# [理解机器学习算法的一点心得](http://blog.csdn.net/dark_scope/article/details/25485893)

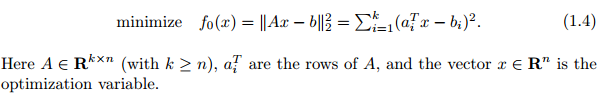
     从Andrew ng的公开课开始，机器学习的算法我接触到的也越来越多，我觉得机器学习算法和传统算法的最大不同就是:不会要求一个问题被100%求解，也就意味着不会有完美的解法，这也是著名的“Essentially, all models are wrong, but some are useful.”所表达的意思。正因为如此，机器学习算法往往不会有一个固定的算法流程，取而代之的把问题转化为最优化的问题，无论是ML(maximum likelihood),MAP(Maximum a Posterior)和EM(Expectation Maximization)，都是这样的。

     然后用不同的方法来优化这个问题，得到尽量好的结果，给人的感觉就像是一个黑盒，实际使用中需要不断地调参实验，但倘若你能理解好算法，至少能让这个盒子透明一点，这也是机器学习算法确实需要使用者去理解算法的原因，举个例子:传统算法比如一些高效的数据结构，我只需要知道一些接口就可以使用，不需要进行太多的理解，了解传统算法更多的是理解算法的思想，开阔思路，增强能力；而机器学习算法，你即使知道接口，也至少要调一些参数来达到实际使用的目的。

     这样一来，阅读各类书籍和paper也就在所难免了，甚至去阅读代码以至于实现加深理解，对于实际使用还是有很大的好处的，因为不是100%求解问题，所以面对不同的应用场景，想要达到最好的效果都需要加以变化。本文记录了一点自己学习的心得，私以为只要你能对算法有一种说得通的解释，就是OK的，不一定要去深挖其数学上的证明(表示完全挖不动啊…………>\_<)

**O.  目的**

     之前说到机器学习算法常常把问题转化为一个最优化问题，理解这个最优化问题的目的能很好地帮助我们理解算法，比如最简单的最小二乘法(Least-squares)：



     (这里的x是参数，和一些机器学习的常用表示里面有出入)

     好多机器学习入门书都是从最小二乘开始引入的，其实这是线性代数(还是概率统计?囧rz)的课本内容嘛。

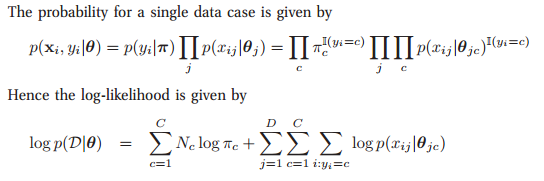
     理解上式应该非常简单呐，括号内的就是目标值和与测试的差，取平方之后抹掉正负，而该式是要最小化这个东西，那么这个优化问题的“目的”就是最小化预测函数在训练集上的误差。

     当然这是最简单的一个例子了，我们接着看朴素贝叶斯分类器的优化目标:

http://img.blog.csdn.net/20140510163300593

     (这里xi,yi是训练集，π和θ是参数)

      无论他后面怎么变化，用了什么优化方式，该算法的目的就是在训练集上最大化这个东西，只不过对于朴素贝叶斯来说，它加入了非常强的假设来简化问题而已。



      然后朴素贝叶斯用了一系列的参数来描述这个需要优化的概率值，为了达到目的还是用了log来变换一下，但对于你来说，只需要记住他的“目的”，就可以很容易地理解算法了。

**一.  趋势**

        接下来要讲的是"趋势"，广义上来说和目的是一回事，但算法的优化目标的一些部分是与算法总体目的相对分割的，比如一些正则化(regularization)的项，这些项对于算法实际使用效果往往有着重大影响，但并不绝对大的方向(目的)，所以“趋势”我们单独开 一章来讲。

        我们还是从最简单的 L2-norm regularization 来开启这个话题吧，把这个项加到最小二乘后面:

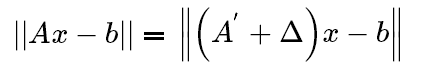
http://img.blog.csdn.net/20140510165142296

        虽然也能把该式表示为标准的最小二乘结构，但对理解算法并无帮助，我们不这样做。

        可以看到该式的第二项是想要参数的平方和，而整个是Minimize的，所以直观来说就是想要学到的参数的绝对值越小越好，这就是我理解的“趋势”

        可是为什么让参数平方和越小能防止over-fitting呢？这里就有很多解释了，比如加入该项是对数据的原始分布加了个高斯分布作为先验(有证明的貌似)，但像我这种数学渣渣还是走intuition的方向吧，这样理解：(这是Convex Optimizition课上提到的，我也不知道是否是对的，但能够说通)

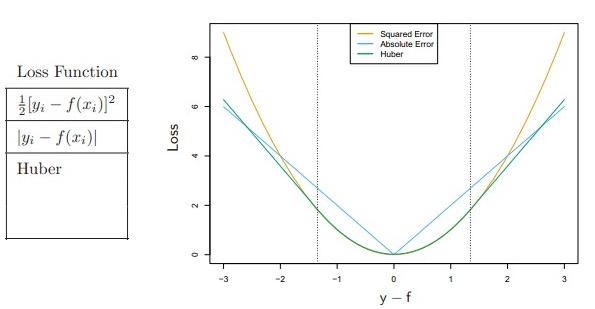
        我们得到的训练数据是有测量误差的，记为delta,参数为x，要优化的为:||Ax-y||，其实是||(A'+delta)x-y||=||A'x-y+delta\*x||：



        所以参数x的值越小，误差delta对于模型的影响就越小，所以能增加模型的泛化能力。

**二.  还是趋势**

        再写上面一章就略长了，新开一段…………还是讲趋势，对于最小二乘，其实是Loss function一种，也就是我们想要最小化的东西，除此之外还有其它的一些Loss function，其选择同样也会影响算法的效果。(这里的xi和yi又是训练集了，不是参数，略乱，见谅)



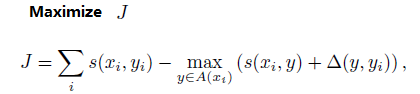
         上面的[Huber](http://en.wikipedia.org/wiki/Huber_Loss_Function" \t "_blank)是在一个阀值内是二次的，阀值外则是线性的

         这能体现什么趋势呢？可以明显看到，对于偏差很大的case来说（|y-f|>2），平方项【1】的要比绝对值【2】的惩罚大不少，这意味着【1】对于极端outlier的容忍能力更差，离太远了简直是没法承受的，对算法带来的影响就是要去满足这个outlier，从而带来一些问题。而在一定阀值以内的时候，平方项【1】的惩罚却比绝对值【2】还要小。综合来看，相对于绝对值，平方项的趋势就是去满足outlier,把绝大多数训练数据的loss降低到够小的范围即可。(略绕，但应该不难理解)

         Huber的优点就是既对outlier有容忍力(大于阀值其增长是线性的)，又不至于全是线性增长，对误差重要程度没有太大区分(小于阀值是二次的)，所以Boyd在公开课上就说:对于绝大多数使用二次Loss function的地方来说，换用Huber基本上都会有更好的效果

**三.  一个复杂点的例子**

        前段时间组内读书会有大大分享了一片论文，开始读着无比顺畅，但就是到了其中一步无法理解，考虑了很久，就用我的“趋势”分析法^\_^理解了下来，这里就不给上下文了，论文叫<Learning Continuous Phrase Representations and Syntactic Parsing with Recursive Neural Networks>，有兴趣可以去看，我现在单把那一个公式提出来分析其目的(趋势)



        其中的s()是表示传入参数的一个得分值，A( xi )表示对于xi来说所有可能的 y 结构，Δ(y,yi) 是对结构 y 和 yi 相异程度 的惩罚项目，Δ( yi , yi)=0

        这个式子很难理解就在于maximize里还减去一个max，而且max里面还不是norm的结构，乍一看是和以前见过的有巨大差异

        但仔细思考其实可以发现，A( xi )之中是有 yi 的，即训练数据。所以 max() 那一项最小的取值就是 s( xi,yi )，不会比这个小，那这个式子的目的是什么呢？

        作者坚定地认为训练数据就是最好最正确的，其得分就该是最高的，所以一旦max项里面选出来的是比 s( xi,yi )大的，就对其进行惩罚，最后该式的目的就是在所有 xi 可能对应的结构 y 中，训练数据 yi 应该是最好的。与此同时加入Δ项，是为了使与 yi 结构更接近的 y 得分更高( 这个这么理解：算法给所有结构加了一个上限在那，超过了就砍头，那么Δ(y,yi)值越小，剩下的可喘息的部分就越大，也就是得分就越高)

        这个式子和经验里看到的有很大差异，但通过分析他的目的和趋势，就可以较好地理解算法和里面一些参数的意义，从而到达我们学习算法的目的

**四.  尾巴**

        本来打算磨好久的，居然几个小时就搞定了，真是顺利啊

        这是我理解算法的一点小心得，可能会有错的地方，求指正啊~~~~~

       【ref】:

        【1】.《Convex Optimization 》（Byod）

        【2】.《Machine Learning - A Probabilistic Perspective》

        【3】.《The Elements of Statistical Learning》

        【4】:Learning Continuous Phrase Representations and Syntactic Parsing with Recursive Neural Networks