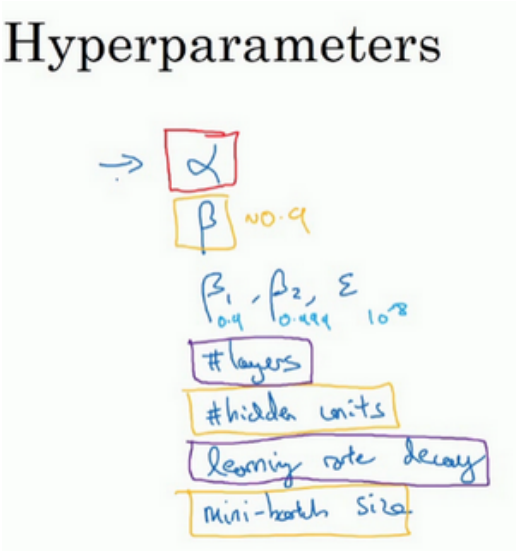
3.1 调试处理（Tuning process）

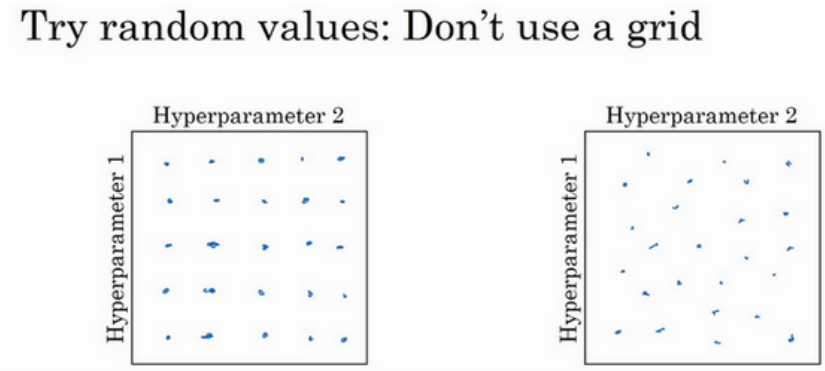


关于训练深度最难的事情之一是你要处理的参数的数量，从学习速率a到 **Momentum**（动量梯度下降法）的参数β。如果使用 **Momentum** 或 **Adam** 优化算法的参数，β_1，β_2和ε，也许你还得选择层数，也许你还得选择不同层中隐藏单元的数量，也许你还想使用学习率衰减。所以，你使用的不是单一的学习率a。接着，当然你可能还需要选择 **mini-batch** 的大小。

结果证实一些超参数比其它的更为重要，我认为，最为广泛的学习应用是a，学习速率是需要调试的最重要的超参数。

除了a，还有一些参数需要调试，例如 **Momentum** 参数β，0.9 就是个很好的默认值。我还会调试 **mini-batch** 的大小，以确保最优算法运行有效。我还会经常调试隐藏单元，我用橙色圈住的这些，这三个是我觉得其次比较重要的，相对于a而言。重要性排第三位的是其他因素，层数有时会产生很大的影响，学习率衰减也是如此。当应用 **Adam** 算法时，事实上，我从不调试β_1，β_2和ε，我总是选定其分别为 0.9，0.999 和10^{-8}，如果你想的话也可以调试它们。

但希望你粗略了解到哪些超参数较为重要，a无疑是最重要的，接下来是我用橙色圈住的那些，然后是我用紫色圈住的那些，但这不是严格且快速的标准，我认为，其它深度学习的研究者可能会很不同意我的观点或有着不同的直觉。

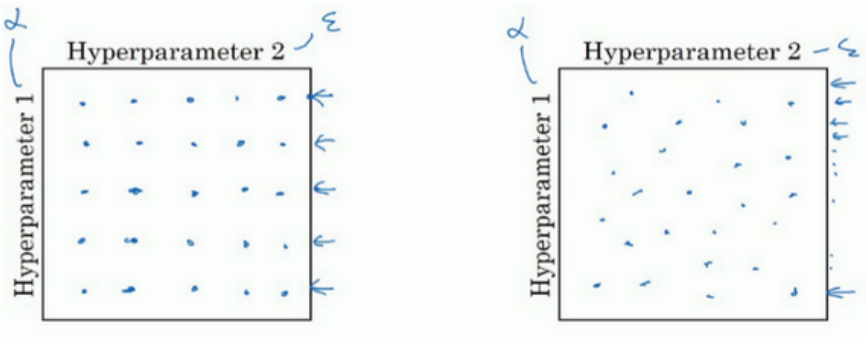


现在，如果你尝试调整一些超参数，该如何选择调试值呢？在早一代的机器学习算法中，如果你有两个超参数，这里我会称之为超参 1，超参 2，常见的做法是在网格中取样点，像这样，然后系统的研究这些数值。这里我放置的是 5×5 的网格，实践证明，网格可以是 5×5，也可多可少，但对于这个例子，你可以尝试这所有的 25 个点，然后选择哪个参数效果最好。当参数的数量相对较少时，这个方法很实用。

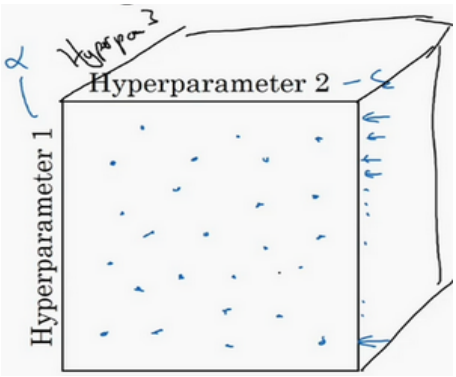
在深度学习领域，我们常做的，我推荐你采用下面的做法，随机选择点，所以你可以选择同等数量的点，对吗？25 个点，接着，用这些随机取的点试验超参数的效果。之所以这么做是因为，对于你要解决的问题而言，你很难提前知道哪个超参数最重要，正如你之前看到的，一些超参数的确要比其它的更重要。

举个例子，假设超参数 1 是α（学习速率），取一个极端的例子，假设超参数 2 是 **Adam** 算法中，分母中的ε。在这种情况下，α的取值很重要，而ε取值则无关紧要。如果你在网格中取点，接着，你试验了α的 5 个取值，那你会发现，无论ε取何值，结果基本上都是一样的。所以，你知道共有 25 种模型，但进行试验的α值只有 5 个，我认为这是很重要的。

对比而言，如果你随机取值，你会试验 25 个独立的α，似乎你更有可能发现效果做好的那个。

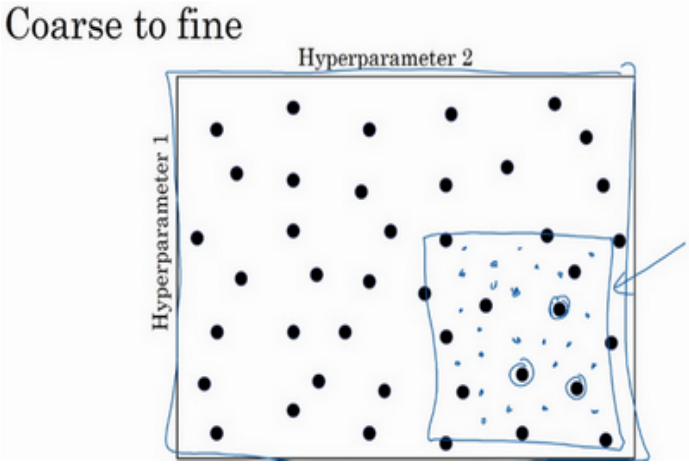


我已经解释了两个参数的情况，实践中，你搜索的超参数可能不止两个。假如，你有三个超参数，这时你搜索的不是一个方格，而是一个立方体，超参数 3 代表第三维，接着，在三维立方体中取值，你会试验大量的更多的值，三个超参数中每个都是。



实践中，你搜索的可能不止三个超参数有时很难预知，哪个是最重要的超参数，对于你的具体应用而言，随机取值而不是网格取值表明，你探究了更多重要超参数的潜在值，无论结果是什么。

当你给超参数取值时，另一个惯例是采用由粗糙到精细的策略。



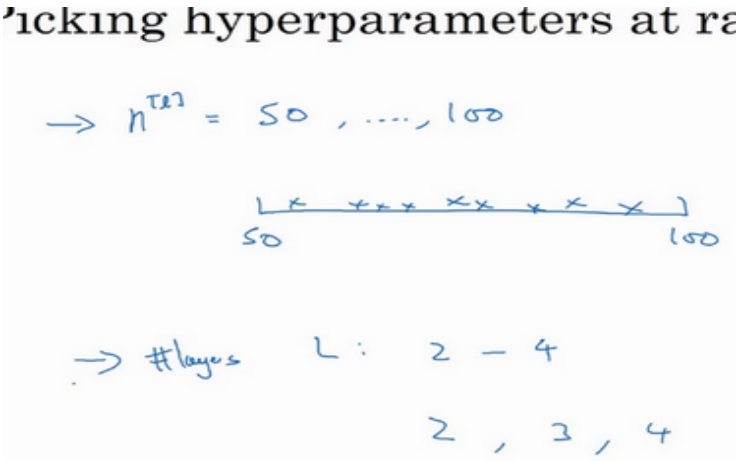
比如在二维的那个例子中，你进行了取值，也许你会发现效果最好的某个点，也许这个点周围的其他一些点效果也很好，那在接下来要做的是放大这块小区域（小蓝色方框内），然后在其中更密集得取值或随机取值，聚集更多的资源，在这个蓝色的方格中搜索，如果你怀疑这些超参数在这个区域的最优结果，那在整个的方格中进行粗略搜索后，你会知道接下来应该聚焦到更小的方格中。在更小的方格中，你可以更密集得取点。所以这种从粗到细的搜索也经常使用。

通过试验超参数的不同取值，你可以选择对训练集目标而言的最优值，或对于开发集而言的最优值，或在超参搜索过程中你最想优化的东西。

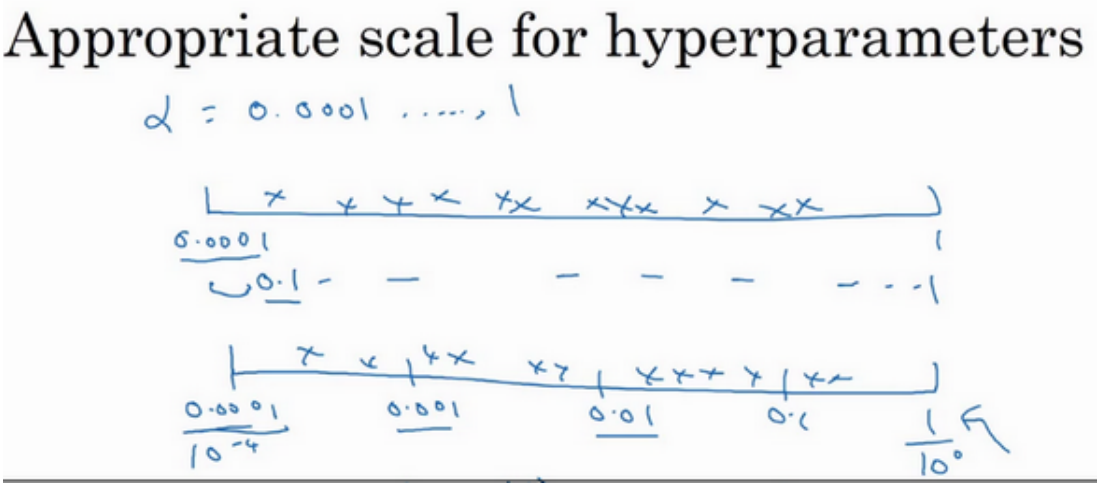
我希望，这能给你提供一种方法去系统地组织超参数搜索过程。另一个关键点是随机取值和精确搜索，考虑使用由粗糙到精细的搜索过程。但超参数的搜索内容还不止这些，在下一个视频中，我会继续讲解关于如何选择超参数取值的合理范围。

3.2 为超参数选择合适的范围（Using an appropriate scale to pick hyperparameters）

在上一个视频中，你已经看到了在超参数范围中，随机取值可以提升你的搜索效率。但随机取值并不是在有效范围内的随机均匀取值，而是选择合适的标尺，用于探究这些超参数，这很重要。在这个视频中，我会教你怎么做。



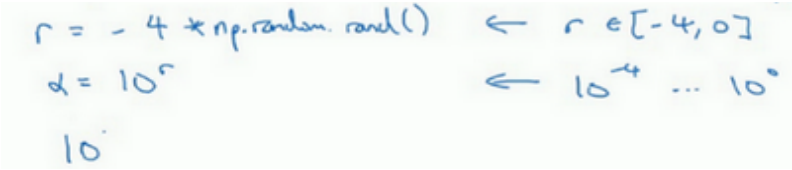
假设你要选取隐藏单元的数量$n^{[l]}$，假设，你选取的取值范围是从 50 到 100 中某点，这种情况下，看到这条从 50-100 的数轴，你可以随机在其取点，这是一个搜索特定超参数的很直观的方式。或者，如果你要选取神经网络的层数，我们称之为字母L，你也许会选择层数为 2 到 4 中的某个值，接着顺着 2，3，4 随机均匀取样才比较合理，你还可以应用网格搜索，你会觉得 2，3，4，这三个数值是合理的，这是在几个在你考虑范围内随机均匀取值的例子，这些取值还蛮合理的，但对某些超参数而言不适用。



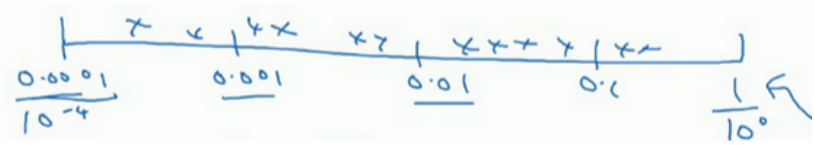
看看这个例子，假设你在搜索超参数a（学习速率），假设你怀疑其值最小是 0.0001 或最大是 1。如果你画一条从 0.0001 到 1 的数轴，沿其随机均匀取值，那 90%的数值将会落在

0.1 到 1 之间，结果就是，在 0.1 到 1 之间，应用了 90%的资源，而在 0.0001 到 0.1 之间，只有 10%的搜索资源，这看上去不太对。

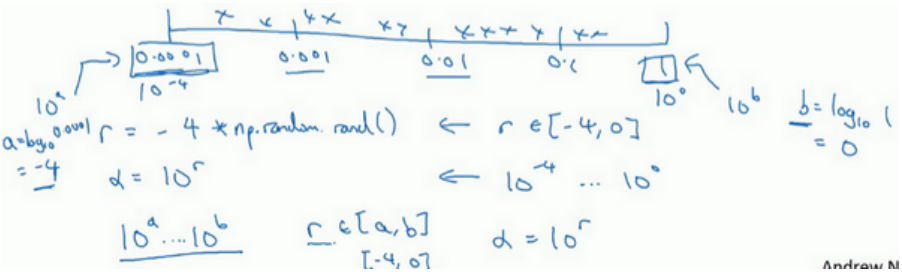
反而，用对数标尺搜索超参数的方式会更合理，因此这里不使用线性轴，分别依次取 0.0001，0.001，0.01，0.1，1，在对数轴上均匀随机取点，这样，在 0.0001 到 0.001 之间，就会有更多的搜索资源可用，还有在 0.001 到 0.01 之间等等。



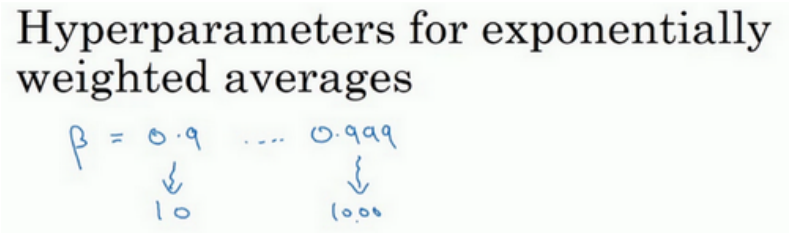
所以在 Python 中，你可以这样做，使 r=-4*np.random.rand()，然后a随机取值，$a = 10^r$，所以，第一行可以得出$r \in [-4, 0]$，那么$a \in [10^{-4}, 10^0]$，所以最左边的数字是10^{-4}，最右边是10^0。



更常见的情况是，如果你在10^a和10^b之间取值，在此例中，这是10^a（0.0001），你可以通过0.0001算出a的值，即-4，在右边的值是10^b，你可以算出b的值1，即 0。你要做的就是在$[a, b]$区间随机均匀地给r取值，这个例子中$r \in [-4,0]$，然后你可以设置a的值，基于随机取样的超参数$a = 10^r$。



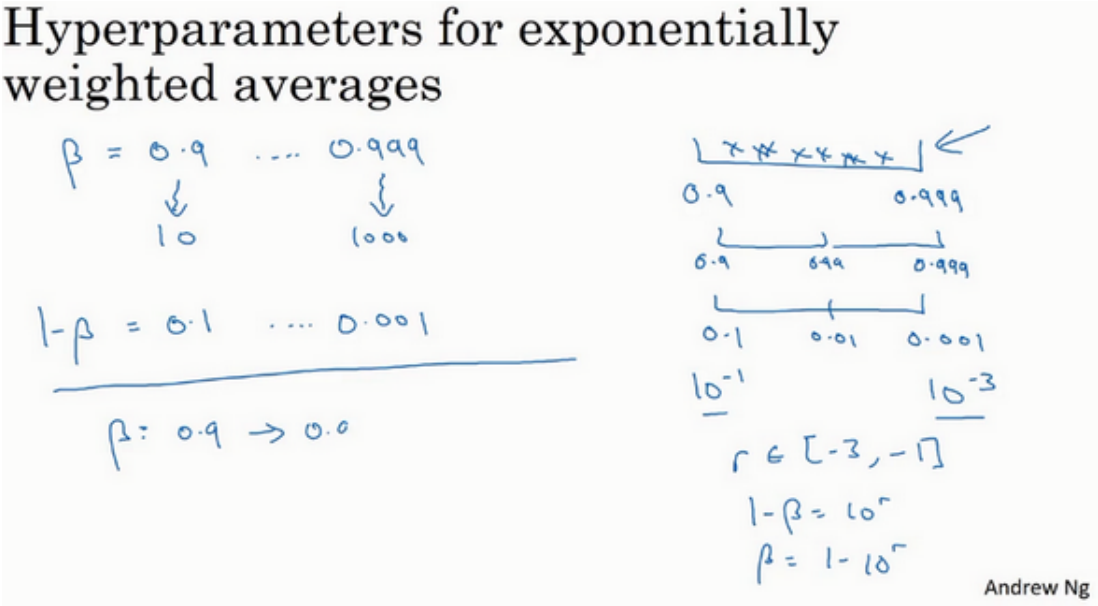
所以总结一下，在对数坐标下取值，取最小值的对数就得到a的值，取最大值的对数就得到b值，所以现在你在对数轴上的10^a到10^b区间取值，在a，b间随意均匀的选取r值，将超参数设置为10^r，这就是在对数轴上取值的过程。



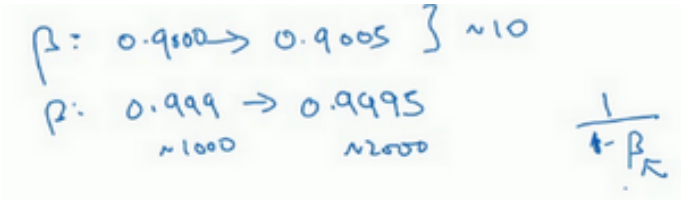
最后，另一个棘手的例子是给β 取值，用于计算指数的加权平均值。假设你认为β是 0.9

到 0.999 之间的某个值，也许这就是你想搜索的范围。**记住这一点，当计算指数的加权平均值时，取 0.9 就像在 10 个值中计算平均值，有点类似于计算 10 天的温度平均值，而取 0.999 就是在 1000 个值中取平均。**

所以和上张幻灯片上的内容类似，如果你想在 0.9 到 0.999 区间搜索，那就不能用线性轴取值，对吧？不要随机均匀在此区间取值，所以考虑这个问题最好的方法就是，我们要探究的是**$1-\beta$，此值在 0.1 到 0.001 区间内**，所以我们会给$1-\beta$取值，大概是从 0.1 到 0.001，应用之前幻灯片中介绍的方法，这是10^{-1}，这是10^{-3}，值得注意的是，在之前的幻灯片里，我们把最小值写在左边，最大值写在右边，但在这里，我们颠倒了大小。这里，左边的是最大值，右边的是最小值。所以你要做的就是在$[-3,-1]$里随机均匀的给 r 取值。你设定了$1-\beta=10^{r}$，所以$\beta=1-10^{r}$，然后这就变成了在特定的选择范围内超参数随机取值。希望用这种方式得到想要的结果，你在 0.9 到 0.99 区间探究的资源，和在 0.99 到 0.999 区间探究的一样多。



所以，如果你想研究更多正式的数学证明，关于为什么我们要这样做，为什么用线性轴取值不是个好办法，**这是因为当β 接近 1 时，所得结果的灵敏度会变化，即使β有微小的变化。所以β 在 0.9 到 0.9005 之间取值，无关紧要，你的结果几乎不会变化。**



但β值如果在 0.999 到 0.9995 之间，这会对你的算法产生巨大影响，对吧？在这两种情

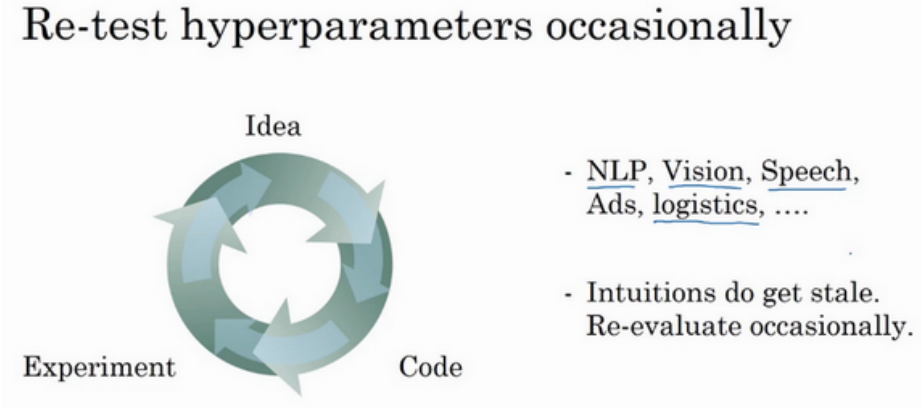
况下，是根据大概 10 个值取平均。但这里，它是指数的加权平均值，基于 1000 个值，现在是 2000 个值，因为这个公式$\frac{1}{1-\beta}$，当β接近 1 时，β就会对细微的变化变得很敏感。**所以整个取值过程中，你需要更加密集地取值，在β 接近 1 的区间内，或者说，当$1-\beta$ 接近于 0 时，这样，你就可以更加有效的分布取样点，更有效率的探究可能的结果。**

希望能帮助你选择合适的标尺，来给超参数取值。如果你没有在超参数选择中作出正确的标尺决定，别担心，即使你在均匀的标尺上取值，如果数值总量较多的话，你也会得到还不错的结果，尤其是应用从粗到细的搜索方法，在之后的迭代中，你还是会聚焦到有用的超参数取值范围上。

希望这会对你的超参数搜索有帮助，下一个视频中，我们将会分享一些关于如何组建搜索过程的思考，希望它能使你的工作更高效。

3.3 超参数调试实践：Pandas VS Caviar（Hyperparameters tuning in practice: Pandas vs. Caviar）

到现在为止，你已经听了许多关于如何搜索最优超参数的内容，在结束我们关于超参数搜索的讨论之前，我想最后和你分享一些建议和技巧，关于如何组织你的超参数搜索过程。

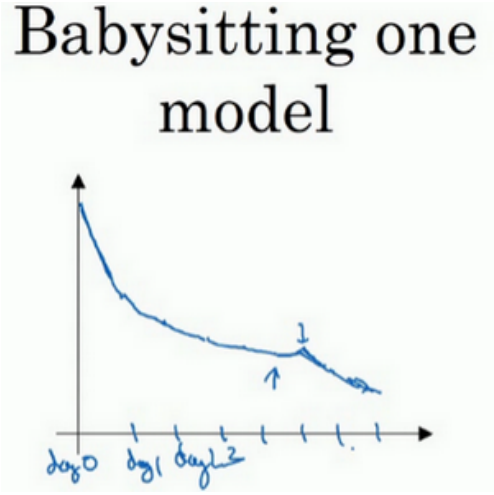


如今的深度学习已经应用到许多不同的领域，某个应用领域的超参数设定，有可能通用于另一领域，不同的应用领域出现相互交融。比如，我曾经看到过计算机视觉领域中涌现的巧妙方法，比如说 **Confonets** 或 **ResNets**，这我们会在后续课程中讲到。它还成功应用于语音识别，我还看到过最初起源于语音识别的想法成功应用于 **NLP** 等等。

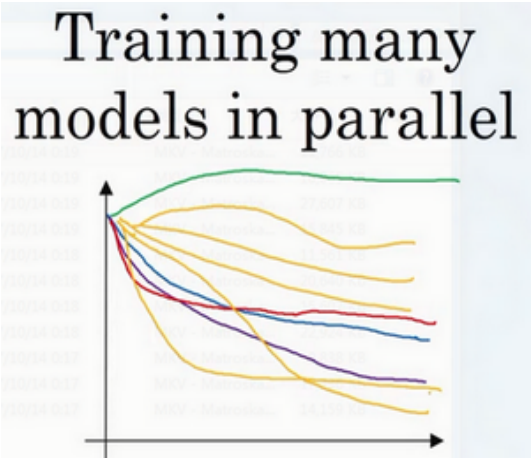
深度学习领域中，发展很好的一点是，不同应用领域的人们会阅读越来越多其它研究领域的文章，跨领域去寻找灵感。

就超参数的设定而言，我见到过有些直觉想法变得很缺乏新意，所以，即使你只研究一个问题，比如说逻辑学，你也许已经找到一组很好的参数设置，并继续发展算法，或许在几个月的过程中，观察到你的数据会逐渐改变，或也许只是在你的数据中心更新了服务器，正因为有了这些变化，你原来的超参数的设定不再好用，所以我建议，**或许只是重新测试或评估你的超参数，至少每隔几个月一次，以确保你对数值依然很满意。**

最后，关于如何搜索超参数的问题，我见过大概两种重要的思想流派或人们通常采用的两种重要但不同的方式。

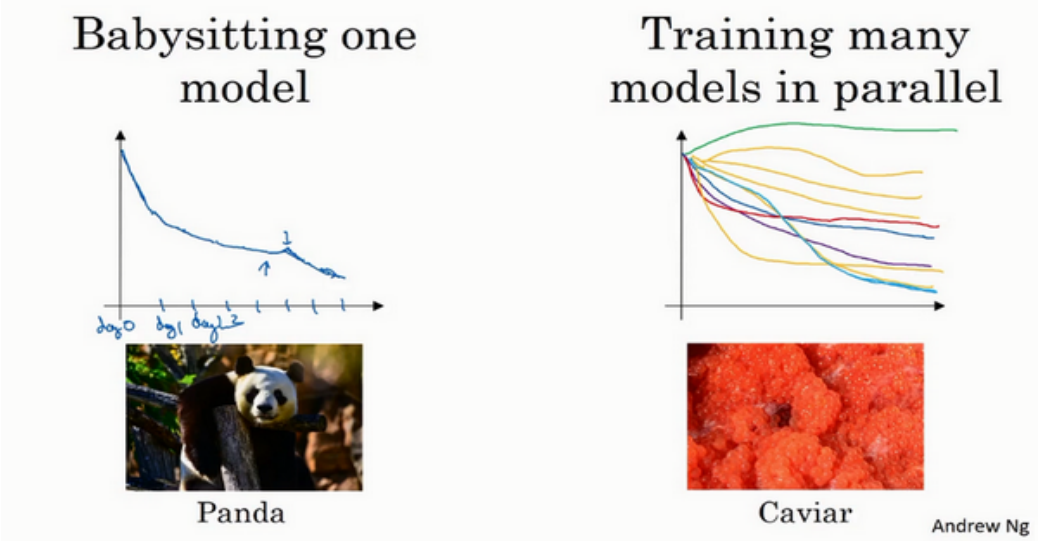


一种是你照看一个模型，通常是有庞大的数据组，但没有许多计算资源或足够的 **CPU** 和 **GPU** 的前提下，基本而言，你只可以一次负担起试验一个模型或一小批模型，在这种情况下，即使当它在试验时，你也可以逐渐改良。比如，第 0 天，你将随机参数初始化，然后开始试验，然后你逐渐观察自己的学习曲线，也许是损失函数 J，或者数据设置误差或其它的东西，在第 1 天内逐渐减少，那这一天末的时候，你可能会说，看，它学习得真不错。我试着增加一点学习速率，看看它会怎样，也许结果证明它做得更好，那是你第二天的表现。两天后，你会说，它依旧做得不错，也许我现在可以填充下 **Momentum** 或减少变量。然后进入第三天，每天，你都会观察它，不断调整你的参数。也许有一天，你会发现你的学习率太大了，所以你可能又回归之前的模型，像这样，但你可以说是在每天花时间照看此模型，即使是它在许多天或许多星期的试验过程中。所以这是一个人们照料一个模型的方法，观察它的表现，耐心地调试学习率，但那通常是因为你没有足够的计算能力，不能在同一时间试验大量模型时才采取的办法。



另一种方法则是同时试验多种模型，你设置了一些超参数，尽管让它自己运行，或者是一天甚至多天，然后你会获得像这样的学习曲线，这可以是损失函数 J 或实验误差或损失或

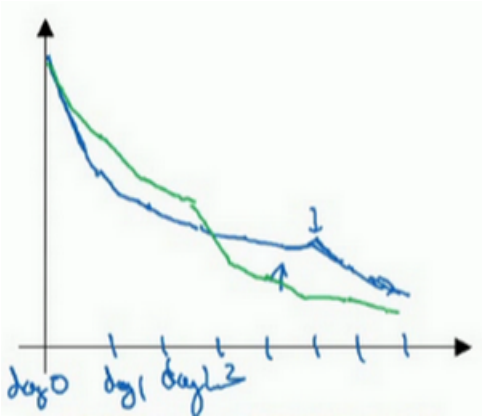
数据误差的损失，但都是你曲线轨迹的度量。同时你可以开始一个有着不同超参数设定的不同模型，所以，你的第二个模型会生成一个不同的学习曲线，也许是像这样的一条（紫色曲线），我会说这条看起来更好些。与此同时，你可以试验第三种模型，其可能产生一条像这样的学习曲线（红色曲线），还有另一条（绿色曲线），也许这条有所偏离，像这样，等等。或者你可以同时平行试验许多不同的模型，橙色的线就是不同的模型。用这种方式你可以试验许多不同的参数设定，然后只是最后快速选择工作效果最好的那个。在这个例子中，也许这条看起来是最好的（下方绿色曲线）。



打个比方，我把左边的方法称为熊猫方式。当熊猫有了孩子，他们的孩子非常少，一次通常只有一个，然后他们花费很多精力抚养熊猫宝宝以确保其能成活，所以，这的确是一种照料，一种模型类似于一只熊猫宝宝。对比而言，右边的方式更像鱼类的行为，我称之为鱼子酱方式。在交配季节，有些鱼类会产下一亿颗卵，但鱼类繁殖的方式是，它们会产生很多卵，但不对其中任何一个多加照料，只是希望其中一个，或其中一群，能够表现出色。我猜，这就是哺乳动物繁衍和鱼类，很多爬虫类动物繁衍的区别。我将称之为熊猫方式与鱼子酱方式，因为这很有趣，更容易记住。

所以这两种方式的选择，是由你拥有的计算资源决定的，如果你拥有足够的计算机去平行试验许多模型，那绝对采用鱼子酱方式，尝试许多不同的超参数，看效果怎么样。但在一些应用领域，比如在线广告设置和计算机视觉应用领域，那里的数据太多了，你需要试验大量的模型，所以同时试验大量的模型是很困难的，它的确是依赖于应用的过程。但我看到那些应用熊猫方式多一些的组织，那里，你会像对婴儿一样照看一个模型，调试参数，试着让它工作运转。尽管，当然，甚至是在熊猫方式中，试验一个模型，观察它工作与否，也许第

二或第三个星期后，也许我应该建立一个不同的模型（绿色曲线），像熊猫那样照料它，我猜，这样一生中可以培育几个孩子，即使它们一次只有一个孩子或孩子的数量很少。



所以希望你能学会如何进行超参数的搜索过程，现在，还有另一种技巧，能使你的神经网络变得更加坚实，它并不是对所有的神经网络都适用，但当适用时，它可以使超参数搜索变得容易许多并加速试验过程，我们在下个视频中再讲解这个技巧。