

# Implementación de BLAS para Optimización de Álgebra Matricial

Arantza Ivonne Pineda Sandoval, Ana Luisa Masetto Herrera, Alexis Solis Cancino

2019-02-16

## 1 Aceleración de Algoritmos de Álgebra Matricial en R

La multiplicación de matrices en R se maneja, en un nivel muy bajo, por la biblioteca que implementa las subrutinas de álgebra lineal básica (BLAS, por sus siglas en inglés). Cualquier versión de R que se descarga de CRAN viene con lo que se conoce como una *implementación de referencia* de BLAS. Esta implementación ciertamente funciona, produce lo que todos concuerdan que son las respuestas correctas, pero de ninguna manera está optimizado. Utilizando ciertas operaciones que se consideran como *benchmarks*, lo siguiente es lo que tardamos en hacer operaciones matriciales en la versión no optimizada de R:

```
R Benchmark 2.5
=====
Number of times each test is run_____ : 3

I. Matrix calculation
-----
Creation, transp., deformation of a 2500x2500 matrix (sec): 1.361
2400x2400 normal distributed random matrix ^1000_____ (sec): 0.2206666666666666
Sorting of 7,000,000 random values_____ (sec): 0.9146666666666667
2800x2800 cross-product matrix (b = a' * a)_____ (sec): 0.6163333333333335
Linear regr. over a 3000x3000 matrix (c = a \ b')_____ (sec): 0.3096666666666666
-----
Trimmed geom. mean (2 extremes eliminated): 0.55887405184253

II. Matrix functions
-----
FFT over 2,400,000 random values_____ (sec): 0.58
Eigenvalues of a 640x640 random matrix_____ (sec): 0.564
Determinant of a 2500x2500 random matrix_____ (sec): 0.2906666666666667
Cholesky decomposition of a 3000x3000 matrix_____ (sec): 0.3280000000000001
Inverse of a 1600x1600 random matrix_____ (sec): 0.3640000000000002
-----
Trimmed geom. mean (2 extremes eliminated): 0.406834814755095

III. Programming
-----
3,500,000 Fibonacci numbers calculation (vector calc)(sec): 0.3546666666666667
Creation of a 3000x3000 Hilbert matrix (matrix calc) (sec): 0.3800000000000003
Grand common divisors of 400,000 pairs (recursion)_____ (sec): 0.4286666666666665
Creation of a 500x500 Toeplitz matrix (loops)_____ (sec): 0.07033333333333328
Escoufier's method on a 45x45 matrix (mixed)_____ (sec): 0.3800000000000003
-----
Trimmed geom. mean (2 extremes eliminated): 0.371360626249155

Total time for all 15 tests_____ (sec): 7.162666666666667
Overall mean (sum of I, II and III trimmed means/3)_____ (sec): 0.438711914946038
--- End of test ---
```

Figure 1: Benchmark de álgebra matricial versión no optimizada (comercial) de R. El total de 15 pruebas se realiza en un total de 7.16 segundos.

Recientemente, sitios como *RBloggers* sugieren que un gran beneficio de su versión (comercial) de R es que estaba vinculada a una mejor biblioteca de álgebra lineal. El resumen rápido es que realmente sólo hace una diferencia para pruebas de referencia (*benchmarks*) bastante artificiales. Para el trabajo “normal” de día a día es poco probable que se vea una diferencia la mayor parte

del tiempo.

En la versión optimizada de R, el tiempo mejora de 7.16 segundos a 6.89 segundos:

```
R Benchmark 2.5
=====
Number of times each test is run_____: 3

I. Matrix calculation
-----
Creation, transp., deformation of a 2500x2500 matrix (sec): 1.25
2400x2400 normal distributed random matrix ^1000_____ (sec): 0.2233333333333334
Sorting of 7,000,000 random values_____ (sec): 0.9456666666666667
2800x2800 cross-product matrix (b = a' * a)_____ (sec): 0.5739999999999999
Linear regr. over a 3000x3000 matrix (c = a \ b')_____ (sec): 0.3263333333333333
-----
Trimmed geom. mean (2 extremes eliminated): 0.561612980258779

II. Matrix functions
-----
FFT over 2,400,000 random values_____ (sec): 0.4829999999999999
Eigenvalues of a 640x640 random matrix_____ (sec): 0.508
Determinant of a 2500x2500 random matrix_____ (sec): 0.2803333333333332
Cholesky decomposition of a 3000x3000 matrix_____ (sec): 0.3489999999999999
Inverse of a 1600x1600 random matrix_____ (sec): 0.322
-----
Trimmed geom. mean (2 extremes eliminated): 0.37862516575666

III. Programming
-----
3,500,000 Fibonacci numbers calculation (vector calc)(sec): 0.3440000000000001
Creation of a 3000x3000 Hilbert matrix (matrix calc) (sec): 0.4063333333333334
Grand common divisors of 400,000 pairs (recursion)_____ (sec): 0.443
Creation of a 500x500 Toeplitz matrix (loops)_____ (sec): 0.06866666666666653
Escoufier's method on a 45x45 matrix (mixed)_____ (sec): 0.375
-----
Trimmed geom. mean (2 extremes eliminated): 0.374246189438055

Total time for all 15 tests_____ (sec): 6.898666666666667
Overall mean (sum of I, II and III trimmed means/3)_____ (sec): 0.430131584060761
--- End of test ---

MacBook-Air-de-Ivonne-319:Desktop ivonnepineda$
```

Figure 2: Benchmark de álgebra matricial versión no optimizada (comercial) de R. El total de 15 pruebas se realiza en un total de 6.89 segundos.

Apple proporciona dos versiones de BLAS, la de referencia y una biblioteca BLAS optimizada (*vecLib*). La instalación de R para MacOS descargada de CRAN se envía con las bibliotecas *vecLib* y BLAS de referencia (la no optimizada se usa por defecto).

Para obtener los resultados de arriba, corrimos las pruebas primero para la versión no optimizada y después con la versión *vecLib*. Para cambiar a la versión optimizada, seguimos los pasos de [R: Use faster vecLib](https://gist.github.com/nicebread/6920c8287d7bffb03007)<sup>1</sup>. El código que se tiene que poner en la terminal es el siguiente:

<sup>1</sup><https://gist.github.com/nicebread/6920c8287d7bffb03007>

```
# Para cambiar a la librería optimizada vecLib
cd /Library/Frameworks/R.framework/Resources/lib
ln -sf /System/Library/Frameworks/Accelerate.framework/
    Frameworks/vecLib.framework/Versions/Current/libBLAS.dylib libRblas.dylib

# Para regresar a la librería por defecto
cd /Library/Frameworks/R.framework/Resources/lib
ln -sf libRblas.0.dylib libRblas.dylib
```

Además, el R script para generar las pruebas se incluye en el archivo `script_blas.R`.

## 1.1 OpenBLAS y Librería benchmarkme

Se encontró que la instalación de OpenBLAS llevó a una aceleración de 2 veces (de alrededor de 150 a 70 segundos). La mayoría de la ganancia de velocidad provino de las pruebas de álgebra matricial, como se puede ver en la siguiente figura. Hay que tener en cuenta que los resultados de dichas pruebas dependen en gran medida de las especificaciones de cada computadora. Sin embargo, muestra claramente que los puntos de referencia de “programación” (por ejemplo, el cálculo de los 3,500,000 números de Fibonacci) no son mucho más rápidos, mientras que los cálculos y las funciones de matrices reciben un aumento sustancial de la velocidad. Esto demuestra que la aceleración que se puede esperar de BLAS depende en gran medida del tipo de cálculos que se realicen.

```
library(benchmarkme)
bench <- benchmarkme::benchmark_std()

## # Programming benchmarks (5 tests):

## 3,500,000 Fibonacci numbers calculation (vector calc): 0.68 (sec).
## Grand common divisors of 1,000,000 pairs (recursion): 0.83 (sec).
## Creation of a 3,500 x 3,500 Hilbert matrix (matrix calc): 0.257 (sec).
## Creation of a 3,000 x 3,000 Toeplitz matrix (loops): 1.4 (sec).
## Escoufier's method on a 60 x 60 matrix (mixed): 0.977 (sec).

## # Matrix calculation benchmarks (5 tests):

## Creation, transp., deformation of a 5,000 x 5,000 matrix: 0.69 (sec).
## 2,500 x 2,500 normal distributed random matrix^1,000: 0.87 (sec).
## Sorting of 7,000,000 random values: 0.767 (sec).
## 2,500 x 2,500 cross-product matrix (b = a' * a): 10.8 (sec).
## Linear regr. over a 5,000 x 500 matrix (c = a \ b'): 0.863 (sec).

## # Matrix function benchmarks (5 tests):

## Cholesky decomposition of a 3,000 x 3,000 matrix: 5.95 (sec).
## Determinant of a 2,500 x 2,500 random matrix: 2.05 (sec).
## Eigenvalues of a 640 x 640 random matrix: 0.607 (sec).
## FFT over 2,500,000 random values: 0.28 (sec).
## Inverse of a 1,600 x 1,600 random matrix: 2.11 (sec).
```

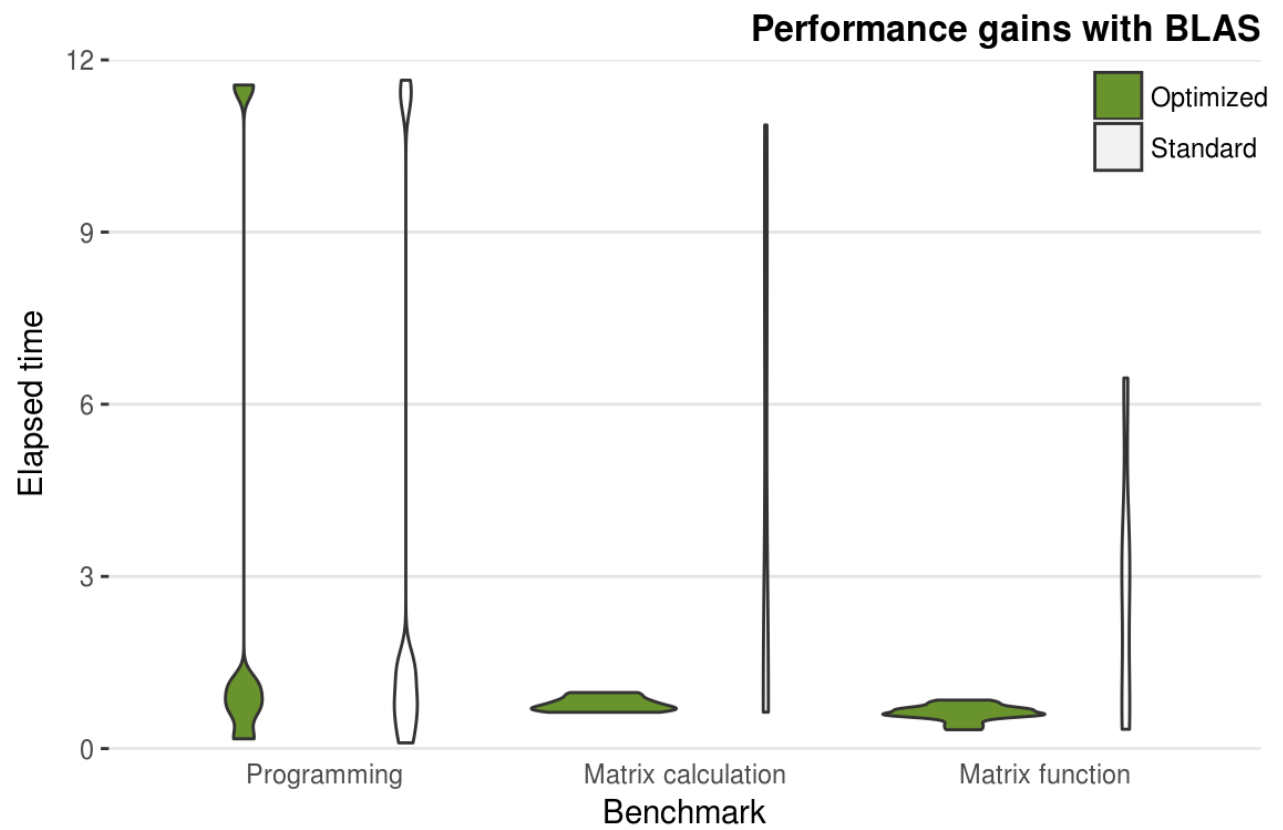


Figure 3: Ganancias de rendimiento obtenidas al cambiar la biblioteca BLAS subyacente.