克服过拟合和提高泛化能力的20条技巧和诀窍

<https://blog.csdn.net/u012968002/article/details/55212082>

作者：[visionshop](https://me.csdn.net/u012968002" \t "_blank)

你是如何提升[深度学习](http://lib.csdn.net/base/deeplearning" \o "深度学习知识库" \t "_blank)模型的效果？

这是我经常被问到的一个问题。   
有时候也会换一种问法：   
我该如何提高模型的准确率呢？   
……或者反过来问：   
如果我的网络模型效果不好，我该怎么办？   
通常我的回答是“具体原因我不清楚，但我有一些想法可以试试”。   
然后我会列举一些我认为能够提升性能的方法。   
为了避免重复罗列这些内容，我打算在本文中把它们都写出来。   
这些想法不仅可以用于深度学习，事实上可以用在任何[机器学习](http://lib.csdn.net/base/machinelearning" \o "机器学习知识库" \t "_blank)的[算法](http://lib.csdn.net/base/datastructure)上。

  
如何提升深度学习的性能 ，[Pedro Ribeiro Simoes](https://www.flickr.com/photos/pedrosimoes7/8654817332/" \t "_blank)拍摄

**前言：提升算法性能的想法**

这个列表并不完整，却是很好的出发点。

我的目的是给大家抛出一些想法供大家尝试，或许有那么一两个有效的方法。 往往只需要尝试一个想法就能得到提升。 如果你用下面某一种想法取得了好效果，请在评论区给我留言！ 如果你还有其它想法或是对这些想法有拓展，也请告诉大家，或许会对我们大家有帮助！

我把这个列表划分为四块：

* **从数据上提升性能**
* **从算法上提升性能**
* **从算法调优上提升性能**
* **从模型融合上提升性能**

性能提升的力度按上表的顺序从上到下依次递减。举个例子，新的建模方法或者更多的数据带来的效果提升往往好于调出最优的参数。但这并不是绝对的，只是大多数情况下如此。

我在文章中添加了不少博客教程和相关的[经典神经网络问题](ftp://ftp.sas.com/pub/neural/FAQ.html" \t "_blank)。 其中有一些想法只是针对人工神经网络，但大多数想法都是通用性的。你可以将它们与其它技术结合起来使用。   
我们开始吧。

**1.从数据上提升性能**

调整训练数据或是问题的抽象定义方法可能会带来巨大的效果改善。甚至是最显著的改善。下面是概览：

1. 收集更多的数据
2. 产生更多的数据
3. 对数据做缩放
4. 对数据做变换
5. 特征选择
6. 重新定义问题

**1）收集更多的数据**

你还能收集到更多的训练数据吗？ 你的模型的质量往往取决于你的训练数据的质量。你需要确保使用的数据是针对问题最有效的数据。你还希望数据尽可能多。深度学习和其它现代的非线性机器学习模型在[**大数据**](http://lib.csdn.net/base/hadoop)集上的效果更好，尤其是深度学习。这也是深度学习方法令人兴奋的主要原因之一。 请看下面的图片：



什么是深度学习？幻灯片来自Andrew Ng

不总是数据阅读效果越好，多数情况下如此。如果让我选择，我会选择要更多的数据。相关阅读：

* [数据集压倒算法](https://www.edge.org/response-detail/26587)

**2）产生更多的数据**

深度学习算法往往在数据量大的时候效果好。   
我们在上一节已经提到过这一点。   
如果由于某些原因你得不到更多的数据，也可以制造一些数据。

* 如果你的数据是数值型的向量，那么随机生成已有向量的变形向量。
* 如果你的数据是图像，用已有的图像随机生成相似图像。
* 如果你的数据是文本，做法你懂得……

这类做法通常被称为数据扩展或是数据生成。   
你可以使用生成模型，也可以用一些简单的小技巧。   
举个例子，若是用图像数据，简单地随机选择和平移已有的图像就能取得很大的提升。它能提升模型的泛化能力，如果新的数据中包含这类变换就能得到很好的处理。   
有时候是往数据中增加噪声，这相当于是一种规则方法，避免过拟合训练数据。   
相关阅读：

* [深度学习中的图像数据扩充](http://machinelearningmastery.com/image-augmentation-deep-learning-keras/)
* [训练含有噪声的数据](ftp://ftp.sas.com/pub/neural/FAQ3.html#A_jitter)

**3）对数据做缩放**

此方法简单有效。   
使用神经网络模型的一条经验法宝就是：   
将数据缩放到激活函数的阈值范围。   
如果你使用sigmoid激活函数，将数据缩放到0~1之间。如果选用tanh激活函数，将值域控制在-1~1之间。   
输入、输出数据都经过同样的变换。比如，如果在输出层有一个sigmoid函数将输出值转换为二值数据，则将输出的y归一化为二进制。如果选用的是softmax函数，对y进行归一化还是有效的。   
我还建议你将训练数据扩展生成多个不同的版本：

* 归一化到0 ~ 1
* 归一化到-1 ~ 1
* 标准化

然后在每个数据集上[**测试**](http://lib.csdn.net/base/softwaretest)模型的性能，选用最好的一组生成数据。   
如果更换了激活函数，最好重复做一次这个小实验。   
在模型中不适合计算大的数值。此外，还有许多其它方法来压缩模型中的数据，比如对权重和激活值做归一化，我会在后面介绍这些技巧。   
相关阅读：

* [我需要对输入数据（列向量）做标准化吗?](ftp://ftp.sas.com/pub/neural/FAQ2.html#A_std)
* [如何用Scikit-Learn准备机器学习的输入数据](http://machinelearningmastery.com/prepare-data-machine-learning-python-scikit-learn/)

**4）对数据做变换**

与上一节的方法相关，但是需要更多的工作量。   
你必须真正了解所用到的数据。数据可视化，然后挑出异常值。   
先猜测每一列数据的分布

* 这一列数据是不是倾斜的高斯分布，若是如此，尝试用Box-Cox方法纠正倾斜
* 这一列数据是不是指数分布，若是如此，则进行对数变换
* 这一列数据是不是存在某些特性，但是难以直观地发现，尝试一下对数据平方或者开方
* 是否可以将特征离散化，以便更好地强调一些特征

凭你的直觉，尝试几种方法

* 是否可以用投影的方法对数据预处理，比如PCA？
* 是否可以将多个属性合并为单个值？
* 是否可以发掘某个新的属性，用布尔值表示？
* 是否可以在时间尺度或是其它维度上有些新发现？

神经网络有特征学习的功能，它们能够完成这些事情。   
不过你若是可以将问题的结构更好地呈现出来，网络模型学习的速度就会更快。   
在训练集上快速尝试各种变换方法，看看哪些方法有些，而哪些不起作用。   
相关阅读：

* [如何定义你的机器学习问题](http://machinelearningmastery.com/how-to-define-your-machine-learning-problem/)
* [特征挖掘工程，如何构造特征以及如何提升](http://machinelearningmastery.com/discover-feature-engineering-how-to-engineer-features-and-how-to-get-good-at-it/)
* [如何用Scikit-Learn准备机器学习的输入数据](http://machinelearningmastery.com/prepare-data-machine-learning-python-scikit-learn/)

**5）特征选择**

神经网络受不相关数据的影响很小。   
它们会对此赋予一个趋近于0的权重，几乎忽略此特征对预测值的贡献。   
你是否可以移除训练数据的某些属性呢？   
我们有许多的特征选择方法和特征重要性方法来鉴别哪些特征可以保留，哪些特征需要移除。   
动手试一试，试一试所有的方法。   
如果你的时间充裕，我还是建议在相同的神经网络模型上选择尝试多个方法，看看它们的效果分别如何。

* 也许用更少的特征也能得到同样的、甚至更好的效果。
* 也许所有的特征选择方法都选择抛弃同一部分特征属性。那么就真应该好好审视这些无用的特征。
* 也许选出的这部分特征给你带来了新的启发，构建出更多的新特征。

相关阅读：

* [特征选择入门介绍](http://machinelearningmastery.com/an-introduction-to-feature-selection/)
* [基于Python的机器学习中的特征选择问题](http://machinelearningmastery.com/feature-selection-machine-learning-python/)

**6）问题重构**

在回到你问题的定义上来。   
你所收集到的这些观测数据是描述问题的唯一途径吗？   
也许还有其它的途径。也许其它途径能更清晰地将问题的结构暴露出来。   
我自己非常喜欢这种练习，因为它强迫我们拓宽思路。很难做好。尤其是当你已经投入大量的时间、精力、金钱在现有的方法上。   
即使你列举了3 ~ 5种不同的方式，至少你对最后所选用的方式有充足的信心。

* 也许你可以将时间元素融入到一个窗口之中
* 也许你的分类问题可以转化为回归问题，反之亦然
* 也许可以把二值类型的输出转化为softmax的输出
* 也许你可以对子问题建模

深入思考问题是一个好习惯，最好在选择工具下手之前先完成上述步骤，以减少无效的精力投入。   
无论如何，如果你正束手无策，这个简单的连续能让你思如泉涌。   
另外，你也不必抛弃前期的大量工作，详情可以参见后面的章节。

相关阅读：

* [如何定义机器学习问题](http://machinelearningmastery.com/how-to-define-your-machine-learning-problem/)

**2. 从算法上提升性能**

机器学习总是与算法相关。   
所有的理论和数学知识都在描述从数据中学习决策过程的不同方法（如果我们这里仅讨论预测模型）。   
你选用深度学习来求解，它是不是最合适的技术呢？   
在这一节中，我们会简单地聊一下算法的选择，后续内容会具体介绍如何提升深度学习的效果。   
下面是概览：

1. 算法的筛选
2. 从文献中学习
3. 重采样的方法

我们一条条展开。

**1）算法的筛选**

你事先不可能知道哪种算法对你的问题效果最好。   
如果你已经知道，你可能也就不需要机器学习了。   
你有哪些证据可以证明现在已经采用的方法是最佳选择呢？   
我们来想想这个难题。   
当在所有可能出现的问题上进行效果评测时，没有哪一项单独的算法效果会好于其它算法。所有的算法都是平等的。这就是[天下没有免费的午餐理论](https://en.wikipedia.org/wiki/No_free_lunch_theorem" \t "_blank)的要点。

**也许你选择的算法并不是最适合你的问题。**   
现在，我们不指望解决所有的问题，但当前的热门算法也许并不适合你的数据集。   
我的建议是先收集证据，先假设有其它的合适算法适用于你的问题。   
筛选一些常用的算法，挑出其中适用的几个。

* 尝试一些线性算法，比如逻辑回归和线性判别分析
* 尝试一些树模型，比如CART、随机森林和梯度提升
* 尝试SVM和kNN等算法
* 尝试其它的神经网络模型，比如LVQ、MLP、CNN、LSTM等等

采纳效果较好的几种方法，然后精细调解参数和数据来进一步提升效果。   
将你所选用的深度学习方法与上述这些方法比较，看看是否能击败他们？   
也许你可以放弃深度学习模型转而选择更简单模型，训练的速度也会更快，而且模型易于理解。   
相关阅读：

* [一种数据驱动的机器学习方法](http://machinelearningmastery.com/a-data-driven-approach-to-machine-learning/)
* [面对机器学习问题为何需要筛选算法](http://machinelearningmastery.com/why-you-should-be-spot-checking-algorithms-on-your-machine-learning-problems/)
* [用scikit-learn筛选机器学习的分类算法](http://machinelearningmastery.com/spot-check-classification-machine-learning-algorithms-python-scikit-learn/)

**2）从文献中学习**

从文献中“窃取”思路是一条捷径。   
其它人是否已经做过和你类似的问题，他们使用的是什么方法。   
阅读论文、书籍、问答网站、教程以及Google给你提供的一切信息。   
记下所有的思路，然后沿着这些方向继续探索。   
这并不是重复研究，这是帮助你发现新的思路。

**优先选择已经发表的论文**   
已经有许许多多的聪明人写下了很多有意思的事情。利用好这宝贵的资源吧。   
相关阅读：

* [如何研究一种机器学习算法](http://machinelearningmastery.com/how-to-research-a-machine-learning-algorithm/)
* [Google学术](http://scholar.google.com/)

**3）重采样的方法**

你必须明白自己模型的效果如何。   
你估计的模型效果是否可靠呢？   
深度学习模型的训练速度很慢。   
这就意味着我们不能用标准的黄金法则来评判模型的效果，比如k折交叉验证。

* 也许你只是简单地把数据分为训练集和测试集。如果是这样，就需要保证切分后的数据分布保持不变。单变量统计和数据可视化是不错的方法。
* 也许你们可以扩展硬件来提升效果。举个例子，如果你有一个集群或是AWS的账号，我们可以并行训练n个模型，然后选用它们的均值和方差来获取更稳定的效果。
* 也许你可以选择一部分数据做交叉验证（对于early stopping非常有效）。
* 也许你可以完全独立地保留一部分数据用于模型的验证。

另一方面，也可以让数据集变得更小，采用更强的重采样方法。

* 也许你会看到在采样后的数据集上训练得到的模型效果与在全体数据集上训练得到的效果有很强的相关性。那么，你就可以用小数据集进行模型的选择，然后把最终选定的方法应用于全体数据集上。
* 也许你可以任意限制数据集的规模，采样一部分数据，用它们完成所有的训练任务。

**你必须对模型效果的预测有十足的把握。**   
相关阅读：

* [用Keras评估深度学习模型的效果](http://machinelearningmastery.com/evaluate-performance-deep-learning-models-keras/)
* [用重采样的方法评估机器学习算法的效果](http://machinelearningmastery.com/evaluate-performance-machine-learning-algorithms-python-using-resampling/)

**3. 从算法调优上提升性能**

你通过算法筛选往往总能找出一到两个效果不错的算法。但想要达到这些算法的最佳状态需要耗费数日、数周甚至数月。   
下面是一些想法，在调参时能有助于提升算法的性能。

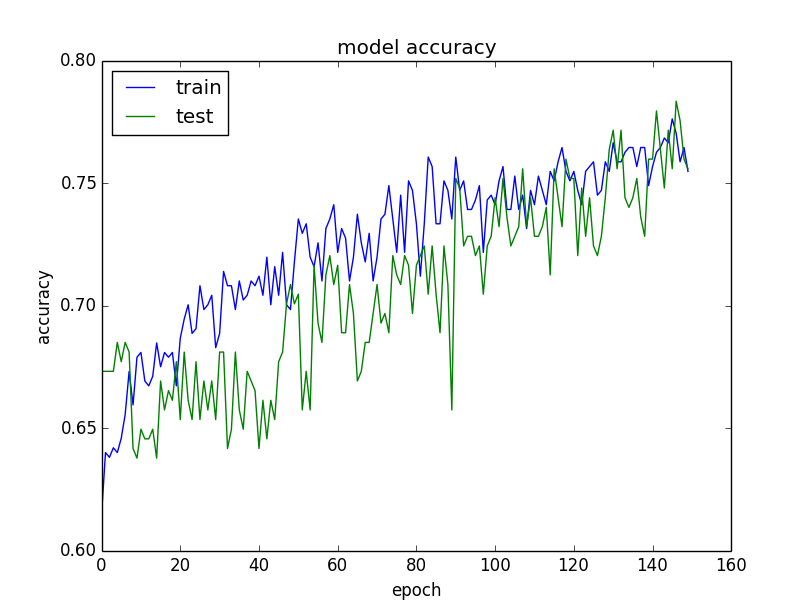
* 模型可诊断性
* 权重的初始化
* 学习率
* 激活函数
* 网络结构
* batch和epoch
* 正则项
* 优化目标
* 提早结束训练

你可能需要指定参数来多次（3-10次甚至更多）训练模型，以得到预计效果最好的一组参数。对每个参数都要不断的尝试。   
有一篇关于超参数最优化的优质博客：

* [如何用Keras网格搜索深度学习模型的超参数](http://machinelearningmastery.com/grid-search-hyperparameters-deep-learning-models-python-keras/)

**1）可诊断性**

只有知道为何模型的性能不再有提升了，才能达到最好的效果。   
是因为模型过拟合呢，还是欠拟合呢？   
**千万牢记这个问题。千万。**   
模型总是处于这两种状态之间，只是程度不同罢了。   
一种快速查看模型性能的方法就是每一步计算模型在训练集和验证集上的表现，将结果绘制成图表。



在训练集和验证集上测试模型的准确率

* 如果训练集的效果好于验证集，说明可能存在过拟合的现象，试一试增加正则项
* 如果训练集和验证集的准确率都很低，说明可能存在欠拟合，你可以继续提升模型的能力，延长训练步骤。
* 如果训练集和验证集的曲线有一个焦点，可能需要用到early stopping的技巧了

经常绘制类似的图表，深入研究并比较不同的方法，以提高模型的性能。

**这些图表也许是你最有价值的诊断工具。**   
另一种有效的诊断方法是研究模型正确预测或是错误预测的样本。   
在某些场景下，这种方法能给你提供一些思路。

* 也许你需要更多的难预测的样本数据
* 也许你可以从训练集中删去那些容易被学习的样本
* 也许你可以有针对性地对不同类型的输入数据训练不同的模型

相关阅读：

* [用Keras展现深度学习模型的训练过程](http://machinelearningmastery.com/display-deep-learning-model-training-history-in-keras/)
* [机器学习算法的过拟合和欠拟合](http://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/)

**2）权重的初始化**

有一条经验规则：用小的随机数初始化权重。   
事实上，这可能已经足够了。但是这是你网络模型的最佳选择吗？   
不同的激活函数也可以有不同的应对策略，但我不记得在实践中存在什么显著的差异。   
保持你的模型结构不变，试一试不同的初始化策略。   
记住，权重值就是你模型需要训练的参数。几组不同的权重值都能取得不错的效果，但你想得到更好的效果。

* 尝试所有的初始化方法，找出最好的一组初始化值
* 试一试用非监督式方法预学习，比如自动编码机
* 尝试用一组现有的模型权重参数，然后重新训练输入和输出层（迁移学习）

记住，修改权重初始化值的方法与修改激活函数或者目标函数的效果相当。   
相关阅读：

* [深度网络模型的初始化](http://deepdish.io/2015/02/24/network-initialization/)

**3）学习率**

调节学习率也能带来效果提升。   
这里也有一些探索的思路：

* 尝试非常大、非常小的学习率
* 根据参考文献，在常规值附近用网格化搜索
* 尝试使用逐步减小的学习率
* 尝试每隔固定训练步骤衰减的学习率
* 尝试增加一个向量值，然后用网格搜索

大的网络模型需要更多的训练步骤，反之亦然。如果你添加了更多的神经节点和网络层，请加大学习率。   
学习率与训练步骤、batch大小和优化方法都有耦合关系。

相关阅读：

* [使用Keras对深度学习模型进行学习率调节](http://machinelearningmastery.com/using-learning-rate-schedules-deep-learning-models-python-keras/)
* [反向传播算法该选用什么学习率？](ftp://ftp.sas.com/pub/neural/FAQ2.html#A_learn_rate)

**4）激活函数**

也许你应该选用ReLU激活函数。   
仅仅因为它们的效果更好。   
在ReLU之前流行sigmoid和tanh，然后是输出层的softmax、线性和sigmoid函数。除此之外，我不建议尝试其它的选择。   
这三种函数都试一试，记得把输入数据归一化到它们的值域范围。   
显然，你需要根据输出内容的形式选择转移函数。   
比方说，将二值分类的sigmoid函数改为回归问题的线性函数，然后对输出值进行再处理。同时，可能需要调整合适的损失函数。在数据转换章节去寻找更多的思路吧。   
相关阅读：

* [为何使用激活函数？](ftp://ftp.sas.com/pub/neural/FAQ2.html#A_act)

**5）网络拓扑结构**

调整网络的拓扑结构也会有一些帮助。   
你需要设计多少个节点，需要几层网络呢？   
别打听了，鬼知道是多少。   
你必须自己找到一组合理的参数配置。

* 试一试加一层有许多节点的隐藏层（拓宽）
* 试一试一个深层的神经网络，每层节点较少（纵深）
* 尝试将上面两种组合
* 尝试模仿近期发表的问题类似的论文
* 尝试拓扑模式和书本上的经典技巧（参考下方的链接）

这是一个难题。越大的网络模型有越强的表达能力，也许你就需要这样一个。   
更多晨的结构提供了抽象特征的更多结构化组合的可能，也许你也需要这样一个网络。   
后期的网络模型需要更多的训练过程，需要不断地调节训练步长和学习率。   
相关阅读：   
下面的链接可能给你提供一些思路：

* [我的网络模型该设计几层呢？](ftp://ftp.sas.com/pub/neural/FAQ3.html#A_hl)
* [我的网络模型该设计几个节点呢？](ftp://ftp.sas.com/pub/neural/FAQ3.html#A_hu)

**6）batch和epoch**

batch的大小决定了梯度值，以及权重更新的频率。一个epoch指的是训练集的所有样本都参与了一轮训练，以batch为序。   
你尝试过不同的batch大小和epoch的次数吗？   
在前文中，我们已经讨论了学习率、网络大小和epoch次数的关系。   
深度学习模型常用小的batch和大的epoch以及反复多次的训练。   
这或许对你的问题会有帮助。

* 尝试将batch大小设置为全体训练集的大小（batch learning）
* 尝试将batch大小设置为1（online learning）
* 用网格搜索尝试不同大小的mini-batch（8，16，32，…）
* 尝试再训练几轮epoch，然后继续训练很多轮epoch

尝试设置一个近似于无限大的epoch次数，然后快照一些中间结果，寻找效果最好的模型。   
有些模型结构对batch的大小很敏感。我觉得多层感知器对batch的大小很不敏感，而LSTM和CNN则非常敏感，但这都是仁者见仁。

相关阅读：

* [什么是批量学习、增量学习和在线学习？](ftp://ftp.sas.com/pub/neural/FAQ2.html#A_styles)
* [直觉上，mini-batch的大小如何影响（随机）梯度下降的效果？](https://www.quora.com/Intuitively-how-does-mini-batch-size-affect-the-performance-of-stochastic-gradient-descent)

**7）正则项**

正则化是克服训练数据过拟合的好方法。   
最近热门的正则化方法是dropout，你试过吗？   
Dropout方法在训练过程中随机地略过一些神经节点，强制让同一层的其它节点接管。简单却有效的方法。

* 权重衰减来惩罚大的权重值
* 激活限制来惩罚大的激活函数值

尝试用各种惩罚措施和惩罚项进行实验，比如L1、L2和两者之和。   
相关阅读：

* [使用Keras对深度学习模型做dropout正则化](http://machinelearningmastery.com/dropout-regularization-deep-learning-models-keras/)
* [什么是权值衰减？](ftp://ftp.sas.com/pub/neural/FAQ3.html#A_decay)

**8）优化方法和损失函数**

以往主要的求解方法是随机梯度下降，然而现在有许许多多的优化器。   
你尝试过不同的优化策略吗？   
随机梯度下降是默认的方法。先用它得到一个结果，然后调节不同的学习率、动量值进行优化。   
许多更高级的优化方法都用到更多的参数，结构更复杂，收敛速度更快。这取决于你的问题，各有利弊吧。   
为了压榨现有方法的更多潜力，你真的需要深入钻研每个参数，然后用网格搜索法测试不同的取值。过程很艰辛，很花时间，但值得去尝试。   
我发现更新/更流行的方法收敛速度更快，能够快速了解某个网络拓扑的潜力，例如：

* [ADAM](https://arxiv.org/abs/1412.6980)
* RMSprop

你也可以探索其它的优化算法，例如更传统的算法（Levenberg-Marquardt）和比较新的算法（基因算法）。其它方法能给SGD创造好的开端，便于后续调优。   
待优化的损失函数则与你需要解决的问题更相关。   
不过，也有一些常用的伎俩（比如回归问题常用MSE和MAE），换个损失函数有时也会带来意外收获。同样，这可能也与你输入数据的尺度以及所使用的激活函数相关。   
相关阅读：

* [梯度下降优化算法概览](http://sebastianruder.com/optimizing-gradient-descent/)
* [什么是共轭梯度和Levenberg-Marquardt？](http://blog.csdn.net/starzhou/article/details/52754436)
* [深度学习的优化方法，2011](http://ai.stanford.edu/~ang/papers/icml11-OptimizationForDeepLearning.pdf)

**9）Early Stopping**

你可以在模型性能开始下降的时候停止训练。   
这帮我们节省了大量时间，也许因此就能使用更精细的重采样方法来评价模型了。   
early stopping也是防止数据过拟合的一种正则化方法，需要你在每轮训练结束后观察模型在训练集和验证集上的效果。   
一旦模型在验证集上的效果下降了，则可以停止训练。   
你也可以设置检查点，保存当时的状态，然后模型可以继续学习。   
相关阅读：

* [如何在Keras给深度学习模型设置check-point](http://machinelearningmastery.com/check-point-deep-learning-models-keras/)
* [什么是early stopping？](ftp://ftp.sas.com/pub/neural/FAQ3.html#A_stop)

**4. 用融合方法提升效果**

你可以将多个模型的预测结果融合。   
继模型调优之后，这是另一个大的提升领域。   
事实上，往往将几个效果还可以的模型的预测结果融合，取得的效果要比多个精细调优的模型分别预测的效果好。   
我们来看一下模型融合的三个主要方向：

1. 模型融合
2. 视角融合
3. stacking

**1）模型融合**

不必挑选出一个模型，而是将它们集成。   
如果你训练了多个深度学习模型，每一个的效果都不错，则将它们的预测结果取均值。   
模型的差异越大，效果越好。举个例子，你可以使用差异很大的网络拓扑和技巧。   
如果每个模型都独立且有效，那么集成后的结果效果更稳定。   
相反的，你也可以反过来做实验。   
每次训练网络模型时，都以不同的方式初始化，最后的权重也收敛到不同的值。多次重复这个过程生成多个网络模型，然后集成这些模型的预测结果。   
它们的预测结果会高度相关，但对于比较难预测的样本也许会有一点提升。   
相关阅读：

* [用scikit-learn集成机器学习算法](http://machinelearningmastery.com/ensemble-machine-learning-algorithms-python-scikit-learn/)
* [如何提升机器学习的效果](http://machinelearningmastery.com/how-to-improve-machine-learning-results/)

**2）视角融合**

如上一节提到的，以不同的角度来训练模型，或是重新刻画问题。   
我们的目的还是得到有用的模型，但是方式不同（如不相关的预测结果）。   
你可以根据上文中提到的方法，对训练数据采取完全不同的缩放和变换技巧。   
所选用的变化方式和问题的刻画角度差异越大，效果提升的可能性也越大。   
简单地对预测结果取均值是一个不错的方式。

**3）stacking**

你还可以学习如何将各个模型的预测结果相融合。   
这被称作是stacked泛化，或者简称为stacking。   
通常，可以用简单的线性回归的方式学习各个模型预测值的权重。   
把各个模型预测结果取均值的方法作为baseline，用带权重的融合作为实验组。

* [Stacked Generalization (Stacking)](http://machine-learning.martinsewell.com/ensembles/stacking/)

**总结**

各抒己见吧

**补充资料**

还有一些非常好的资料，但没有像本文这么全面。   
我在下面列举了一些资料和相关的文章，你感兴趣的话可以深入阅读。

* [神经网络常见问答](ftp://ftp.sas.com/pub/neural/FAQ.html)
* [如何用网格搜索法求解深度学习模型的超参数](http://machinelearningmastery.com/grid-search-hyperparameters-deep-learning-models-python-keras/)
* [深度神经网络必知的技巧](http://lamda.nju.edu.cn/weixs/project/CNNTricks/CNNTricks.html)
* [如何提升深度神经网络的验证准确率？](http://stackoverflow.com/questions/37020754/how-to-increase-validation-accuracy-with-deep-neural-net)