数值优化（Numerical Optimization）学习系列-无梯度优化（Derivative-Free Optimization）

下一步　于 2015-12-27 18:51:19 发布

概述

在实际应用中，有些目标函数的梯度不容易计算，即使使用有限差分等近似算法，也会因为噪声的存在导致结果不精确。无梯度优化算法（DFO-Derivative-Free Optimization）可以在不计算梯度的情况下进行问题的最优化，主要有两类思路，一是根据目标函数的样本进行拟合，对拟合函数进行最优化；二是用一些 启发式算法。
1. 有限差分和误差
2. 基于模型近似的方法
3. 坐标和模式搜索方法
4. 其他DFO方法
5. 总结

有限差分和误差

有限差分方法在某些情况下可能会有一定的误差，例如如果函数值需要通过随机试验进行模拟，此时会引入人为误差或者仪器误差。
因此对问题进行建模时，将误差引入目标函数中，然后利用有限差分和梯度相关算法进行优化。

$$f(x) = h(x) + \phi(x)$$

其中函数h表示某平滑函数，ϕ表示误差分布函数，该函数可以和参数x有关也可以无关。

对误差进行建模后，然后利用中心有限差分方法，进行梯度的计算

$$\frac{\partial f}{\partial x_i} \approx \frac{f(x + \epsilon e_i) - f(x - \epsilon e_i)}{2\epsilon}$$

噪声水平（Noise Level）定义为：
在x附近噪声最大值。$\eta(x;\phi) = sup_{||z-x||\leq\epsilon}|\phi(z)|$

此时使用有限差分方法，近似误差来源于**固有误差和噪声误差。**

基于模型的方法

主要思路是，在第k步迭代时，基于该点进行模型近似，通过采样推导出模型中的参数，基于该模型进行最优化计算。

二次模型近似

在第k步迭代时，构建一个二次模型进行近似

$$m_k(x_k+p)=c+g^Tp+\frac{1}{2}p^TGp$$

，其中g和G分别表示函数f的一阶和二阶梯度。
由于该模型参数c、g和G都是未知的，因此需要1+n+(n+1)n/2=(n+1)(n+2)/2个未知数需要计算。
所以基于点Xk需要采样这么多个点进行未知数计算。
样本 $Y=y^1,y^2...y^q$，假设该集合中的点值都比x_k大。根据拟合等式 $m_k(y^l)=f(y^l)$
此时可以唯一确定模型m，然后利用信赖域或者梯度方法进行最优化。

> 在实际应用中，我们仅需要更新模型M即可，不用每次都重新计算。可以选择合适方便计算的基函数。

算法过程如下

Algorithm 9.1 (Model-Based Derivative-Free Method).

Choose an interpolation set $Y = \{y^1, y^2, \ldots, y^q\}$ such that the linear system defined by (9.7) is nonsingular, and select x_0 as a point in this set such that $f(x_0) \leq f(y^i)$ for all $y^i \in Y$. Choose an initial trust region radius Δ_0, a constant $\eta \in (0, 1)$, and set $k \leftarrow 0$.

repeat until a convergence test is satisfied:
  Form the quadratic model $m_k(x_k + p)$ that satisfies the interpolation conditions (9.7);
  Compute a step p by approximately solving subproblem (9.9);
  Define the trial point as $x_k^+ = x_k + p$;
  Compute the ratio ρ defined by (9.10);
  if $\rho \geq \eta$
    Replace an element of Y by x_k^+;
    Choose $\Delta_{k+1} \geq \Delta_k$;
    Set $x_{k+1} \leftarrow x_k^+$;
    Set $k \leftarrow k + 1$ and go to the next iteration;
  else if the set Y need not be improved
    Choose $\Delta_{k+1} < \Delta_k$;
    Set $x_{k+1} \leftarrow x_k$;
    Set $k \leftarrow k + 1$ and go to the next iteration;
  end (if)

  Invoke a geometry-improving procedure to update Y:
    at least one of the points in Y is replaced by some other point,
    with the goal of improving the conditioning of (9.7);
  Set $\Delta_{k+1} \leftarrow \Delta_k$;
  Choose $\hat{x}$ as an element in Y with lowest function value;
  Set $x_k^+ \leftarrow \hat{x}$ and recompute ρ by (9.10);
  if $\rho \geq \eta$
    Set $x_{k+1} \leftarrow x_k^+$;
  else
    Set $x_{k+1} \leftarrow x_k$;
  end (if)
  Set $k \leftarrow k + 1$;
end (repeat)

算法过程如下

1. 构建插值集合$Y = y^1, y^2 \ldots y^q$需要保证线性方式的解存在。
2. 求解插值方程
3. 根据二次模型进行最优解计算

4. 根据最优解的效果，决定是否采用该解。
5. 根据一个几何过程更新几何Y。

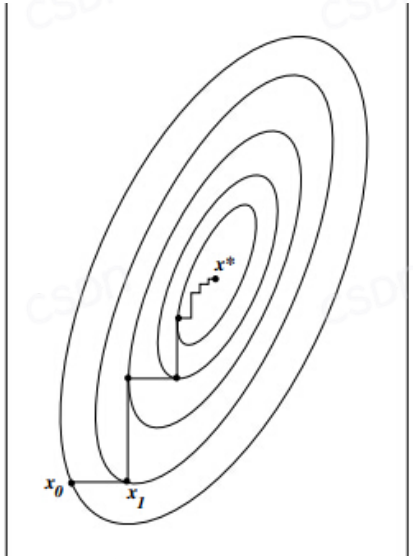
> 二次模型的缺点：样本点选择是O(n^2)的，如果维度越高计算复杂度越大。因此可以考虑线性模型，此时只有O(n+1)个样本需要求解，复杂度会降低。

坐标和模式搜索方法

不同于梯度相关的算法，基于模式搜索方法的搜索方向都是事先确定好的，该方法需要从方向集合中选择一个下降方向作为搜索方向并且更新该方向集合，之后利用线搜索决定步长，逐步迭代得到最优解。
坐标下降是模式搜索方法中的一个特例。

坐标搜索方法（Coordinate SearchMethod）

该方法也称之为坐标下降法或者变量交替方法，主要思路是依次沿着坐标轴方向进行线搜索。
详细过程如下
1. 选择某个迭代点x=(x1,x2…xn)，固定x2…xn，优化x1使得目标函数最小
2. i=2..n 优化x_i使得目标函数最小
3. 重复以上步骤
对于二维情况下，搜索过程如下



> 1. 从上图中可以看出，对于条件数比较大的问题，收敛速度非常低。
> 2. 实际中，如果沿着线性独立的搜索方向搜索，可能不能保证收敛。但是优点是不需要计算梯度，并且对于变量松耦合的情况下，收敛速度可以接受。
> 3. 另外为了进行优化，搜索方向可以选择为$\{e_1, e_2 \dots e_n, e_{n-1} \dots e_1\}$

模式搜索方法

每次搜索方向都是从一个“结构集”中选取，找到某个下降点，进行线搜索，否则修改步长，重复该过程。

Algorithm 9.2 (Pattern-Search).
Given convergence tolerance γ_{tol}, contraction parameter $\theta_{\max}$,
sufficient decrease function $\rho : [0, \infty) \to \mathbf{R}$ with $\rho(t)$ an increasing function of t and $\rho(t)/t \to 0$ as $t \downarrow 0$;
Choose initial point x_0, initial step length $\gamma_0 > \gamma_{\text{tol}}$, initial direction set $\mathcal{D}_0$;
for $k = 1, 2, \ldots$
  if $\gamma_k \leq \gamma_{\text{tol}}$
    stop;
  if $f(x_k + \gamma_k p_k) < f(x_k) - \rho(\gamma_k)$ for some $p_k \in \mathcal{D}_k$
    Set $x_{k+1} \leftarrow x_k + \gamma_k p_k$ for some such p_k;
    Set $\gamma_{k+1} \leftarrow \phi_k \gamma_k$ for some $\phi_k \geq 1$; (* increase step length *)
  else
    Set $x_{k+1} \leftarrow x_k$;
    Set $\gamma_{k+1} \leftarrow \theta_k \gamma_k$, where $0 < \theta_k \leq \theta_{\max} < 1$;
    end (if)
end (for)

该方法会受到噪声点、函数值不精确、不平滑的影响。算法过程如下
算法描述如下
定义
* $\mathcal{D}_k$表示第k迭代的方向集合
* γ_k表示第k步线性搜索参数，即步长，如果找到下降方向，则$x_k + \gamma_k p_k$为最优点
* $\rho(t)$为递增函数，并且当t接近0时，该函数值为0
算法过程
1. 初始化搜索方向集合$\mathcal{D}_0$
2. 循环迭代一下过程，直到搜索步长满足给定阈值。
3. 如果找到满足一定下降条件的搜索方向，则修改最优值点，并且增大步长。
4. 否则减少步长
关键点

1. 初始化搜索方向集合$\mathcal{D}_0$如何选取，需要保证包含最优解的方向。
2. 有理论保证如果搜索方向满足一下条件，则一定能保证收敛。

$$\kappa(\mathcal{D}_k) = \min_{v \in R^n} \max_{p \in \mathcal{D}_k} \frac{v^T p}{||v||||p||} \geq \delta$$

$$\beta_{min} \leq ||p|| \leq \beta_{max}\ p \in \mathcal{D}_k$$

3. 条件1说明需要保证最少有一个搜索方向和最优方向的夹角小于90，即cos(θ) > δ，不能再相反的方向，否则不容易收敛。
4. 条件2说明搜索方向的模不能相差太大，因此搜索步长统一进行缩放。
5. 满足条件的搜索方向有 $\{e_1, e_2 \dots e_n, -e_1 \dots -e_n\}$,供2n个搜索方向或者$\{p_i = \frac{1}{2n}e - e_i, p_{n+1} = \frac{1}{2n}e\}$，供n+1个点
6. 递增函数可以选择为$\rho(t) = Mt^{3/2}$

其他DFO算法

共轭方向算法

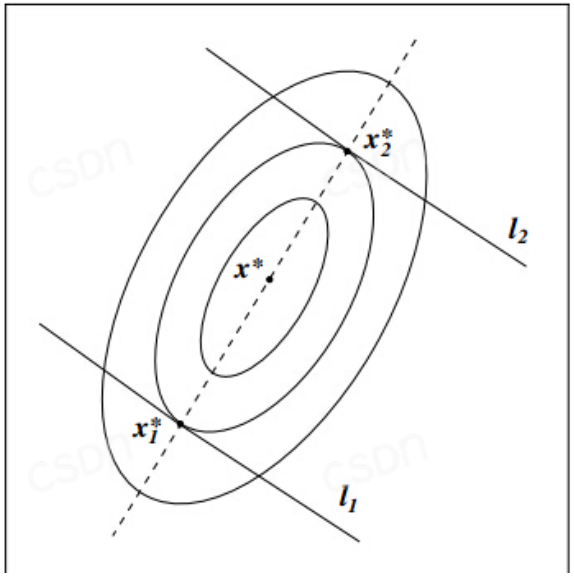
类似于共轭梯度方法，该方法的目标是最优化

$$f(x) = \frac{1}{2}x^T Ax - b^T x$$

，不同点在于共轭方向的计算仅仅依靠函数值得到，不依赖梯度的计算。

Parallel subspace property

通过该方法可以找到一系列共轭方向，并且沿着该方向可以得到最优解，以二维情况为例



如上图如果直线l1和l2平行，并且x1*和x2*是目标函数沿着该直线的最优解，则x1*-x2*共轭于直线的法向量。
因此只要沿着某两个平行子空间寻找最优解，则最优解的差就共轭于该平面的法向量。
假设$\{p_1, p_2 \dots p_l\}$是线性独立的向量，定义两个平行平面

$$s_1 = \{x_1 + \sum_{i=1..l} \alpha_i p_i\}$$

$$s_2 = \{x_2 + \sum_{i=1..l} \alpha_i p_i\}$$

并且目标函数沿着该平面的最优解分布为x1*和x2*，则x2*-x1*共轭于 $p_1, p_2 \ldots p_l$

证明很简单
由于x1*是最优解，则有

$$\frac{\partial f(x_1^* + \alpha_i p_i)}{\partial \alpha_i} = \partial f(x_1^* + \alpha_i p_i) p_i$$

，当 $\alpha_i = 0$，$\nabla f(x_1^*) p_i = 0$，根据最优化条件得到

$$0 = (\nabla f(x_1^*) - \nabla f(x_2^*)) p_i = (Ax_1 - b - Ax_2 + b) p_i = (x_1 - x_2) A p_i$$

根据共轭条件可以得到。

Nelder-Mead 方法

也叫做Nelder-Mead simplex reflection方法。
保存n+1个点，并且这些点构成一个单纯性，在每次循环中搜索使得函数值最低的点，去掉后，用其他更好的点替代。

Implicit Filtering方法

对比于带有噪声的有限微分方法，适用于noise level随着迭代减小的情形。

总结

通过该小结的学习，可以了解到
1. 对于梯度不可求的复杂函数，可以通过DFO的方式进行优化
2. 通过随机试验估计函数值的最优化问题，可以考虑带噪声的有限差分。
3. 了解基于模型的方法，但是复杂度可能会比较大
4. 了解坐标下降法和模式搜索算法
5. 了解基于共轭方向等其他方法。