Practical Optimization Algorithms

徐 翔

数学科学学院
浙江大学

May 17, 2022

第九讲：非线性约束优化问题算法基础

简介(Introduction)

简介

- 一般形式：

$$\min_{x\in R^n} f(x),$$
$$s.t.\ \begin{cases} c_i(x) = 0, & i \in \mathcal{E} \\ c_i(x) \geq 0, & i \in \mathcal{I}. \end{cases}$$

简介

- 一般形式：

$$\min_{x \in R^n} f(x),$$
$$s.t. \quad \begin{cases} c_i(x) = 0, & i \in \mathcal{E} \\ c_i(x) \geq 0, & i \in \mathcal{I}. \end{cases}$$

- 讨论一些算法的基本概念而不是具体某个算法

简介

- 一般形式：

$$\min_{x\in R^n} f(x),$$
$$s.t. \begin{cases} c_i(x) = 0, & i \in \mathcal{E} \\ c_i(x) \geq 0, & i \in \mathcal{I}. \end{cases}$$

- 讨论一些算法的基本概念而不是具体某个算法
- 优化问题的算法分类 (Categorize)

简介

- 一般形式：

$$\min_{x\in R^n} f(x),$$
$$s.t. \begin{cases} c_i(x)=0, & i\in\mathcal{E} \\ c_i(x)\geq 0, & i\in\mathcal{I}. \end{cases}$$

- 讨论一些算法的基本概念而不是具体某个算法
- 优化问题的算法分类 (Categorize)
- 不等式约束问题的组合困难 (Combinatorial difficulty)

简介

- 一般形式：

$$\min_{x\in R^n} f(x),$$
$$s.t. \begin{cases} c_i(x) = 0, & i \in \mathcal{E} \\ c_i(x) \geq 0, & i \in \mathcal{I}. \end{cases}$$

- 讨论一些算法的基本概念而不是具体某个算法
- 优化问题的算法分类 (CATEGORIZE)
- 不等式约束问题的组合困难 (COMBINATORIAL DIFFICULTY)
- 消元法(ELIMINATION OF VARIABLES)

简介

- 一般形式：

$$\min_{x \in R^n} f(x),$$
$$s.t. \begin{cases} c_i(x) = 0, & i \in \mathcal{E} \\ c_i(x) \geq 0, & i \in \mathcal{I}. \end{cases}$$

- 讨论一些算法的基本概念而不是具体某个算法
- 优化问题的算法分类 (Categorize)
- 不等式约束问题的组合困难 (Combinatorial difficulty)
- 消元法(Elimination of Variables)
- 价值函数和Filters (Merit Functions and Filters)

分类

一阶最优性条件（KKT条件）刻画了最优解的特征，但不是有效的求解算法。

分类

一阶最优性条件（KKT条件）刻画了最优解的特征，但不是有效的求解算法。算法通常需要结合优化问题的特点，以及目标函数和约束函数的结构。 有一些特殊的情形，有非常高效的算法。

分类

一阶最优性条件（KKT条件）刻画了最优解的特征，但不是有效的求解算法。算法通常需要结合优化问题的特点，以及目标函数和约束函数的结构。有一些特殊的情形，有非常高效的算法。

- 线性规划(Linear programming)：目标函数f和约束条件c_i都是线性函数

分类

一阶最优性条件（KKT条件）刻画了最优解的特征，但不是有效的求解算法。算法通常需要结合优化问题的特点，以及目标函数和约束函数的结构。有一些特殊的情形，有非常高效的算法。

- 线性规划(Linear programming)：目标函数f和约束条件c_i都是线性函数
- 二次规划(Quadratic programming): 目标函数f是二次函数，约束条件c_i是线性函数

分类

一阶最优性条件（KKT条件）刻画了最优解的特征，但不是有效的求解算法。算法通常需要结合优化问题的特点，以及目标函数和约束函数的结构。有一些特殊的情形，有非常高效的算法。

- 线性规划(Linear programming)：目标函数f和约束条件c_i都是线性函数
- 二次规划(Quadratic programming): 目标函数f是二次函数，约束条件c_i是线性函数
- 线性约束优化(Linearly constrained optimization): 所有约束条件c_i都是线性函数

分类

一阶最优性条件（KKT条件）刻画了最优解的特征，但不是有效的求解算法。算法通常需要结合优化问题的特点，以及目标函数和约束函数的结构。有一些特殊的情形，有非常高效的算法。

- 线性规划(Linear programming)：目标函数f和约束条件c_i都是线性函数
- 二次规划(Quadratic programming): 目标函数f是二次函数，约束条件c_i是线性函数
- 线性约束优化(Linearly constrained optimization): 所有约束条件c_i都是线性函数
- 非线性规划(Nonlinear programming): 约束条件中至少有一个是非线性函数c_i

分类

一阶最优性条件（KKT条件）刻画了最优解的特征，但不是有效的求解算法。算法通常需要结合优化问题的特点，以及目标函数和约束函数的结构。 有一些特殊的情形，有非常高效的算法。

- 线性规划(Linear programming)：目标函数f和约束条件c_i都是线性函数
- 二次规划(Quadratic programming): 目标函数f是二次函数，约束条件c_i是线性函数
- 线性约束优化(Linearly constrained optimization): 所有约束条件c_i都是线性函数
- 非线性规划(Nonlinear programming): 约束条件中至少有一个是非线性函数c_i
- 有界约束优化(Bound-constrained optimization): 所有的约束都是$I_i \leq x_i \leq U_i$的形式，分别称为第i个分量的下界和上界。

分类

一阶最优性条件（KKT条件）刻画了最优解的特征，但不是有效的求解算法。算法通常需要结合优化问题的特点，以及目标函数和约束函数的结构。有一些特殊的情形，有非常高效的算法。

- 线性规划(Linear programming)：目标函数f和约束条件c_i都是线性函数
- 二次规划(Quadratic programming): 目标函数f是二次函数，约束条件c_i是线性函数
- 线性约束优化(Linearly constrained optimization): 所有约束条件c_i都是线性函数
- 非线性规划(Nonlinear programming): 约束条件中至少有一个是非线性函数c_i
- 有界约束优化(Bound-constrained optimization): 所有的约束都是$I_i \le x_i \le U_i$的形式，分别称为第i个分量的下界和上界。
- 凸规划(Convex programming): 目标函数f是凸的，等式约束都是线性的$i \in \mathcal{E}$，不等式约束都是凹的$i \in \mathcal{I}$。

这些分类并不是最小子分类，也不是互相排斥的。通常分的越细，越容易设计算法。

简介

- 通常约束优化问题算法也是迭代算法

简介

- 通常约束优化问题算法也是迭代算法
- 算法通常会产生一个序列x_k，收敛到精确解x^*。同时还会产生另一个序列$\lambda_i^{(k)}$，收敛到精确的拉格朗日乘子λ^*。

简介

- 通常约束优化问题算法也是迭代算法
- 算法通常会产生一个序列x_k，收敛到精确解x^*。同时还会产生另一个序列$\lambda_i^{(k)}$，收敛到精确的拉格朗日乘子λ^*。
- 如何从当前状态$(x_k, \lambda_i^{(k)})$，计算下一个状态$(x_{k+1}, \lambda_i^{(k+1)})$，通常需要用到$f(x_k)$, $c_i(x_k)$以及他们的导数值$\nabla f(x_k)$, $\nabla c_i(x_k)$，甚至有可能前几步的函数和导数值。

简介

- 通常约束优化问题算法也是迭代算法
- 算法通常会产生一个序列x_k，收敛到精确解x^*。同时还会产生另一个序列$\lambda_i^{(k)}$，收敛到精确的拉格朗日乘子λ^*。
- 如何从当前状态$(x_k, \lambda_i^{(k)})$，计算下一个状态$(x_{k+1}, \lambda_i^{(k+1)})$，通常需要用到$f(x_k)$, $c_i(x_k)$以及他们的导数值$\nabla f(x_k)$, $\nabla c_i(x_k)$，甚至有可能前几步的函数和导数值。
- 如何停机？要么已经找到在容忍误差范围的解，要么已经无法改进了。

问题初探

- 首先要看一下是否可以简化，或者可以直接判断。（例如可行集是空集，或者目标函数在可行集中无界）

问题初探

- 首先要看一下是否可以简化，或者可以直接判断。（例如可行集是空集，或者目标函数在可行集中无界）
- 是否可以直接求解KKT条件。（如果知道哪些是活跃集，可以直接求解，当然很多时候仍然是数值求解）

问题初探

- 首先要看一下是否可以简化，或者可以直接判断。（例如可行集是空集，或者目标函数在可行集中无界）
- 是否可以直接求解KKT条件。（如果知道哪些是活跃集，可以直接求解，当然很多时候仍然是数值求解）
- 约束条件是否有隐含条件。一般可以分为“硬”约束条件和“软”约束条件。

问题初探

- 首先要看一下是否可以简化，或者可以直接判断。（例如可行集是空集，或者目标函数在可行集中无界）
- 是否可以直接求解KKT条件。（如果知道哪些是活跃集，可以直接求解，当然很多时候仍然是数值求解）
- 约束条件是否有隐含条件。一般可以分为“硬”约束条件和“软”约束条件。
- 从算法的角度看，“硬”是指x必须要满足的条件，否则目标函数或者约束条件可能无意义。这些条件很有可能没有显式的给出。比如，
  - 一个变量必须是非负的，因为在目标函数或约束函数中需要求其平方根。
  - 再比如有些变量是分数比，自然满足求和等于1。

问题初探

- 首先要看一下是否可以简化，或者可以直接判断。（例如可行集是空集，或者目标函数在可行集中无界）
- 是否可以直接求解KKT条件。（如果知道哪些是活跃集，可以直接求解，当然很多时候仍然是数值求解）
- 约束条件是否有隐含条件。一般可以分为“硬”约束条件和“软”约束条件。
- 从算法的角度看，“硬”是指x必须要满足的条件，否则目标函数或者约束条件可能无意义。这些条件很有可能没有显式的给出。比如，
    - 一个变量必须是非负的，因为在目标函数或约束函数中需要求其平方根。
    - 再比如有些变量是分数比，自然满足求和等于1。
- “软”约束条件，通常指可以引入一个惩罚项到目标函数中，把约束优化问题转化为非约束优化问题。这种引入罚函数的办法，一般会引入坏条件数(ill-conditioning)，有时会对算法设计带来额外困难。

问题初探

可行算法 (feasible algorithm)

- 当有“硬”约束条件时，所有的迭代过程，必须满足“硬”约束条件。

问题初探

可行算法 (feasible algorithm)

- 当有“硬”约束条件时，所有的迭代过程，必须满足“硬”约束条件。
- 相比于不可行算法(infeasible algorithm)，可行算法通常比较慢，而且计算量大。

问题初探

可行算法 (feasible algorithm)

- 当有“硬”约束条件时，所有的迭代过程，必须满足“硬”约束条件。
- 相比于不可行算法(infeasible algorithm)，可行算法通常比较慢，而且计算量大。
- 可行算法不能穿越不可行区域走捷径。

问题初探

可行算法 (feasible algorithm)

- 当有“硬”约束条件时，所有的迭代过程，必须满足“硬”约束条件。
- 相比于不可行算法(infeasible algorithm)，可行算法通常比较慢，而且计算量大。
- 可行算法不能穿越不可行区域走捷径。
- 优点是每一步都可以计算目标函数。不需要引入其他复杂价值函数(Merit function)来考虑不可行区域上的约束不成立。

不等式约束问题的组合困难

- 非线性约束优化问题中的一个主要挑战是不等式约束，即确定哪些是活跃的，哪些是不活跃的。

不等式约束问题的组合困难

- 非线性约束优化问题中的一个主要挑战是不等式约束，即确定哪些是活跃的，哪些是不活跃的。
- 有一种方法（积极集方法的本质）

不等式约束问题的组合困难

- 非线性约束优化问题中的一个主要挑战是不等式约束，即确定哪些是活跃的，哪些是不活跃的。
- 有一种方法（积极集方法的本质）
  - 从最优活跃集$\mathcal{A}^*$（所有的等式约束集）的一个猜测出发，记为工作集$\mathcal{W}$

不等式约束问题的组合困难

- 非线性约束优化问题中的一个主要挑战是不等式约束，即确定哪些是活跃的，哪些是不活跃的。
- 有一种方法（积极集方法的本质）
  - 从最优活跃集$\mathcal{A}^*$（所有的等式约束集）的一个猜测出发，记为工作集$\mathcal{W}$
  - 在工作集上求解最优问题，是否存在拉格朗日乘子，使得局部解x^*存在。

不等式约束问题的组合困难

- 非线性约束优化问题中的一个主要挑战是不等式约束，即确定哪些是活跃的，哪些是不活跃的。
- 有一种方法（积极集方法的本质）
  - 从最优活跃集$\mathcal{A}^*$（所有的等式约束集）的一个猜测出发，记为工作集$\mathcal{W}$
  - 在工作集上求解最优问题，是否存在拉格朗日乘子，使得局部解x^*存在。
  - 如果不存在，调整工作集$\mathcal{W}$，重复以上过程。

不等式约束问题的组合困难

- 非线性约束优化问题中的一个主要挑战是不等式约束，即确定哪些是活跃的，哪些是不活跃的。
- 有一种方法（积极集方法的本质）
  - 从最优活跃集$\mathcal{A}^*$（所有的等式约束集）的一个猜测出发，记为工作集$\mathcal{W}$
  - 在工作集上求解最优问题，是否存在拉格朗日乘子，使得局部解x^*存在。
  - 如果不存在，调整工作集$\mathcal{W}$，重复以上过程。
  - 因为总的说来，求解等式约束要比不等式约束更容易些。

不等式约束问题的组合困难

- 工作集$\mathcal{W}$的可能性很多，可以由$2^{|\mathcal{I}|}$, $|\mathcal{I}|$是不等式约束的个数。

不等式约束问题的组合困难

- 工作集$\mathcal{W}$的可能性很多，可以由$2^{|\mathcal{I}|}$, $|\mathcal{I}|$是不等式约束的个数。
- 运算量会随着不等式约束的个数指数增长。

不等式约束问题的组合困难

- 工作集$\mathcal{W}$的可能性很多，可以由$2^{|\mathcal{I}|}$, $|\mathcal{I}|$是不等式约束的个数。
- 运算量会随着不等式约束的个数指数增长。
- 不可能设计出实用的计算方法。

不等式约束问题的组合困难

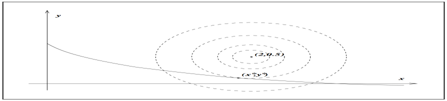
例1：考虑如下的问题

$$
\begin{aligned}
\min_{x,y} f(x,y) \equiv \frac{1}{2}(x-2)^2 + \frac{1}{2}(y-\frac{1}{2})^2 \\
s.t. \qquad (x+1)^{-1} - y - \frac{1}{4} \geq 0, \\
x \geq 0, \\
y \geq 0.
\end{aligned}
$$

不等式约束问题的组合困难

例1：考虑如下的问题

$$
\begin{aligned}
\min_{x,y} f(x,y) \equiv \frac{1}{2}(x-2)^2 + \frac{1}{2}(y-\frac{1}{2})^2 \\
s.t. \qquad (x+1)^{-1} - y - \frac{1}{4} \geq 0, \\
x \geq 0, \\
y \geq 0.
\end{aligned}
$$



精确解在$(x^*, y^*) = (1.953, 0.089)$。

不等式约束问题的组合困难

不等式约束问题的组合困难

- 首先，三个约束条件不能同时为活跃条件，因为这样的可行集为空集。

不等式约束问题的组合困难

- 首先，三个约束条件不能同时为活跃条件，因为这样的可行集为空集。
- 其次，如果有两个约束条件为活跃条件，有三种情况，分别为

不等式约束问题的组合困难

- 首先，三个约束条件不能同时为活跃条件，因为这样的可行集为空集。
- 其次，如果有两个约束条件为活跃条件，有三种情况，分别为
  - $\mathcal{W} = \{1, 2\}$，则可行集为$(0, \frac{1}{4})$.

不等式约束问题的组合困难

- 首先，三个约束条件不能同时为活跃条件，因为这样的可行集为空集。
- 其次，如果有两个约束条件为活跃条件，有三种情况，分别为
  - $\mathcal{W}=\{1,2\}$，则可行集为$(0,\frac{1}{4})$.
  - $\mathcal{W}=\{1,3\}$，则可行集为$(3,0)$.

不等式约束问题的组合困难

- 首先，三个约束条件不能同时为活跃条件，因为这样的可行集为空集。
- 其次，如果有两个约束条件为活跃条件，有三种情况，分别为
  - $\mathcal{W}=\{1,2\}$，则可行集为$(0,\frac{1}{4})$.
  - $\mathcal{W}=\{1,3\}$，则可行集为$(3,0)$.
  - $\mathcal{W}=\{2,3\}$，则可行集为$(0,0)$.

不等式约束问题的组合困难

- 首先，三个约束条件不能同时为活跃条件，因为这样的可行集为空集。
- 其次，如果有两个约束条件为活跃条件，有三种情况，分别为
  - $\mathcal{W}=\{1,2\}$，则可行集为$(0,\frac{1}{4})$.
  - $\mathcal{W}=\{1,3\}$，则可行集为$(3,0)$.
  - $\mathcal{W}=\{2,3\}$，则可行集为$(0,0)$.
- 紧接着，如果只有1个约束条件为活跃的，有三种情况，分别为

不等式约束问题的组合困难

- 首先，三个约束条件不能同时为活跃条件，因为这样的可行集为空集。
- 其次，如果有两个约束条件为活跃条件，有三种情况，分别为
  - $\mathcal{W}=\{1,2\}$，则可行集为$(0,\frac{1}{4})$.
  - $\mathcal{W}=\{1,3\}$，则可行集为$(3,0)$.
  - $\mathcal{W}=\{2,3\}$，则可行集为$(0,0)$.
- 紧接着，如果只有1个约束条件为活跃的，有三种情况，分别为
  - $\mathcal{W}=\{3\}$，则可行集为x轴，此时解为$(x^*,y^*)=(2,0)$，在这一点，无法找到合适的Lagrange乘子满足所有的KKT条件。因为$\lambda_1=\lambda_2=0$.

不等式约束问题的组合困难

- 首先，三个约束条件不能同时为活跃条件，因为这样的可行集为空集。
- 其次，如果有两个约束条件为活跃条件，有三种情况，分别为
  - $\mathcal{W} = \{1, 2\}$，则可行集为$(0, \frac{1}{4})$.
  - $\mathcal{W} = \{1, 3\}$，则可行集为$(3, 0)$.
  - $\mathcal{W} = \{2, 3\}$，则可行集为$(0, 0)$.
- 紧接着，如果只有1个约束条件为活跃的，有三种情况，分别为
  - $\mathcal{W} = \{3\}$，则可行集为x轴，此时解为$(x^*, y^*) = (2, 0)$，在这一点，无法找到合适的Lagrange乘子满足所有的KKT条件。因为$\lambda_1 = \lambda_2 = 0$.
  - $\mathcal{W} = \{2\}$, 则可行集为y轴，此时解为$(x^*, y^*) = (0, \frac{1}{2})$，经检验，在该点无法找到合适的Lagrange乘子满足所有的KKT条件。因为$\lambda_1 = \lambda_3 = 0$.

不等式约束问题的组合困难

- 首先，三个约束条件不能同时为活跃条件，因为这样的可行集为空集。
- 其次，如果有两个约束条件为活跃条件，有三种情况，分别为
  - $\mathcal{W}=\{1,2\}$，则可行集为$(0,\frac{1}{4})$.
  - $\mathcal{W}=\{1,3\}$，则可行集为$(3,0)$.
  - $\mathcal{W}=\{2,3\}$，则可行集为$(0,0)$.
- 紧接着，如果只有1个约束条件为活跃的，有三种情况，分别为
  - $\mathcal{W}=\{3\}$，则可行集为x轴，此时解为$(x^*,y^*)=(2,0)$，在这一点，无法找到合适的Lagrange乘子满足所有的KKT条件。因为$\lambda_1=\lambda_2=0$.
  - $\mathcal{W}=\{2\}$, 则可行集为y轴，此时解为$(x^*,y^*)=(0,\frac{1}{2})$，经检验，在该点无法找到合适的Lagrange乘子满足所有的KKT条件。因为$\lambda_1=\lambda_3=0$.
  - $\mathcal{W}=\{1\}$， 可以找到解$(1.953,0.089)$及$\lambda_1=0.411,\ \lambda_2=\lambda_3=0$。满足所有的KKT条件。

不等式约束问题的组合困难

不等式约束问题的组合困难

- 这个例子告诉我们，即使数量很少的不等式约束，也很难确定活跃集。

不等式约束问题的组合困难

- 这个例子告诉我们，即使数量很少的不等式约束，也很难确定活跃集。
- 有时可以利用目标函数、约束条件及他们的导数，排除一些显而易见的情况。

不等式约束问题的组合困难

- 这个例子告诉我们，即使数量很少的不等式约束，也很难确定活跃集。
- 有时可以利用目标函数、约束条件及他们的导数，排除一些显而易见的情况。
- 事实上，积极集方法就是根据这些信息，构造一些好的工作集猜测，避免了那些显然得不到解的工作集。

不等式约束问题的组合困难

- 这个例子告诉我们，即使数量很少的不等式约束，也很难确定活跃集。
- 有时可以利用目标函数、约束条件及他们的导数，排除一些显而易见的情况。
- 事实上，积极集方法就是根据这些信息，构造一些好的工作集猜测，避免了那些显然得不到解的工作集。
- 还有一种内点法(barrier方法)也可以避免这种组合指数级困难。这种方法产生的序列总是不会靠近不等式约束定义的边界。

利用线性约束条件简单消元

利用线性约束条件简单消元

- 一个自然的方法：将约束条件消去，把约束优化问题转化为非约束优化问题。

利用线性约束条件简单消元

- 一个自然的方法：将约束条件消去，把约束优化问题转化为非约束优化问题。
- 或者消去部分约束条件

利用线性约束条件简单消元

- 一个自然的方法：将约束条件消去，把约束优化问题转化为非约束优化问题。
- 或者消去部分约束条件
- 消去变量或约束条件时要很小心，因为有可能会改变问题或引入坏条件数问题。一个安全的消元

$$
\begin{aligned}
&\min f(x_1, x_2, x_3, x_4, x_5)\\
s.t.\quad & x_1 + x_3^2 - x_4 x_5 = 0,\\
& -x_2 + x_4 + x_3^2 = 0.
\end{aligned}
$$

可以设置$x_1 = x_4 x_5 - x_3^2$，$x_2 = x_3^2 + x_4$，将原问题转化为无约束优化问题

$$\min h(x_3, x_4, x_5) = f(x_4 x_5 - x_3^2, x_3^2 + x_4, x_3, x_4, x_5)$$

非线性约束消元

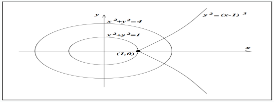
考虑如下问题

$$\min x^2 + y^2 \quad s.t. \quad (x-1)^3 = y^2.$$

非线性约束消元

考虑如下问题

$$\min x^2 + y^2 \quad s.t. \quad (x-1)^3 = y^2.$$

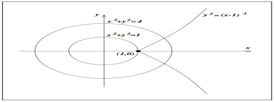


可以看出，问题的解是$(1,0)$。

非线性约束消元

考虑如下问题

$$\min x^2 + y^2 \quad s.t. \quad (x-1)^3 = y^2.$$



可以看出，问题的解是$(1,0)$。
但是如果消去y，转为无约束优化问题

$$\min h(x) = x^2 + (x-1)^3$$

很显然当$x \to -\infty$时，$h(x) \to -\infty$。

利用线性约束条件简单消元

考虑如下问题

$$\min \quad f(x) \quad s.t. \quad Ax = b,$$

其中 $A \in R^{m\times n}$ 且 $m \leq n$.

利用线性约束条件简单消元

考虑如下问题

$$\min \quad f(x) \quad s.t. \quad Ax = b,$$

其中$A \in R^{m\times n}$且 $m \leq n$. 假设 $rank(A) = m$，即可以找到m个线性无关列。

利用线性约束条件简单消元

考虑如下问题

$$\min \quad f(x) \quad s.t. \quad Ax = b,$$

其中$A \in R^{m\times n}$且 $m \leq n$. 假设 $rank(A) = m$，即可以找到m个线性无关列。可以找一个$n \times n$的初等列交换矩阵P，使得前m列正好线性无关。

利用线性约束条件简单消元

考虑如下问题

$$\min \quad f(x) \quad s.t. \quad Ax = b,$$

其中$A \in R^{m\times n}$且 $m \le n$. 假设 $rank(A) = m$，即可以找到m个线性无关列。可以找一个$n \times n$的初等列交换矩阵P，使得前m列正好线性无关。

$$AP = [B|N],$$

其中N表示A中剩下的$n-m$列。把x中的分量也分为 $x_B \in R^m$, $x_N \in R^{n-m}$, 并且

$$\begin{bmatrix} x_B \\ x_N \end{bmatrix} = P^T x,$$

这里把x_B称为基本变量， B称为基矩阵。

利用线性约束条件简单消元

注意到 $PP^T = I$, 我们可以重写约束条件$Ax = b$

$$b = Ax = AP(P^Tx) = Bx_B + Nx_N.$$

利用线性约束条件简单消元

注意到 $PP^T = I$, 我们可以重写约束条件$Ax = b$

$$b = Ax = AP(P^T x) = Bx_B + Nx_N.$$

这样，我们可以得到基本变量

$$x_B = B^{-1}b - B^{-1}Nx_N.$$

利用线性约束条件简单消元

注意到 $PP^T = I$, 我们可以重写约束条件$Ax = b$

$$b = Ax = AP(P^T x) = Bx_B + Nx_N.$$

这样，我们可以得到基本变量

$$x_B = B^{-1}b - B^{-1}Nx_N.$$

然后，我们可以通过任选x_N，通过上面的公式得到x_B，自动满足约束条件。

利用线性约束条件简单消元

注意到 $PP^T = I$, 我们可以重写约束条件$Ax = b$

$$b = Ax = AP(P^T x) = Bx_B + Nx_N.$$

这样，我们可以得到基本变量

$$x_B = B^{-1}b - B^{-1}Nx_N.$$

然后，我们可以通过任选x_N，通过上面的公式得到x_B，自动满足约束条件。而相应的约束优化问题等价于

$$\min_{x_N} \quad h(x_N) \equiv f\left(P\begin{bmatrix} B^{-1}b - B^{-1}Nx_N \\ x_N \end{bmatrix}\right)$$

利用线性约束条件简单消元

注意到 $PP^T = I$, 我们可以重写约束条件$Ax = b$

$$b = Ax = AP(P^T x) = Bx_B + Nx_N.$$

这样，我们可以得到基本变量

$$x_B = B^{-1}b - B^{-1}Nx_N.$$

然后，我们可以通过任选x_N，通过上面的公式得到x_B，自动满足约束条件。而相应的约束优化问题等价于

$$\min_{x_N} \quad h(x_N) \equiv f\left(P\begin{bmatrix} B^{-1}b - B^{-1}Nx_N \\ x_N \end{bmatrix}\right)$$

我们把这种方法叫做简单消元法。也就是对于一个非线性约束优化问题，如果所有的约束都是线性等式约束，那么其实等价于无约束优化问题。

线性约束条件的简化一般策略

- 选择合适的矩阵 $Y \in R^{n\times m}$ 和 $Z \in R^{n\times(n-m)}$ 满足

$$[Y|Z] \in R^{n\times n} \text{ is nonsingular }, AZ = 0.$$

线性约束条件的简化一般策略

- 选择合适的矩阵 $Y \in R^{n\times m}$ 和 $Z \in R^{n\times(n-m)}$ 满足

$$[Y|Z] \in R^{n\times n} \text{ is nonsingular }, AZ = 0.$$

- Z其实是矩阵A的零空间中的基，组成的矩阵。

线性约束条件的简化一般策略

- 选择合适的矩阵 $Y \in R^{n\times m}$ 和 $Z \in R^{n\times(n-m)}$ 满足

$$[Y|Z] \in R^{n\times n} \text{ is nonsingular }, AZ = 0.$$

- Z其实是矩阵A的零空间中的基，组成的矩阵。
- 由于$x = Yx_Y + Zx_Z$，代入到等式约束中，可以得到

$$Ax = (AY)x_Y = b.$$

线性约束条件的简化一般策略

- 选择合适的矩阵 $Y \in R^{n\times m}$ 和 $Z \in R^{n\times(n-m)}$ 满足

$$[Y|Z] \in R^{n\times n} \text{ is nonsingular }, AZ = 0.$$

- Z其实是矩阵A的零空间中的基，组成的矩阵。
- 由于$x = Yx_Y + Zx_Z$，代入到等式约束中，可以得到

$$Ax = (AY)x_Y = b.$$

- 如果AY是非奇异的，则

$$x_Y = (AY)^{-1}b.$$

线性约束条件的简化一般策略

- 选择合适的矩阵 $Y \in R^{n\times m}$ 和 $Z \in R^{n\times(n-m)}$ 满足

$$[Y|Z] \in R^{n\times n} \text{ is nonsingular }, AZ = 0.$$

- Z其实是矩阵A的零空间中的基，组成的矩阵。
- 由于$x = Yx_Y + Zx_Z$，代入到等式约束中，可以得到

$$Ax = (AY)x_Y = b.$$

- 如果AY是非奇异的，则

$$x_Y = (AY)^{-1}b.$$

这样的话，我们可以得到如下的
无约束优化问题

$$\min_{x_Z} \quad f\big(Y(AY)^{-1}b + Zx_Z\big)$$

线性约束条件的简化一般策略

- 选择合适的矩阵 $Y \in R^{n\times m}$ 和 $Z \in R^{n\times(n-m)}$ 满足

$$[Y|Z] \in R^{n\times n} \text{ is nonsingular }, AZ = 0.$$

- Z其实是矩阵A的零空间中的基，组成的矩阵。
- 由于$x = Yx_Y + Zx_Z$，代入到等式约束中，可以得到

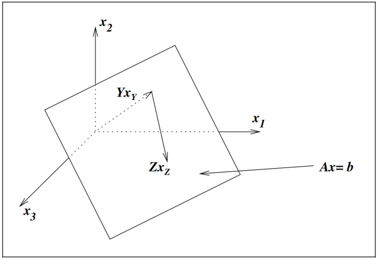
$$Ax = (AY)x_Y = b.$$

- 如果AY是非奇异的，则

$$x_Y = (AY)^{-1}b.$$

这样的话，我们可以得到如下的无约束优化问题

$$\min_{x_Z} \quad f\left(Y(AY)^{-1}b + Zx_Z\right)$$



线性约束条件的简化一般策略

如何选取Y 和 Z?

线性约束条件的简化一般策略

如何选取Y 和 Z?

- 要求是AY的条件数不能太大(well-conditioned)，因为涉及到$(AY)^{-1}$。

线性约束条件的简化一般策略

如何选取Y 和 Z?

- 要求是AY的条件数不能太大(well-conditioned)，因为涉及到$(AY)^{-1}$。
- 通常我们可以通过对A的QR分解得到Y和Z。

线性约束条件的简化一般策略

如何选取Y 和 Z?

- 要求是AY的条件数不能太大(well-conditioned)，因为涉及到$(AY)^{-1}$。
- 通常我们可以通过对A的QR分解得到Y和Z。

$$A^T P = [Q_1 \ \ Q_2] \begin{bmatrix} R \\ 0 \end{bmatrix}$$

其中P是$m \times m$的初等列交换矩阵（满足 $PP^T = I$）， $[Q_1 \ \ Q_2]$是正交矩阵。 R是上三角矩阵。

线性约束条件的简化一般策略

如何选取Y 和 Z?

- 要求是AY的条件数不能太大(well-conditioned)，因为涉及到$(AY)^{-1}$。
- 通常我们可以通过对A的QR分解得到Y和Z。

$$A^T P = [Q_1 \;\; Q_2] \begin{bmatrix} R \\ 0 \end{bmatrix}$$

  其中P是$m \times m$的初等列交换矩阵（满足 $PP^T = I$）， $[Q_1 \;\; Q_2]$是正交矩阵。 R是上三角矩阵。

- 这样我们可以取 $Y = Q_1$, $Z = Q_2$. 显然满足

$$AY = PR^T, AZ = 0$$

线性约束条件的简化一般策略

如何选取Y 和 Z?

- 要求是AY的条件数不能太大(well-conditioned)，因为涉及到$(AY)^{-1}$。
- 通常我们可以通过对A的QR分解得到Y和Z。

$$A^T P = [Q_1 \;\; Q_2] \begin{bmatrix} R \\ 0 \end{bmatrix}$$

  其中P是$m \times m$的初等列交换矩阵（满足 $PP^T = I$），$[Q_1 \;\; Q_2]$是正交矩阵。R是上三角矩阵。
- 这样我们可以取 $Y = Q_1$, $Z = Q_2$. 显然满足

$$AY = PR^T, AZ = 0$$

- 优点一，AY的条件数与R的条件数一样大。

线性约束条件的简化一般策略

如何选取Y 和 Z?

- 要求是AY的条件数不能太大(well-conditioned)，因为涉及到$(AY)^{-1}$。
- 通常我们可以通过对A的QR分解得到Y和Z。

$$A^T P = [Q_1 \;\; Q_2] \begin{bmatrix} R \\ 0 \end{bmatrix}$$

其中P是$m \times m$的初等列交换矩阵（满足 $PP^T = I$），$[Q_1 \;\; Q_2]$是正交矩阵。R是上三角矩阵。

- 这样我们可以取 $Y = Q_1, Z = Q_2$. 显然满足

$$AY = PR^T, AZ = 0$$

- 优点一，AY的条件数与R的条件数一样大。
优点二，$Ax = b$可以转化为$x = Q_1 R^{-T} P^T b + Q_2 x_Z$。计算$R^{-T} P^T b$运算量很小，只需要追赶法中的一小步即可(三角代入，triangular substitution)。

线性约束条件的简化一般策略

- 简单的计算可以得到 $Q_1R^{-T}P^Tb = A^T(AA^T)^{-1}b$.

线性约束条件的简化一般策略

- 简单的计算可以得到 $Q_1R^{-T}P^Tb = A^T(AA^T)^{-1}b$.
- 可以看作是如下最小范数问题的解

$$\min \|x\|_2^2 \quad s.t. \quad Ax = b.$$

线性约束条件的简化一般策略

- 简单的计算可以得到 $Q_1R^{-T}P^Tb = A^T(AA^T)^{-1}b.$
- 可以看作是如下最小范数问题的解

$$\min \|x\|_2^2 \quad s.t. \quad Ax = b.$$

- 从数值计算的角度看，通过正交基分解做消元法是很稳定的，主要的计算量在A的QR分解。

线性约束条件的简化一般策略

- 简单的计算可以得到 $Q_1R^{-T}P^Tb = A^T(AA^T)^{-1}b.$
- 可以看作是如下最小范数问题的解

$$\min \|x\|_2^2 \quad s.t. \quad Ax = b.$$

- 从数值计算的角度看，通过正交基分解做消元法是很稳定的，主要的计算量在A的QR分解。
- 对于大规模稀疏矩阵A的QR分解， 运算量会比较大，比简单的高斯消元法计算量要大很多，需要设计其他的消元策略（结合QR分解和高斯消元法）。

不等式约束的影响

不等式约束的影响

- 当出现不等式约束时，甚用消元法

不等式约束的影响

- 当出现不等式约束时，甚用消元法
- 例如考虑如下问题

$$\begin{aligned} \min \quad & \sin(x_1+x_2)+x_3^2+\frac{1}{3}(x_4+x_5^4+\frac{x_6}{2}) \\ s.t. \quad & 8x_1-6x_2+x_3+9x_4+4x_5=6, \\ & 3x_1+2x_2-x_4+6x_5+4x_6=-4, \\ & x \geq 0. \end{aligned}$$

不等式约束的影响

- 当出现不等式约束时，甚用消元法
- 例如考虑如下问题

$$\begin{aligned} \min \quad & \sin(x_1+x_2)+x_3^2+\frac{1}{3}(x_4+x_5^4+\frac{x_6}{2}) \\ s.t. \quad & 8x_1-6x_2+x_3+9x_4+4x_5=6, \\ & 3x_1+2x_2-x_4+6x_5+4x_6=-4, \\ & x \geq 0. \end{aligned}$$

- 通过消元法(消去x_3, x_6)后转化为

$$\begin{aligned} \min_{x_1,x_2,x_4,x_4} \quad & \sin(x_1+x_2)+(8x_1-6x_2+9x_4+4x_5-6)^2 \\ & +\frac{1}{3}\Big(x_4+x_5^4-[\frac{1}{2}+\frac{3}{8}x_1+\frac{1}{4}x_2-\frac{1}{8}x_4+\frac{3}{4}x_5]\Big) \\ s.t. \quad & 8x_1-6x_2+9x_4+4x_5 \leq 6 \\ & \frac{3}{4}x_1+\frac{1}{2}x_2-\frac{1}{4}x_4+\frac{3}{2}x_5 \leq -1 \\ & (x_1,x_2,x_4,x_5) \geq 0. \end{aligned}$$

不等式约束的影响

- 当出现不等式约束时，甚用消元法
- 例如考虑如下问题

$$\begin{aligned} \min \quad & \sin(x_1+x_2)+x_3^2+\frac{1}{3}(x_4+x_5^4+\frac{x_6}{2}) \\ s.t. \quad & 8x_1-6x_2+x_3+9x_4+4x_5=6, \\ & 3x_1+2x_2-x_4+6x_5+4x_6=-4, \\ & x\geq 0. \end{aligned}$$

- 通过消元法(消去x_3, x_6)后转化为

$$\begin{aligned} \min_{x_1,x_2,x_4,x_4} \quad & \sin(x_1+x_2)+(8x_1-6x_2+9x_4+4x_5-6)^2 \\ & +\frac{1}{3}\left(x_4+x_5^4-[\frac{1}{2}+\frac{3}{8}x_1+\frac{1}{4}x_2-\frac{1}{8}x_4+\frac{3}{4}x_5]\right) \\ s.t. \quad & 8x_1-6x_2+9x_4+4x_5\leq 6 \\ & \frac{3}{4}x_1+\frac{1}{2}x_2-\frac{1}{4}x_4+\frac{3}{2}x_5\leq -1 \\ & (x_1,x_2,x_4,x_5)\geq 0. \end{aligned}$$

价值函数和Filters

- 问题：如果在迭代算法中，产生了一步使得目标函数显著下降，但是却违反了约束条件，我们接受这一次迭代吗?

价值函数和Filters

- 问题：如果在迭代算法中，产生了一步使得目标函数显著下降，但是却违反了约束条件，我们接受这一次迭代吗?
- 通常需要权衡两个目标(目标函数与约束条件)。价值函数和Filters，是分别衡量这两个目标的两种方式。

价值函数和Filters

- 问题：如果在迭代算法中，产生了一步使得目标函数显著下降，但是却违反了约束条件，我们接受这一次迭代吗?
- 通常需要权衡两个目标(目标函数与约束条件)。价值函数和Filters，是分别衡量这两个目标的两种方式。
- 在实际优化算法中，一次更新被接受，需要同时满足：价值函数要有显著下降，并且被 Filter 接受。

价值函数和Filters

- 问题：如果在迭代算法中，产生了一步使得目标函数显著下降，但是却违反了约束条件，我们接受这一次迭代吗?
- 通常需要权衡两个目标(目标函数与约束条件)。价值函数和Filters，是分别衡量这两个目标的两种方式。
- 在实际优化算法中，一次更新被接受，需要同时满足：价值函数要有显著下降，并且被 Filter 接受。
- 在无约束优化中，目标函数就是价值函数的最好选择。

价值函数和Filters

- 问题：如果在迭代算法中，产生了一步使得目标函数显著下降，但是却违反了约束条件，我们接受这一次迭代吗?
- 通常需要权衡两个目标(目标函数与约束条件)。价值函数和Filters，是分别衡量这两个目标的两种方式。
- 在实际优化算法中，一次更新被接受，需要同时满足：价值函数要有显著下降，并且被 Filter 接受。
- 在无约束优化中，目标函数就是价值函数的最好选择。
- 在可行算法中，由于不允许违反约束条件，一直在可行域中搜索，目标函数仍旧是一个合适的价值函数。

价值函数和Filters

- 问题：如果在迭代算法中，产生了一步使得目标函数显著下降，但是却违反了约束条件，我们接受这一次迭代吗?
- 通常需要权衡两个目标(目标函数与约束条件)。价值函数和Filters，是分别衡量这两个目标的两种方式。
- 在实际优化算法中，一次更新被接受，需要同时满足：价值函数要有显著下降，并且被 Filter 接受。
- 在无约束优化中，目标函数就是价值函数的最好选择。
- 在可行算法中，由于不允许违反约束条件，一直在可行域中搜索，目标函数仍旧是一个合适的价值函数。
- 只有当允许违反约束条件的算法中，才需要平衡两个目标。

价值函数和Filters

- 一个常用的非线性规划问题的带ℓ_1罚函数的价值函数

$$\phi_1(x;\mu) = f(x) + \mu \sum_{i\in\mathcal{E}} |c_i(x)| + \mu \sum_{i\in\mathcal{I}} [c_i(x)]^-$$

其中 $[z]^{-1} = \max\{0, -z\}$, $\mu > 0$是惩罚参数。

价值函数和Filters

- 一个常用的非线性规划问题的带ℓ_1罚函数的价值函数

$$\phi_1(x;\mu)=f(x)+\mu\sum_{i\in\mathcal{E}}|c_i(x)|+\mu\sum_{i\in\mathcal{I}}[c_i(x)]^-$$

  其中 $[z]^{-1}=\max\{0,-z\}$, $\mu>0$是惩罚参数。

- μ 代表了违反约束条件在整个优化中的权重。

价值函数和Filters

- 一个常用的非线性规划问题的带ℓ_1罚函数的价值函数

$$\phi_1(x;\mu) = f(x) + \mu \sum_{i\in\mathcal{E}} |c_i(x)| + \mu \sum_{i\in\mathcal{I}} [c_i(x)]^-$$

其中 $[z]^{-1} = \max\{0, -z\}$, $\mu > 0$是惩罚参数。

- μ 代表了违反约束条件在整个优化中的权重。
- ℓ_1罚函数是不可微的，但是有很好的性质，因为这个价值函数的解是精确的原问题的解。

价值函数和Filters

定义：精确的价值函数

如果存在一个$\mu^* > 0$，使得对于任何$\mu > \mu^*$，原非线性约束优化问题的局部解都是无约束优化问题$\phi(x;\mu)$的局部解。则称$\phi(x;\mu)$为精确的价值函数。

价值函数和Filters

定义：精确的价值函数

如果存在一个$\mu^* > 0$，使得对于任何$\mu > \mu^*$，原非线性约束优化问题的局部解都是无约束优化问题$\phi(x;\mu)$的局部解。则称$\phi(x;\mu)$为精确的价值函数。

- 可以证明，ℓ_1罚函数的价值函数是个精确的价值函数。

价值函数和Filters

定义：精确的价值函数

如果存在一个$\mu^* > 0$，使得对于任何$\mu > \mu^*$，原非线性约束优化问题的局部解都是无约束优化问题$\phi(x;\mu)$的局部解。则称$\phi(x;\mu)$为精确的价值函数。

- 可以证明，ℓ_1罚函数的价值函数是个精确的价值函数。
- 其他还有一些精确的价值函数如，对于只有等式约束的ℓ_2罚函数

$$\phi_2(x;\mu) = f(x) + \mu\|c_i(x)\|_2.$$

但是这个价值函数还是不可微的，因为在$c_i(x) = 0$处，导数没有定义。

价值函数和Filters

定义：精确的价值函数

如果存在一个$\mu^* > 0$，使得对于任何$\mu > \mu^*$，原非线性约束优化问题的局部解都是无约束优化问题$\phi(x;\mu)$的局部解。则称$\phi(x;\mu)$为精确的价值函数。

- 可以证明，ℓ_1罚函数的价值函数是个精确的价值函数。
- 其他还有一些精确的价值函数如，对于只有等式约束的ℓ_2罚函数
$$\phi_2(x;\mu) = f(x) + \mu\|c_i(x)\|_2.$$
但是这个价值函数还是不可微的，因为在$c_i(x) = 0$处，导数没有定义。
- 有一些价值函数是精确的且是光滑的（可微的）。例如对于只有等式约束的问题 可以定义(Fletcher's argumented Lagrangian)
$$\phi_F(x;\mu) = f(x) - \lambda(x)^T c(x) + \frac{1}{2}c_i(x)^2,$$
其中$\mu > 0$是惩罚参数，
$$\lambda(x) = [A(x)A(x)^T]^{-1}A(x)\nabla f(x), \quad A(x) = \nabla c_i(x).$$

价值函数和Filters

- 还有一种价值函数(标准的拉格朗日函数+ℓ_2罚函数)

$$\mathcal{L}_A(x,\lambda;\mu) = f(x) - \lambda^T c(x) + \frac{1}{2}\|c(x)\|_2^2$$

价值函数和Filters

- 还有一种价值函数(标准的拉格朗日函数+ℓ_2罚函数)

$$\mathcal{L}_A(x, \lambda; \mu) = f(x) - \lambda^T c(x) + \frac{1}{2}\|c(x)\|_2^2$$

- 严格说来$\mathcal{L}_A$不是一个精确的价值函数。因为原问题的解(x^*, λ^*)不是$\mathcal{L}_A$的一个局部解，而仅仅是一个平衡点(stationary point)。

价值函数和Filters

- 还有一种价值函数(标准的拉格朗日函数$+\ell_2$罚函数)

$$\mathcal{L}_A(x,\lambda;\mu)=f(x)-\lambda^T c(x)+\frac{1}{2}\|c(x)\|_2^2$$

- 严格说来$\mathcal{L}_A$不是一个精确的价值函数。因为原问题的解(x^*,λ^*)不是$\mathcal{L}_A$的一个局部解，而仅仅是一个平衡点(stationary point)。
- 虽然在有些情况下，使用$\mathcal{L}_A$可以成功求出解（需要不停调整λ和μ），但一般说来不考虑这种情况，仍然考虑ϕ_1和ϕ_2，不光滑但是是精确的价值函数。

价值函数和Filters

- 还有一种价值函数(标准的拉格朗日函数+ℓ_2罚函数)

$$\mathcal{L}_A(x,\lambda;\mu) = f(x) - \lambda^T c(x) + \frac{1}{2}\|c(x)\|_2^2$$

- 严格说来$\mathcal{L}_A$不是一个精确的价值函数。因为原问题的解(x^*,λ^*)不是$\mathcal{L}_A$的一个局部解，而仅仅是一个平衡点(stationary point)。
- 虽然在有些情况下，使用$\mathcal{L}_A$可以成功求出解（需要不停调整λ和μ），但一般说来不考虑这种情况，仍然考虑ϕ_1和ϕ_2，不光滑但是是精确的价值函数。
- ϕ_1或ϕ_2虽然不可微，但是存在方向导数$D(\phi(x;\mu);p)$。当某一步沿着p方向线搜索后迭代得到$x^+ = x + \alpha p$能是价值函数有显著下降，则可以接受这步迭代。

$$\phi_1(x+\alpha p;\mu) \le \phi(x;\mu) + \eta\alpha D(\phi(x;\mu);p)$$

其中$\eta \in (0,1)$

Filters

- Filter的概念来源于多目标优化问题，主要作用是处理是否接受迭代步。

Filters

- Filter的概念来源于多目标优化问题，主要作用是处理是否接受迭代步。
- 带约束的非线性优化问题其实可以分解为：

$$\color{blue}{\min_x \ f(x)} \ \text{和} \ \color{red}{\min_x \ h(x)} = \sum_{i\in\mathcal{E}} |c_i(x)| + \sum_{i\in\mathcal{I}} [c_i(x)]^-$$

Filters

- Filter的概念来源于多目标优化问题，主要作用是处理是否接受迭代步。
- 带约束的非线性优化问题其实可以分解为：

$$\min_x \; f(x) \text{ 和 } \min_x \; h(x) = \sum_{i\in\mathcal{E}} |c_i(x)| + \sum_{i\in\mathcal{I}} [c_i(x)]^-$$

- Filter是保持$f(x)$和$h(x)$独立的两个目标函数。

Filters

- Filter的概念来源于多目标优化问题，主要作用是处理是否接受迭代步。
- 带约束的非线性优化问题其实可以分解为：

$$\min_x \; f(x) \text{ 和 } \min_x \; h(x) = \sum_{i\in\mathcal{E}} |c_i(x)| + \sum_{i\in\mathcal{I}} [c_i(x)]^-$$

- Filter是保持$f(x)$和$h(x)$独立的两个目标函数。
- Filter准则：如果迭代产生的x^+，使得$(f(x^+), h(x^+))$没有被之前的$(f(x_l), h(x_l))$控制(dominate)，则我们接受x^+。

定义

Filters

- Filter的概念来源于多目标优化问题，主要作用是处理是否接受迭代步。
- 带约束的非线性优化问题其实可以分解为：

$$\min_{x} \; f(x) \text{ 和 } \min_{x} \; h(x) = \sum_{i\in\mathcal{E}} |c_i(x)| + \sum_{i\in\mathcal{I}} [c_i(x)]^-$$

- Filter是保持$f(x)$和$h(x)$独立的两个目标函数。
- Filter准则：如果迭代产生的x^+，使得 $(f(x^+), h(x^+))$没有被之前的$(f(x_l), h(x_l))$控制(dominate)，则我们接受x^+。

定义

- (f_k, h_k) 控制 (f_l, h_l)，是指$f_k \le f_l$, $h_k \le h_l$同时成立。

Filters

- Filter的概念来源于多目标优化问题，主要作用是处理是否接受迭代步。
- 带约束的非线性优化问题其实可以分解为：

$$\min_x \; f(x) \text{ 和 } \min_x \; h(x) = \sum_{i\in\mathcal{E}} |c_i(x)| + \sum_{i\in\mathcal{I}} [c_i(x)]^-$$

- Filter是保持$f(x)$和$h(x)$独立的两个目标函数。
- Filter准则：如果迭代产生的x^+，使得 $(f(x^+), h(x^+))$没有被之前的$(f(x_l), h(x_l))$控制(dominate)，则我们接受x^+。

定义

- (f_k, h_k) 控制 (f_l, h_l)，是指$f_k \le f_l$, $h_k \le h_l$同时成立。
- Filter是所有不能被互相控制的(f_l, h_l)组成的列表。

Filters

- Filter的概念来源于多目标优化问题，主要作用是处理是否接受迭代步。
- 带约束的非线性优化问题其实可以分解为：

$$\min_x \; f(x) \text{ 和 } \min_x \; h(x) = \sum_{i\in\mathcal{E}} |c_i(x)| + \sum_{i\in\mathcal{I}} [c_i(x)]^-$$

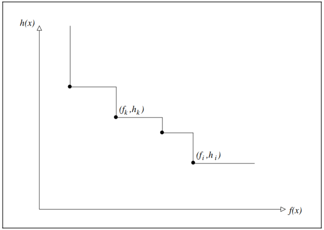
- Filter是保持$f(x)$和$h(x)$独立的两个目标函数。
- Filter准则：如果迭代产生的x^+，使得 $(f(x^+), h(x^+))$没有被之前的$(f(x_l), h(x_l))$控制(dominate)，则我们接受x^+。

定义

- (f_k, h_k) 控制 (f_l, h_l)，是指$f_k \le f_l$, $h_k \le h_l$同时成立。
- Filter是所有不能被互相控制的(f_l, h_l)组成的列表。
- x_k可以被Filter接受是指，(f_k, h_k)不能被Filter中的任何(f_l, h_l)控制。

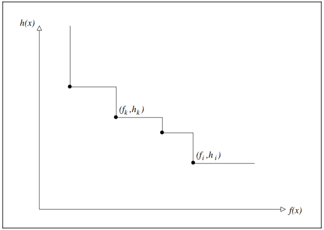
Filters

例子：



Filters

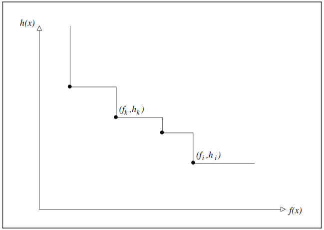
例子：



- Filter中已有4对，每个点都是一块无限矩形区域的左下角点。

Filters

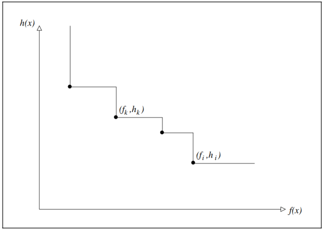
例子：

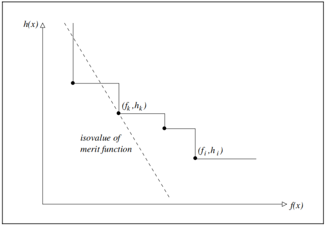


- Filter中已有4对，每个点都是一块无限矩形区域的左下角点。
- 只有实线左下方的点才能被Filter接受。

Filters

例子：

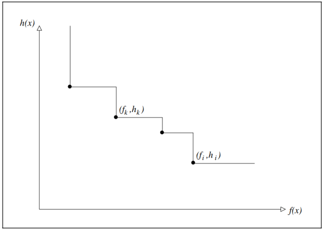


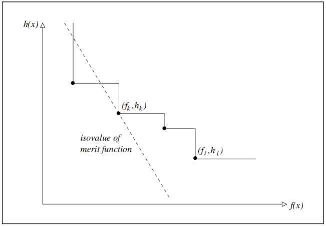


- Filter中已有4对，每个点都是一块无限矩形区域的左下角点。
- 只有实线左下方的点才能被Filter接受。

Filters

例子：

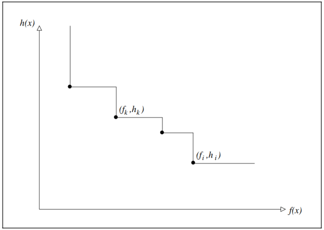


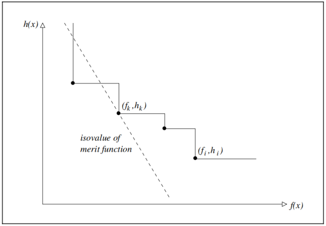


- Filter中已有4对，每个点都是一块无限矩形区域的左下角点。
- 只有实线左下方的点才能被Filter接受。
- 比较价值函数和Filter的可接受的范围，显然两种方法给出的可接受范围是不一样的。

Filters

例子：

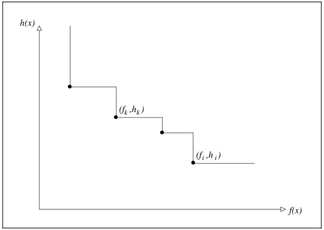


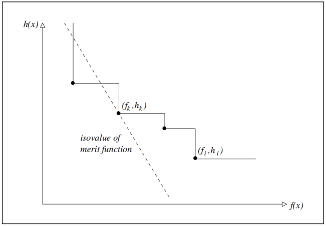


- Filter中已有4对，每个点都是一块无限矩形区域的左下角点。
- 只有实线左下方的点才能被Filter接受。
- 比较价值函数和Filter的可接受的范围， 显然两种方法给出的可接受范围是不一样的。
- 如果先搜索框架下的尝试步$x^+ = x_k + \alpha_k p_k$可以被接受，就取$x_{k+1} = x^+$，否则重新搜索。

Filters

例子：





- Filter中已有4对，每个点都是一块无限矩形区域的左下角点。
- 只有实线左下方的点才能被Filter接受。
- 比较价值函数和Filter的可接受的范围，显然两种方法给出的可接受范围是不一样的。
- 如果先搜索框架下的尝试步$x^+ = x_k + \alpha_k p_k$可以被接受，就取$x_{k+1} = x^+$，否则重新搜索。在信赖域方法中，如果尝试步没被接受，则缩小信赖域范围，重新尝试。

Filters

在实际优化算法中，可以增强Filter达到全局收敛性

- 我们不接受非常近的更新。

Filters

在实际优化算法中，可以增强Filter达到全局收敛性

- 我们不接受非常近的更新。把接受标准提高

  $f(x^+) \le f_j - \beta h_j$, 或 $h(x^+) \le h_j - \beta h_j$, $\forall j$ in Filter, $\beta \in (0, 1)$ is small.

Filters

在实际优化算法中，可以增强Filter达到全局收敛性

- 我们不接受非常近的更新。把接受标准提高

$$f(x^+) \leq f_j - \beta h_j, \text{ 或 } h(x^+) \leq h_j - \beta h_j, \ \forall j \text{ in Filter}, \ \beta \in (0,1) \text{ is small.}$$

- 在算法实施中(线搜索或信赖域)，有可能只会产生非常小的下降(insufficient decrease)。这个时候需要*feasibility restoration phase*。

Filters

在实际优化算法中，可以增强Filter达到全局收敛性

- 我们不接受非常近的更新。把接受标准提高

  $$f(x^+) \le f_j - \beta h_j, \text{ 或 } h(x^+) \le h_j - \beta h_j, \ \forall j \text{ in Filter}, \ \beta \in (0,1) \text{ is small.}$$

- 在算法实施中(线搜索或信赖域)，有可能只会产生非常小的下降(insufficient decrease)。这个时候需要*feasibility restoration phase*。
- *Feasibility restoration phase* aims exclusively to reduce the constraint violation （单独减少违反约束条件）, that is, to find an approximate solution to the problem

  $$\min_x h(x)$$

价值函数和Filters

Algorithm 1: (一般Filter方法)

初始点x_0和初始的信赖域半径Δ_0;设$k = 0$;

价值函数和Filters

Algorithm 1: (一般Filter方法)

初始点x_0和初始的信赖域半径Δ_0;设$k = 0$;

Repeat 直到停机准则条件满足

  if 找不到满足条件的下降

价值函数和Filters

ALGORITHM 1: (一般Filter方法)

初始点x_0和初始的信赖域半径Δ_0;设$k=0$;

Repeat 直到停机准则条件满足

  if 找不到满足条件的下降

    计算x_{k+1} using the feasibility restoration phase;

价值函数和Filters

ALGORITHM 1: (一般Filter方法)

初始点x_0和初始的信赖域半径Δ_0;设$k=0$;

Repeat 直到停机准则条件满足

    if 找不到满足条件的下降

        计算x_{k+1} using the feasibility restoration phase;

    else

        计算一次尝试 $x^+ = x_k + p_k$;

价值函数和Filters

ALGORITHM 1: (一般Filter方法)

初始点x_0和初始的信赖域半径Δ_0;设$k=0$;

Repeat 直到停机准则条件满足

  if 找不到满足条件的下降

    计算x_{k+1} using the feasibility restoration phase;

  else

    计算一次尝试 $x^+ = x_k + p_k$;

    if (f^+, h^+)可以被Filter接受

价值函数和Filters

Algorithm 1: (一般Filter方法)

初始点x_0和初始的信赖域半径Δ_0;设$k=0$;

Repeat 直到停机准则条件满足

  if 找不到满足条件的下降

    计算x_{k+1} using the feasibility restoration phase;

  else

    计算一次尝试 $x^+ = x_k + p_k$;

    if (f^+, h^+)可以被Filter接受

      设$x_{k+1} = x^+$ 并把 (f_{k+1}, h_{k+1})加入到Filter;

价值函数和Filters

Algorithm 1: (一般Filter方法)

初始点x_0和初始的信赖域半径Δ_0;设$k=0$;

Repeat 直到停机准则条件满足

  if 找不到满足条件的下降

    计算x_{k+1} using the feasibility restoration phase;

  else

    计算一次尝试 $x^+ = x_k + p_k$;

    if (f^+, h^+)可以被Filter接受

      设$x_{k+1} = x^+$ 并把 (f_{k+1}, h_{k+1})加入到Filter;

      选择合适的$\Delta_{k+1} \geq \Delta_k$;

价值函数和Filters

ALGORITHM 1: (一般Filter方法)

初始点x_0和初始的信赖域半径Δ_0;设$k=0$;

Repeat 直到停机准则条件满足

  if 找不到满足条件的下降

    计算x_{k+1} using the feasibility restoration phase;

  else

    计算一次尝试 $x^+ = x_k + p_k$;

    if (f^+, h^+)可以被Filter接受

      设$x_{k+1} = x^+$ 并把 (f_{k+1}, h_{k+1})加入到Filter;

      选择合适的$\Delta_{k+1} \geq \Delta_k$;

      删掉Filter中被(f_{k+1}, h_{k+1})的其他对,

价值函数和Filters

Algorithm 1: (一般Filter方法)

初始点x_0和初始的信赖域半径Δ_0;设$k=0$;

Repeat 直到停机准则条件满足

  if 找不到满足条件的下降

    计算x_{k+1} using the feasibility restoration phase;

  else

    计算一次尝试 $x^+ = x_k + p_k$;

    if (f^+, h^+)可以被Filter接受

      设$x_{k+1} = x^+$ 并把 (f_{k+1}, h_{k+1})加入到Filter;

      选择合适的$\Delta_{k+1} \geq \Delta_k$;

      删掉Filter中被(f_{k+1}, h_{k+1})的其他对,

    else

      拒绝尝试步，设$x_{k+1} = x_k$;

价值函数和Filters

ALGORITHM 1: (一般Filter方法)

初始点x_0和初始的信赖域半径Δ_0;设$k=0$;

Repeat 直到停机准则条件满足

  if 找不到满足条件的下降

    计算x_{k+1} using the feasibility restoration phase;

  else

    计算一次尝试 $x^+ = x_k + p_k$;

    if (f^+, h^+)可以被Filter接受

      设$x_{k+1} = x^+$ 并把 (f_{k+1}, h_{k+1})加入到Filter;

      选择合适的$\Delta_{k+1} \geq \Delta_k$;

      删掉Filter中被(f_{k+1}, h_{k+1})的其他对,

    else

      拒绝尝试步，设$x_{k+1} = x_k$;

      选择合适的$\Delta_{k+1} < \Delta_k$

价值函数和Filters

Algorithm 1: (一般Filter方法)

初始点x_0和初始的信赖域半径Δ_0;设$k=0$;
Repeat 直到停机准则条件满足
  if 找不到满足条件的下降
    计算x_{k+1} using the feasibility restoration phase;
  else
    计算一次尝试 $x^+ = x_k + p_k$;
    if (f^+, h^+)可以被Filter接受
      设$x_{k+1} = x^+$ 并把 (f_{k+1}, h_{k+1})加入到Filter;
      选择合适的$\Delta_{k+1} \geq \Delta_k$;
      删掉Filter中被(f_{k+1}, h_{k+1})的其他对,
    else
      拒绝尝试步，设$x_{k+1} = x_k$;
      选择合适的$\Delta_{k+1} < \Delta_k$
    end if

价值函数和Filters

ALGORITHM 1: (一般Filter方法)

初始点x_0和初始的信赖域半径Δ_0;设$k=0$;

Repeat 直到停机准则条件满足

  if 找不到满足条件的下降

    计算x_{k+1} using the feasibility restoration phase;

  else

    计算一次尝试 $x^+ = x_k + p_k$;

    if (f^+, h^+)可以被Filter接受

      设$x_{k+1} = x^+$ 并把 (f_{k+1}, h_{k+1})加入到Filter;

      选择合适的$\Delta_{k+1} \geq \Delta_k$;

      删掉Filter中被(f_{k+1}, h_{k+1})的其他对,

    else

      拒绝尝试步，设$x_{k+1} = x_k$;

      选择合适的$\Delta_{k+1} < \Delta_k$

    end if

  end if

价值函数和Filters

Algorithm 1: (一般Filter方法)

初始点x_0和初始的信赖域半径Δ_0;设$k=0$;
Repeat 直到停机准则条件满足
  if 找不到满足条件的下降
    计算x_{k+1} using the feasibility restoration phase;
  else
    计算一次尝试 $x^+ = x_k + p_k$;
    if (f^+, h^+)可以被Filter接受
      设$x_{k+1} = x^+$ 并把 (f_{k+1}, h_{k+1})加入到Filter;
      选择合适的$\Delta_{k+1} \geq \Delta_k$;
      删掉Filter中被(f_{k+1}, h_{k+1})的其他对,
    else
      拒绝尝试步，设$x_{k+1} = x_k$;
      选择合适的$\Delta_{k+1} < \Delta_k$
    end if
  end if
end repeat

Thanks For Your Attention