# [机器学习算法中如何选取超参数：学习速率、正则项系数、minibatch size](http://blog.csdn.net/u012162613/article/details/44265967)

分类： [Machine Learning](http://blog.csdn.net/u012162613/article/category/2760005)2015-03-14 23:49 3096人阅读 [评论](http://blog.csdn.net/u012162613/article/details/44265967#comments)(1) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/u012162613/article/details/44265967#report)

[超参数](http://www.csdn.net/tag/%e8%b6%85%e5%8f%82%e6%95%b0)[机器学习](http://www.csdn.net/tag/%e6%9c%ba%e5%99%a8%e5%ad%a6%e4%b9%a0)

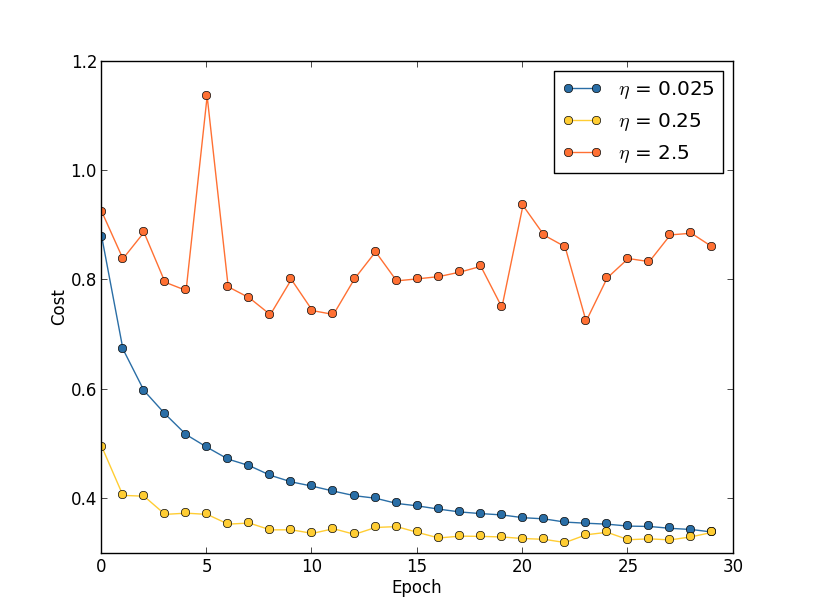
目录[(?)](http://blog.csdn.net/u012162613/article/details/44265967" \o "系统根据文章中H1到H6标签自动生成文章目录)[[+]](http://blog.csdn.net/u012162613/article/details/44265967)

本文是[《Neural networks and deep learning》概览](http://blog.csdn.net/u012162613/article/details/44220115) 中第三章的一部分，讲机器学习算法中，如何选取初始的超参数的值。（本文会不断补充）

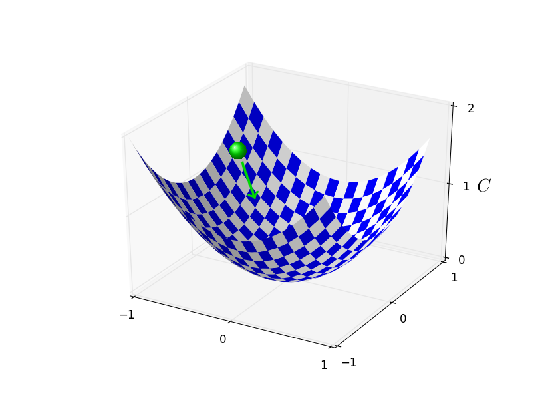
## ****学习速率（learning rate，η）****

运用梯度下降算法进行优化时，权重的更新规则中，在梯度项前会乘以一个系数，这个系数就叫学习速率η。下面讨论在训练时选取η的策略。

* 固定的学习速率。如果学习速率太小，则会使收敛过慢，如果学习速率太大，则会导致代价函数振荡，如下图所示。就下图来说，一个比较好的策略是先将学习速率设置为0.25，然后在训练到第20个Epoch时，学习速率改为0.025。



关于为什么学习速率太大时会振荡，看看这张图就知道了，绿色的球和箭头代表当前所处的位置，以及梯度的方向，学习速率越大，那么往箭头方向前进得越多，如果太大则会导致直接跨过谷底到达另一端，所谓“步子太大，迈过山谷”。



在实践中，怎么粗略地确定一个比较好的学习速率呢？好像也只能通过尝试。你可以先把学习速率设置为0.01，然后观察training cost的走向，如果cost在减小，那你可以逐步地调大学习速率，试试0.1，1.0….如果cost在增大，那就得减小学习速率，试试0.001，0.0001….经过一番尝试之后，你可以大概确定学习速率的合适的值。

为什么是根据training cost来确定学习速率，而不是根据validation accuracy来确定呢？这里直接引用一段话，有兴趣可以看看：

This all seems quite straightforward. However, using the training cost to pick η appears to contradict what I said earlier in this section, namely, that we’d pick hyper-parameters by evaluating performance using our held-out validation data. In fact, we’ll use validation accuracy to pick the regularization hyper-parameter, the mini-batch size, and network parameters such as the number of layers and hidden neurons, and so on. Why do things differently for the learning rate? Frankly, this choice is my personal aesthetic preference, and is perhaps somewhat idiosyncratic. The reasoning is that the other hyper-parameters are intended to improve the final classification accuracy on the test set, and so it makes sense to select them on the basis of validation accuracy. However, the learning rate is only incidentally meant to impact the final classification accuracy. It’s primary purpose is really to control the step size in gradient descent, and monitoring the training cost is the best way to detect if the step size is too big. With that said, this is a personal aesthetic preference. Early on during learning the training cost usually only decreases if the validation accuracy improves, and so in practice it’s unlikely to make much difference which criterion you use.

## ****Early Stopping****

所谓early stopping，即在每一个epoch结束时（一个epoch即对所有训练数据的一轮遍历）计算 validation data的accuracy，当accuracy不再提高时，就停止训练。这是很自然的做法，因为accuracy不再提高了，训练下去也没用。另外，这样做还能防止overfitting。

**那么，怎么样才算是validation accuracy不再提高呢？**并不是说validation accuracy一降下来，它就是“不再提高”，因为可能经过这个epoch后，accuracy降低了，但是随后的epoch又让accuracy升上去了，所以不能根据一两次的连续降低就判断“不再提高”。正确的做法是，在训练的过程中，记录最佳的validation accuracy，当连续10次epoch（或者更多次）没达到最佳accuracy时，你可以认为“不再提高”，此时使用early stopping。这个策略就叫**“ no-improvement-in-n”**，n即epoch的次数，可以根据实际情况取10、20、30….

## ****可变的学习速率****

在前面我们讲了怎么寻找比较好的learning rate，方法就是不断尝试。在一开始的时候，我们可以将其设大一点，这样就可以使weights快一点发生改变，从而让你看出cost曲线的走向（上升or下降），进一步地你就可以决定增大还是减小learning rate。

但是问题是，找出这个合适的learning rate之后，我们前面的做法是在训练这个网络的整个过程都使用这个learning rate。这显然不是好的方法，在优化的过程中，learning rate应该是逐步减小的，越接近“山谷”的时候，迈的“步伐”应该越小。

在讲前面那张cost曲线图时，我们说可以先将learning rate设置为0.25，到了第20个epoch时候设置为0.025。这是人工的调节，而且是在画出那张cost曲线图之后做出的决策。能不能让程序在训练过程中自动地决定在哪个时候减小learning rate？

答案是肯定的，而且做法很多。**一个简单有效的做法**就是，当validation accuracy满足 no-improvement-in-n规则时，本来我们是要early stopping的，但是我们可以不stop，而是让learning rate减半，之后让程序继续跑。下一次validation accuracy又满足no-improvement-in-n规则时，我们同样再将learning rate减半（此时变为原始learni rate的四分之一）…继续这个过程，直到learning rate变为原来的1/1024再终止程序。（1/1024还是1/512还是其他可以根据实际确定）。【PS：也可以选择每一次将learning rate除以10，而不是除以2.】

A readable recent paper which demonstrates the benefits of variable learning rates in attacking MNIST.《Deep Big Simple Neural Nets Excel on HandwrittenDigit Recognition》

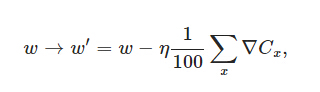
## ****正则项系数（regularization parameter, λ）****

正则项系数初始值应该设置为多少，好像也没有一个比较好的准则。建议一开始将正则项系数λ设置为0，先确定一个比较好的learning rate。然后固定该learning rate，给λ一个值（比如1.0），然后根据validation accuracy，将λ增大或者减小10倍（增减10倍是粗调节，当你确定了λ的合适的数量级后，比如λ = 0.01,再进一步地细调节，比如调节为0.02，0.03，0.009之类。）

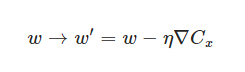
在《Neural Networks：Tricks of the Trade》中的第三章『A Simple Trick for Estimating the Weight Decay Parameter』中，有关于如何估计权重衰减项系数的讨论，有基础的读者可以看一下。

## ****Mini-batch size****

首先说一下采用mini-batch时的权重更新规则。比如mini-batch size设为100，则权重更新的规则为：



也就是将100个样本的梯度求均值，替代online learning方法中单个样本的梯度值：



当采用mini-batch时，我们可以将一个batch里的所有样本放在一个矩阵里，利用线性代数库来加速梯度的计算，这是工程实现中的一个优化方法。

**那么，size要多大？**一个大的batch，可以充分利用矩阵、线性代数库来进行计算的加速，batch越小，则加速效果可能越不明显。当然batch也不是越大越好，太大了，权重的更新就会不那么频繁，导致优化过程太漫长。所以mini-batch size选多少，不是一成不变的，根据你的数据集规模、你的设备计算能力去选。

The way to go is therefore to use some acceptable (but not necessarily optimal) values for the other hyper-parameters, and then trial a number of different mini-batch sizes, scaling η as above. Plot the validation accuracy versus time (as in, real elapsed time, not epoch!), and choose whichever mini-batch size gives you the most rapid improvement in performance. With the mini-batch size chosen you can then proceed to optimize the other hyper-parameters.

## ****更多资料****

LeCun在1998年的论文《Efficient BackProp》

Bengio在2012年的论文《Practical recommendations for gradient-based training of deep architectures》,给出了一些建议，包括梯度下降、选取超参数的详细细节。

以上两篇论文都被收录在了2012年的书《Neural Networks: Tricks of the Trade》里面，这本书里还给出了很多其他的tricks。