Covariate shift.

* 1. Covariate shift描述训练集和测试集的分布不同的问题.



1. 考虑中间层a , 其受w[1],b[1]影响.
2. BatchNorm可以减少隐藏值(W[1],b[1])的变化.
   1. 无论W[1],b[1]怎么变化,其中间变量z[1]永远是均值0方差1的分布.



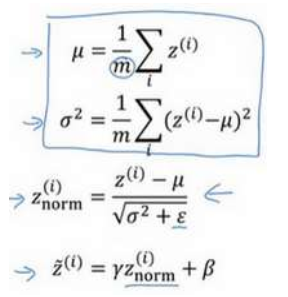
* 1. 基于此, a的值较为稳定.抑制了Covariate shift影响.

## 原因三) 轻微正则化效果

BatchNorm的γ,β的引入会带来一定的噪声.这些噪声起到一定程度(轻微)的正则化.

1. Minibatch太大时, 减少了噪声, BatchNorm的正则化效果被削弱(更加不明显).

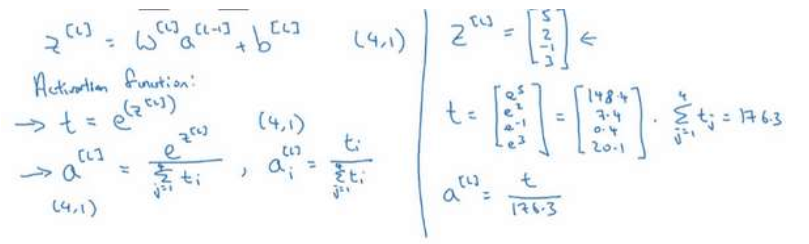
# 七 测试是BatchNorm



1. 超参数μ,σ采用指数加权平均的方法更新.
   * 1. 在训练时, 用指数甲醛平均方法更新.
     2. 在测试时, 会采用训练时候最后保存的那组参数
2. 超参数γ,β.
3. 采用梯度更新的方式(Adam)

# 八 softmax回归

## 8.1 何为softmax



1. Softmax是一种计算概率的方法.
2. 把得分转化成概率的一种方法.
3. 本身并没有损失函数等内容.
4. 损失函数是衡量真值和预测值的差异的函数.