

【Math】常见的几种最优化方法

Poll 机器学习算法那些事 2018-12-20

作者: Poll

链接:

<https://www.cnblogs.com/maybe2030/p/4751804.html>

编辑: 石头

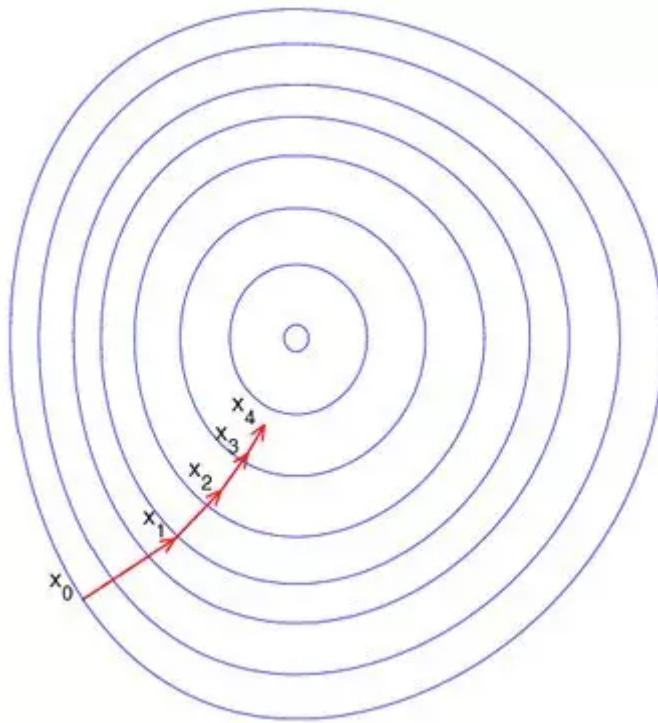
目录

1. 梯度下降法
2. 牛顿法和拟牛顿法
3. 共轭梯度法
4. 启发式优化方法
5. 解决约束优化问题——拉格朗日乘数法

我们每个人都会在我们的生活或者工作中遇到各种各样的最优化问题，比如每个企业和个人都要考虑的一个问题“在一定成本下，如何使利润最大化”等。最优化方法是一种数学方法，它是研究在给定约束之下如何寻求某些因素(的量)，以使某一(或某些)指标达到最优的一些学科的总称。随着学习的深入，博主越来越发现最优化方法的重要性，学习和工作中遇到的大多问题都可以建模成一种最优化模型进行求解，比如我们现在学习的机器学习算法，大部分的机器学习算法的本质都是建立优化模型，通过最优化方法对目标函数（或损失函数）进行优化，从而训练出最好的模型。常见的最优化方法有梯度下降法、牛顿法和拟牛顿法、共轭梯度法等等。

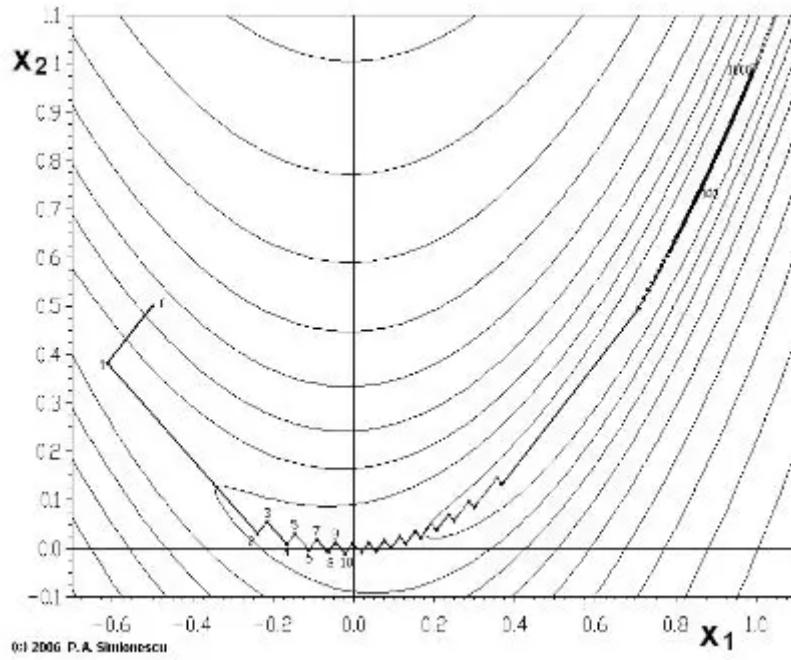
1. 梯度下降法 (Gradient Descent)

梯度下降法是最早最简单，也是最为常用的最优化方法。梯度下降法实现简单，当目标函数是凸函数时，梯度下降法的解是全局解。一般情况下，其解不保证是全局最优解，梯度下降法的速度也未必是最快的。**梯度下降法的优化思想是用当前位置负梯度方向作为搜索方向，因为该方向为当前位置的最快下降方向，所以也被称为是“最速下降法”。**最速下降法越接近目标值，步长越小，前进越慢。梯度下降法的搜索迭代示意图如下图所示：



梯度下降法的缺点：

- (1) 靠近极小值时收敛速度减慢，如下图所示；
- (2) 直线搜索时可能会产生一些问题；
- (3) 可能会“之字形”地下降。



从上图可以看出，梯度下降法在接近最优解的区域收敛速度明显变慢，利用梯度下降法求解需要很多次的迭代。

在机器学习中，基于基本的梯度下降法发展了两种梯度下降方法，分别为随机梯度下降法和批量梯度下降法。

比如对一个线性回归 (Linear Logistics) 模型, 假设下面的 $h(x)$ 是要拟合的函数, $J(\theta)$ 为损失函数, θ 是参数, 要迭代求解的值, θ 求解出来了, 那最终要拟合的函数 $h(\theta)$ 就出来了。其中 m 是训练集的样本个数, n 是特征的个数。

1) 批量梯度下降法 (Batch Gradient Descent, BGD)

(1) 将 $J(\theta)$ 对 θ 求偏导, 得到每个 θ 对应的梯度:

$$\sum_{i=1}^m \frac{\partial J(\theta, x_i)}{\partial \theta_j}$$

(2) 由于是要最小化风险函数, 所以按每个参数 θ 的梯度负方向, 来更新每个 θ :

$$\theta_j := \theta_j - \eta \cdot \sum_{i=1}^m \frac{\partial J(\theta, x_i)}{\partial \theta_j}$$

(3) 从上面公式可以注意到, 它得到的是一个全局最优解, 但是每迭代一步, 都要用到训练集所有的数据, 如果 m 很大, 那么可想而知这种方法的迭代速度会相当的慢。所以, 这就引入了另外一种方法——随机梯度下降。

对于批量梯度下降法, 样本个数 m , x 为 n 维向量, 一次迭代需要把 m 个样本全部带入计算, 迭代一次计算量为 $m \cdot n^2$ 。

2) 随机梯度下降 (Stochastic Gradient Descent, SGD)

(1) 上面的风险函数可以写成如下这种形式, 损失函数对应的是训练集中每个样本的梯度, 而上面批量梯度下降对应的是所有的训练样本:

$$\frac{\partial J(\theta, x_i)}{\partial \theta_j}$$

(2) 每个样本的损失函数, 对 θ 求偏导得到对应梯度, 来更新 θ :

$$\theta_j := \theta_j - \eta \cdot \frac{\partial J(\theta, x_i)}{\partial \theta_j}$$

(3) 随机梯度下降是通过每个样本来迭代更新一次, 如果样本量很大的情况 (例如几十万), 那么可能只用其中几万条或者几千条的样本, 就已经将 θ 迭代到最优解了, 对比上面的批量梯度下降, 迭代一次需要用到十几万训练样本, 一次迭代不可能最优, 如果迭代10次的话就需要遍历训练样本10次。但是, SGD伴随的一个问题是噪音较BGD要多, 使得SGD并不是每次迭代都向着整体最优化方向。

随机梯度下降每次迭代只使用一个样本, 迭代一次计算量为 n^2 , 当样本个数 m 很大的时候, 随机梯度下降迭代一次的速度要远高于批量梯度下降方法。**两者的关系可以这样理解: 随机梯度下降方法以损失很小的一部分精确度和增加一定数量的迭代次数为代价, 换取了总体的优化效率的提升。增加的迭代次数远远小于样本的数量。**

对批量梯度下降法和随机梯度下降法的总结:

批量梯度下降---最小化所有训练样本的损失函数，使得最终求解的是全局的最优解，即求解的参数是使得风险函数最小，但是对于大规模样本问题效率低下。

随机梯度下降---最小化每条样本的损失函数，虽然不是每次迭代得到的损失函数都向着全局最优方向，但是大的整体的方向是向全局最优解的，最终的结果往往是在全局最优解附近，适用于大规模训练样本情况。

2. 牛顿法和拟牛顿法 (Newton's method & Quasi-Newton Methods)

1) 牛顿法 (Newton's method)

牛顿法是一种在实数域和复数域上近似求解方程的方法。方法使用函数 $f(x)$ 的泰勒级数的前面几项来寻找方程 $f(x) = 0$ 的根。牛顿法最大的特点就在于它的收敛速度很快。

具体步骤：

首先，选择一个接近函数 $f(x)$ 零点的 x_0 ，计算相应的 $f(x_0)$ 和切线斜率 $f'(x_0)$ （这里 f' 表示函数 f 的导数）。然后我们计算穿过点 $(x_0, f(x_0))$ 并且斜率为 $f'(x_0)$ 的直线和 x 轴的交点的 x 坐标，也就是求如下方程的解：

$$x \cdot f'(x_0) + f(x_0) - x_0 \cdot f'(x_0) = 0$$

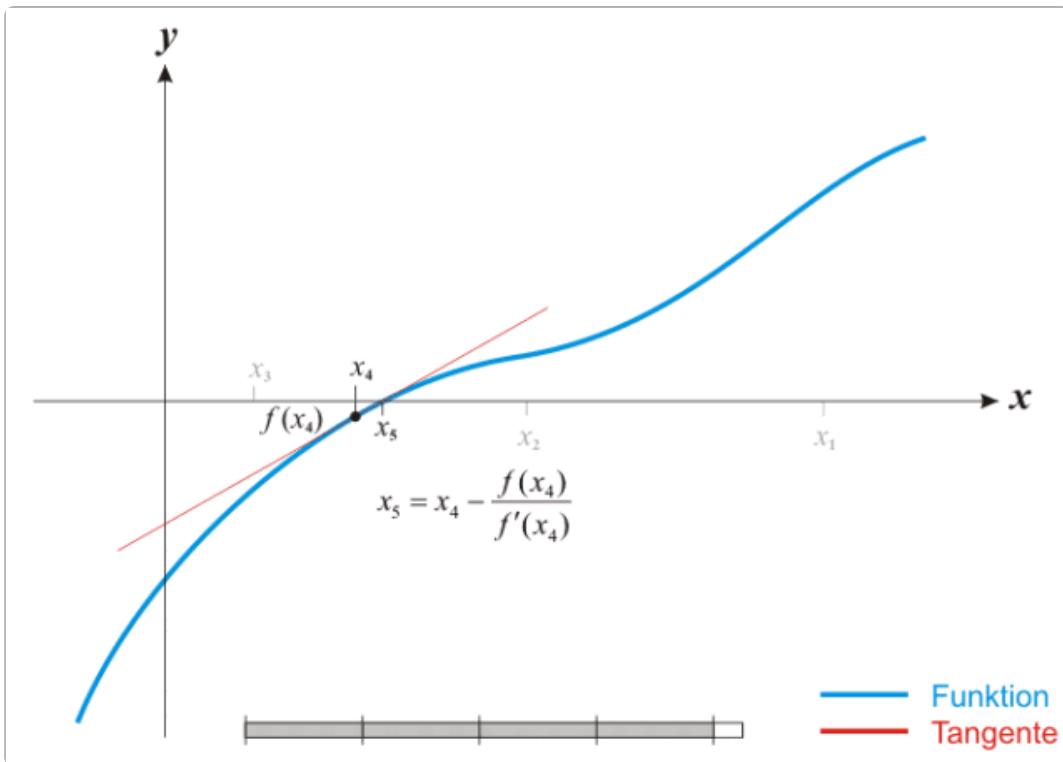
我们将新求得的 x 坐标命名为 x_1 ，通常 x_1 会比 x_0 更接近方程 $f(x) = 0$ 的解。因此我们现在可以利用 x_1 开始下一轮迭代。迭代公式可化简为如下所示：

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

已经证明，如果 f' 是连续的，并且待求的零点 x 是孤立的，那么在零点 x 周围存在一个区域，只要初始值 x_0 位于这个邻近区域内，那么牛顿法必定收敛。并且，如果 $f'(x)$ 不为0，那么牛顿法将具有平方收敛的性能。粗略的说，这意味着每迭代一次，牛顿法结果的有效数字将增加一倍。

由于牛顿法是基于当前位置的切线来确定下一次的位置，所以牛顿法又被很形象地称为是“切线法”。牛顿法的搜索路径（二维情况）如下图所示：

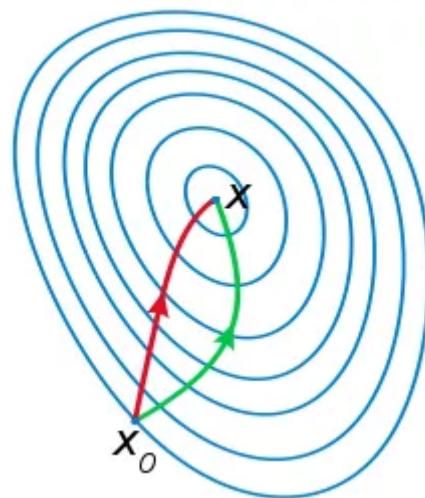
牛顿法搜索动态示例图：



关于牛顿法和梯度下降法的效率对比：

从本质上来看，牛顿法是二阶收敛，梯度下降是一阶收敛，所以牛顿法就更快。如果更通俗地说的话，比如你想找一条最短的路径走到一个盆地的最底部，梯度下降法每次只从你当前所处位置选一个坡度最大的方向走一步，牛顿法在选择方向时，不仅会考虑坡度是否够大，还会考虑你走了一步之后，坡度是否会变得更大。所以，可以说牛顿法比梯度下降法看得更远一点，能更快地走到最底部。（牛顿法目光更加长远，所以少走弯路；相对而言，梯度下降法只考虑了局部的最优，没有全局思想。）

根据wiki上的解释，从几何上说，牛顿法就是用一个二次曲面去拟合你当前所处位置的局部曲面，而梯度下降法是用一个平面去拟合当前的局部曲面，通常情况下，二次曲面的拟合会比平面更好，所以牛顿法选择的下降路径会更符合真实的最优下降路径。



注：红色的牛顿法的迭代路径，绿色的是梯度下降法的迭代路径。

牛顿法的优缺点总结：

优点：二阶收敛，收敛速度快；

缺点：牛顿法是一种迭代算法，每一步都要求解目标函数的Hessian矩阵的逆矩阵，计算比较复杂。

2) 拟牛顿法 (Quasi-Newton Methods)

拟牛顿法是求解非线性优化问题最有效的方法之一，于20世纪50年代由美国Argonne国家实验室的物理学家W. C. Davidon所提出来。Davidon设计的这种算法在当时看来是非线性优化领域最具创造性的发明之一。不久R. Fletcher和M. J. D. Powell证实了这种新的算法远比其他方法快速和可靠，使得非线性优化这门学科在一夜之间突飞猛进。

拟牛顿法的本质思想是改善牛顿法每次都要求解复杂的Hessian矩阵的逆矩阵的缺陷，它使用正定矩阵来近似Hessian矩阵的逆，从而简化了运算的复杂度。拟牛顿法和最速下降法一样只要求每一步迭代时知道目标函数的梯度。通过测量梯度的变化，构造一个目标函数的模型使之足以产生超线性收敛性。这类方法大大优于最速下降法，尤其对于困难的问题。另外，因为拟牛顿法不需要二阶导数的信息，所以有时比牛顿法更为有效。如今，优化软件中包含了大量的拟牛顿算法用来解决无约束，约束，和大规模的优化问题。

具体步骤：

拟牛顿法的基本思想如下。首先构造目标函数在当前迭代 x_k 的二次模型：

$$\begin{aligned} m_k(p) &= f(x_k) + \nabla f(x_k)^T p + \frac{p^T B_k p}{2} \\ p_k &= -B_k^{-1} \nabla f(x_k) \end{aligned}$$

这里 B_k 是一个对称正定矩阵，于是我们取这个二次模型的最优解作为搜索方向，并且得到新的迭代点：

$$x_{k+1} = x_k + a_k p_k$$

其中我们要求步长 a_k 满足Wolfe条件。这样的迭代与牛顿法类似，区别就在于用近似的Hesse矩阵 B_k 代替真实的Hesse矩阵。所以拟牛顿法最关键的地方就是每一步迭代中矩阵 B_k 的更新。现在假设得到一个新的迭代 x_{k+1} ，并得到一个新的二次模型：

$$m_{k+1}(p) = f(x_{k+1}) + \nabla f(x_{k+1})^T p + \frac{p^T B_{k+1} p}{2}$$

我们尽可能地利用上一步的信息来选取 B_k 。具体地，我们要求

$$\nabla f(x_{k+1}) - \nabla f(x_k) = a_k B_{k+1} p_k$$

从而得到

$$B_{k+1}(x_{k+1} - x_k) = \nabla f(x_{k+1}) - \nabla f(x_k)$$

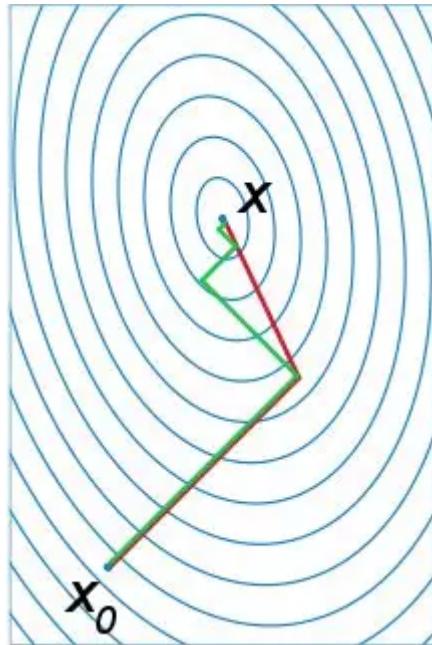
这个公式被称为割线方程。常用的拟牛顿法有DFP算法和BFGS算法。

3. 共轭梯度法 (Conjugate Gradient)

共轭梯度法是介于最速下降法与牛顿法之间的一个方法，它仅需利用一阶导数信息，但克服了最速下降法收敛慢的缺点，又避免了牛顿法需要存储和计算Hesse矩阵并求逆的缺点，共轭梯度法不仅是解决大型线性方程组最有用的方法之一，也是解大型非线性最优化最有效的算法之一。 在各种优化算法中，共轭梯度法是非常重要的一种。其优点是所需存储量小，具有步收敛性，稳定性高，而且不需要任何外来参数。

具体的实现步骤请参加wiki百科共轭梯度法。

下图为共轭梯度法和梯度下降法搜索最优解的路径对比示意图：



注：绿色为梯度下降法，红色代表共轭梯度法

MATLAB代码：

```
function [x] = conjgrad(A, b, x)
r=b-A*x;
p=r;
rsold=r'*r;

for i=1:length(b)
    Ap=A*p;
    alpha=rsold/(p'*Ap);
    x=x+alpha*p;
    r=r-alpha*Ap;
    rsnew=r'*r;
    if sqrt(rsnew)<1e-10
        break;
    end
    p=r+(rsnew/rsold)*p;
    rsold=rsnew;
end
```

4. 启发式优化方法

启发式方法指人在解决问题时所采取的一种根据经验规则进行发现的方法。其特点是在解决问题时，利用过去的经验，选择已经行之有效的方法，而不是系统地、以确定的步骤去寻求答案。启发式优化方法种类繁多，包括经典的模拟退火方法、遗传算法、蚁群算法以及粒子群算法等等。

还有一种特殊的优化算法被称之为多目标优化算法，它主要针对同时优化多个目标（两个及两个以上）的优化问题，这方面比较经典的算法有NSGAII算法、MOEA/D算法以及人工免疫算法等。

5. 解决约束优化问题——拉格朗日乘数法

请参考公众号文章《支持向量机（三）：图解KKT条件和拉格朗日乘子法》和博文《拉格朗日乘数法》

推荐阅读资料

《支持向量机（三）：图解KKT条件和拉格朗日乘子法》



-END-



长按二维码关注

机器学习算法那些事
微信：beautifulife244

砥砺前行 不忘初心