Report on fast update for the (1+1)-Active-CMA-ES

Tamames Louis

December 2021

Faster (1+1)-Active-CMA-ES

Using the article [2] I add faster updates inside the (1+1)-Active-CMA-ES by updating the inverse of the covariance matrix at each iteration instead of calculating the inverse directly. This new update of the inverse of the covariance matrix is permitted by Sherman-Morrison formula and the Woodbury matrix identity. In total there are three different updates to add to the original algorithm, thanks to those modifications the complexity of the two first updates will be $\mathcal{O}(n^2)$ per iteration instead of $\mathcal{O}(n^3)$ when we inverse the matrix, and the last one will be $\mathcal{O}(m_a^3 + m_a \times n^2)$ instead of $\mathcal{O}(n^3 + m_a \times n^2)$, with n the dimension of the problem, m the number of constraints and m_a the number of violated constraints. This means that we will archive better performance when $n \gg m_a$ and similar complexity when the dimension and the active constraints have the same magnitude. I use the algorithm presented in the article [1], don't hesitate to read it for more information about the original updates.

First, the covariance matrix update of a successful candidate become :

$$\begin{aligned}
\boldsymbol{C}_{(t+1)} &= (1 - c_{cov}^{+})\boldsymbol{C}_{(t)} + c_{cov}^{+}\boldsymbol{s}\boldsymbol{s}^T \\
\boldsymbol{A}_{(t+1)} &= \sqrt{1 - c_{cov}^{+}}\boldsymbol{A}_{(t)} + \frac{\sqrt{1 - c_{cov}^{+}}}{||\boldsymbol{w}||^2}\Big(\sqrt{1 + \frac{c_{cov}^{+}||\boldsymbol{w}||^2}{1 - c_{cov}^{+}}} - 1\Big)\boldsymbol{s}\boldsymbol{w}^T \\
\boldsymbol{A}_{(t+1)}^{-1} &= \frac{1}{\sqrt{\alpha}}\boldsymbol{A}_{(t)}^{-1} - \frac{1}{\sqrt{\alpha}\,\|\boldsymbol{w}\|^2}\Big(1 - \frac{1}{\sqrt{1 + \frac{\beta}{\alpha}\,\|\boldsymbol{w}\|^2}}\Big)\boldsymbol{w}(\boldsymbol{w}^T\boldsymbol{A}_{(t)}^{-1})
\end{aligned} \tag{1}$$

Second, the covariance matrix update of the fifth order ancestor is now :

$$\begin{aligned}
\boldsymbol{C}_{(t+1)} &= (1 + c_{cov}^{-})\boldsymbol{C}_{(t)} - c_{cov}^{-}(\boldsymbol{A}_{(t)}\boldsymbol{z})(\boldsymbol{A}_{(t)}\boldsymbol{z})^T \\
\boldsymbol{A}_{(t+1)} &= \sqrt{1 + c_{cov}^{-}}\boldsymbol{A}_{(t)} + \frac{\sqrt{1 + c_{cov}^{-}}}{||\boldsymbol{z}||^2}\Big(\sqrt{1 - \frac{c_{cov}^{-}||\boldsymbol{z}||^2}{1 + c_{cov}^{-}}} - 1\Big)\boldsymbol{A}_{(t)}\boldsymbol{z}\boldsymbol{z}^T \\
\boldsymbol{A}_{(t+1)}^{-1} &= \frac{1}{\sqrt{\alpha}}\boldsymbol{A}_{(t)}^{-1} - \frac{1}{\sqrt{\alpha}\,\|\boldsymbol{z}\|^2}\Big(1 - \frac{1}{\sqrt{1 - \frac{\beta}{\alpha}\,\|\boldsymbol{z}\|^2}}\Big)\boldsymbol{z}(\boldsymbol{z}^T\boldsymbol{A}_{(t)}^{-1})
\end{aligned} \tag{2}$$

The proof of those two updates are presented in [2], using the Cholesky factorisation and Sherman-Morrison formula. We can use them because we have $\boldsymbol{A}_{(0)}$ invertible so all $\boldsymbol{A}_{(t)}$ are too, $\forall t \in \mathbb{N}$.
The statement of the Sherman Morrison formula is :
Let $A \in \mathbb{R}^{n \times n}$ be an invertible square matrix and $u, v \in \mathbb{R}^n$ two column vectors, then $A + \boldsymbol{u}\boldsymbol{v}^T$ is invertible if and only if $1 + \boldsymbol{v}^T\boldsymbol{A}^{-1}\boldsymbol{u} \neq 0$, and we have:

$$(\boldsymbol{A} + \boldsymbol{u}\boldsymbol{v}^T)^{-1} = \boldsymbol{A}^{-1} - \frac{(\boldsymbol{A}^{-1}\boldsymbol{u})(\boldsymbol{v}^T\boldsymbol{A}^{-1})}{1 + \boldsymbol{v}^T\boldsymbol{A}^{-1}\boldsymbol{u}}$$

Computationally speaking it is important to follow the calculation order given by the brackets, if not the algorithm will not reach the desired complexity because of extra cost in matrix multiplication.

Remark 1 *If you read [2] for more details, be careful the notation are slightly different, for example they use* $\boldsymbol{v}$ *instead of* $\boldsymbol{z}$*. I stick with the newer notation of [1].*

The last update to modify is the covariance update when the candidate violate at least a constraint, it have not been consider in any article until now. I use the Woodbury matrix identity, given by:

$$(A + VCW)^{-1} = A^{-1} - A^{-1}V(C^{-1} + WA^{-1}V)^{-1}WA^{-1} \tag{3}$$

We start by defining an exponentially fading record, $\boldsymbol{v}_j$ for $j \in \mathcal{I}_t$ where $\mathcal{I}_t$ is the set of violated constraints at step (t):

$$\boldsymbol{v}_j \leftarrow (1 - c_c)\boldsymbol{v}_j + c_c\boldsymbol{A}\boldsymbol{z}$$

Then, we have from [1] the active update given by:

$$\boldsymbol{A}_{(t+1)} = \boldsymbol{A}_{(t)} - \frac{\beta}{m_a(\mathbf{y})} \sum_{j=1}^{m} \mathbf{1}_{g_j(\mathbf{y})>0} \frac{\boldsymbol{v}_j \boldsymbol{w}_j^{\mathrm{T}}}{\|\boldsymbol{w}_j\|^2} \tag{4}$$

Where $\boldsymbol{w}_j = \boldsymbol{A}^{-1}\boldsymbol{v}_j$.
In order to use the Woodbury matrix identity we set:

- $m_a(\mathbf{y}) = \sum_{j=1}^{m} \mathbf{1}_{g_j(\mathbf{y})>0}$ the number of violated constraints at $\mathbf{y}$
- $C = \mathbb{I}_{m_a}$ the identity matrix of shape $m_a \times m_a$
- $V = -\frac{\beta}{m_a}\Big(\boldsymbol{v}_j\Big)_{j \in \mathcal{I}_t}$ a matrix in $\mathbb{R}^{n \times m_a}$
- $W = \Big(\frac{\boldsymbol{w}_j^{\mathrm{T}}}{\|\boldsymbol{w}_j\|^2}\Big)_{j \in \mathcal{I}_t}$ a matrix in $\mathbb{R}^{m_a \times n}$

So using equation 4 and the Woodbury matrix identity 3, we have:

$$\begin{aligned}
\boldsymbol{A}_{(t+1)}^{-1} &= (\boldsymbol{A}_{(t)} - \frac{\beta}{m_a(\mathbf{y})} \sum_{j=1}^{m} \mathbf{1}_{g_j(\mathbf{y})>0} \frac{\boldsymbol{v}_j \boldsymbol{w}_j^{\mathrm{T}}}{\|\boldsymbol{w}_j\|^2})^{-1} \\
&= (\boldsymbol{A}_{(t)} + VCW)^{-1} \\
&= \boldsymbol{A}_{(t)}^{-1} - \boldsymbol{A}_{(t)}^{-1} V (C^{-1} + W\boldsymbol{A}_{(t)}^{-1} V)^{-1} W \boldsymbol{A}_{(t)}^{-1}
\end{aligned} \tag{5}$$

We have the inverse of $\boldsymbol{A}_{(t)}$ since we calculate it at each iteration and $\boldsymbol{A}_0 = \mathbb{I}_n$. While coding those updates you should be careful to do the matrix products in the good order to archive the desired cost of $\mathcal{O}(m_a^3 + m_a \times n^2)$. The figure 1 show the performance of the modificated algorithm against the classic implementation.

No implementation of the faster update of the (1+1)-Active-CMA-ES with constraint handling existed until now, I code and post it on my GitHub, `https://github.com/LsTam91/elitist_constrained_es`. Where you can also find the implementation in python of the (1+1)-Active-CMA-ES and some example of numerical performances on test problems.

References

[1] Dirk V. Arnold and Nikolaus Hansen. A (1+1)-CMA-ES for Constrained Optimisation. pages 297–304, July 2012.

[2] Thorsten Suttorp, Nikolaus Hansen, and Christian Igel. Efficient Covariance Matrix Update for Variable Metric Evolution Strategies. *Machine Learning*, 2009.

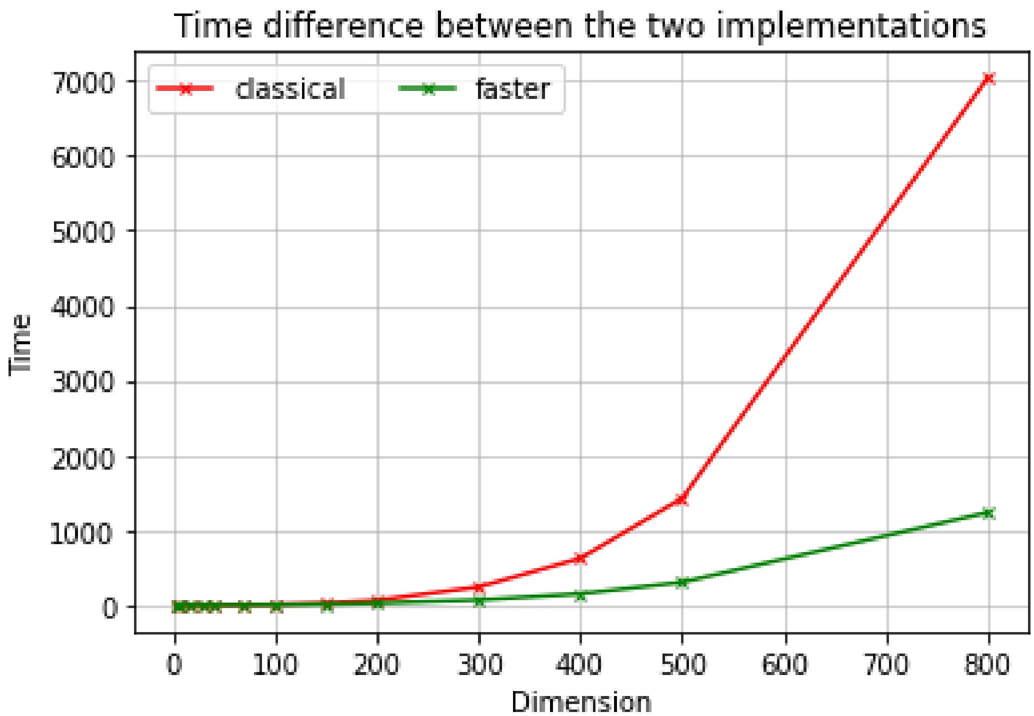

Figure 1: Performance of the classic implementation presented in the (1+1)-Active-CMA-ES paper, in comparison with my faster implementation updating the inverse of the covariance matrix. I run both algorithm in the sphere problem with one constraints, taking the time cost according to the dimension. The optimisation stops when the algorithm reach a precision of 10^{-6} to the optimum.