Scientific Software
Homework 5: Expression templates

Emil Loevbak, Joris Tavernier & Pieter Appeltans Leuven, December 6, 2018

1 Introduction – Linear Discriminant Analysis & Given Code

See homework 4.

2 Assignment

The interesting context aside, the homework is about expression templates. You will create expression templates for the matrix–vector product from SRDA

$$\mathbf{y} = (X^T X + \beta I)\mathbf{x}.$$

During the homework consider the following items: assume that explicitly storing the transpose of the matrix in the memory is too expensive ; assume that your code should work in the general matrix case (Do not compute the matrix–matrix product $A = X^T X$); the matrix–vector product from SRDA is used in an iterative method (CG).

1. Create an expression template for the transpose of a matrix. Assume that the following code creates the transpose

   ```
   transpose(C);
   ```

2. Create an expression template for the matrix–vector multiplication. Assume that the following code executes the matrix–vector multiplication

   ```
   y=A*x;
   ```

3. Create a functor or lambda function to to investigate the performance of the previously declared expression templates for the following matrix–vector product

   ```
   y=transpose(X)*(X*x)+beta*x;
   ```

   with x and y vectors; beta the parameter β; and X a matrix.

4. Compare the performance of the previous matrix–vector operation with a functor implementing the following code

   ```
   t=X*x;
   y=transpose(X)*t+beta*x;
   ```

   with x, y and t vectors; beta the parameter β; and X a matrix.

5. Derive the theoretical complexity for the matrix–vector multiplications from task 3 and 4 for a square N by N matrix X. Confirm these complexities by measuring the execution time of the matrix–vector multiplications in task 3 and 4 for a N by N randomly filled matrix X. Create a file `time_xtx.cpp` that performs these timings with a command line argument for N. You may use the `time_mv` functionality from `tws_util.hpp` provided on Toledo. Plot the execution times as well as the theoretical complexity in function of N. Try values for N in the range $[2, \ldots, 2^{10}]$ or even larger N if feasible. There is no need to do a perfect fit, but scale the theoretical complexity so you can at least compare the slope with the slope of the execution time in a logarithmically scaled plot.

6. (**extra**) Can you find an more efficient way to implement the matrix–vector multiplication $\mathbf{y} = (X^T X + \beta I)\mathbf{x}$? (Hint: the matrix–vector product $\mathbf{y} = X^T X\mathbf{x}$ can be implemented efficiently from a data access point of view.)

3 Practical information

The deadline for the submission is **31 December 12:00**. This deadline is strict! Do not wait until the last minute to submit, as we will not accept technical issues as an excuse for late submissions.

We expect you to submit your code together with a PDF containing design choices for the expressions and functors. Please also mention the total amount of time spent on the assignment in your report. This should be submitted in a ZIP file with the name `lastname_firstname_studentnumber.zip`, i.e., if your name is John Smith and your student number is r0123456, your file should be called `smith_john_r0123456.zip`. Name your main program files `srda.cpp` and `time_xtx.cpp`; and include the commands for compiling, linking and running of your program in comments at the top of the file. If this is more than a couple instructions, you can include a Makefile as an alternative. The question in the previous section marked extra are not necessary to achieve a reasonable passing grade for this assignment. However if you wish to go for extra marks, you can complete this question.

Some specific points to pay attention to:

- In your discussion, we expect you to be specific. This means that you should link concepts such as stability, condition, computational or memory complexity and so on to specific lines in your code (e.g. a specific operation in a given line of code, number of nested loops, . . .) to explain your design choices.
- This assignment is an opportunity to demonstrate that you have understood the concepts from the first three C++ exercise sessions. Have a look at the bullet-point list at the top of the two assignment sheets to see if there are concepts that you have not yet discussed, which apply to this assignment. Try to specifically use the key words that appear in these lists.
- Motivate your implementation decisions. Which concepts from the exercise sessions have you applied to your program?
- Provide detailed explanation of the advantages and/or disadvantages of using expression templates in general and specifically for the matrix–vector multiplications described in task 3 and 4.

References

[1] Deng Cai, Xiaofei He, and Jiawei Han. Srda: An efficient algorithm for large-scale discriminant analysis. *Knowledge and Data Engineering, IEEE Transactions on*, 20(1):1–12, 2008.

[2] James W Demmel. *Applied numerical linear algebra.* Siam, 1997.

[3] Geoffrey McLachlan. *Discriminant analysis and statistical pattern recognition*, volume 544. John Wiley & Sons, 2004.

[4] Christopher C Paige and Michael A Saunders. Lsqr: An algorithm for sparse linear equations and sparse least squares. *ACM transactions on mathematical software*, 8(1):43–71, 1982.