牛顿迭代,BFGS,L-BFGS等优化算法

2015年7月27日 星期一　　下午10:34

一,简介

牛顿迭代法,DFP算法,BFGS,L-BFGS 这一些列的算法,都是优化算法,类似sgd之类的 都是用来逼近目标函数的.

牛顿迭代法是基础,但是计算复杂度太高,后面的都是对牛顿法的精简版,去模拟牛顿法,模拟牛顿法要满足 拟牛顿条件

二,详述: 牛顿迭代法 (更详细化简见博客)

1,简介

利用一阶导,二阶导去求解,寻找在当前状态/当前参数 x_k 下,进一步逼近极值点的值 x_(k+1) ,有点类似sgd,但是求解的原理完全不一样

2,数学推导 (以特征维度只有1为例,也就是自变量只有个x,x是一个数)

在当前状态/当前参数 x_k 的附近,做二阶泰勒展开

设 x_k 为当前的极小点估计值, 则

$$\varphi(x) = f(x_k) + f'(x_k)(x - x_k) + \frac{1}{2}f''(x_k)(x - x_k)^2 \tag{1.2}$$

对φ(x) 求导,令导数=0　一系列化简,得出

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

,那么如果从第0轮开始,每轮都做如此的迭代,则:

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}, \quad k = 0, 1, \cdots$$

3,数学推导 (拓展到特征维度N为例,也就是自变量是x, x是一个N维向量)

将一阶导,二阶导表示出来

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_N} \end{bmatrix}, \quad \nabla^2 f = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_N} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_N} \\ & & \ddots & \\ \frac{\partial^2 f}{\partial x_N \partial x_1} & \frac{\partial^2 f}{\partial x_N \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_N^2} \end{bmatrix}_{N \times N}$$

g 表示 gradient, H 表示 Hessian

g表示一阶导的向量 H表示二阶导的矩阵 叫做海森矩阵

如果矩阵中那些混合偏导是可以交换顺序的,(我理解是 这些的 特征的各个维度之间是互相独立的) ,那么 海森矩阵H就是对称矩阵

同上化简, 可以得到

$$\mathrm{x}_{k+1} = \mathrm{x}_k - H_k^{-1} \cdot \mathrm{g}_k, \quad k = 0, 1, \cdots$$

这就得到了最原始的牛顿迭代法

4,算法过程

算法 1.1 (牛顿法)

1. 给定初值 x_0 和精度阈值 ϵ, 并令 $k := 0$.
2. 计算 g_k 和 H_k.
3. 若 $\|\mathrm{g}_k\| < \epsilon$, 则停止迭代; 否则确定搜索方向 $\mathrm{d}_k = -H_k^{-1} \cdot \mathrm{g}_k$.
4. 计算新的迭代点 $\mathrm{x}_{k+1} := \mathrm{x}_k + \mathrm{d}_k$.
5. 令 $k := k + 1$, 转至步 2.

5,阻尼牛顿法 (增加求步长的式子)

由于步长是固定的,牛顿迭代法容易不熟练,跑偏等

所以加入了个 "一维搜索"(line search), 就是每次迭代找出 **最优**的步长

$$\lambda_k = arg \min_{\lambda \in \mathbb{R}} f(\mathrm{x}_k + \lambda \mathrm{d}_k).$$

给出一个**阻尼牛顿法**的完整算法描述.

算法 1.2 (阻尼牛顿法)

1. 给定初值 x_0 和精度阈值 ϵ, 并令 $k := 0$.
2. 计算 g_k 和 H_k.
3. 若 $\|\mathrm{g}_k\| < \epsilon$, 则停止迭代; 否则确定搜索方向 $\mathrm{d}_k = -H_k^{-1} \cdot \mathrm{g}_k$.
4. 利用 (1.13) 得到步长 λ_k, 并令 $\mathrm{x}_{k+1} := \mathrm{x}_k + \lambda_k \mathrm{d}_k$.
5. 令 $k := k + 1$, 转至步 2.

三,详述:牛顿迭代条件

由于牛顿法计算复杂度高,就想办法对牛顿法做近似,主要是对海森矩阵近似,不同的近似方法衍生出了不同的算法, DFP BFGS L-BFGS都是拟牛顿法的.

B 表示对海森矩阵 H 本身的近似, 而用 D 表示对海森矩阵的逆

H^{-1} 的近似, 即 $B \approx H$, $D \approx H^{-1}$

记

$$\mathrm{s}_k = \mathrm{x}_{k+1} - \mathrm{x}_k, \quad \mathrm{y}_k = \mathrm{g}_{k+1} - \mathrm{g}_k$$

拟牛顿条件对海森矩阵做了一定的近似,变成:

$$\mathrm{y}_k = B_{k+1} \cdot \mathrm{s}_k$$

或者

$$\mathrm{s}_k = D_{k+1} \cdot \mathrm{y}_k.$$

四,DPF算法

也是类似这样的迭代形式

$$D_{k+1} = D_k + \Delta D_k, \quad k = 0, 1, 2, \cdots$$

$D_{k+1} = D_k + \Delta D_k, \quad k = 0, 1, 2,$

初始D0一般是单位阵,关键如何构造

ΔD_k

我们采用"待定法",猜

ΔD_k

与 sk yk有某种关系 ,构造类似

$$\Delta D_k = \alpha \mathbf{u}\mathbf{u}^T + \beta \mathbf{v}\mathbf{v}^T$$

的形式 ,

其中, α, β 为待定系数, $\mathbf{u}, \mathbf{v} \in \mathbb{R}^N$ 为待定向量

增加一系列假设来做近似, 经过推导最终得到

$$\Delta D_k = \frac{\mathbf{s}_k \mathbf{s}_k^T}{\mathbf{s}_k^T \mathbf{y}_k} - \frac{D_k \mathbf{y}_k \mathbf{y}_k^T D_k}{\mathbf{y}_k^T D_k \mathbf{y}_k}$$

算法过程:

算法 2.1 (DFP 算法)

1. 给定初值 $\mathbf{x}_0$ 和精度阀值 ϵ, 并令 $D_0 = I$, $k := 0$.

2. 确定搜索方向 $\mathbf{d}_k = -D_k \cdot \mathbf{g}_k$.

3. 利用 (1.13) 得到步长 λ_k, 令 $\mathbf{s}_k = \lambda_k \mathbf{d}_k$, $\mathbf{x}_{k+1} := \mathbf{x}_k + \mathbf{s}_k$

4. 若 $\|\mathbf{g}_{k+1}\| < \epsilon$, 则算法结束.

5. 计算 $\mathbf{y}_k = \mathbf{g}_{k+1} - \mathbf{g}_k$.

6. 计算 $D_{k+1} = D_k + \frac{\mathbf{s}_k \mathbf{s}_k^T}{\mathbf{s}_k^T \mathbf{y}_k} - \frac{D_k \mathbf{y}_k \mathbf{y}_k^T D_k}{\mathbf{y}_k^T D_k \mathbf{y}_k}$.

7. 令 $k := k + 1$, 转至步 2.

五,BFGS 算法

1,简介

BFGS 性能更佳,已成为求解无约束非线性优化的最常用算法之一, 相比DFP只是互换了 sk与yk的位置

2,详述

过程不管了,直接写出推导结果

$$\Delta B_k = \frac{\mathbf{y}_k \mathbf{y}_k^T}{\mathbf{y}_k^T \mathbf{s}_k} - \frac{B_k \mathbf{s}_k \mathbf{s}_k^T B_k}{\mathbf{s}_k^T B_k \mathbf{s}_k}$$

注意B 和D矩阵的含义

算法 2.3 (BFGS 算法 (II))

1. 给定初值 $\mathbf{x}_0$ 和精度阀值 ϵ, 并令 $D_0 = I$, $k := 0$.

2. 确定搜索方向 $\mathbf{d}_k = -D_k \cdot \mathbf{g}_k$.

3. 利用 (1.13) 得到步长 λ_k, 令 $\mathbf{s}_k = \lambda_k \mathbf{d}_k$, $\mathbf{x}_{k+1} := \mathbf{x}_k + \mathbf{s}_k$

4. 若 $\|\mathbf{g}_{k+1}\| < \epsilon$, 则算法结束.

5. 计算 $\mathbf{y}_k = \mathbf{g}_{k+1} - \mathbf{g}_k$.

6. 计算 $D_{k+1} = \left(I - \frac{\mathbf{s}_k \mathbf{y}_k^T}{\mathbf{y}_k^T \mathbf{s}_k}\right) D_k \left(I - \frac{\mathbf{y}_k \mathbf{s}_k^T}{\mathbf{y}_k^T \mathbf{s}_k}\right) + \frac{\mathbf{s}_k \mathbf{s}_k^T}{\mathbf{y}_k^T \mathbf{s}_k}$.

7. 令 $k := k + 1$, 转至步 2.

六,非精确搜索 (求λ时候,不要求是必须最小值了 满足一定条件就行了)

一,简介

求λ时候,不要求是必须最小值了 满足一定条件就行了

二,详述

均采用 (1.13) 来计算步长 λ_k, 其实这是一种**精确搜索**. 实际应用中, 还有像 Wolfe 型搜索、Armijo 搜索以及满足 Goldstein 条件的**非精确搜索**. 这里我们以 Wolfe 搜索为例, 简单做个介绍.

设 $\widetilde{\beta} \in (0, \frac{1}{2})$, $\beta \in (\widetilde{\beta}, 1)$, 所谓的 Wolfe 搜索是指 λ_k 满足如下 **Wolfe 条件** ([7])

$$\begin{cases} f(\mathbf{x}_k + \lambda_k \mathbf{d}_k) & \leq \quad f(\mathbf{x}_k) + \widetilde{\beta} \lambda_k \mathbf{d}_k^T \mathbf{g}_k; \\ \mathbf{d}_k^T \mathbf{g}(\mathbf{x}_k + \lambda_k \mathbf{d}_k) & \geq \quad \beta \mathbf{d}_k^T \mathbf{g}_k. \end{cases} \tag{2.41}$$

带非精确搜索的拟牛顿法的研究是从 1976 年 Powell 的工作开始的, 他证明了带 Wolfe 搜索的 BFGS 算法的全局收敛性和超线性收敛性.

七,L-BFGS

(一),简介

是BFGS的进一步升级版, 主要是由于BFGS等方法要求存的数太多了,大数据的话内存放不下,就做了limited-memory

主要优化是:

1,以前存海森矩阵,现在不存,现算

2,存的sk 和 yk向量 也可以省, 比较小的k对应的向量可以丢弃,(也就是比较旧的历史信息)

(二),详述

1, 之前要计算Dk-1时候,是存Dk,直接用Dk计算的现在要像下面一样用向量的乘代替了

2, 抛弃部分旧的历史信息

$$\begin{aligned} D_{k+1} \quad = \quad & (V_k^T V_{k-1}^T \cdots V_1^T V_0^T) D_0 (V_0 V_1 \cdots V_{k-1} V_k) \\ & + (V_k^T V_{k-1}^T \cdots V_2^T V_1^T)(\rho_0 \mathbf{s}_0 \mathbf{s}_0^T)(V_1 V_2 \cdots V_{k-1} V_k) \end{aligned}$$

$$
\begin{aligned}
&+ (V_k^T V_{k-1}^T \cdots V_3^T V_2^T)(\rho_1 \mathbf{s}_1 \mathbf{s}_1^T)(V_2 V_3 \cdots V_{k-1} V_k) \\
&+ \cdots \\
&+ (V_k^T V_{k-1}^T)(\rho_{k-2} \mathbf{s}_{k-2} \mathbf{s}_{k-2}^T)(V_{k-1} V_k) \\
&+ V_k^T (\rho_{k-1} \mathbf{s}_{k-1} \mathbf{s}_{k-1}^T) V_k \\
&+ \rho_k \mathbf{s}_k \mathbf{s}_k^T.
\end{aligned}
$$

变换成了

$$
\begin{aligned}
D_{k+1} \quad = \quad & (V_k^T V_{k-1}^T \cdots V_{k-m+2}^T V_{k-m+1}^T) D_0 (V_{k-m+1} V_{k-m+2} \cdots V_{k-1} V_k) \\
&+ (V_k^T V_{k-1}^T \cdots V_{k-m+3}^T V_{k-m+2}^T)(\rho_0 \mathbf{s}_0 \mathbf{s}_0^T)(V_{k-m+2} V_{k-m+3} \cdots V_{k-1} V_k) \\
&+ (V_k^T V_{k-1}^T \cdots V_{k-m+4}^T V_{k-m+3}^T)(\rho_1 \mathbf{s}_1 \mathbf{s}_1^T)(V_{k-m+3} V_{k-m+4} \cdots V_{k-1} V_k) \\
&+ \cdots \\
&+ (V_k^T V_{k-1}^T)(\rho_{k-2} \mathbf{s}_{k-2} \mathbf{s}_{k-2}^T)(V_{k-1} V_k) \\
&+ V_k^T (\rho_{k-1} \mathbf{s}_{k-1} \mathbf{s}_{k-1}^T) V_k \\
&+ \rho_k \mathbf{s}_k \mathbf{s}_k^T.
\end{aligned}
$$

3,算法过程

算法运行的时候,不需要进行上面的求解,因为其实算法中,我们只是通过他得到梯度方向就行了,连步长都是现调整的

所以,有如下算法过程,只要求得

$$D_k \mathbf{g}_k$$

就行了

算法 2.4 ($D_k \cdot \mathbf{g}_k$ 的快速算法)

Step 1 初始化.

$$\delta = \begin{cases} 0, & 若\ k \le m \\ k - m, & 若\ k > m \end{cases}; \quad L = \begin{cases} k, & 若\ k \le m \\ m, & 若\ k > m \end{cases}; \quad \mathbf{q}_L = \mathbf{g}_k.$$

Step 2 后向循环.

FOR $i = L-1, L-2, \cdots, 1, 0$ **DO**

{

$j = i + \delta;$

$\alpha_i = \rho_j \mathbf{s}_j^T \mathbf{q}_{i+1};$ // α_i 需要存下来,前向循环要用!

$\mathbf{q}_i = \mathbf{q}_{i+1} - \alpha_i \mathbf{y}_j.$

}

Step 3 前向循环.

$\mathbf{r}_0 = D_0 \cdot \mathbf{q}_0;$

FOR $i = 0, 1, \cdots, L-2, L-1$ **DO**

{

$j = i + \delta;$

$$\beta_j = \rho_j \mathbf{y}_j^T \mathbf{r}_i;$$

$$\mathbf{r}_{i+1} = \mathbf{r}_i + (\alpha_i - \beta_i)\mathbf{s}_j.$$

}

最后算出的 $\mathbf{r}_L$ 即为 $H_k \cdot \mathbf{g}_k$ 的值.

八,应用范围与优缺点 (待续)

1,对目标函数的要求,必须有一阶导,二阶导? 必须是凸函数?

2,牛顿法与梯度下降的对比:

梯度下降只考虑一阶导,只考虑目标函数在迭代点的局部特征

牛顿法考虑二阶导,收敛速度会更快,

牛顿法的缺点:(仅仅是最原始的牛顿法)

1, 对目标函数要求严格, 必须有连续的一阶导,二阶导,海森矩阵必须正定 (正定和对称有啥关系? tmd总是忘)

2, 计算复杂度高

L-BFGS在维基上说,应用有 log-linear (MaxEnt) models and conditional random fields

ℓ_2

-regularization .

伍海洋的新技术,实现硬盘上的东西访问速度像内存一样,这样的话可以直接把海森矩阵存在硬盘吗??

梯度下降缺点:(tornadomeet的博客)

SGD优点：实现简单，当训练样本足够多时优化速度非常快。

SGD缺点：需要人为调整很多参数，比如学习率，收敛准则等。另外，它是序列的方法，不利于GPU并行或分布式处理。

参考资料:

这一套博客讲的很好:

http://blog.csdn.net/itplus/article/details/21896453 牛顿法

http://blog.csdn.net/itplus/article/details/21896619 拟牛顿条件

http://blog.csdn.net/itplus/article/details/21896981 DFP算法

http://blog.csdn.net/itplus/article/details/21897443 BFGS

http://blog.csdn.net/itplus/article/details/21897715 L-BFGS

http://www.cnblogs.com/kemaswill/p/3352898.html L-BFGS 很naive的解释

http://en.wikipedia.org/wiki/Broyden%E2%80%93Fletcher%E2%80%93Goldfarb%E2%80%93Shanno_algorithm 没细看

http://en.wikipedia.org/wiki/Limited-memory_BFGS 大体上上面的博客都涵盖了 这里有应用的介绍 还有些变形

http://www.cnblogs.com/tornadomeet/archive/2013/05/02/3053916.html tornadomeet 的 博客