

VIO 第三章作业

曾卓

05 2021

1 参数估计

1.1 请绘制样例代码中 LM 阻尼因子 μ 随着迭代变化的曲线图

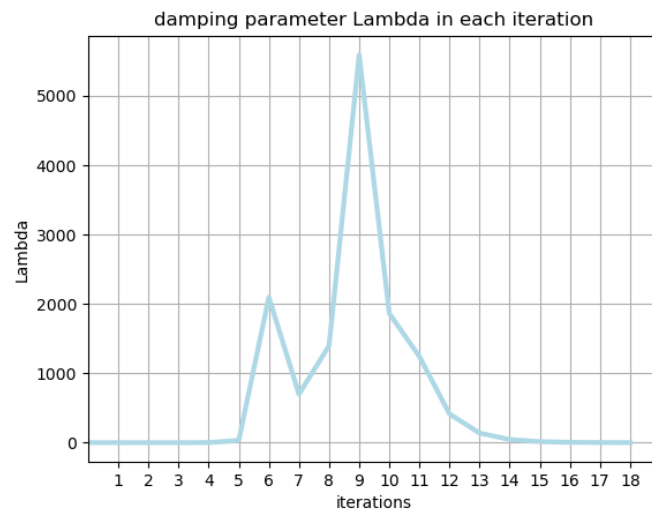


图 1: 阻尼因子 μ 随着迭代变化的曲线图

1.2 将曲线函数改成 $y = ax^2 + bx + c$, 请修改样例代码中残差计算, 雅克比计算等函数, 完成曲线参数估计

修改残差项和雅可比即可

```
residual__(0) = abc(0)*x__*x__ + abc(1)*x__ + abc(2) - y__;
```

```
jaco_abc « x__ * x__, x__, 1;
```

如果采用原始数据, 拟合效果较差, 可以通过以下操作改进:

- 1、增加数据点数, 如 $N=1000$ (原始 $N=100$)
- 2、增大步长以增大数据范围, 如 $x=i/10$ (原始 $x=i/100$)
- 3、减小噪声方差, 如 $w_sigma=0.01$ (原始 $w_sigma=0.1$)

```

Test CurveFitting start...
iter: 0 , chi= 61493.7 , Lambda= 0.001
iter: 1 , chi= 91.3952 , Lambda= 0.000333333
iter: 2 , chi= 91.395 , Lambda= 0.000222222
problem solve cost: 0.157363 ms
makeHessian cost: 0.065848 ms
-----After optimization, we got these parameters :
10.6107 19.6183 9.99517
-----ground truth:
10 20 10

```

图 2: 参数估计结果

1.3 实现其他更优秀的阻尼因子策略，并给出实验对比

对 $y = e^{(ax^2+bx+c)}$ 进行拟合

原代码采用的是类似 Nielsen 的方法，但是对 alpha 进行了一定的限制 ($\leq 2/3$)，避免其过大从而使 lambda 下降得更快一些。

参数拟合结果如下：

```

Test CurveFitting start...
iter: 0 , chi= 36048.3 , Lambda= 0.001
iter: 1 , chi= 30015.5 , Lambda= 699.051
iter: 2 , chi= 13421.2 , Lambda= 1864.14
iter: 3 , chi= 7273.96 , Lambda= 1242.76
iter: 4 , chi= 269.255 , Lambda= 414.252
iter: 5 , chi= 105.473 , Lambda= 138.084
iter: 6 , chi= 100.845 , Lambda= 46.028
iter: 7 , chi= 95.9439 , Lambda= 15.3427
iter: 8 , chi= 92.3017 , Lambda= 5.11423
iter: 9 , chi= 91.442 , Lambda= 1.70474
iter: 10 , chi= 91.3963 , Lambda= 0.568247
iter: 11 , chi= 91.3959 , Lambda= 0.378832
problem solve cost: 0.485304 ms
makeHessian cost: 0.264057 ms
-----After optimization, we got these parameters :
0.941939 2.09453 0.965586
-----ground truth:
1 2 1

```

采用 ppt 中的 Marquardt 的方法

```

//// method from Marquardt, 1963
double rho = (currentChi_ - tempChi) / scale;
if (rho > 0.75 && isfinite(tempChi)) {
    currentLambda_ /= 3;
    currentChi_ = tempChi;
    return true;
} else if (rho < 0.25) {
    currentLambda_ *= 2;
}
return false;

```

结果：效果欠佳，结束迭代的原因应该是错误次数过多，所以必须把 false_cnt 改成更大的值。

```

Test CurveFitting start...
iter: 0 , chi= 36048.3 , Lambda= 0.001
iter: 1 , chi= 16035.7 , Lambda= 349.525
problem solve cost: 0.756021 ms
makeHessian cost: 0.047206 ms
-----After optimization, we got these parameters :
0.720116 0.87538 1.13788
-----ground truth:
1 2 1

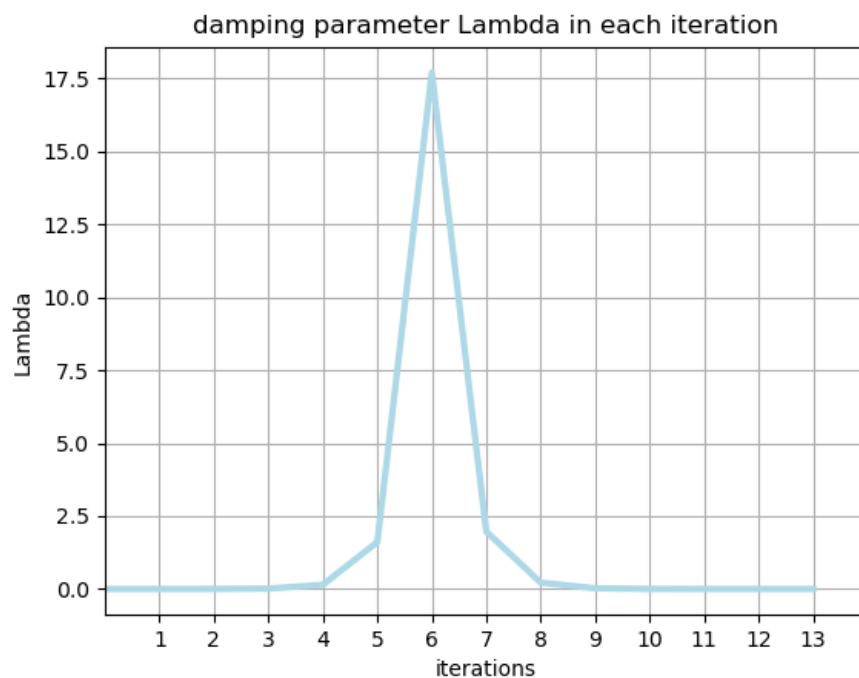
```

采用 Gavin 论文中的第一种方法

```
//// method 1 from Gavin
//// need to change the delta_x and scale with additional diag(Hessian_)
double L = 2.;
if (rho > 0 && isfinite(tempChi)){
    currentLambda_ = max(currentLambda_ / L, 1e-7);
    currentChi_ = tempChi;
    return true;
}else{
    currentLambda_ = min(currentLambda_ * L , 1e7);
    return false;
}
```

结果:效果还行,但是 Lambda 很快便减小为一个很小的值,使其接近与 GN 算法,应该在调整参数 L 和初始值后采用了论文中建议的 L 值后效果如下:

```
Test CurveFitting start...
iter: 0 , chi= 36048.3 , Lambda= 1e-05
iter: 1 , chi= 1521.82 , Lambda= 1.9684
iter: 2 , chi= 940.205 , Lambda= 0.218711
iter: 3 , chi= 689.792 , Lambda= 0.0243012
iter: 4 , chi= 216.426 , Lambda= 0.00270014
iter: 5 , chi= 101.56 , Lambda= 0.000300015
iter: 6 , chi= 92.361 , Lambda= 3.3335e-05
iter: 7 , chi= 91.4 , Lambda= 3.70389e-06
iter: 8 , chi= 91.3959 , Lambda= 4.11544e-07
problem solve cost: 0.426983 ms
makeHessian cost: 0.19941 ms
-----After optimization, we got these parameters :
0.941957 2.0945 0.965601
-----ground truth:
1 2 1
```



采用 Gavin 论文中的第二种方法

```
//// method 2 from Gavin
// currentLambda_ *= maxDiagonal;
double temp_bx = b_.transpose() * delta_x_;
double alpha = temp_bx / ((tempChi - currentChi_) / 2 + 2 * temp_bx);
cout << "alpha is : " << alpha << endl;
alpha = min(alpha, 1e-1);
//// compute rho(alpha * delta_x_)
//// roll back to compute Chi with alpha * delta_x_
//// rest procedure is same as before
RollbackStates();
delta_x_ *= alpha;
UpdateStates();
tempChi = 0.0;
for (auto edge: edges_) {
    edge.second->ComputeResidual();
    tempChi += edge.second->Chi2();
}
scale = 0;
scale = delta_x_.transpose() * (currentLambda_ * delta_x_ + b_);
scale += 1e-3;
double rho = (currentChi_ - tempChi) / scale;
if (rho > 0 && isfinite(tempChi)) // last step was good, 误差在下降
{
    currentLambda_ = max(currentLambda_ / (1 + alpha), 1e-7);
    currentChi_ = tempChi;
    return true;
} else {
    currentLambda_ += abs(tempChi - currentChi_) / (2 * alpha);
    return false;
}
```

结果：效果欠佳，alpha 在初值 0, 0, 0 下几乎为零，改用其它初值结果也不理想，应该还有点问题需要完善。

2 公式推导，根据课程知识，完成 F,G 中如下两项的推导过程

$$\begin{pmatrix} \alpha_{k+1} \\ \theta_{k+1} \\ \rho_{k+1} \\ b_{k+1}^a \\ b_{k+1}^g \end{pmatrix} = F \begin{pmatrix} \delta \alpha_k \\ \delta \theta_k \\ \delta \rho_k \\ \delta b_k^a \\ \delta b_k^g \end{pmatrix} + G \begin{pmatrix} n_k^a \\ n_k^g \\ n_{k+1}^a \\ n_{k+1}^g \\ n_{b_k^a} \\ n_{b_k^g} \end{pmatrix} \quad f_1: \alpha_{k+1} = \alpha_k + \beta_k \delta t + \frac{1}{2} a \cdot \delta t^2$$

$$a = \frac{1}{2} (q_{b_i b_k} (a_k^b - b_k^a) + q_{b_i b_{k+1}} (a_{k+1}^b - b_k^a))$$

$$\textcircled{1} \alpha_{k+1} + \delta \alpha_{k+1} = \alpha_k + \beta_k \delta t + \frac{1}{4} (q_{b_i b_k} (a_k^b - b_k^a) + q_{b_i b_k} \otimes \left[\frac{1}{2} \underline{w} \delta t \right] (a_{k+1}^b - b_k^a) \overset{A}{\delta t^2}) \delta t^2$$

$$\textcircled{2} \underline{w} = \frac{1}{2} (w^{b_{k+1}} - b_k^g - \delta b_k^g + w^{b_k} - b_k^g - \delta b_k^g) = w - \delta b_k^g$$

$$\textcircled{3} \therefore \overset{A}{\delta t^2} = \left[q_{b_i b_k} \otimes \left[\frac{1}{2} (w - \delta b_k^g) \delta t \right] (a_{k+1}^b - b_k^a) \right]$$

$$\textcircled{4} = \left[R_{b_i b_k} \cdot \exp((w - \delta b_k^g)^\top \delta t) (a_{k+1}^b - b_k^a) \right]$$

$$\textcircled{5} = \left[R_{b_i b_k} \exp(w^\top \delta t) \cdot \exp(-(\underline{J}_r(w \delta t) \delta b_k^g)^\top \delta t) (a_{k+1}^b - b_k^a) \right]$$

$$\textcircled{6} = \left[R_{b_i b_{k+1}} (I - (\underline{J}_r(w \delta t) \delta b_k^g)^\top \delta t) (a_{k+1}^b - b_k^a) \right]$$

$$\textcircled{7} \therefore \frac{\partial \alpha_{k+1}}{\partial b_k^g} = \lim_{\delta b_k^g \rightarrow 0} \frac{\delta \alpha_{k+1}}{\delta b_k^g} = \lim_{\delta b_k^g \rightarrow 0} \frac{\frac{1}{4} [R_{b_i b_{k+1}} (-\underline{J}_r(w \delta t) \delta b_k^g)^\top \delta t (a_{k+1}^b - b_k^a)] \delta t^2}{\delta b_k^g}$$

$$\textcircled{8} = \frac{1}{4} R_{b_i b_{k+1}} (a_{k+1}^b - b_k^a)^\top \delta t^2 \cdot \underline{J}_r(w \delta t) \delta t \approx -\frac{1}{4} R_{b_i b_{k+1}} (a_{k+1}^b - b_k^a)^\top \delta t^2 (-\delta t) \quad \left(\begin{array}{l} \text{和 } f_{35} \text{ 很像，分别是} \\ \text{速度 } w t, \text{ 位移 } \frac{1}{2} a t^2 \\ \text{差了一个 } \frac{1}{2} t, \text{ 而已} \end{array} \right)$$

图 3: f_{15} 推导结果

$$\begin{pmatrix} \alpha_{k+1} \\ \theta_{k+1} \\ p_{k+1} \\ b_{k+1}^g \\ b_{k+1}^a \end{pmatrix} = F \cdot \begin{pmatrix} \alpha_k \\ \theta_k \\ p_k \\ b_k^g \\ b_k^a \end{pmatrix} + G \cdot \begin{pmatrix} n_k^a \\ n_k^g \\ n_{k+1}^a \\ n_{k+1}^g \\ n_{k+1}^a \end{pmatrix}$$

$$g, : \alpha_{k+1} = \alpha_k + \beta_k \cdot \delta t + \frac{1}{2} a \cdot \delta t^2$$

$$a = \frac{1}{2} \left(q_{b_k b_k} (a_{b_k}^b - b_k^a) + q_{b_k b_{k+1}} (a_{b_{k+1}}^b - b_k^a) \right)$$

$$\alpha_{k+1} + \delta \alpha_{k+1} = \alpha_k + \beta_k \delta t + \frac{1}{4} \left[q_{b_k b_k} (a_{b_k}^b - b_k^a) + q_{b_k b_{k+1}} \otimes \begin{bmatrix} 1 \\ \frac{1}{2} \underline{w} \delta t \end{bmatrix} (a_{b_{k+1}}^b - b_k^a) \right] \delta t^2$$

$$\underline{w} = \frac{1}{2} \left[(\bar{w}_k + n_k^g - b_k^g) + (\bar{w}_{k+1} - b_k^g) \right] = w + \frac{1}{2} n_k^g$$

$$\underline{A} = R_{b_k b_{k+1}} \cdot \exp \left[\left(w + \frac{1}{2} n_k^g \right)^T \delta t \right] \cdot (a_{b_{k+1}}^b - b_k^a)$$

$$= R_{b_k b_{k+1}} \cdot \exp [w \delta t]^T \cdot \exp \left[\left(J_r(w \delta t) \cdot \frac{1}{2} n_k^g \right)^T \delta t \right] \cdot (a_{b_{k+1}}^b - b_k^a)$$

$$= R_{b_k b_{k+1}} \cdot \left(I + \left(J_r(w \delta t) \cdot \frac{1}{2} n_k^g \right)^T \delta t \right) (a_{b_{k+1}}^b - b_k^a)$$

$$\therefore \frac{\partial \alpha_{k+1}}{\partial n_k^g} = \frac{\partial \frac{1}{4} R_{b_k b_{k+1}} \cdot \left(J_r(w \delta t) \cdot \frac{1}{2} n_k^g \right)^T \delta t (a_{b_{k+1}}^b - b_k^a) \delta t^2}{\partial n_k^g}$$

$$= -\frac{1}{4} R_{b_k b_{k+1}} (a_{b_{k+1}}^b - b_k^a)^T \overset{\approx I}{J_r(w \delta t)} \cdot \delta t^2 \cdot \left(\frac{1}{2} \delta t \right)$$

$$\approx -\frac{1}{4} R_{b_k b_{k+1}} (a_{b_{k+1}}^b - b_k^a)^T \delta t^2 \cdot \left(\frac{1}{2} \delta t \right)$$

图 4: g_{12} 推导结果

3 证明式 9

$$(J^T J + M I) \Delta x = -J^T f, \quad F' = (J^T f)^T$$

通常 EVD 是 $X = V \Lambda V^{-1}$, 但是由于 $J^T J$ 是 symmetric, semi-positive definite. 可以证明此时特征值可以选取为相互正交的, (即为 SVD)

即 $V^{-1} = V^T \therefore J^T J = V \Lambda V^T, \quad V V^T = I$

$$(V \Lambda V^T + M V V^T) \Delta x = -F'^T$$

$$V (\Lambda + M I) V^T \Delta x = -F'^T$$

$$\Delta x = -V (\Lambda + M I)^{-1} V^T F'^T$$

展开右式 即得

$$\Delta x = -\sum_{j=1}^n \frac{1}{\lambda_j + M} \cdot V_j \underbrace{V_j^T \cdot F'^T}_{\text{标量, 位置元所谓}}$$

$$= -\sum_{j=1}^n \frac{V_j^T \cdot F'^T}{\lambda_j + M} \cdot V_j$$