理解牛顿法

SIGAI

其它机器学习、深度学习算法的全面系统讲解可以阅读**《机器学习-原理、算法与应用》**，清华大学出版社，**雷明著**，由SIGAI公众号作者倾力打造。

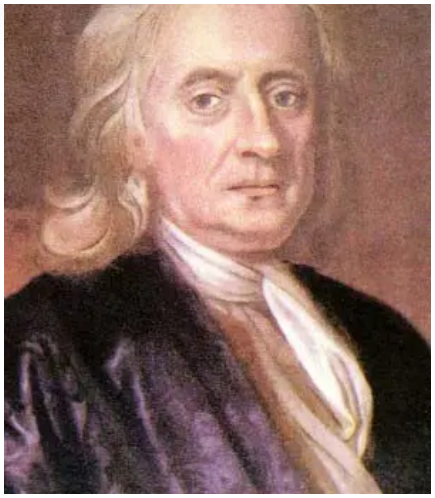
- 书的购买链接
- 书的勘误，优化，源代码资源

导言

牛顿法是数值优化算法中的大家族，她和她的改进型在很多实际问题中得到了应用。在机器学习中，牛顿法是和梯度下降法地位相当的的主要优化算法。在本文中，SIGAI将为大家深入浅出的系统讲述牛顿法的原理与应用。

牛顿法的起源

牛顿法以伟大的英国科学家牛顿命名，牛顿不仅是伟大的物理学家，是近代物理的奠基人，还是伟大的数学家，他和德国数学家莱布尼兹并列发明了微积分，这是数学历史上最有划时代意义的成果之一，奠定了近代和现代数学的基石。在数学中，也有很多以牛顿命名的公式和定理，牛顿法就是其中之一。



牛顿法不仅可以用来求解函数的极值问题，还可以用来求解方程的根，二者在本质上是一个问题，因为求解函数极值的思路是寻找导数为0的点，这就是求解方程。在本文中，我们介绍的是求解函数极值的牛顿法。

在SIGAI之前关于最优方法的系列文章“理解梯度下降法”，“理解凸优化”中，我们介绍了最优化的基本概念和原理，以及迭代法的思想，如果对这些概念还不清楚，请先阅读这两篇文章。和梯度下降法一样，牛顿法也是寻找导数为0的点，同样是一种迭代法。核心思想是在某点处用二次函数来近似目标函数，得到导数为0的方程，求解该方程，得到下一个迭代点。因为是用二次函数近似，因此可能会有误差，需要反复这样迭代，直到到达导数为0的点处。下面我们开始具体的推导，先考虑一元函数的情况，然后推广到多元函数。

一元函数的情况

为了能让大家更好的理解推导过程的原理，首先考虑一元函数的情况。根据一元函数的泰勒展开公式，我们对目标函数在x0点处做泰勒展开，有：

$$f(x)=f(x_0)+f^{'}(x_0)(x-x_0)+\frac{1}{2}f^{''}(x_0)(x-x_0)^2+\ldots+\frac{1}{n!}f^{(n)}(x_0)(x-x_0)^n\ldots$$

如果忽略2次以上的项，则有：

$$f(x)=f(x_0)+f^{'}(x_0)(x-x_0)+\frac{1}{2}f^{''}(x_0)(x-x_0)^2$$

现在我们在x0点处，要以它为基础，找到导数为0的点，即导数为0。对上面等式两边同时求导，并令导数为0，可以得到下面的方程：

$$f^{'}(x)=f^{'}(x_0)+f^{''}(x_0)(x-x_0)=0$$

可以解得：

$$x=x_0-\frac{f^{'}(x_0)}{f^{''}(x_0)}$$

$$x_{t+1} = x_t - \frac{f^{'}\left(x_t\right)}{f^{"}\left(x_t\right)}$$

给定初始迭代点x0，反复用上面的公式进行迭代，直到达到导数为0的点或者达到最大迭代次数。

多元函数的情况

下面推广到多元函数的情况，如果读者对梯度，Hessian的概念还不清楚，请先去看微积分教材，或者阅读SIGAI之前关于最优化的公众号文章。根据多元函数的泰勒展开公式，我们对目标函数在x0点处做泰勒展开，有：

$$f\left(\mathrm{x}\right) = f\left(\mathrm{x}_0\right) + \nabla f\left(\mathrm{x}_0\right)^{\mathrm{T}}\left(\mathrm{x} - \mathrm{x}_0\right) + \frac{1}{2}\left(\mathrm{x} - \mathrm{x}_0\right)^{\mathrm{T}} \nabla^2 f\left(\mathrm{x}_0\right)\left(\mathrm{x} - \mathrm{x}_0\right) + o\left(\left(\mathrm{x} - \mathrm{x}_0\right)^2\right)$$

忽略二次及以上的项，并对上式两边同时求梯度，得到函数的导数（梯度向量）为：

$$\nabla\left(\mathrm{x}\right) = \nabla f\left(\mathrm{x}_0\right) + \nabla^2 f\left(\mathrm{x}_0\right)\left(\mathrm{x} - \mathrm{x}_0\right)$$

其中 $\nabla^2 f(x_0)$ 即为Hessian矩阵，在后面我们写成H。令函数的梯度为0，则有：

$$\nabla f\left(\mathrm{x}_0\right) + \nabla^2 f\left(\mathrm{x}_0\right)\left(\mathrm{x} - \mathrm{x}_0\right) = 0 \Rightarrow \mathrm{x} = \mathrm{x}_0 - \left(\nabla^2 f\left(\mathrm{x}_0\right)\right)^{-1} \nabla f\left(\mathrm{x}_0\right)$$

这是一个线性方程组的解。如果将梯度向量简写为g，上面的公式可以简写为：

$$\mathrm{x} = \mathrm{x}_0 - \mathrm{H}^{-1}\mathrm{g}$$

从初始点x0处开始，反复计算函数在处的Hessian矩阵和梯度向量，然后用下述公式进行迭代：

$$\mathrm{x}_{k+1} = \mathrm{x}_k - \mathrm{H}_k^{-1}\mathrm{g}_k$$

最终会到达函数的驻点处。其中 $-\boldsymbol{H}^{-1}\boldsymbol{g}$ 称为牛顿方向。迭代终止的条件是梯度的模接近于0，或者函数值下降小于指定阈值。

实现细节

根据上面的推导，我们可以得到牛顿法的完整流程为：

1.给定初始值 $\boldsymbol{x_0}$ 和精度阈值 $\boldsymbol{\varepsilon}$ ，设置k = 0

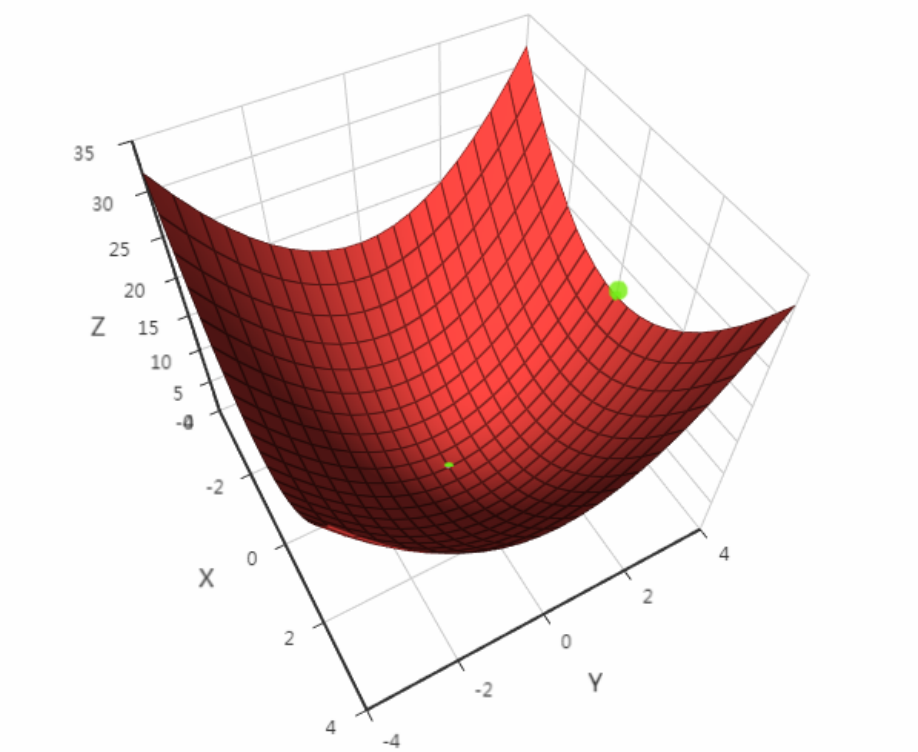
2.计算梯度 $\boldsymbol{g_k}$ 和矩阵 $\boldsymbol{H_k}$

3.如果$\| \boldsymbol{g_k} \| < \boldsymbol{\varepsilon}$ 即在此点处梯度的值接近于0，则达到极值点处，停止迭代

4.计算搜索方向 $\boldsymbol{d_k} = -\boldsymbol{H}^{-1}\boldsymbol{g_k}$

6.令k = k + 1，返回步骤2

其中 γ 是一个人工设定的接近于0的常数，和梯度下降法一样，需要这个参数的原因是保证 X_{k+1} 在 X_k 的邻域内，从而可以忽略泰勒展开的高次项。如果目标函数是二次函数，Hessian矩阵是一个常数矩阵，对于任意给定的初始点，牛顿法只需要一步迭代就可以收敛到极值点。下图是对x*x+y*y用牛顿法求解的一个例子：



如果我们把步长设置的足够大（在这里为1），则算法一步就收敛了。在这里，初始迭代位置为(0,4)，最优解为(0,0)。

对于不带约束条件的问题，我们可以将x的初始值设定为任意值，最简单的，可以设置为全0的向量。迭代终止的判定规则和梯度下降法相同，是检查梯度是否接近于0。

牛顿法并不能保证每一步迭代时函数值下降，也不保证一定收敛。为此，提出了一些补救措施，其中的一种是直线搜索（line search）技术，即搜索最优步长。具体做法是让 γ 取一些典型的离散值，如0.0001,0.001,0.01等，比较取哪个值时函数值下降最快，作为最优步长。

和梯度下降法相比牛顿法有更快的收敛速度，但每一步迭代的成本也更高。在每次迭代中，除了要计算梯度向量还要计算Hessian矩阵，并求解Hessian矩阵的逆矩阵。实际实现时一般不直接求Hessian矩阵的逆矩阵，而是求解如下方程组：

$$\mathrm{H}_k \mathrm{d} = -\mathrm{g}_k$$

求解这个线性方程组一般使用迭代法，如共轭梯度法，当然也可以使用其他算法。

面临的问题

牛顿法面临的另外一个问题是Hessian矩阵可能不可逆，从而导致这种方法失效。此外，求解Hessian矩阵的逆矩阵或者求解线性方程组计算量大，需要耗费大量的时间。为此，提出了拟牛顿法这种改进方案，在后面会介绍。

除此之外，牛顿法在每次迭代时序列xi可能不会收敛到一个最优解，它甚至不能保证函数值会按照这个序列递减。解决第一个问题可以通过调整牛顿方向的步长来实现，目前常用的方法有两种：直线搜索和可信区域法。可信域牛顿法在后面也会介绍。

可信域牛顿法

可信域牛顿法（Trust Region Newton Methods）可以求解带界限约束的最优化问题，是对牛顿法的改进。在可信域牛顿法的每一步迭代中，有一个迭代序列 $\boldsymbol{x_k}$ ，一个可信域的大小 $\boldsymbol{\Delta_k}$ ，以及一个二次目标函数：

$$q_k\left(\mathrm{s}\right)=\left(\nabla f\left(\mathrm{x}_k\right)\right)^{\mathrm{T}}\mathrm{s}+\frac{1}{2}\mathrm{s}^{\mathrm{T}}\nabla^2 f\left(\mathrm{x}_k\right)\mathrm{s}$$

这个式子可以通过泰勒展开得到，忽略二次以上的项，这是对函数下降值：

$$f\left(\mathrm{x}_k+\mathrm{s}\right)-f\left(\mathrm{x}_k\right)$$

的近似。算法寻找一个 $\boldsymbol{S_k}$ ，在满足约束条件 $||S|| \leq \boldsymbol{\Delta_k}$ 下近似最小化 $\boldsymbol{q_k(s)}$

。接下来检查如下比值以更新$\boldsymbol{x_k}$和$\boldsymbol{\Delta_k}$：

$$\rho_k=\frac{f\left(\mathrm{x}_k+\mathrm{s}_k\right)-f\left(\mathrm{x}_k\right)}{q_k\left(\mathrm{s}_k\right)}$$

这是函数值的实际减少量和二次近似模型预测方向导致的函数减少量的比值。迭代方向可以接受的条件是 $\boldsymbol{p_k}$ 足够大，由此得到参数的更新规则为：

$$\mathrm{x}_{k+1}=\begin{cases}\mathrm{x}_k+\mathrm{s}_k, \rho_k>\eta_0\\ \mathrm{x}_k, \quad \rho_k\leq\eta_0\end{cases}$$

其中 $\boldsymbol{\eta_0}$ 是一个人工设定的值。 $\boldsymbol{\Delta_k}$ 的更新规则取决于人工设定的正常数 $\boldsymbol{\eta_1}$ 和 $\boldsymbol{\eta_2}$ ，其中：

$$\eta_1<\eta_2<1$$

而$\boldsymbol{\Delta_k}$的更新率取决于人工设定的正常数 $\boldsymbol{\sigma_1,\sigma_2,\sigma_3}$ ，其中：

$$\sigma_1<\sigma_2<1<\sigma_3$$

可行域的边界$\boldsymbol{\Delta_k}$更新规则为：

$$\Delta_{k+1}\in\left[\sigma_1\Delta_k,\sigma_3\Delta_k\right],if\ \rho_k\in\left(\eta_1,\eta_2\right)$$

$$\Delta_{k+1}\in\left[\Delta_k,\sigma_3\Delta_k\right],if\ \rho_k\geq\eta_2$$

在牛顿法的每一步迭代中，动态调整可信域，确保序列收敛。

拟牛顿法

牛顿法在每次迭代时需要计算出Hessian矩阵，然后求解一个以该矩阵为系数矩阵的线性方程组，这非常耗时，另外Hessian矩阵可能不可逆。为此提出了一些改进的方法，典型的代表是拟牛顿法（Quasi-Newton）。

拟牛顿法的思想是不计算目标函数的Hessian矩阵然后求逆矩阵，而是通过其他手段得到Hessian矩阵或其逆矩阵的近似矩阵。具体做法是构造一个近似Hessian矩阵或其逆矩阵的正定对称矩阵，用该矩阵进行牛顿法的迭代。将函数在 $\boldsymbol{x_{k+1}}$ 点处进行泰勒展开，忽略二次以上的项，有：

$$f\left(\mathrm{x}\right)\approx f\left(\mathrm{x}_{k+1}\right)+\nabla f\left(\mathrm{x}_{k+1}\right)^{\mathrm{T}}\left(\mathrm{x}-\mathrm{x}_{k+1}\right)+\frac{1}{2}\left(\mathrm{x}-\mathrm{x}_{k+1}\right)^{\mathrm{T}}\nabla^2 f\left(\mathrm{x}_{k+1}\right)\left(\mathrm{x}-\mathrm{x}_{k+1}\right)$$

对上式两边同时取梯度，有：

$$\nabla f\left(\mathrm{x}\right)\approx\nabla f\left(\mathrm{x}_{k+1}\right)+\nabla^2 f\left(\mathrm{x}_{k+1}\right)\left(\mathrm{x}-\mathrm{x}_{k+1}\right)$$

令 $\boldsymbol{X}=\boldsymbol{X_k}$，有：

$$\nabla f\left(\mathrm{x}_{k+1}\right)-\nabla f\left(\mathrm{x}_k\right)\approx\nabla^2 f\left(\mathrm{x}_{k+1}\right)\left(\mathrm{x}_{k+1}-\mathrm{x}_k\right)$$

这可以简写为：

$$\mathrm{g}_{k+1}-\mathrm{g}_k\approx\mathrm{H}_{k+1}\left(\mathrm{x}_{k+1}-\mathrm{x}_k\right)$$

如果令：

$$\mathrm{s}_k=\mathrm{x}_{k+1}-\mathrm{x}_k$$

$$\mathrm{y}_k=\mathrm{g}_{k+1}-\mathrm{g}_k$$

上式可以简写为：

即：

算法。下图列出了常用的拟牛顿法的迭代公式（图片来自维基百科）：

BFGS算法是它的四个发明人Broyden，Fletcher，Goldfarb和Shanno名字首字母的简写。算法的思想是构造Hessian矩阵的近似矩阵：

并迭代更新这个矩阵：

该矩阵的初始值 B_{0} 为单位阵I。这样，要解决的问题就是每次的修正矩阵 \Delta B_{k} 的构造。其计算公式为：

其中：

因此有：

算法的完整流程为：

2.确定搜索方向 d_{k}=-B_{k}^{-1}g_{k}

3.搜索得到步长 \lambda_{k}，令 S_{k}=\lambda_{k}d_{k},X_{k+1}=X_{k}+S_{k}

4.如果 ||g_{k+1}||<\varepsilon，则迭代结束

5.计算 Y_{k}=g_{k+1}-g_{k}

6.计算 B_{k+1}=B_{k}+\frac{Y_{k}Y_{k}^{T}}{Y_{k}^{Y}S_{k}}-\frac{B_{k}S_{k}S_{k}^{T}B_{k}}{S_{k}^{t}B_{k}xS_{k}}

7.令K=K+1，返回步骤2

每一步迭代需要计算nxn的矩阵 D_{k}，当n很大时，存储该矩阵非常耗费内存。为此提出了改进方案L-BFGS，其思想是不存储完整的矩阵D_{k}，只存储向量 S_{k} 和 Y_{k} 。

实际应用

下面介绍牛顿法在机器学习中的实际应用。在这里我们以线性支持向量机和liblinear为例。liblinear是一个线性算法的开源库，其作者是台湾大学林智仁教授和他的学生，与libsvm的作者相同。这个库支持logistic回归和线性支持向量机两类算法，包括各种损失函数和正则化的版本。L1正则化L2损失函数线性支持向量机训练时求解如下最优化问题：

目标函数的前半部分其中为L1范数的正则化项，后半部分括号里为合页损失函数。在liblinear中，求解上述问题采用了坐标下降法，这是一种分治法，每次挑选出一部分变量进行优化，将其他变量固定住不动。如果只挑选出一个变量 W_{j} 进行优化，要求解的子问题为：

其中向量e的第j个分量为1，其他分量全为0。上式中最后一步对函数用二阶泰勒展开近似代替。上面子问题的求解采用牛顿法。求解整个问题的坐标下降法流程为（这里只列出了和牛顿法相关的步骤）：

设置各个参数的初始值

如果w还不是最优值，则循环

循环，对j = 1, 2, ..., n

更新参数值:

结束循环

结束循环

下面来看源代码实现。函数solve_l1r_l2_svc实现求解L1正则化L2损失函数支持向量机原问题的坐标下降法。在这里我们重点看牛顿方向的计算，直线搜索，参数更新这三步，其他的可以忽略掉。代码如下:

知乎

切换

知乎

切换

知乎

切换

知乎

切换

知乎

切换

推荐阅读

[1] 机器学习-波澜壮阔40年 SIGAI 2018.4.13.

[4] 基于深度学习的目标检测算法综述 SIGAI 2018.4.24.

[5] 卷积神经网络为什么能够称霸计算机视觉领域？ SIGAI 2018.4.26.

[6] 用一张图理解SVM的脉络 SIGAI 2018.4.28.

[7] 人脸检测算法综述 SIGAI 2018.5.3.

[8] 理解神经网络的激活函数 SIGAI 2018.5.5.

[9] 深度卷积神经网络演化历史及结构改进脉络-40页长文全面解读 SIGAI 2018.5.8.

[10] 理解梯度下降法 SIGAI 2018.5.11

[11] 循环神经网络综述—语音识别与自然语言处理的利器 SIGAI 2018.5.15

[12] 理解凸优化 SIGAI 2018.5.18

[13]【实验】理解SVM的核函数和参数 SIGAI 2018.5.22

[14]] [SIGAI综述] 行人检测算法 SIGAI 2018.5.25

[15] 机器学习在自动驾驶中的应用—以百度阿波罗平台为例 SIGAI 2018.5.29

更多干货请关注V X公众号：SIGAI

编辑于 2019-10-21 10:04

机器学习 算法 凸优化

写下你的评论...

能用用约等吗，泰勒n不趋近无穷没法用等号啊。太不严谨了吧。

2018-11-26 回复 1

看你看我

后面有个无穷小

2019-06-21 回复 喜欢

稻壳特溯

理解梯度下降法 链接不可达了呀

2019-12-31 回复 喜欢

柳树上有只小鸟

有一点点迷惑，我们一开始让泰勒展开式中的一阶导等于零，然后推出一个递推的式子，怎么能证明新得到的x要比原来的x更好呢?

2019-12-02 回复 喜欢

雷明

文章里说的很清楚，牛顿法不能保证每次迭代时目标函数值一定下降，更不能保证一定收敛

2020-09-12 回复 2

云玩家

不能证明, 因为更好只是有可能, 比如梯度下降, 也只是可能能达到最优点, 也可能到达鞍点等等. 这里的思想就是泰勒展开近似, 把泰勒展开前几项 "当成" 是原来的函数, 一般来说 "很可能" 会更好, 但不一定.

2020-03-14 回复 2

zxfzxfzxf

终于有一个文章讲的是为什么拟牛顿法用k+1而不是k了！！！，感谢

2019-11-21 回复 喜欢

雷明

拟牛顿法之所以要在k+1点出进行泰勒展开，是因为我们要推导的是这各点处的Hessian矩阵的近似矩阵所需要满足的条件，在我即将出版的《机器学习的数学》中有详细的说明

2020-09-12 回复 喜欢

Jack-Ma

二级导数连续的时候，Hessian矩阵是对称阵，才能求多元素迭代公式

2019-10-07 回复 喜欢

堇然

讲的通俗易懂啊棒!

2019-03-07 回复 喜欢

幻觉在北京

提到的共轭梯度法，并不是用来求解线性方程组的吧？是为了减少迭代次数来减少求解线性方程的次数，我的理解

2021-10-18 回复 喜欢

许诺

原来公式是一张图，显示有点慢哦

2018-06-12 回复 喜欢

SIGAI 作者

是的，为了让公式显示的更清晰，小编试了很多种方法，我们会在努力把图片缩小同时保证清晰度

2018-06-13 回复 1

shift

为什么重复迭代就可以求最小值

2021-04-24 回复 喜欢

知乎 切换

blue
文章写的很好，可是我看不懂，怎么办啊
2020-10-08 回复 喜欢

遇见娇妍
一元函数情况中，x1对应的点不就是导数为零的点吗？为什么还需要迭代呀？
2022-10-04 回复 喜欢

遇见娇妍
还有，多元函数的多元体现在哪，f(x)是多元函数吗
2022-10-04 回复 喜欢

Leon
讲得好棒啊！！！
2020-08-05 回复 喜欢

懵b小狮子
妈的，终于找到一篇写的从头到尾看的贼拉顺畅的牛顿法，真的写得很好。
2021-12-10 回复 喜欢

写下你的评论...

推荐阅读

$$\text{for } i = 0, \ldots, L-1$$
$$\{$$
$$j = i + \delta\ ;$$
$$\beta_j = \rho_j \mathbf{y}_j^T \mathbf{r}_i\ ;$$
$$\mathbf{r}_{i+1} = \mathbf{r}_i + (\alpha_i - \beta_i)\mathbf{s}_j\ ;$$

牛顿法

大咸鱼

牛顿法进阶

在之前的文章 线搜索与信赖域中，我们介绍过，牛顿法是线搜索的一种，目标函数二阶近似的极值，就是每次迭代牛顿法的方向，用公式表示为 abla^2 f \left(x_k \right) p_k^N = - abla f \le...

王金戈 发表于数值最优化

牛顿法和拟牛顿法

牛顿法（Newton method）和拟牛顿法（quasi Newton method）是求解无约束最优化问题的常用方法，有收敛速度快的优点。牛顿法是迭代算法，每一步都需求解目标函数的海塞矩阵（Hessian...

Pikac... 发表于机器学习中...

牛顿法和拟

导言牛顿法和约束最优化问常用方法为：敛速度快的优法，每一步需Hessian 矩阵

多鱼