当你在应用机器学习时你应该想什么

张皓

zhangh0214@gmail.com

如今，机器学习变得十分诱人，它已在网页搜索，商品推荐，垃圾邮件检测，语音识别，图像识别，自然语言处理等诸多领域发挥重要作用．和以往我们显式地通过编程告诉计算机如何进行计算不同，机器学习是一种数据驱动方法(data-driven approach)．然而，有时候机器学习像是一种"魔术"，即使是给定相同的数据，一位机器学习领域专家和一位新手训练得到的结果可能相去甚远．本文简要讨论了实际应用机器学习时九个需要注意的重要方面．

我该选什么学习算法?

这可能是你面对一个具体应用场景想到的第一个问题．你可能会想"机器学习里面这么多算法，究竟哪个算法最好"．很"不幸"的是，没有免费午餐定理(No Free Lunch Theorem)告诉我们对于任意两个学习算法，如果其中一个在某些问题上比另一个好，那么一定存在一些问题另一个学习算法更好．因此如果考虑所有可能问题，所有算法都一样好．

"好吧"，你可能会接着想，"没有免费午餐定理假定所有问题都有相同机会发生，但我只关心对我现在面对的问题，哪个算法更好"．又很"不幸"的是，有可能你把机器学习里面所谓"十大算法"都试了一遍，然后感觉机器学习"这东西根本没用，这些算法我都试了，没一个效果好的"．前一段时间"约战比武"的话题很热，其实机器学习和练武术有点像，把太极二十四式朝对方打一遍结果对方应声倒下这是不可能的．机器学习算法是有限的，而现实应用问题是无限的，以有限的套路应对无限的变化，一定是会存在有的问题你无法用现有的算法解决的，岂有不败之理?

因此，该选什么学习算法要和你要解决的具体问题相结合．不同的学习算法有不同的归纳偏好(inductive bias)，你使用的算法的归纳偏好是否适应要解决的具体问题直接决定了学得模型的性能，有时你可能需要改造现有算法以应对你要解决的现实问题．

我该选什么目标函数?

目标函数用于刻画什么样的模型是好的，和选择学习算法一样，选什么目标函数也要和具体问题相结合．大部分的教材和文献重点在呈现算法，对目标函数和优化方法的选择通常使用缺省值．以分类问题为例，我们缺省地优化分类均等代价(cost)下的错误率．但是在你所要解决的问题中，问问自己这些问题:

- 你真正关心的是错误率吗(比如还有查准率(precision)，查全率(recall)等其他指标)?
- 数据中有没有类别不平衡(class-imbalance)的现象(比如信用卡欺诈检测中，欺诈用户数远小于正常用户数)?
- 不同类型错误所造成的损失是一样的吗(比如医疗诊断中，错误地把患者诊断为健康人与错误地把健康人诊断为患者的代价截然不同)?
- 还有没有其他类型的代价(除了误分类代价外，还有测试代价，标记代价，特征(feature)代价等)?

除此之外还有其他的一些问题．这些问题存不存在，要不要考虑，该怎么考虑都是和你要处理的实际问题有关，最终体现在你的目标函数之中．

我该选什么优化方法?

"我是谁? 我从哪里来? 我往哪里去?"是哲学里避不开的三个问题, 而"选什么算法? 选什么目标函数? 选什么优化方法?"是机器学习里经常遇到的三个问题, 这三者的组合就构成了一个机器学习解决方案基本框架. 通常, 我们使用现有的数值优化方法对目标函数进行优化, 比如梯度下降(gradient descent), 牛顿法等.

但是当目标函数非凸(non-convex), 有多个局部极小(local minima)时, 选什么优化方法会对结果产生直接影响. 比如深度学习中通常目标函数有多个局部极小, 使用不同的参数初始化方法(如高斯(Gaussian)随机初始化, Xavier 初始化, MSRA 初始化等), 不同的优化方法(如 SGD, 带动量(momentum)的 SGD, RMSProp, ADAM 等), 或不同的学习率(learning rate)策略等, 都会对结果有很大影响.

另一方面, 即使目标函数是凸函数, 设计合适的优化方法可能会使你的训练过程有质的飞跃. 比如 SVM 的优化是一个二次规划(quadratic programming)问题, 可以通过调用现成的 QP 软件包进行优化. 然而, 这个二次规划问题的大小和训练样本数成线性关系, 当数据量很大时将导致巨大的开销. 为了避开这个障碍, 人们根据 SVM 的特点设计出了 SMO (sequential minimal optimization)这样的高效优化方法.

不要偷看测试数据

我们希望学习器能从训练数据中尽可能学出适用于所有潜在样本(sample)的普遍规律, 从而能很好的泛化(generalize)到新样本. 为了评估模型的泛化能力, 我们通常收集一部分数据作为测试集(testing set)计算测试误差用以作为泛化误差的近似. 为了能得到较准的近似, 我们不能在训练阶段以任何方式偷看测试数据.

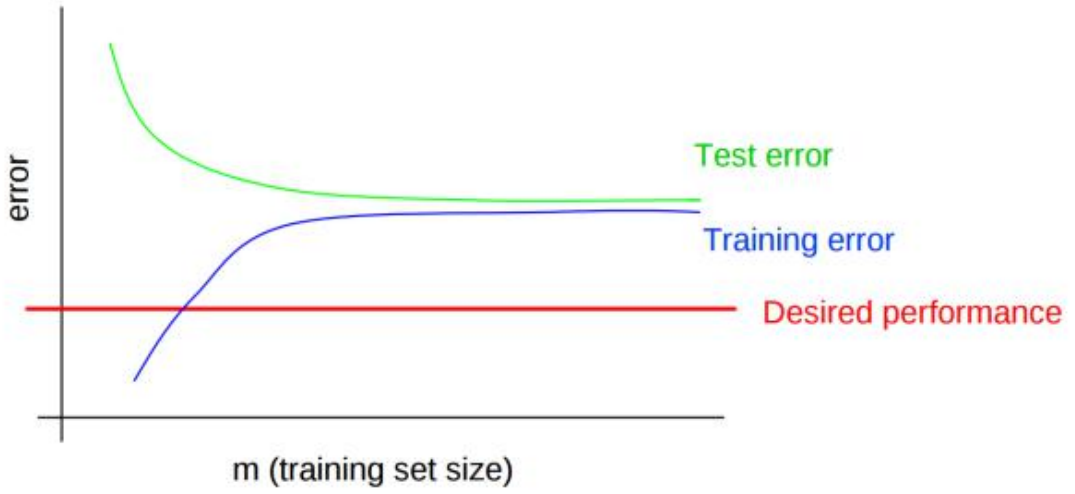
一个常见错误做法是用测试数据调模型的参数. 这相当于老师在考试前向学生透露考试原题, 这虽然可能会使学生在这次考试中拿到高分, 但这不能有效反应学生是否对这门课学的很好, 获得了对所学知识举一反三的能力, 得到的将是过于乐观的估计结果. 调参的正确做法是从训练数据中再划分出一部分作为验证集(validation set), 用训练集(training set)剩下的数据做训练, 用验证集调参.

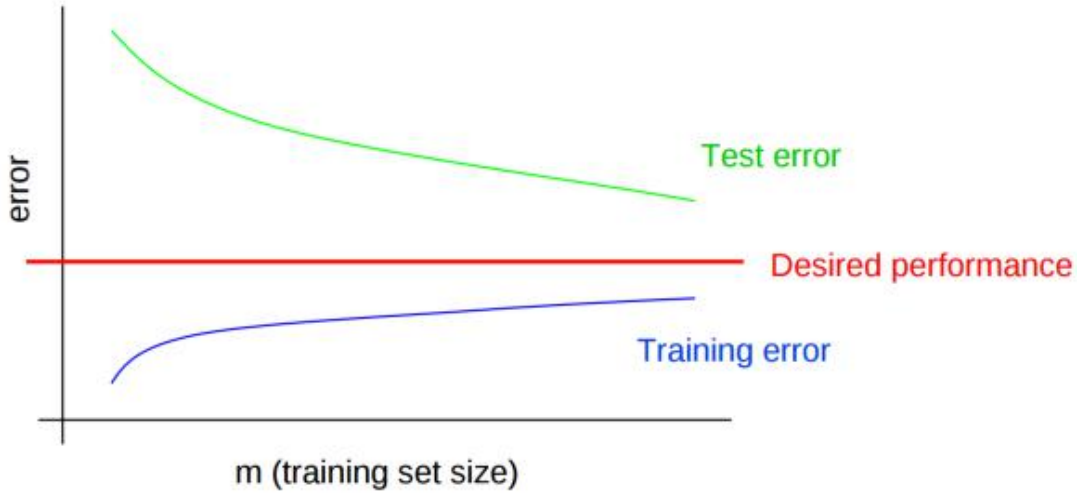
另一个常见错误是用测试数据参加训练数据预处理(data pre-processing). 通常, 数据在输入给模型之前会经过一些预处理, 比如减去各维的均值, 除以各维的方差等. 如果这个均值和方差是由训练集和测试集数据一起计算得到的, 这相当于间接偷看了测试数据. 这相当于老师在考前向学生划重点, 虽然比直接透露原题好一些, 由于学生知道了考试范围(即测试数据分布), 也会使我们得到过于乐观的估计. 正确做法是只从训练数据中计算预处理所用的统计量, 将这个量应用于测试集.

欠拟合/过拟合: 认准你的敌人

欠拟合(underfitting)通常是由于学习器的学习能力不足, 过拟合(overfitting)通常是由于学习能力过于强大. 两者都会影响模型的泛化能力, 但是解决这两个问题的方法迥然不同. 解决欠拟合可以通过使用更复杂的模型, 增加训练轮数等. 缓解过拟合可以通过使用简单模型, 正则化(regularization), 训练早停(early-stopping)等. 欠拟合比较容易解决, 而过拟合是无法彻底避免的, 我们只能缓和, 减小其风险.

因此, 问题的关键在于认准你当前的敌人是欠拟合还是过拟合. 判断欠拟合或过拟合最简单直接的方法是画出学习曲线(learning curve). 欠拟合的表现是: 训练误差很低(甚至为 0), 而测试误差很高, 两者有很大的差距. 而过拟合的表现是: 训练误差和测试误差很接近, 但都很高, 下图是两个例子, 你能看出来哪个处于过拟合, 哪个处于欠拟合吗?





机器智能+人类智能

经过前面这几部分你可能已经意识到了，机器学习不是把数据扔给机器然后自己可以撒手不管．机器学习不是"空手套白狼"，如果我们对问题/数据认识的越深刻，我们越容易找到归纳假设与之匹配的学习算法，学习算法也越容易学到数据背后的潜在规律．

数据中特征的好坏直接影响学习算法的性能．如果数据之间相互独立并且与数据的标记有很好的相关性，学习过程将相对容易．但很多情况下，你手中数据的原始特征并没有这么好的性质，特征和标记之间是一个非常复杂的映射关系．这时候机器智能需要人类智能的配合，我们需要从原始数据中构造合适的特征．这个过程叫做特征工程(feature engineering)，这通常需要领域知识和你对这个问题的认识．

你可能会想"既然机器学习可以学到从数据的表示到标记的映射，那么我们能不能让机器自动地从数据的原始特征中学习到合适的表示呢"，表示学习(representation learning)就专门研究这方面的内容．深度学习的最大优点就在于其表示学习能力，通过很多层的堆叠，深度神经网络可以对输入数据进行逐层加工，从而把初始的，与输出目标关系不密切的表示转化为与输出目标关系密切的表示．这样将低层表示转化为高层表示后，用"简单模型"即可完成复杂的学习任务．因此，深度学习才能在计算机视觉，自然语言处理，语音识别这样数据的原始表示和数据的合适表示相差很大的任务上大放异彩．

高维"灾难"

经过上一节你可能会想"既然特征工程这么重要，那我就把想到的所有的特征组合都作为数据的特征不就好了吗". 这么做的结果会使特征维数增加，一方面会增加存储和计算开销，更重要的是，它会招来机器学习担忧的另一头猛兽：维数灾难(curse of dimensionality).

由于你能拿到手中的训练数据是有限的，当维数增加时，输入空间(input space)的大小随维数指数级增加，训练数据占整个数据空间的比例将急剧下降，这将导致模型的泛化变得更困难. 在高维空间中，样本数据将变得十分稀疏，许多的相似性度量在高维都会失效. 比如下图中，最左边的是原图，右边三张图看上去差别很大，但和原图都有着相同的欧氏距离.

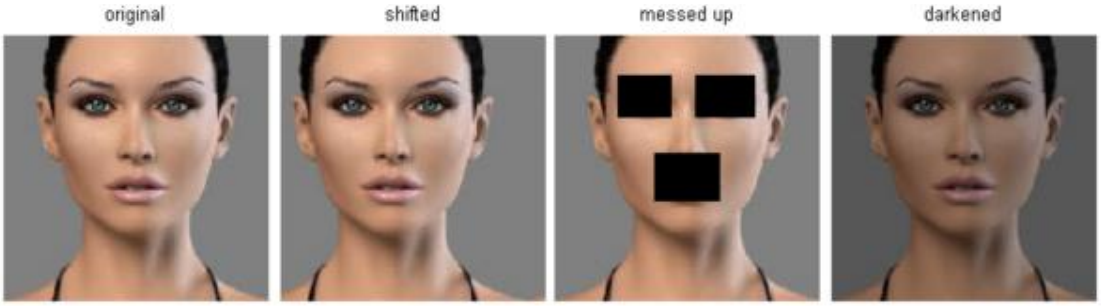
解决维数灾难的一个重要途径是降维(dimension reduction)，即通过一些手段将原始高维空间数据转变为一个低维子空间，在这个子空间中样本密度大幅提高，距离计算也更容易. 特征选择(feature selection)和低维投影(如 PCA)是用来处理高维数据的两大主流技术.

数据！数据！

看了前面各部分的讨论可能你已经有点受不了了："机器学习没有通式通法，要根据任务具体情况具体分析；要小心翼翼，不能偷看测试数据，要设法避免过拟合和欠拟合；特征工程很重要，但是特征又不能太多... 哦天呐！这太麻烦了！有没有什么能提升性能直截了当的方法啊!". 有！那就是收集更多的数据. 通常情况下，你有很多的数据但是学习算法一般和你设计出了很好的学习算法但手中数据不足相比，前者效果会更好. 这是因为收集更多的数据是缓解过拟合最直接有效的方法.

收集更多的数据通常有下面两种办法：一种方法是收集更多的真实数据，这是最直接有效的方法. 但有时收集数据很昂贵，或者我们拿到的是第二手数据，数据就这么多. 这就需要另一个方法：从现有数据中生成更多数据，用生成的"伪造"数据当作更多的真实数据进行训练. 这个过程叫做数据扩充(data augmentation). 以图像数据做分类任务为例，把图像水平翻转，移动一定位置，旋转一定角度，或做一点色彩变化等，这些操作通常都不会影响这幅图像对应的标记. 并且你可以尝试这些操作的组合，理论上讲，你可以通过这些组合得到无穷多的训练样本.

在收集/生成数据过程中一个要注意的关键问题是：确保你的数据服从同一分布. 这比如说老师教了学生一学期的英语结果期末考试却是考俄语，虽然英语和俄语在语言学上有一定的相似性，但是这给学习过程增加了很多困难. 迁移学习(transfer learning)旨在研究数据分布变化情况下的学习算法的泛化能力，但如果你的目的不是做迁移学习方面的科学研究，建议你确保你的训练和测试数据服从同一分布，这会使问题简单许多.



集成学习：以柔克刚

面对大多数的任务，集成学习应当是你的必备招式．集成学习的目的在于结合多个弱学习器的优点构成一个强学习器，"三个臭皮匠，顶个诸葛亮"讲的就是这个道理．在各种机器学习/数据挖掘竞赛中，那些取得冠军的队伍通常会使用集成学习，有的甚至用成百上千个个体学习器来做集成．通常我们面对一个实际问题时会训练得到多个学习器，然后使用模型选择(model selection)方法来选择一个最优的学习器．而集成学习则相当于把这些学习器都利用起来，因此，集成学习的实际计算开销并不比使用单一学习器大很多．

目前的集成学习方法大致可以分为两大类，以 Boosting 为代表的集成学习方法中每个个体学习器相互有强依赖关系，必须序列化生成；以 Bagging 为代表的集成学习方法中每个个体学习器不存在强依赖关系，可以并行化生成．而从偏差-方差分解(bias-variance decomposition)的角度看，Boosting 主要关注降低偏差，而 Bagging 主要关注降低方差．欠拟合/过拟合这两个敌人是不是又一次出现在你眼前了呢?

参考文献

[1]. Domingos, Pedro. "A few useful things to know about machine learning." Communications of the ACM 55.10 (2012): 78-87.

[2]. Glorot, Xavier, and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks." Aistats. Vol. 9. 2010.

[3]. He, Kaiming, et al. "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification." Proceedings of the IEEE international conference on computer vision. 2015.

[4]. Kingma, Diederik, and Jimmy Ba. "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980 (2014).

[5]. Li, Fei-Fei, Andrej Karpathy, and Justin Johnson. CS231n: Convolutional Neural Networks for Visual Recognition. Stanford. 2016.

[6]. Lin, Hsuan-Tien. Machine Learning Foundations. National Taiwan University.

[7]. Lin, Hsuan-Tien. Machine Learning Techniques. National Taiwan University.

[8]. Murphy, Kevin P. Machine learning: a probabilistic perspective. MIT press, 2012.

[9]. Ng, Andrew. Machine learning yearning. Draft, 2016.

[10]. Ng, Andrew. CS229: Machine Learning. Stanford.

[11]. Platt, John. "Sequential minimal optimization: A fast algorithm for training support vector machines." (1998).

[12]. Tieleman, Tijmen, and Geoffrey Hinton. "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude." COURSERA: Neural networks for machine learning 4.2 (2012).

[13]. Wei, Xiu-Shen. Must Know Tips/Tricks in Deep Neural Networks. http://lamda.nju.edu.cn/weixs/project/CNNTricks/CNNTricks.html.

[14]. Wolpert, David H. "The lack of a priori distinctions between learning algorithms." Neural computation 8.7 (1996): 1341-1390.

[15]. Wolpert, David H., and William G. Macready. No free lunch theorems for search. Vol. 10. Technical Report SFI-TR-95-02-010, Santa Fe Institute, 1995.

[16]. Zhou, Zhi-Hua. Ensemble methods: foundations and algorithms. CRC press, 2012.

[17]. Zhou, Zhi-Hua. "Learnware: on the future of machine learning." Frontiers of Computer Science 10.4 (2016): 589-590.
[18]. 周志华 著. 机器学习, 北京: 清华大学出版社, 2016 年 1 月.