数值优化：理解L-BFGS

V1.0

翻译：tianshan

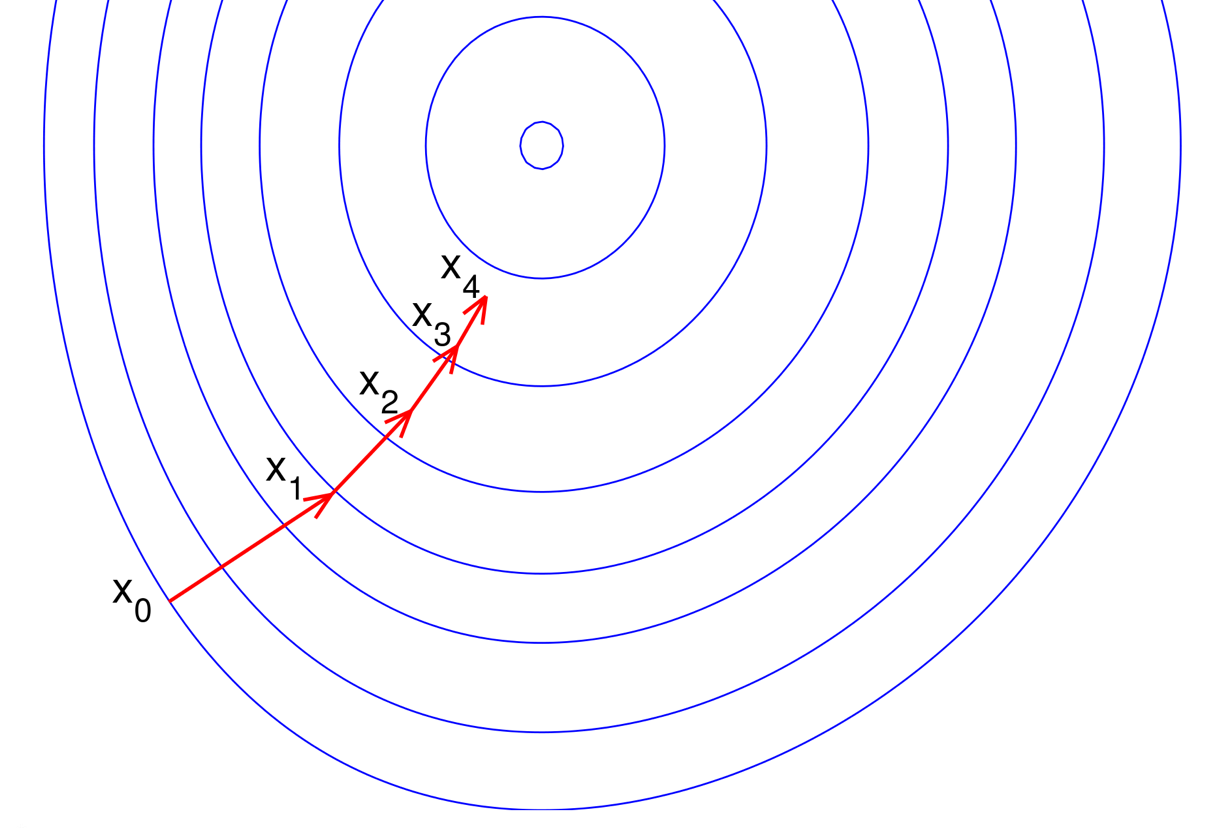
原文：<http://aria42.com/blog/2014/12/understanding-lbfgs/>

数值优化是大部分机器学习算法的核心。当你有了一个数据集并且定义好了模型，模型的参数估计就变成了最小化一些多元函数，输入x是高维空间的模型参数。换句话说，如果你解决了：

其中是所设定目标中的最合适的模型参数1。

本篇将主要讲述无约束最小化算法L-BFGS的动机。L-BFGS常用来在ML问题中，使得’batch’优化成为可能。对于更大的问题，一般采用基于随机梯度下降的在线方法，因为可以用更少的数据达到收敛。在上一篇博文中，我提到了其中的一些技术，包括我最喜爱的[AdaDelta](http://www.matthewzeiler.com/pubs/googleTR2012/googleTR2012.pdf)。

注意：本篇需要读者掌握多元微积分知识，比如梯度和海森矩阵。



# 牛顿法

大部分的数值优化方法是迭代算法，考虑一系列的来最终逼近真正的全局最小值。假定，我们有一个估计值，然后估计下一个值，使得。

牛顿法是围绕f的二次逼近来找到附近的一个点。假定f是二次可导的，我们可以使用f在一个固定点X附近的泰勒展开来得到f的二次逼近

其中和是f在点xn的梯度和海森值，在条件下。这是一维的泰勒展开。

为了简化记号，我们开始考虑生成这样一个二次逼近的序列hn。不失一般性，

，上式修改为

其中和代表f在点xn处的梯度和海森。

我们需要选择来最小化f在点的局部二次逼近。对上式求的偏导数

当时，是的极值。如果我们假定是[半正定](http://en.wikipedia.org/wiki/Positive-definite_matrix)，那么我们知道这个同样也是的全局最小值。的解为：

这意味着是一个移动的好方向。事实上，我们设定，对一定的会足够小于

# 迭代算法

根据上文，得到这样一个迭代算法：

步长的计算可以使用任何的[线性](http://en.wikipedia.org/wiki/Line_search)搜索算法。其中最简单的就是[回溯线性搜索](http://en.wikipedia.org/wiki/Backtracking_line_search)，不断的尝试更小的直到函数值最够小。

从软件工程的角度，我们可以把NewtonRaphson当作任何二次求导函数的黑盒，满足如下的Java接口：

|  |
| --- |
| **public** **interface** **TwiceDifferentiableFunction** **{**  *// compute f(x)*  **public** **double** **valueAt(double[]** x**);**  *// compute grad f(x)*  **public** **double[]** **gradientAt(double[]** x**);**  *// compute inverse hessian H^-1*  **public** **double[][]** **inverseHessian(double[]** x**);**  **}** |

通过一个冗长的数学证明，对于一个凸函数，无论初始值x0如何选择，通过上文的过程总可以收敛到一个全局最小值。对于ML中的非凸函数问题（几乎所有的隐变量模型个或者深度网络），上文的过程同样可以工作但是只保证收敛到局部最小值。事实上，对一个非凸优化，我们需要更多的关注初始化过程和其他算法细节。

# 大海森矩阵

NewtonRaphson的核心问题是，我们需要计算海森矩阵的逆3。对于ML应用，f的输入的维度和模型的参数有关。在视觉应用中，经常会有数亿甚至十亿的参数。所以，计算海森矩阵或者他的逆是不明智的。对于很多函数，海森矩阵可能并不能分析计算，更不用说可表示了。

因为这些原因，在实践中很少用NewtonRaphson来优化函数。幸运的是，上述的算法在是点xn的海森矩阵逆的近似值时还可以工作，并且仍然是一个好的近似。

# 拟牛顿法

假想如果不求点xn的海森矩阵逆，只求一个近似值。我们可以把NewtonRaphson归纳为QuasiUpdate策略，以此来生成序列。

// Compute search direction and step-size

// Store the input and gradient deltas

// Update inverse hessian

我们假定QuasiUpdate只需要前一个海森逆的估计以及输入和梯度差（对应sn和yn）。注意如果QuasiUpdate只返回，我们可以回复出精确的NewtonRaphson。

在软件中，我们可以用QuasiNewton黑盒优化任意的可导方程（在不计算二次导的情况下）。Java的实现如下，

|  |
| --- |
| **public** **interface** **DifferentiableFunction** **{**  *// compute f(x)*  **public** **double** **valueAt(double[]** x**);**  *// compute grad f(x)*  **public** **double[]** **gradientAt(double[]** x**);**  **}**  **public** **interface** **QuasiNewtonApproximation** **{**  *// update the H^{-1} estimate (using x\_{n+1}-x\_n and grad\_{n+1}-grad\_n)*  **public** **void** **update(double[]** deltaX**,** **double[]** deltaGrad**);**  *// H^{-1} (direction) using the current H^{-1} estimate*  **public** **double[]** **inverseHessianMultiply(double[]** direction**);**  **}** |

注意我们只在乘上梯度方向时才用到海森矩阵。这对下面描述L-BFGS很有用，我们不需要在内存中保留海森矩阵的近似。这里是我写的[Java 8](https://github.com/aria42/java8-optimize/tree/master/src/optimize)和[golang](https://github.com/aria42/taskar/blob/master/optimize/newton.go)写的抽象。

# 海森矩阵的近似

QuasiUpdate的形式是什么样的？如果每次QuasiUpdate返回的都是一致的矩阵（忽略输入），那就和[梯度下降](http://en.wikipedia.org/wiki/Gradient_descent)一样了，因为搜索方向总是。虽然这样的方法对凸函数f来说确实可以收敛到，但直观上来看这样的方法并不需要f的二阶导信息。

现在考虑作为点xn附近f的近似：

横切条件（Secant Condition）

的一个好的性质是它的梯度和f在xn和xn-1的梯度一致。换句话说就是：

结合上两式

使用的梯度并抵消后

这个式子就是所谓的”secant conditions”，这保证了和海森矩阵近似，至少在的导数差。假设是可逆的（在半正定的情况下是成立的），然后两边同乘以

其中是梯度的差，是输入的差。

# 对称性

回想一下，海森矩阵代表二阶偏导数：、无论偏导的顺序是什么，这个海森矩阵是对称的。

# BFGS更新

直观上来讲， 我们需要满足两个条件：

* Sn和yn满足横切条件
* 是对称的

在上述条件下，我们给出的变化。这使人想起MIRA更新， where we have conditions on any good solution but all other things equal, want the ‘smallest’ change.

这里用的范式是[加权费罗尼乌斯范数](http://mathworld.wolfram.com/FrobeniusNorm.html)4。这个优化问题的解法如下

其中。证明过程比较复杂，主要使一些符号运算。不知道任何简单的推到方式。



这个更新就是大家熟知的Broyden–Fletcher–Goldfarb–Shanno (BFGS)更新，以四位作者的名字命名。更新中的有些事需要大家注意：

* 当是半正定时也是。假设初始值的是半正定，由此可得每个海森矩阵的估计也是半正定的。因为初始矩阵可以任意选择，包括单位矩阵，所以这很容易满足。
* 上式指定了和之间的循环关系。我们只需要来重建。

最后一点很重要，对于方向d，我们有了不计算就可以得到的过程。重复应用上式，我们可以得到：

// Compute right product

,*…*,1:

// Compute center

// Compute left product

,*…*,1:

return r

我们只在乘法中用到，所以我们只需要在这个过程中使用BFGS来逼近拟牛顿法。

# L-BFGS：在有限的内存中使用BFGS

BFGS这种拟牛顿近似的好处是不需要分析计算一个函数的海森矩阵。然而，我们仍然需要维护每一轮迭代的和向量。NewtonRaphson算法的一个核心的担忧就是维护海森矩阵的内存开销，BFGS拟牛顿法并没有解决这个问题，内存开销可能反而增长了。

L-BFGS算法，完整名字是limited BFGS，仅仅简单的在BFGSMultiply更新中截断并只使用最后m个输入的差和梯度差。这意味着，我们只需要存储和来计算更新。中间的乘法仍然可以使用任何对称的半正定矩阵和或者。

# L-BFGS的变体

有很多L-BFGS的变体可以应用在实践中。对于不可导的函数，有个[othant-wise变种](http://research.microsoft.com/en-us/um/people/jfgao/paper/icml07scalable.pdf)（[代码](http://research.microsoft.com/en-us/downloads/b1eb1016-1738-4bd5-83a9-370c9d498a03/)）可以用来训练L1正则损失。

一个在大数据集上不使用L-BFGS的原因是，在线的方法可以收敛的更快。事实上有[在线版本的L-BFGS](http://jmlr.org/proceedings/papers/v2/schraudolph07a/schraudolph07a.pdf)，但是据我所知，对于特别大的数据集，还没有出现表现比SGD变种（包括[AdaGrad](http://www.magicbroom.info/Papers/DuchiHaSi10.pdf)或者AdaDelta）更好的。

1. 假定存在一个唯一的全局最小化函数。事实上， 除非是凸函数，否则参数会导向迭代函数的另一面。
2. 我们知道当梯度为零时，是局部极值，在海森矩阵有正的曲率的情况下。如果f是凸的，我们知道海森举证总是半正定的，并且我们知道有一个唯一的全局最小值。
3. 正如我们看到的，对于方向d，我们真的需要乘上。
4. 我有意把权重矩阵W用来权衡范数，因为你可以用很多选择得到相同的解决方案。特别的是，对于正定的W，满足，我们得到相同的解决方案。