Homework3 Coding Part

Yuguang Yue

October 20, 2017

Problem 1,2. Compare trust-region method and coordinate descent.

Proof. Because the objective function has matrix variable

$$f(B,C)=\sum_i\sum_j g_{ij}(a_{ij}-(e_i^TB)(Ce_j))^2+\frac{\mu}{2}(||B||_F^2+||C||_F^2)$$

it is hard to write down the second order approximation using Hessian matrix. Therefore, we treat individual rows or columns as variable instead of the original matrix. Since there is no constraint on each row or column, we can rewrite the problem with respect to B's row and C's column, specifically

$$f(B_i,C_j)=\sum_i\sum_j g_{ij}(a_{ij}-B_iC_j)^2+\frac{\mu}{2}(\sum_i||B_i||^2+\sum_j||C_j||^2)$$

then the variables becomes

$$v=[B_1,B_2,\cdots,B_n,C_1,\cdots,C_n]$$

taking derivative w.r.t v is equivalent to

$$g=\frac{\partial f}{\partial v}=[\frac{\partial f}{\partial B_1},\cdots,\frac{\partial f}{\partial B_n},\frac{\partial f}{\partial C_1},\cdots,\frac{\partial f}{\partial C_n}]$$

which is also the first order term of $m_k(p)$ in trust region method.

In order to get second order term, we use Hessian w.r.t v to approximate, which is equivalent to

$$B=\begin{bmatrix}\frac{\partial^2 f}{\partial B_1\partial B_1^T} & \cdots & \frac{\partial^2 f}{\partial B_1\partial B_n^T} & \frac{\partial^2 f}{\partial B_1\partial C_1^T} & \frac{\partial^2 f}{\partial B_1\partial C_n^T}\\ \vdots & \vdots & \vdots & \vdots & \vdots\\ \frac{\partial^2 f}{\partial C_n\partial B_1^T} & \cdots & \frac{\partial^2 f}{\partial C_n\partial B_n^T} & \frac{\partial^2 f}{\partial C_n\partial C_1^T} & \frac{\partial^2 f}{\partial C_n\partial C_n^T}\end{bmatrix}$$

For each entry, we have

$$\frac{\partial f}{\partial B_i}=2\sum_j g_{ij}(B_i'C_j-a_{ij})C_j+\mu B_i$$

because $B_i'C_jC_j = C_j'B_iC_j = C_jC_j'B_i$, we can further get

$$\frac{\partial f}{\partial B_i} = (\mu I + 2\sum_j g_{ij}C_jC_j')B_i - 2\sum_j g_{ij}a_{ij}C_j \tag{1}$$

similarly, we can get

$$\frac{\partial f}{\partial C_j} = (\mu I + 2\sum_i g_{ij}B_iB_i')C_j - 2\sum_j g_{ij}a_{ij}B_j \tag{2}$$

it is clear from (1) and (2) that $\frac{\partial^2 f}{\partial B_i \partial B_j'} = 0$ and $\frac{\partial^2 f}{\partial C_i \partial C_j'} = 0$, so the Hessian is block-wise diagonalized with right upper and left lower parts non-zero. Specifically, we have

$$\frac{\partial^2 f}{\partial B_i \partial B_i'} = \mu I + 2\sum_j g_{ij}C_jC_j' \tag{3}$$

$$\frac{\partial^2 f}{\partial C_i \partial C_i'} = \mu I + 2\sum_i g_{ij}B_iB_i' \tag{4}$$

$$\frac{\partial^2 f}{\partial B_i \partial C_j'} = -2g_{ij}a_{ij}I + 2g_{ij}C_j'B_iI + 2g_{ij}B_iC_j' \tag{5}$$

I apply the same scheme from lecture 8 and Cauchy point method to get the roughly optimal p_k. For coordinate descent, just solve (1) and (2) iteratively to update B_i and C_j to get optimum.

Data Generation: I generate $n = 10, r = 5$ matrix with sparsity ratio 5%, the algorithm can be largely accelerated by the sparsity property of G, however I treat it as normal matrix here. And I choose $\mu = 0.05$.

Result: In my experiment, coordinate descent outperform trust region largely. They achieve the same optimum value 0.0509, while coordinate descent uses 40 iterations with 0.378 seconds, trust-region uses 297 iterations with 1.106 seconds. Though they achieve same optimum value, the matrix they returned are totally different. It's obvious that the matrices returned from coordinate descent is very sparse, while the matrices from trust-region has very small values but not zero. I think this is because I used the Cauchy point calculation to get direction instead of exact direction. □