Rutgers University
Summer Research Program

Algorithm Recurrence of First Paper

Han Liu

July 31, 2019

1 Introduction

In the process of training and predicting a gaussian process (GP), we need to get three quantities: the mean of predicted input, the log marginal likelihood function and its derivative. Their expressions are given as following:

$$u_{f|D}(\hat{x}) = u(\hat{x}) + k_{X\hat{x}}^T K^{-1} y \tag{1}$$

$$logp(y|K) = -\frac{1}{2} log((2\pi)^k |K|) - \frac{1}{2} y^T K^{-1} y \tag{2}$$

$$\frac{\partial}{\partial \theta_j} logp(y|X,\theta) = \frac{1}{2} y^T K^{-1} \frac{\partial K}{\partial \theta_j} K^{-1} y - \frac{1}{2} tr(K^{-1} \frac{\partial K}{\partial \theta_j}) \tag{3}$$

These equation have three operations in common that dominate its time complexity: $K^{-1}y$, $log|K|$, $tr(K^{-1}\frac{\partial K}{\partial \theta_j})$. Before that, I use Cholesky decomposition of K to compute all three quantities (see *GP_code.py*), but it is very computational expensive, thus we need some parallel computional algorithm.

I think the innovative part of this paper is to use matrix-matrix multipy to combine three algorithms (Conjugate Gradient, Lancos Algorithm, Pivoted Cholesky Decomposition) together. The most important formula is as following:

$$[u_0 \quad u_1 \quad ... \quad u_t] = K^{-1}[y \quad z_1 \quad ... \quad z_t] \tag{4}$$

The whole algorithm flow chart can be seen in Figure 1. In the following section, I will implement all the three algorithms individually by MATLAB.

2 Implementation of mBCG

In this section, I firstly implement standard preconditioned conjugate gradient algorithm, then I try to implement modified conjugate gradient algorithm given in Equation 4.

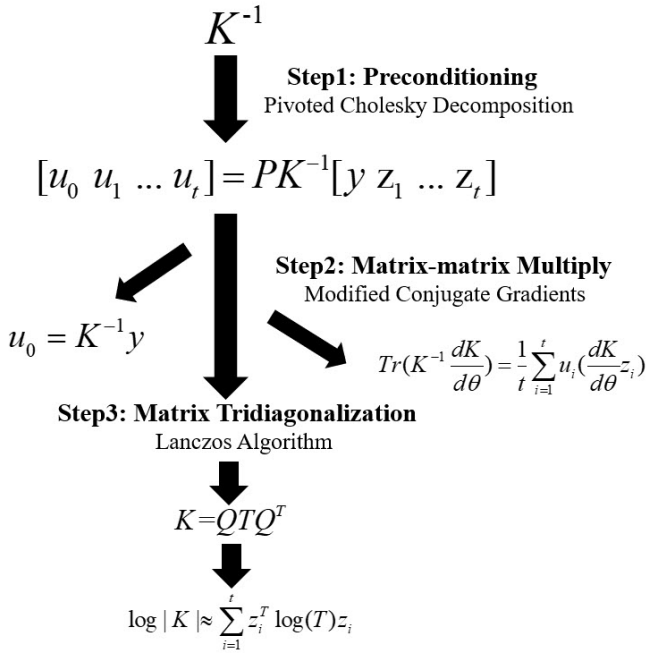


Figure 1: Overall Algorithm Flow Chart

2.1 Standard PCG

For the implementation of standard preconditioned conjugate gradient (PCG), I carefully follow the **Algorithm 1** in the paper. But unfortunately, I failed to get the correct results. It bothered me almost entire day, and I refer many literature, and try to implement this algorithm in a different dimension. For example, **Conjugate Gradient Method** [1] from Stanford University specify the process as following:

CG algorithm

(follows C. T. Kelley)

$x := 0, \quad r := b, \quad \rho_0 := \|r\|^2$

for $k = 1, \ldots, N_{\max}$
  quit if $\sqrt{\rho_{k-1}} \leq \epsilon\|b\|$
  if $k = 1$ then $p := r$; else $p := r + (\rho_{k-1}/\rho_{k-2})p$
  $w := Ap$
  $\alpha := \rho_{k-1}/p^T w$
  $x := x + \alpha p$
  $r := r - \alpha w$
  $\rho_k := \|r\|^2$

Figure 2: Conjugate Gradient Method

The MATLAB code is given in $PCG\_stanford.m$. Finally, I found there are two mistakes in the paper, which can be seen in the following figure:

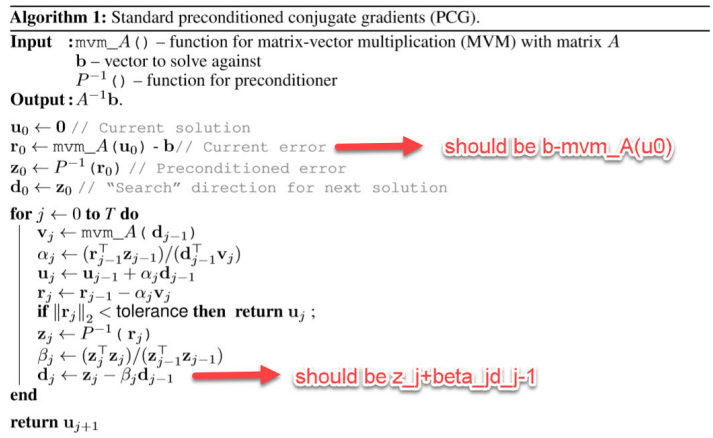


Figure 3: Correction of PCG Algorithm

A simple test for the following input is given in Figure 4:

$$A = \begin{bmatrix} 6 & 3 & 0 \\ 3 & 6 & -6 \\ 0 & -6 & 11 \end{bmatrix}, b = \begin{bmatrix} 3 \\ 1 \\ 2 \end{bmatrix} \tag{5}$$

```
ans =

   -3.3704
    8.0741
    5.2222

>> inv(mvm_A)*b

ans =

   -3.3704
    8.0741
    5.2222
```

Figure 4: Matlab Test Results

2.2 Modified PCG

To implement a modified (batched) PCG, we need to convert MVM to MMM. There are several modifications of original algrithm, and some proofs are given in appendix. The MATLAB code is as following:

```
%Modified preconditioned conjugate gradients

mmm_A=  [ -3.3704    0.2593   -3.6296;
8.0741    0.4815    7.9259;
5.2222    0.4444    4.7778];
B=[4 3 2;7 1 8;9 2 5];
col=size(B,2);
U=zeros(3,3);
R=B-mmm_A*U;
Z=R;
D=Z;
alpha=zeros(3,1);
beta=zeros(3,1);
t=3;        %number of iteration
e=0.0001;

for j=1:3
    V=mmm_A*D;
    alpha=(dot(R,Z)./dot(D,V))';
    U= (U'+diag(alpha)*D')';
    R= (R'-diag(alpha)*V')';
    norm_matrix=zeros(1,col);
    for i=1:col
        norm_matrix(i)=norm(R(:,i),2);
    end
    if any(any(norm_matrix<e))
        break
    end
    Z_old=dot(Z,Z);
    Z=R;
    beta=(dot(Z,Z)./Z_old)';
    D=(Z'+diag(beta)*D')';
end
display(U)
```

3 Implementation of Lanczos Algorithm

To implement Lanczos Algorithm for tridiagonalizing a symmetric matrix A, I refer the alogorithm given in **Algorithm 2**. Specifically, we have:

$$\begin{aligned} \forall i \quad [T_i]_{j,j} &= 1/[\alpha_j]_i + [\beta_{j-1}]_i/[\alpha_{j-1}]_i \\ [T_i]_{j-1,j}, [T_i]_{j,j-1} &= \sqrt{[\beta_{j-1}]}/[\alpha_j]_i \end{aligned} \tag{6}$$

I have implemented this algorithm in MATLAB as following:

```
function [ T ] = Lanczos_Algorithm( alpha,beta )
    col=size(alpha,2);  %iteration
    row=size(alpha,1);  %third-dimension of T
    T=zeros(col,col,row);
    for i=1:row
        for j=1:col
```

```
            if j==1
                T(1,1,i)=1/alpha(i,1);
            else
                T(j,j,i)=1/alpha(i,j)+beta(i,j-1)/alpha(i,j-1);
                T(j-1,j,i)=sqrt(beta(i,j-1))/alpha(i,j-1);
                T(j,j-1,i)= T(j-1,j,i);
            end
        end
    end
end
```

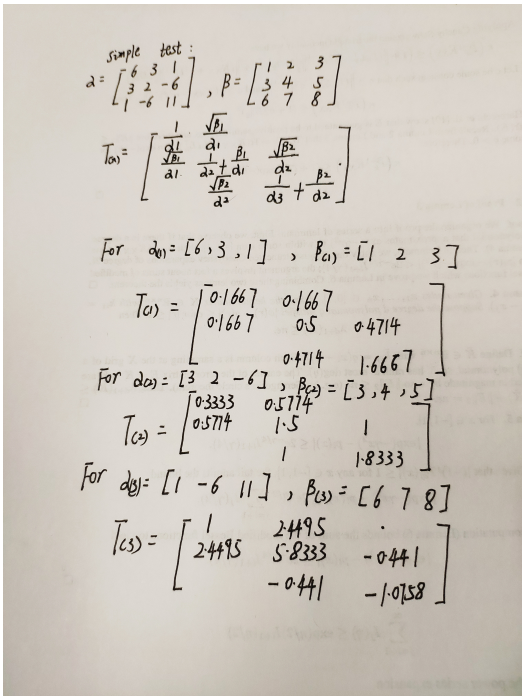
To test the correctness of this algorithm, I have choose two matrixes:

$$\alpha = \begin{bmatrix} 6 & 3 & 1 \\ 3 & 2 & -6 \\ 1 & -6 & 11 \end{bmatrix}, \beta = \begin{bmatrix} 1 & 2 & 3 \\ 3 & 4 & 5 \\ 6 & 7 & 8 \end{bmatrix} \tag{7}$$

The calculation results from calculator and program operation results are as following:

```
T(:, :, 1) =

    0.1667    0.1667         0
    0.1667    0.5000    0.4714
         0    0.4714    1.6667


T(:, :, 2) =

    0.3333    0.5774         0
    0.5774    1.5000    1.0000
         0    1.0000    1.8333


T(:, :, 3) =

    1.0000    2.4495         0
    2.4495    5.8333   -0.4410
         0   -0.4410   -1.0758
```

(a) MATLAB Results



(b) Calculator Results

Figure 5: Comparison of Two Implementations

4 Implementation of Pivoted Cholesky Decomposition

The preconditioning is an operation of utilizing hardware more efficiently. In this paper, the author use **Pivoted Cholesky Decomposition (PCG)**. Specifically, PCG algorithm will produce a low-

rank approximation of a positive definite matrix, which is:

$$K_{XX} = L_k L_k^T \tag{8}$$

The pseudo-code of PCG algorithm is as following [2]:

Algorithm: cost $O(nm^2)$

Data: matrix $\mathbf{A} = [a_{i,j}] \in \mathbb{R}^{n \times n}$ and error tolerance $\varepsilon > 0$
Result: low-rank approximation $\mathbf{A}_m = \sum_{i=1}^{m} \boldsymbol{\ell}_i \boldsymbol{\ell}_i^T$ such that $\text{trace}(\mathbf{A} - \mathbf{A}_m) \leq \varepsilon$

begin
- set $m := 1$;
- set $\mathbf{d} := \text{diag}(\mathbf{A})$ and $error := \|\mathbf{d}\|_1$;
- initialize $\boldsymbol{\pi} := (1, 2, \ldots, n)$;
- **while** $error > \varepsilon$ **do**
  - set $i := \arg\max\{d_{\pi_j} : j = m, m+1, \ldots, n\}$;
  - swap π_m and π_i;
  - set $\ell_{m,\pi_m} := \sqrt{d_{\pi_m}}$;
  - **for** $m + 1 \leq i \leq n$ **do**
    - compute $\ell_{m,\pi_i} := \left(a_{\pi_m,\pi_i} - \sum_{j=1}^{m-1} \ell_{j,\pi_m} \ell_{j,\pi_i} \right) \Big/ \ell_{m,\pi_m}$;
    - update $d_{\pi_i} := d_{\pi_i} - \ell_{m,\pi_m} \ell_{m,\pi_i}$;
  - compute $error := \sum_{i=m+1}^{n} d_{\pi_i}$;
  - increase $m := m + 1$;

end

Figure 6: Pivoted Cholesky Decomposition Algorithm

I implement this algorithm by MATLAB:

```
%Implementation of Pivited Cholesky Composition
A=[6 3 0;
3 6 -6;
0 -6 11];    %semi-positive matrix

e=0.1;             %Error
n=size(A,1);

m=1;
d=diag(A);
error=norm(d,1);
v=randperm(n);
Pi = sort(v);
l=zeros(n,n);   %triangular matrix we want

while error>e
    [argvalue, i] = max(d(Pi(m:end)));
    a=Pi(m);
    Pi(m)=Pi(i);
    Pi(i)=a;
```

```
21      l(m,Pi(m))=sqrt(d(Pi(m)));
22      for i = (m+1:n)
23          S=0;
24          for j =(1:m-1)
25              l(j,Pi(m))
26              l(j,Pi(i))
27              S=S+l(j,Pi(m)).*l(j,Pi(i));
28          end
29          l(m,Pi(i))=(A(Pi(m),Pi(i))-S)./l(m,Pi(m));
30          d(Pi(i))=d(Pi(i))-l(m,Pi(m)).*l(m,Pi(i));
31      end
32  %compute error
33      error_s=0;
34      for i = (m+1:n)
35          error_s=error_s+d(Pi(i));
36      end
37      error=error_s;
38      m=m+1;
39  end
40
41  S_Am=zeros(3,3);
42  for i=(1:m-1)
43      S_Am=S_Am+l(:,i)*l(:,i)';
44  end
45  Am=S_Am;
```

References

[1] https://stanford.edu/class/ee364b/lectures/conj_grad_slides.pdf

[2] Helmut Harbrecht, Michael Peters, and Reinhold Schneider, "The pivoted Cholesky decomposition and its application to stochastic PDEs", p7. http://www.mathe.tu-freiberg.de/naspde2010/sites/default/files/harbrecht.pdf

A Derivation of mBCG

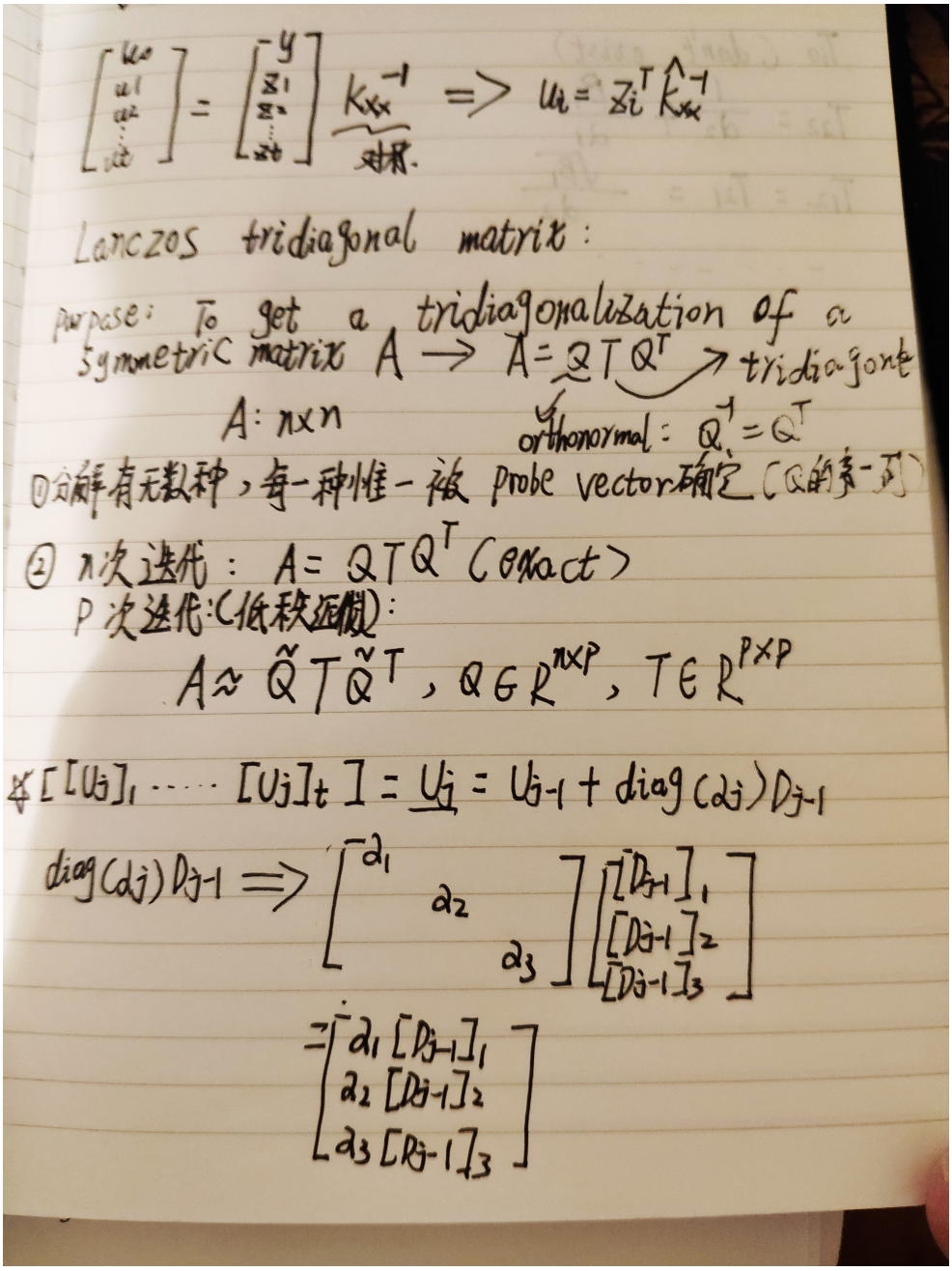


Figure 7: Derivation of mBCG