概述

如何提高深度学习（和机器学习）的性能

* 选取一个方向
  + 数据
  + 算法
  + 调参
  + 模型集成
* 在某一方向里选取一种方法
* 在选取的方法中选取一件事情去尝试
* 比较结果，如果性能有提升，则保留
* 不断重复
* **从上往下，效果提升越来越小**

原文地址：[How To Improve Deep Learning Performance](http://machinelearningmastery.com/improve-deep-learning-performance/)

中文翻译：<http://blog.csdn.net/han_xiaoyang/article/details/52654879>

### 可参考资源：

[Neural Network FAQ](ftp://ftp.sas.com/pub/neural/FAQ.html)

[How to Grid Search Hyperparameters for Deep Learning Models in Python With Keras](http://machinelearningmastery.com/grid-search-hyperparameters-deep-learning-models-python-keras/)

[Must Know Tips/Tricks in Deep Neural Networks](http://lamda.nju.edu.cn/weixs/project/CNNTricks/CNNTricks.html)

[How to increase validation accuracy with deep neural net?](http://stackoverflow.com/questions/37020754/how-to-increase-validation-accuracy-with-deep-neural-net)

参数调优

1. 通过数据提升性能

(1) 获取更多数据

* 获得最好的数据

[Datasets Over Algorithms](https://www.edge.org/response-detail/26587)

1. 创造更多数据

* 如果你的数据是数值型向量，可以随机构造已有向量的修改版本
* 如果你的数据是图片，可以随机构造已有图片的修改版本(平移、截取、旋转等)
* 如果你的数据是文本，类似的操作……

[Image Augmentation for Deep Learning With Keras](http://machinelearningmastery.com/image-augmentation-deep-learning-keras/)

[What is jitter? (Training with noise)](ftp://ftp.sas.com/pub/neural/FAQ3.html" \l "A_jitter)

1. 重放缩你的数据

* 归一化到0和1的区间
* 重放缩到－1和1的区间
* 标准化（标准化数据使之成为零均值，单位标准差）

[Should I standardize the input variables (column vectors)?](ftp://ftp.sas.com/pub/neural/FAQ2.html" \l "A_std)

[How To Prepare Your Data For Machine Learning in Python with Scikit-Learn](http://machinelearningmastery.com/prepare-data-machine-learning-python-scikit-learn/)

1. 转换你的数据

**猜测每一列数据的单变量分布：**

* 列数据看起来像偏斜的高斯分布吗？考虑用Box-Cox变换调整偏态
* 列数据看起来像指数分布吗？考虑用对数变换
* 列数据看起来有一些特征，但是它们被一些明显的东西遮盖了，尝试取平方或者开平方根来转换数据
* 你能离散化一个特征或者以某种方式组合特征，来更好地突出一些特征吗？

**依靠你的直觉，尝试以下方法：**

* 你能利用类似PCA的投影方法来预处理数据吗？
* 你能综合多维特征至一个单一数值(特征)吗？
* 你能用一个新的布尔标签去发现问题中存在一些有趣的方面吗？
* 你能用其他方法探索出目前场景下的其他特殊结构吗？

[How to Define Your Machine Learning Problem](http://machinelearningmastery.com/how-to-define-your-machine-learning-problem/)

[Discover Feature Engineering, How to Engineer Features and How to Get Good at It](http://machinelearningmastery.com/discover-feature-engineering-how-to-engineer-features-and-how-to-get-good-at-it/)

[How To Prepare Your Data For Machine Learning in Python with Scikit-Learn](http://machinelearningmastery.com/prepare-data-machine-learning-python-scikit-learn/)

1. 特征选取

* 或许你利用更少的特征就能达到同等甚至更好的性能。而且，这将使模型变得更快！
* 或许所有的特征选择方法都剔除了同样的特征子集。很好，这些方法在没用的特征上达成了一致
* 或许筛选过后的特征子集，能带给特征工程的新思路

[An Introduction to Feature Selection](http://machinelearningmastery.com/an-introduction-to-feature-selection/)

[Feature Selection For Machine Learning in Python](http://machinelearningmastery.com/feature-selection-machine-learning-python/)

(6) 重架构你的问题

* 看看能够在一个时间窗（时间周期）内对已有的特征/数据做一个合并
* 或许你的分类问题可以成为一个回归问题(有时候是回归到分类)
* 或许你的二元输出可以变成softmax输出
* 或许你可以转而对子问题进行建模

[How to Define Your Machine Learning Problem](http://machinelearningmastery.com/how-to-define-your-machine-learning-problem/)

2. 通过算法提升性能

1. 对算法进行抽样调查

**原则：Maybe your chosen algorithms is not the best for your problem**

**[no free lunch theorem](https://en.wikipedia.org/wiki/No_free_lunch_theorem)**

* 首先尝试评估一些线性方法，例如逻辑回归（logistic regression）和线性判别分析（linear discriminate analysis）
* 评估一些树类模型，例如CART， 随机森林（Random Forest）和Gradient Boosting
* 评估一些实例方法，例如支持向量机（SVM）和K-近邻（KNN）
* 评估一些其他的神经网络方法，例如LVQ, MLP, CNN, LSTM, hybrids等

[A Data-Driven Approach to Machine Learning](http://machinelearningmastery.com/a-data-driven-approach-to-machine-learning/)

[Why you should be Spot-Checking Algorithms on your Machine Learning Problems](http://machinelearningmastery.com/why-you-should-be-spot-checking-algorithms-on-your-machine-learning-problems/)

[Spot-Check Classification Machine Learning Algorithms in Python with scikit-learn](http://machinelearningmastery.com/spot-check-classification-machine-learning-algorithms-python-scikit-learn/)

1. 借鉴已有文献

[How to Research a Machine Learning Algorithm](http://machinelearningmastery.com/how-to-research-a-machine-learning-algorithm/)

[Google Scholar](http://scholar.google.com/)

1. 重采样方法

**深度学习模型在训练阶段非常缓慢，无法用一些常用的方法，例如k层交叉验证，去估计模型的性能：**

* 或许你在使用一个简单的训练集／测试集分割（保持训练集和测试集分布一致），这是常规套路。如果是这样，你需要确保这种分割针对你的问题具有代表性。单变量统计和可视化是一个好的开始
* 或许你能利用硬件来加速估计的过程。例如，如果你有集群或者AWS云端服务（Amazon Web Services）账号，你可以并行地训练n个模型，然后获取结果的均值和标准差来得到更鲁棒的估计
* 或许你可以利用hold-out验证方法来了解模型在训练后的性能（这在早停法early stopping中很有用，后面会讲到）
* 或许你可以先隐藏一个完全没用过的验证集，等到你已经完成模型选择之后再使用它

**让数据集变得更小，以及使用更强的重采样方法：**

* 有些情况下你会发现在训练集的一部分样本上训练得到的模型的性能，和在整个数据集上训练得到的模型的性能有很强的相关性。也许你可以先在小数据集上完成模型选择和参数调优，然后再将最终的方法扩展到全部数据集上
* 或许你可以用某些方式限制数据集，只取一部分样本，然后用它进行全部的建模过程（如何对数据进行采样，使用采集的样本作为全部训练集）
* 先创造一些和原始数据很相似的数据（包括特征空间和数据分布，创造的数据应力求简单，并和原始数据任务相同），然后用创造的数据先训练一个基本模型，然后将得到的模型应用到原始数据

[Evaluate the Performance Of Deep Learning Models in Keras](http://machinelearningmastery.com/evaluate-performance-deep-learning-models-keras/)

[Evaluate the Performance of Machine Learning Algorithms in Python using Resampling](http://machinelearningmastery.com/evaluate-performance-machine-learning-algorithms-python-using-resampling/)

1. 通过算法调参提升性能

**关于超参数优化请参阅博文：**

[How to Grid Search Hyperparameters for Deep Learning Models in Python With Keras](http://machinelearningmastery.com/grid-search-hyperparameters-deep-learning-models-python-keras/)

1. 诊断（Diagnostics）

**你的模型是过拟合还是欠拟合？永远牢记这个问题。永远！**

**模型总是会遇到过拟合或者欠拟合，只是程度不同罢了。一个快速了解模型学习行为的方法是，在每个周期，评估模型在训练集和验证集上的表现：**

* 如果训练集上的模型总是优于验证集上的模型，你可能遇到了过拟合，你可以使用诸如正则化的方法
* 如果训练集和验证集上的模型都很差，你可能遇到了欠拟合，你可以提升网络的容量，以及训练更多或者更久
* 如果有一个拐点存在，在那之后训练集上的模型开始优于验证集上的模型，你可能需要使用早停法

**另一个有用的诊断是网络模型判定对和判定错的观察值：**

* 对于难以训练的样本，或许你需要更多的数据
* 或许你应该剔除训练集中易于建模的多余的样本
* 也许可以尝试对训练集划分不同的区域，在特定区域中用更专长的模型

[Display Deep Learning Model Training History in Keras](http://machinelearningmastery.com/display-deep-learning-model-training-history-in-keras/)

[Overfitting and Underfitting With Machine Learning Algorithms](http://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/)

1. 权重初始化（Weight Initialization）

**经验法则通常是：用小的随机数进行初始化**

* 尝试所有不同的初始化方法，考察是否有一种方法在其他情况不变的情况下(效果)更优
* 尝试用无监督的方法，例如自动编码（autoencoder），来进行预先学习
* 尝试使用一个已经存在的模型，只是针对你的问题重新训练输入层和输出层（迁移学习transfer learning）

**注意一点：改变权重初始化方法和激活函数，甚至优化函数/损失函数紧密相关**

[Initialization of deep networks](http://deepdish.io/2015/02/24/network-initialization/)

1. 学习速率（Learning Rate）

* 实验很大和很小的学习率
* 格点搜索文献里常见的学习速率值，考察你能学习多深的网络
* 尝试随周期递减的学习率
* 尝试经过固定周期数后按比例减小的学习率
* 尝试增加一个动量项（momentum term），然后对学习速率和动量同时进行格点搜索

[Using Learning Rate Schedules for Deep Learning Models in Python with Keras](http://machinelearningmastery.com/using-learning-rate-schedules-deep-learning-models-python-keras/)

[What learning rate should be used for backprop?](ftp://ftp.sas.com/pub/neural/FAQ2.html" \l "A_learn_rate)

1. 激活函数

* 尝试全部三种激活函数（ReLU，Sigmoid，Tanh等），并且重缩放你的数据以满足激活函数的边界：

[Why use activation functions?](ftp://ftp.sas.com/pub/neural/FAQ2.html" \l "A_act)

1. 网络拓扑（Network Topology）

* 尝试一个隐藏层和许多神经元（广度模型）
* 尝试一个深的网络，但是每层只有很少的神经元（深度模型）
* 尝试上述两种方法的组合
* 借鉴研究问题与你的类似的论文里面的结构
* 尝试拓扑模式（扇出fan out然后扇入fan in）和书籍论文里的经验法则

[How many hidden layers should I use?](ftp://ftp.sas.com/pub/neural/FAQ3.html" \l "A_hl)

[How many hidden units should I use?](ftp://ftp.sas.com/pub/neural/FAQ3.html" \l "A_hu)

1. 批次和周期（Batches and Epochs）

* 尝试选取与训练数据同大小的batch size，但注意一下内存（批次学习batch learning）
* 尝试选取1作为batch size（在线学习online learning）
* 尝试用格点搜索不同的小的batch size（4，8，16，32，…）
* 分别尝试训练少量周期和大量周期

[What are batch, incremental, on-line … learning?](ftp://ftp.sas.com/pub/neural/FAQ2.html" \l "A_styles)

[Intuitively, how does mini-batch size affect the performance of (stochastic) gradient descent?](https://www.quora.com/Intuitively-how-does-mini-batch-size-affect-the-performance-of-stochastic-gradient-descent)

1. 正则化

**可以从dropout方法开始：**

* 格点搜索不同的丢失比例
* 分别在输入，隐藏层和输出层中试验dropout方法
* dropout方法也有一些拓展，比如你也可以尝试[drop connect](http://cs.nyu.edu/~wanli/dropc/)方法

**可以尝试其他更传统的神经网络正则化方法：**

* 权重衰减（Weight decay）去惩罚大的权重
* 激活约束（Activation constraint）去惩罚大的激活值

[Dropout Regularization in Deep Learning Models With Keras](http://machinelearningmastery.com/dropout-regularization-deep-learning-models-keras/)

[What is Weight Decay?](ftp://ftp.sas.com/pub/neural/FAQ3.html" \l "A_decay)

1. 优化和损失

[An overview of gradient descent optimization algorithms](http://sebastianruder.com/optimizing-gradient-descent/)

[What are conjugate gradients, Levenberg-Marquardt, etc.?](ftp://ftp.sas.com/pub/neural/FAQ2.html" \l "A_numanal)

[On Optimization Methods for Deep Learning, 2011 PDF](http://ai.stanford.edu/~ang/papers/icml11-OptimizationForDeepLearning.pdf)

1. 早停法

[How to Check-Point Deep Learning Models in Keras](http://machinelearningmastery.com/check-point-deep-learning-models-keras/)

[What is early stopping?](ftp://ftp.sas.com/pub/neural/FAQ3.html" \l "A_stop)

4. 通过模型集成提升性能

(1) 组合模型

[Ensemble Machine Learning Algorithms in Python with scikit-learn](http://machinelearningmastery.com/ensemble-machine-learning-algorithms-python-scikit-learn/)

[How to Improve Machine Learning Results](http://machinelearningmastery.com/how-to-improve-machine-learning-results/)

1. 组合视角
2. 堆叠（Stacking）

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

有关神经网络几点：

1) 神经网络结构

第一，神经网络理论上可以映射任何关系，表达能力超强（[universal function approximation](https://en.wikipedia.org/wiki/Universal_approximation_theorem)）

第二，对于自己的数据，要根据实际情况选择一个合适的网络结构

第三，对不同结构神经网络，使用之前要有一个预期的效果

2) 神经网络最常见的三个问题

● 梯度弥散：梯度可视化查看梯度变化情况（可以从AutoEncoder，ResNet和Inception等角度考虑）

● 局部极小值：权重初始化和优化算法角度解决

● 过拟合（或欠拟合）：参考上述3

3) 参数调整

第一，首先要知道是**过拟合还是欠拟合**

第二，参考上述3