Паралелизация на алгоритъма SINCO

Теодора Христева

Резюме: *Проблемът със селектираната обратна ковариация се среща в много практически приложения. SINCO(Sparse INverse Covariance) е алгоритъм за решаване такив вид задачи. Основното предимство на този алгоритъм е простотата на неговото прилагане и потенциала му за паралелизиране. Алгоритъмът е разпаралелен с използване на технологията OpenMP, направени са измервания на бързодействието и анализ на резултатите.*

Ключови думи: *SINCO, OpenMP, паралелелизация, алгоритми*

Parallelization of the SINCO algorithm

Teodora Hristeva

Abstract*: Sparse inverse covariance selection problem is encountered in many practical applications. SINCO is an algorithm for the Sparse INverse COvariance problem. The main advantage of this algorithm is the simplicity of its implementation and his parallelizing potential. The algorithm is parallelized using OpenMP, the speedup is measured and the results are analyzed.*

Key words: *SINCO, OpenMP, parallelization, algorithm*

1. INTRODUCTION

Sparse inverse covariance selection problem is encountered in many practical applications. SINCO is an algorithm for the Sparse INverse COvariance problem. It is also an algorithm that allows parallelization.

2. SINCO ALGORITHM

In each step, the algorithm consists of two phases: search phase and update phase. In the search phase, the method computes potential updates for all coordinates Xij and selects the best coordinate producing the most increase in the objective function. In the update phase, using the Xij information the method updates the coordinate in the inverse covariance matrix which is a rank two update X*ij

Next step is to update the maintained estimate of ⟦W=X⟧^(-1) using Sherman-Morrison formula for the rank-two update. This step is necessary as values of W are extensively used in the search phase of the algorithm. Each sub problem of the search phase will be a one-dimensional problem as

$$f_{ij}(\alpha) = \max \frac{n}{2} \log\det(X+\alpha(e_i e_j^T + e_j e_i^T)) - \langle S, X+\alpha(e_i e_j^T + e_j e_i^T)\rangle - \lambda|X+\alpha(e_i e_j^T + e_j e_i^T)|.$$

The algorithm is a variant of a greedy coordinate descent algorithm for a convex problem and so it enjoys known convergence guarantees and a sub linear convergence rate. We can see that, since the updates are computed for each coordinate separately, it lends itself to distributed parallel setting. The data can be distributed among processors, assuming different distributions and methods for updating the inverse update step.

Our implementation of the parallelization is done with OpenMP and corresponds to the division of workload between different threads using the sections or parallel for directives. The first phase of the algorithm involves a simple search done on all coordinates, so we can just make the for loop parallel. In order to divide the compute load in the coordinate search phase, we need only to use #pragma parallel for for the search loop, however each thread although threads are now searching in the shared memory space, in order to save the results (maximum coordinates of each thread) we need to privatize the variables corresponding to

coordinates and function value of sections corresponding to thread's territory.

Outline of SINCO algorithm

1: Set $k = 0, X^{(k)} = I, W^{(k)} = I$
2: while $f(X^{k+1}) - f(X^k) > \epsilon$ do
3: for $\forall (i,j), 1 \le i \le j \le p$ do
4: Compute f_{ij} corresponding to updated value of the coordinate (i,j)
5: end for
6: Choose the best update α_{ij} for updating $(i,j) \leftarrow \arg\max_{(i,j)} f_{ij}$
7: Update coordinate $X_{ij}^{(k+1)} \leftarrow X_{ij}^{(k)} + \alpha_{ij}(e_i e_j^T + e_j e_i^T)$
8: Get $W^{(k+1)}$ by updating $W^{(k)}$ by Sherman-Morrison-Woodbury formula
9: $k \leftarrow k+1$
10: end while

Fig. 1 SINCO algorithm, described sequentially.

```
#pragma omp parallel for private(jj,upIndex)

#pragma omp critical
{
if (fnew > funmax ) {
    funmax=fnew;
    imax=i;
    jmax=j;
    alphamax=AllSteps(i,j).alpha;
    updatemax=AllSteps(i,j).update;}
}
```

Fig.2 Example of using pragmas in parallelism.

The important aspect that needs a specialized treatment is the update of the global best coordinate variables which will maintain the best coordinate information through all threads. To prevent race conditions we need the lines corresponding to this section in a critical statement so that no two threads execute these instructions simultaneously.

For the second phase which is updating the matrix with two rank one updates, as all the information is shared, there is no communication involved, however a barrier is needed before the second phase to ensure that all threads have synchronized information about the coordinates staged for updating and also the magnitudes of the updates. The update loop needs to be modified so that threads divide the load on updating the matrix coordinates of the inverse matrix to account for parallel environment.

3. RESULTS

From Fig. 3 can be seen that as the number of threads increases, the execution time increases. This happens when we have dependencies between iterations and it is necessary to use a critical section or barrier - this causes a consistent execution of the specified region. This option parallelizes the cycles automatically.

Threads	PA
2	197.88
4	371.19
8	1064.72

Fig.3 Starting the algorithm with a different number of threads.

There is another type of distribution of iterations shown in Fig. 4. In the block-cyclic distribution the 4 processors calculate the values of different elements of the matrix. This eliminates waiting in a critical section to retrieve a value from an adjacent item. As we can see with a parallel optimization algorithm, the execution time is less than the time of the parallel algorithm.

1	2	1	2
3	4	3	4
1	2	1	2
3	4	3	4

Fig. 4 Block-cyclic distribution

There is another type of distribution of iterations shown in Fig. 4. In the block-cyclic distribution the 4 processors calculate the values of different elements of the matrix. This eliminates waiting in a critical section to retrieve a value from an adjacent item. As we can see with a parallel optimization algorithm, the execution time is less than the time of the parallel algorithm.

4. CONCLUSION

The main advantage of this algorithm is the simplicity of its implementation and his parallelizing potential. This makes it suitable for solving a variety of tasks where optimization of calculations is required.

REFERENCES

[1] Friedman, J.; Hastie, T. & Tibshirani, R. (2008), 'Sparse inverse covariance estimation with the graphical lasso', Biostatistics 9 (3), 432--441.
[2] Bach, F., 2008b.Consistency of the group Lasso andmultiple kernel learning. Journal of Machine Learning Research 9, 1179-1225
[3] Bach, F., 2010. Self-concordant analysis for logistic regression. Electronic Journal of Statistics 4, 384–414.
[4] Rish, Irina and G. Grabarnik. “Sparse Modeling: Theory, Algorithms, and Applications.” (2014).
[5] Bach, F., Jenatton, R., Mairal, J., Obozinski, G., 2012. Optimization with sparsity-inducing penalties. Foundations and Trends in Machine Learning 4 (1), 1–106

За автора:

Теодора Христева, hristeva@tu-plovdiv.bg
Асистент, докторант
Катедра Компютърни системи и технологии
Факултет по Електроника и Автоматика
Технически университет – София,
Филиал Пловдив

About the author:

Teodora Hristeva, hristeva@tu-plovdiv.bg
Assistant, Phd Student.
Department Computer Systems and Technologies
Faculty of Electronics and Automation
Technical University - Sofia,
Plovdiv Branch