优化算法

张伯雷

南京邮电大学 计算机学院、通达学院

https://bolei-zhang.github.io/course/opt.html

目录

- 无约束优化算法
  - 线搜索算法
- 约束优化算法
  - 牛顿法
  - 拉格朗日法
- 随机优化算法

预备知识

- 泰勒展开

$$f(a)+\frac{f'(a)}{1!}(x-a)+\frac{f''(a)}{2!}(x-a)^2+\frac{f'''(a)}{3!}(x-a)^3+\cdots,$$

- KKT条件

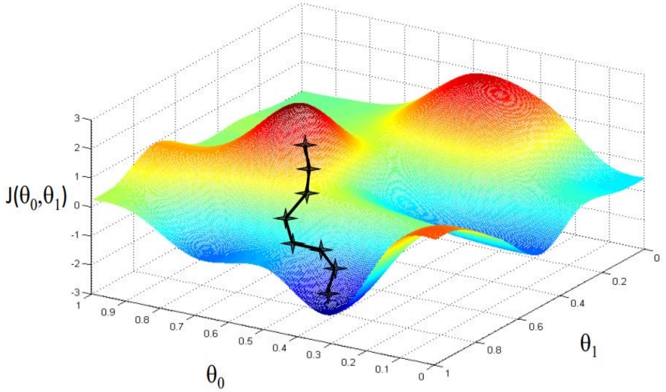
$$\begin{array}{rcll}
f_i(x^\star) & \leq & 0, & i=1,\ldots,m \\
h_i(x^\star) & = & 0, & i=1,\ldots,p \\
\lambda_i^\star & \geq & 0, & i=1,\ldots,m \\
\lambda_i^\star f_i(x^\star) & = & 0, & i=1,\ldots,m \\
\nabla f_0(x^\star)+\sum_{i=1}^m \lambda_i^\star \nabla f_i(x^\star)+\sum_{i=1}^p \nu_i^\star \nabla h_i(x^\star) & = & 0, &
\end{array}$$

优化算法

- 所有的算法的本质都在求解KKT条件
- 大多数优化算法都是迭代算法
- 假设：$f_0(x)$是光滑的，容易求梯度、Hessian矩阵
- **线搜索算法：**
  - 一旦确定了搜索的方向，下一 步即沿着该方向寻找下一个迭代点

线搜索算法

- 线搜索算法
  - 梯度下降法
  - 牛顿法



- 迭代算法（下山法）

$$x^{k+1} = x^k + \alpha^k d^k$$

  - 步长： $\alpha^k \in R$
  - 方向： $d^k \in R^n$

$$\varphi(\alpha) = f_0(x^k + \alpha d^k)$$

$$\alpha^k = \arg\min_{\alpha} \varphi(\alpha) = \arg\min_{\alpha} f_0(x^k + \alpha d^k)$$

一维、关于α的凸函数（amijo准则）

梯度下降法

- 负梯度定义

$$d^k = -\nabla f(x^k)$$

- 重复

$$\alpha^k = \arg\min_{\alpha} f_0(x^k + \alpha^k d^k)$$
$$x^{k+1} = x^k + \alpha^k d^k$$

直到收敛

二次函数的梯度法

设二次函数$f(x,y)=x^2+10y^2$, 初始点(x^0,y^0) 取为$(10,1)$, 取固定步长$\alpha_k=0.085$. 我们使用梯度法$x^{k+1}=x^k-\alpha_k\nabla f(x^k)$ 进行15 次迭代.

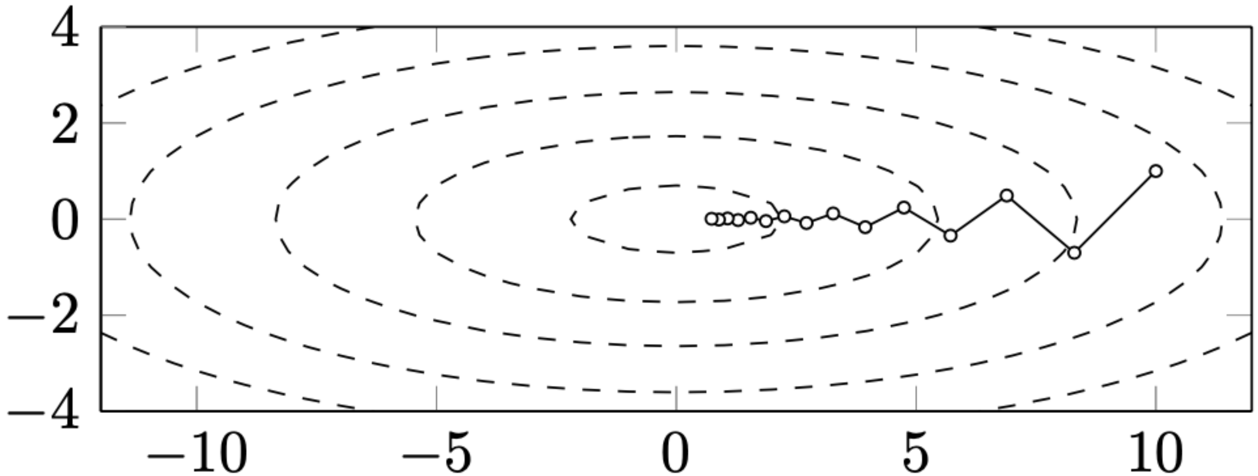


Figure: 梯度法的前15 次迭代

梯度法在凸函数上的收敛性

- 收敛性的证明思路

$$\frac{||x^{k+1}-x^*||}{||x^k-x^*||}<1$$

- 线性收敛

$$\frac{||x^{k+1}-x^*||}{||x^0-x^*||}<c^k$$

牛顿法

- 考虑$f(x)$在x^k的二阶泰勒展开

$$f(x^k + d^k) = f(x^k) + \nabla f(x^k)^{\mathrm{T}} d^k + \frac{1}{2}(d^k)^{\mathrm{T}} \nabla^2 f(x^k) d^k + o(\|d^k\|^2).$$

  - 忽略其中的高阶项，求关于d^k的函数的稳定点

$$\nabla^2 f\left(x^k\right) d^k = -\nabla f\left(x^k\right)$$

  - 牛顿法更新公式

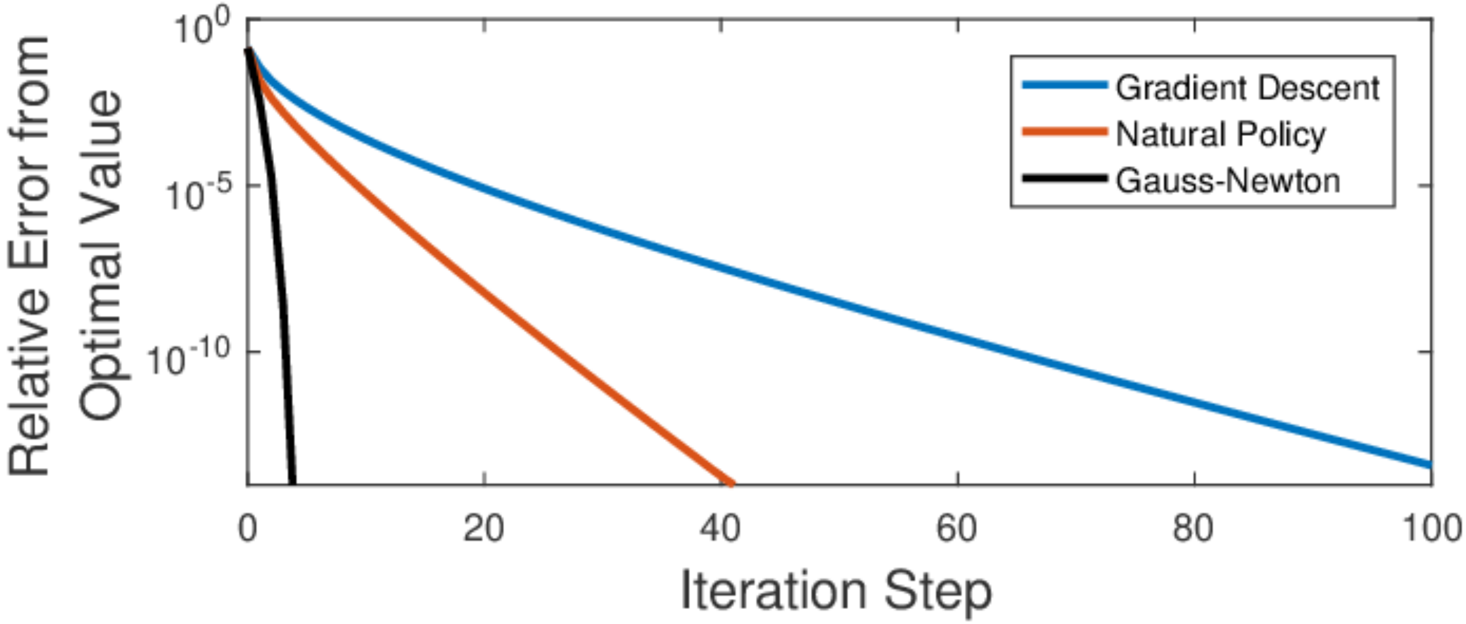
$$x^{k+1} = x^k - \nabla^2 f(x^k)^{-1} \nabla f(x^k).$$

  - 步长恒为1（经典牛顿法）

经典牛顿法的收敛性

- 二次收敛

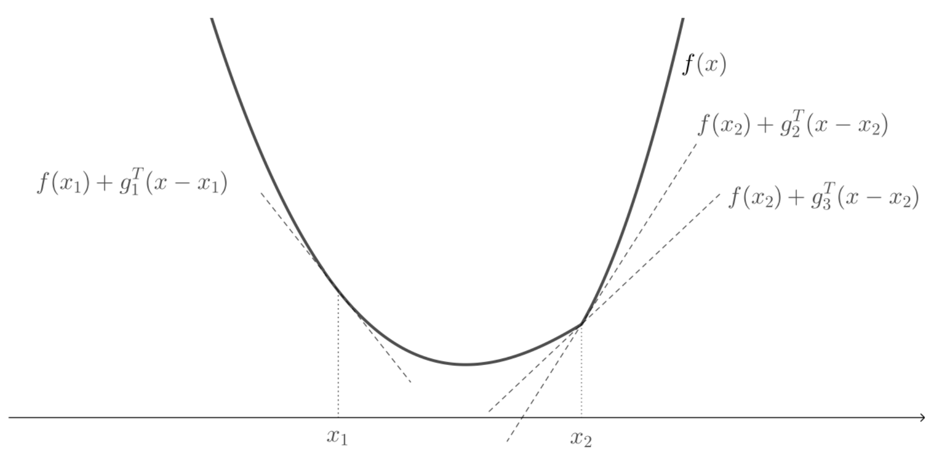
$$\frac{||x^{k+1}-x^*||}{||x^k-x^*||^2}<1$$



次梯度法

- $f_0(x)$为凸函数，但不可微
- 次梯度

$$f(y) \geqslant f(x) + g^T(y - x),$$



- 次梯度算法

$$x^{k+1} = x^k - \alpha^k g^k$$

约束优化问题

$$\min_{x} \quad f(x)$$
$$\text{s.t.} \quad x \in \mathcal{X}$$

- 相比于无约束优化问题
  - 约束优化问题中x不能随便取值，梯度下降法所得点不一定在可行域内
  - 最优解处目标函数的梯度不一定为零向量

1. 等式约束的凸优化问题

- 优化问题

$$\min f_0(x)$$
$$\mathrm{s.t.}\, Ax = b$$

- KKT条件

$$Ax^* = b$$
$$\nabla f^*(x) + A^T v^* = 0$$

例：等式约束的QP问题

$$\begin{array}{ll}\text{minimize} & (1/2)x^T Px + q^T x + r \\ \text{subject to} & Ax = b,\end{array}$$

$P \in S_+^n$

1. 等式约束的牛顿法

- 假设x^k为可行解，需要找到下一个可行解

$$\min f_0\left(x^k+d\right)$$
$$\text{s.t.}\ \ A\left(x^k+d\right)=0$$

- 对x^k+d进行二阶泰勒展开

$$\min f_0\left(x^k\right)+\nabla f^T\left(x^k\right)d+\frac{1}{2}d^T\nabla^2 f^T\left(x^k\right)d$$
$$\text{s.t.}\ \ Ad=0$$

- 算法

$$\alpha^k=\arg\min_{\alpha\geq 0} f(x^k+\alpha^k d^k)$$
$$x^{k+1}=x^k+\alpha^k d^k$$

2. 拉格朗日法

- 算法

$$x^{k+1} = x^k - \alpha^k\left(\nabla f\left(x^k\right) + A^T v^k\right)$$

$$v^{k+1} = v^k + \alpha^k\left(Ax^k - b\right)$$

- 鞍点解释
  - $(x^*, v^*) = \arg\min_x \max_v L(x, v)$
  - $(v^*, x^*) = \arg\max_v \min_x L(x, v)$
  - $x^* = \arg\min_x L(x, v^*)$：假设$v^*$已知（用$v^k$代替），求解无约束优化问题
  - $v^* = \arg\max_x L(x^*, v)$：假设$x^*$已知（用$x^k$代替），求解无约束优化问题

优化算法

张伯雷

南京邮电大学 计算机学院、通达学院

https://bolei-zhang.github.io/course/opt.html

目录

- 无约束优化算法
  - 线搜索算法
- 约束优化算法
  - 牛顿法
  - 拉格朗日法
- 随机优化算法

2. 拉格朗日法

- 算法

$$x^{k+1} = x^k - \alpha^k\left(\nabla f\left(x^k\right) + A^T v^k\right)$$

$$v^{k+1} = v^k + \alpha^k\left(Ax^k - b\right)$$

- 鞍点解释
  - $(x^*, v^*) = \arg\min_x \max_v L(x, v)$
  - $(v^*, x^*) = \arg\max_v \min_x L(x, v)$
  - $x^* = \arg\min_x L(x, v^*)$：假设$v^*$已知（用$v^k$ 代替），求解无约束优化问题
  - $v^* = \arg\max_x L(x^*, v)$：假设$x^*$已知（用$x^k$ 代替），求解无约束优化问题

增广拉格朗日函数

- 增广拉格朗日函数

$$L_C(x,v) = f(x) + v^T(Ax-b) + \frac{c}{2}||Ax-b||_2^2$$

- 增广拉格朗日函数为以下问题的拉格朗日函数

$$\min f(x) + \frac{c}{2}||Ax-b||_2^2$$

$$s.t. Ax = b$$

与以下问题的原问题最优解和对偶问题最优解都是一样的

$$\min f(x)$$

$$s.t. Ax = b$$

增广拉格朗日法(1)

$$\min f(x) + \frac{c}{2}||Ax - b||_2^2$$

$$s.t. Ax = b$$

算法:

$$x^{k+1} = x^k - \alpha^k \nabla_x L_C(x^k, v^k)$$

$$v^{k+1} = v^k + \alpha^k \left(Ax^k - b\right)$$

增广拉格朗日法(2)

$$\min f(x) + \frac{c}{2}||Ax - b||_2^2$$

$$s.t. Ax = b$$

算法:

$$x^{k+1} = \arg\min_x L_C(x, v^k)$$

$$v^{k+1} = v^k + C\left(Ax^{k+1} - b\right)$$

不等式约束的凸优化问题

$$
\begin{aligned}
Ax^\star = b, \quad f_i(x^\star) &\leq 0, \quad i = 1, \ldots, m \\
\lambda^\star &\succeq 0 \\
\nabla f_0(x^\star) + \sum_{i=1}^m \lambda_i^\star \nabla f_i(x^\star) + A^T \nu^\star &= 0 \\
\lambda_i^\star f_i(x^\star) &= 0, \quad i = 1, \ldots, m.
\end{aligned}
$$

- 内点法
- 屏障法
- …

随机优化算法

- 监督学习模型
  - 假定 (X, y) 服从概率分布 P，其中X为输入， y为标签.
  - 决定一个最优的函数 φ 使得期望风险 $E[L(\varphi(x), y)]$ 最小，其中 $L(\cdot,\cdot)$ 表示损失函数，用来衡量预测的准确度，函数 φ 为某个函数空间中的预测函数.
  - 在实际问 题中我们并不知道真实的概率分布 P，而是随机采样得到的一个数据集 $D = \{(x_1, y_1), (x_2, y_2), \cdots, (x_N, y_N)\}$.然后用经验风险来近似期望风险， 并将预测函数 $\varphi(\cdot)$ 参数化为 $\varphi(\cdot;\theta)$ 以缩小要找的预测函数的范围，即要求解下面的极小化问题:

$$\min_x \frac{1}{N} \sum_{i=1}^{N} L(\varphi(x_i, \theta), y_i)$$

  - 由于数据规模巨大，计算目标函数的梯度变得非常困难

随机梯度下降（Stochastic Gradient Descent）

$$\min_{x\in\mathbb{R}^n} \quad f(x) \stackrel{\text{def}}{=\!=} \frac{1}{N}\sum_{i=1}^{N} f_i(x),$$

- 假设每一个 $f_i(x)$ 是凸的、可微的

$$x^{k+1} = x^k - \alpha_k \nabla f(x^k), \qquad \nabla f(x^k) = \frac{1}{N}\sum_{i=1}^{N} \nabla f_i(x^k).$$

- 随机梯度下降法

$$x^{k+1} = x^k - \alpha_k \nabla f_{s_k}(x^k),$$

$$x^{k+1} = x^k - \frac{\alpha_k}{|\mathcal{I}_k|}\sum_{s\in\mathcal{I}_k} \nabla f_s(x^k),$$

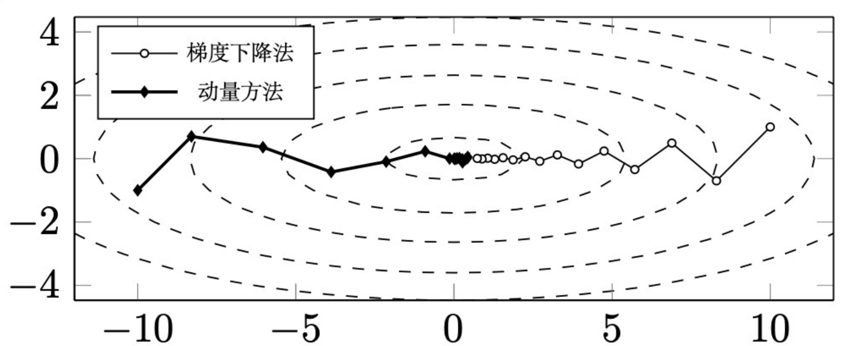
其中 s_k 是从 $\{1, 2, \cdots, \mathrm{N}\}$ 中随机等可能地抽取的一个样本。常用的形式是小批量(mini-batch)随机梯度法，即随机选择一个元素个数很少的集合 $I_k \subset \{1,2,\cdots,N\}$，然后执行迭代格式

动量方法(momentum)

- 在算法迭代时一定程度上保留之前更新的方向，同时利用当前计算的梯度调整最终的更新方向.

$$v^{k+1} = \mu_k v^k - \alpha_k \nabla f_{s_k}(x^k),$$
$$x^{k+1} = x^k + v^{k+1}.$$

- 在计算当前点的随机梯度 $\nabla f_{s_k}(x_k)$ 后，将其和上一步更新方向 v_k 做线性组合来得到新的更新方向 v_{k+1}
- 在普通的梯度法中，每一步迭代只用到了当前点的梯度估计，动量方法的更新方向还使用了之前的梯度信息。



Nesterov方法

- 先对点施加速度的作用，再求梯度，可以理解为对标准动量方法做了一个校正.

$$
\begin{aligned}
y^{k+1} &= x^k + \mu_k(x^k - x^{k-1}), \\
x^{k+1} &= y^{k+1} - \alpha_k \nabla f_{s_k}(y^{k+1}),
\end{aligned}
$$

AdaGrad算法(adaptive subgradient methods)

- 传统梯度算法只有一个统一的步长α_k来调节每一步迭代，它没有针对每一个分量考虑
- 当梯度的某个分量较大时，可以推断出在该方向上函数变化比较剧烈，此时应该用小步长
- 当梯度的某个分量较小时，在该方向上函数比较平缓，此时应该用大步长.
- $g_k = \nabla f_{s_k}(x_k)$
- 当G_k的某分量较大时，该分量变化比较剧烈，因此应采用小步长，反之亦然

$$x^{k+1} = x^k - \frac{\alpha}{\sqrt{G^k + \varepsilon \mathbf{1}_n}} \odot g^k,$$

$$G^{k+1} = G^k + g^{k+1} \odot g^{k+1},$$

谢谢！！