SLAM 中的优化理论

0 引言

优化理论在 SLAM 以及深度学习中都有非常重要的应用。本篇文章的主要内容，就是总结归纳一下相关的内容，包括线性最小二乘，非线性最小二乘，常用优化算法等。

最小二乘方法主要的目的是当没有一种精确的方法求解时，找到一种近似求解的方法。比如，找到一种方法最小化系统的残差。最小二乘也可以被用来拟合一个带噪声数据的模型，或者对复杂数据拟合出一个简单的模型。比如，将高维的数据映射到一个低维的空间。最下二乘有一个显著的统计特性，即测量值与期望值之间的残差满足正态分布时，最小二乘问题转换为一个最大似然估计问题。

1 最小二乘

线性最小二乘试图对一个超定线性系统（比如一个线性系统由一个 $m\times n$ 的矩阵 A 描述 $Ax=b$ ， $m>¿$ n，其中 x 为 $n\times 1$ ， b 为 $m\times 1$ ）找到一个最小二乘解。有意思的是，我们的样本数据往往比待求解的参数多很多，所以我们面对的往往是一个超定问题。最小二乘最小化残差的平方欧式范数：

$$min\|b-A\,x\|^2=min\|r\|^2 \quad (1)$$

其中 r 表示残差(向量)。

线性最小二乘可以用来拟合任意的线性函数。

1.1 线性最小二乘的存在与唯一性证明（正则方程求解）

针对上面的超定线性系统 $Ax=b$ ，系数矩阵 A（ $m\times n$ ）的秩最大为 n。因此当 A 为满秩矩阵时，系统只有唯一解。当 A 不是满秩矩阵时，系统有无穷解。

如果当 A 为满秩矩阵，那么目标函数可表示为：

$$\|\acute{r}\|^2=r^Tr=(b-Ax)^T(b-Ax)=b^Tb-2x^TA^Tb+x^TA^TAx \quad (2)$$

两边对 $\acute{x}$ 求导可得：

$$-2A^Tb+2A^TAx=0 \quad (3)$$

化简后：

$$A^TAx=A^Tb \quad (4)$$

注意这里的 A^TA 变为了一个 $n\times n$ 的矩阵，因此可以求解：

$$x=(A^TA)^{-1}A^Tb \quad (5)$$

解的形式也可以用 A 的伪逆形式表示：

$$x=A^{¿}+¿b \quad (6)$$

其中

$$A^{¿}+¿=(A^TA)^{-1}A^T \quad (7)$$

1.2 Cholesky decomposition 求解

对上面的超定线性系统更快更稳定的求解方式是先将 $A^T A$ 作 Cholesky decomposition 分解为一个上三角矩阵 R 和它的转置：

$$A^T A = R^T R \quad (8)$$

将式(4)重新改写为：

$$R^T Rx = A^T b \quad (9)$$

将式(9)中的 Rx 替换为 α ， $A^T b$ 替换为 β 后：

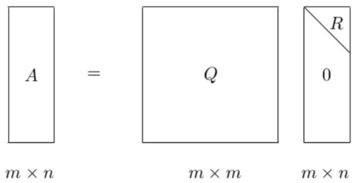
$$R^T \alpha = \beta \quad (10)$$

通过式(10)可以求得 α ，又因为 $Rx = \alpha$ ，所以又可以求得 x 。

但是这里使用正则方程或者 Cholesky 分解求解的效果很好，但是我们假定了这是个超定线性方程。而且只有当 $A^T A$ 是可逆的，才可以求解。如果矩阵 A 是病态（奇异矩阵）的，那么 $A^T A$ 往往会更差。这时候矩阵 $A^T A$ 求逆或者做 Cholesky 分解，都可能失败。计算中为了提高效率，通常先计算出 $A^T A$ 。

1.3 QR 求解

QR 分解是将一个矩阵 A 分解为一个正交矩阵 Q 和一个上三角矩阵 R 的乘积。



QR 分解的计算方式有很多，例如 Givens 旋转，Householder 变换，Gram-Schmidt 正交化等。因为

$$A = QR \quad (11)$$

两边乘上 Q^T ，

$$Q^T A = Q^T QR \quad (12)$$

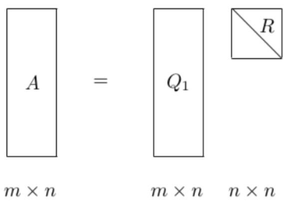
因为 Q 为正交阵，所以 $Q^T Q = I$ ，因此有，

$$Q^T A = R \quad (13)$$

这里 Q^T 表示 Householder 变换的各种集合，

$$Q^T = H_n H_{n-1} \ldots H_2 H_1 \quad (14)$$

我们知道 Q 是一个 $m \times m$ 的正交阵，但是这个 R 是一个 $m \times n$ 的上三角矩阵。但是 R 的下半部分全 0，不为 0 的行数是前 n 行。因此将 Q 拆分为 Q_1 ， Q_2 两部分。 Q_1 是一个 $m \times n$ 的矩阵， Q_2 是一个 0 矩阵，R 也去掉 0 的部分，变成一个 $n \times n$ 标准上三角阵 R' 。



因此式(11)QR 分解变为：

$$A = Q_1 R' \quad (15)$$

这里 Q_1 也是正交矩阵。因此在 $Ax = b$ 等号左右两边乘上 Q_1^T 可得：

$$Q_1^T A x = Q_1^T b \quad (16)$$

将式(15)代入式(16)中可得:

$$R' x = Q_1^T b \quad (17)$$

因此可求得:

$$x = R'^{-1} Q_1^T b \quad (18)$$

1.4 SVD 求解

SVD 将矩阵 A 分解为 U，Σ，V ,其中 U,V 是正交矩阵，U 是 $m \times m$ 的矩阵，V 是 $n \times n$ 的矩阵， Σ 是对角矩阵 $diag(\sigma_1, \sigma_2, \sigma_3, \ldots, \sigma_n)$，$\Sigma$ 为 $m \times n$ 。这里的奇异值是有序的:

$$A = U \Sigma V^T \quad (19) \qquad \sigma_1 \geq \sigma_2 \geq \sigma_3 \geq \ldots \geq \sigma_n > 0$$

将式(19)代入式(5)中得:

$$x = \left(\left(U \Sigma V^T \right)^T U \Sigma V^T \right)^{-1} \left(U \Sigma V^T \right)^T b \quad (20)$$

由于 A 是 $m \times n$ 不是方阵，所以对 A 进行 SVD 分解时， U 拆分为 U_1 ， U_2 两部分。 U_1 为前 n 列，同时需要将 Σ 为 $m \times n$ 截断为 $n \times n$ 的对角阵 S，因此:

$$A = \begin{pmatrix} U_1 & U_2 \end{pmatrix} \begin{pmatrix} s \\ 0 \end{pmatrix} V^T = U_1 S V^T \quad (21)$$

2 非线性最小二乘

对于非线性最小二乘问题，会有类似 $A(x)x=b$ 这种形式，即系数矩阵也依赖要求解的参数 x 。因此对于非线性最小二乘问题需要迭代的方式求解，一般待求解的参数 x 会给一个初值然后通过迭代的方式逐步得到一个近似的局部最优解。如果非线性问题有好几个极小值，那么最后收敛到的不能确定是哪一个极小值，这取决于给定的初始值。

定义非线性最小二乘的目标函数形式：

$$\rho(x)=\frac{1}{2}\sum_i \left(b_i-f_i(x)\right)^2 \quad (22)$$

$$r_i(x)=b_i-f_i(x) \quad (23)$$

$$\rho(x)=\frac{1}{2}\left[r_i(x)\right]^T r_i(x) \quad (24)$$

雅各比矩阵 $J(x)$ 的定义：

$$J_{i,j}(x)=\left[\frac{\partial r_i}{x_j}\right] \quad (25)$$

在目标函数 $\rho(x)$ 的最小值处，目标函数的梯度为 0：

$$\nabla\rho(x)=\frac{1}{2}\left[\frac{\partial\left[r_i(x)\right]^T}{x_j}r(x)+\left[r_i(x)\right]^T\frac{\partial r(x)}{x_j}\right]=\left[J(x)\right]^T r(x)$$

$$\nabla\rho(x)=\left[J(x)\right]^T r(x)=0 \quad (26)$$

目标函数的海森矩阵：

$$\nabla^2\rho(x)=\frac{\partial\left[J(x)\right]^T}{x_j}r(x)+\left[J(x)\right]^T\frac{\partial r(x)}{x_j}\cong\left[J(x)\right]^T J(x) \quad (27)$$

记目标函数的梯度为 $g(x)$ ，目标函数的海森矩阵为 $H(x)$ ：

$$g(x)=\left[J(x)\right]^T r(x)\equiv\rho(x)'=\begin{bmatrix}\frac{\partial\rho}{\partial x_1}(x)\\ \dots\\ \frac{\partial\rho}{\partial x_n}(x)\end{bmatrix} \quad (28)$$

$$H(x)=\left[J(x)\right]^T J(x)\equiv\rho(x)''=¿\quad\left[\frac{\partial^2\rho}{\partial x_i\partial x_j}(x)\right] \quad (29)$$

假设目标函数是平滑的，对它进行泰勒展开：

$$\rho(x+h)=\rho(x)+h^T g(x)+\frac{1}{2}h^T H(x)h+O(\|h\|^3) \quad (30)$$

2.1 最速下降法

最速下降法时一个通用的函数优化方法，并不是专门解决最小二乘的方法。最速下降法的每一次迭代包括三个步骤：

1． 选择下降方向 h_{dd} ,不仅使得 $\rho(x_{k+1})<\rho(x_k)$ ，而且要下降得最快
2． 选择步长 α
3． 更新参数： $x_{k+1}=x_k+\alpha h_{dd}$

Algorithm 2.4. Descent Method

```
begin
  k := 0;  x := x0;  found := false                  {Starting point}
  while not found and k < kmax
    hdd := search_direction(x)                       {From x and downhill}
    if no such h exists
      found := true                                  {x is stationary}
    else
      α := line_search(x, hdd)                       {from x in direction hdd}
      x := x + αhdd;  k := k+1                       {next iterate}
end                                                  {... of descent algorithm }
```

对变化后的目标函数做一阶泰勒展开：

$$\rho(x+\alpha h_{dd})=\rho(x)+\alpha h_{dd}{}^T g(x)+O(\alpha^2) \quad (31)$$

这里 α 足够小， $0<\alpha<1$ 。

如果要说明 h_{dd} 是下降的方向，那么 $\alpha h_{dd}{}^T g(x)$ 必须是小于 0 的，这样才会导致 $\rho(x+\alpha h_{dd})<\rho(x)$ 。因此有：

$$\rho(x+\alpha h_{dd})<\rho(x) \quad (32)$$

$$h_{dd}^{T} g(x) < 0 \quad (33)$$

如果这一步找不到一个方向 h_{dd} 使得函数下降，那么 h_{dd} 不存在，此时 $g(x)=0$ 。说明找到了一个最小值。

最速下降的方向确定：

最速下降的方向选择，通过函数的相对增益确定，即函数值的变化率与自变量变化率之比。

$$\lim_{\alpha \to 0} \frac{\rho(x) - \rho(x + \alpha h_{dd})}{\alpha \|h_{dd}\|} \quad (34)$$

将式(31)代入式(34)中可得：

$$\lim_{\alpha \to 0} \frac{\rho(x) - \rho(x) - \alpha h_{dd}^{T} g(x)}{\alpha \|h_{dd}\|} = \frac{-1}{\|h_{dd}\|} h_{dd}^{T} g(x) \quad (35)$$

因为 h_{dd}^{T} 与 $g(x)$ 是两个向量，因此变成了向量点乘，继续化简：

$$\frac{-1}{\|h_{dd}\|} h_{dd}^{T} g(x) = \frac{-1}{\|h_{dd}\|} \|h_{dd}^{T}\| \|g(x)\| \cos\theta = -\|g(x)\| \cos\theta \quad (36)$$

这里 θ 表示方向 h_{dd} 与梯度 $g(x)$ 之间的夹角。当 $\theta = \pi$ 时(h_{dd} 为梯度的反方向)，$\cos\theta = -1$ ，函数的相对增益达到最大为 $\|g(x)\|$ 。因此有：

$$h_{dd} = -g(x) \quad (37)$$

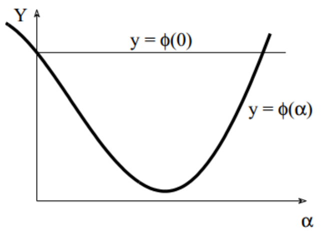
步长的确定(line search)：

为了研究我们需要每次迭代沿着下降方向走多远，这里考察下迭代之后的目标函数：

$$\varnothing(\alpha) = \rho(x + \alpha h_{dd}) \quad (38)$$

这里 x ，h_{dd} 都是固定的，$\alpha \geq 0$ 。我们必须满足下降条件(32)，改写下：

$$\varnothing(\alpha) < \varnothing(0) \quad (39)$$



步长 α 一般初值给定为 1，有以下两种情况可以调整：

1. 当 α 很小时，导致目标函数的相对增益太小（本质就是下降的幅度太小），我们就增大 α 。

2. 当 α 太大时，导致了 $\varnothing(\alpha) \geq \varnothing(0)$ ，这时我们需要减小 α ，以使得满足下降条件(39)

这个步长确定的过程，即 line search 的过程是一个迭代的过程，这个过程产生了一系列的 $\alpha_1, \alpha_2, \alpha_3, \alpha_4 \cdots$ 我们在这些值中，找到一个最好的满足下降条件(32)。

2.1 高斯牛顿法

高斯牛顿方法是一种高效的方法，它是基于目标函数的一阶导数推导的，在某些特殊情况下，它具有平方收敛的速度。高斯牛顿方法的基础是在工作点 x 领域上对残差函数 $r(x)$ 的线性近似：

$$r(x+h) \cong l(h) \equiv r(x) + J_r(x) h \quad (40)$$

由此得到：

$$\rho(x+h) \cong L(h) \equiv \frac{1}{2} l(h)^T l(h) = \frac{1}{2} \left(r(x) + J_r(x) h\right)^T \left(r(x) + J_r(x) h\right)$$

$$¿\frac{1}{2}r(x)^T r(x)+h^T J_r(x)^T r(x)+\frac{1}{2}h^T J_r(x)^T J_r(x)h$$

$$¿\rho(x)+h^T J_r(x)^T r(x)+\frac{1}{2}h^T J_r(x)^T J_r(x)h$$

因此有：

$$\rho(x+h)\cong L(h)\equiv\rho(x)+h^T J_r(x)^T r(x)+\frac{1}{2}h^T J_r(x)^T J_r(x)h \quad (41)$$

容易看出 $L(h)$ 的梯度和海森矩阵：

$$L(h)'=J_r(x)^T r(x)+J_r(x)^T J_r(x)h \quad (42)$$

$$L(h)''=J_r(x)^T J_r(x) \quad (43)$$

如果 $\rho(x+h)$ 有极小值，那么应该有一阶导数为 0：

$$L(h)'=J_r(x)^T r(x)+J_r(x)^T J_r(x)h=0$$

求得：

$$J_r(x)^T J_r(x)h=-J_r(x)^T r(x) \quad (44)$$

这个式子跟式(4)形式一摸一样，这个也称为正则方程。因此可以求得高斯牛顿的迭代方向 h_{GN} ，每一步的步长为 1，算法流程跟最速下降法一致。

2.1 Levenberg-Marquardt 方法

LM 方法是一种基于信赖域的方法，但是跟高斯牛顿方法有些类似，其每一步迭代的方向 h_{LM} 由以下方式计算得到：

$$\left(J_r{}^T J_r+\mu I\right)h_{LM}=-g \quad (45)$$

这里 $J_r=J_r(x)$ ， $g=J_r(x)^T r(x)$ ， μ 表示一个阻尼系数， $\mu\geq 0$ 。阻尼系数 μ 有几个作用：

1. 保证 $\left(J_r{}^T J_r+\mu I\right)$ 是正定的矩阵，使得函数下降
2. 如果 μ 很大，可以得到 $h_{LM}\approx-\frac{1}{\mu}g$ ，比如在最速下降法中走了很小一步
3. 如果 μ 比较小， $h_{LM}\approx h_{GN}$ ，在迭代的最后阶段（ x 接近于极值点 $x^{¿}$ ）是比较好的迭代方式，如果目标函数 $\rho(x^{¿})$ 在极值点处为 0 或者很小时，迭代的收敛速度接近于平方收敛。

阻尼系数 μ 受增益率控制：

$$\varrho = \frac{\rho(x) - \rho(x + h_{LM})}{L(0) - L(h_{LM})}$$

如果增益 ϱ 比较大，说明 $L(h_{LM})$ 是一个对 $\rho(x+h_{LM})$ 很好的近似，我们减少 μ 使得下一次迭代更加接近于高斯牛顿迭代，增益 ϱ 比较小，表示 $L(h_{LM})$ 对 $\rho(x+h_{LM})$ 的近似很差，我们应该对 μ 加倍，使得下一次迭代更加接近于最速下降的迭代方向，并且减少迭代的步长。

Algorithm 3.21. Marquardt's Method 3)

```
begin
  k := 0;   ν := 2;   x := x_0
  A := J_f(x)^T J_f(x);   g := J_f(x)^T f(x)
  found := (||g||_∞ ≤ ε_1);   μ := τ * max{a_ii}
  while (not found) and (k < k_max)
    k := k+1;   Solve (A + μI)h_M = −g
    if ||h_M|| ≤ ε_2||x||
      found := true
    else
      x_new := x + h_M
      ϱ := (F(x) − F(x_new))/(L(0) − L(h_M))          {cf. (3.15)}
      if ϱ > 0                                        {step acceptable}
        x := x_new
        A := J_f(x)^T J_f(x);   g := J_f(x)^T f(x)
        found := (||g||_∞ ≤ ε_1)
        μ := μ * max{1/3, 1 − (2ϱ − 1)^3};   ν := 2
      else
        μ := μ * ν;   ν := 2 * ν
end
```