Non-convex optimisation: CMA-ES with gradient

A combination of evaluation strategy and gradient descent

semester project

Author:
Huajian Qiu

Supervisor:
Benoit Guillard
Pascal Fua

Computer Vision Laboratory (CVLAB)
École polytechnique fédérale de Lausanne (EPFL)

Submitted on July 26, 2020
EPFL, Lausanne

Abstract

In this project, a series of new optimisers combining one of the most successful evolutionary strategy (CMA-ES) with another gradient descent family optimiser is proposed. These optimisers share the same framework that a gradient-based local optimiser is injected into CMA-ES. This combination is proven to be effective in three out-of-shelf benchmark functions and one modified harder function in low dimensional case. Its scalability to high dimension(less than 500) is checked in Ackley benchmark function case. Enough experiments show the new optimisers get much better performance than the original optimisers in both cases, but require more time and function evaluations. Therefore, this project provides a framework/direction to construct better alternative optimiser to find global minimas for differentiable machine learning optimisation problems less than 500 dimensions.

Contents

Chapter 1

Introduction

Non-convex optimisation problems are almost everywhere in many science and engineering domain, especially with the widespread application of neural network in computer vision community. The most recent examples include finding the latent code representation in Geometric Deep Learning [11, 12].

These examples share the same properties that the objective function is non-convex, differentiable and often the dimension of domain is not too high. Therefore, give these prior knowledge, proposing a new specific optimiser exploiting these properties is possible, rather than directly using the more general out-of-shelf optimiser like Adam[9], Nesterov accelerated gradient[5], RMSprop[1], to name a few among others.

Compared to gradient based method, stochastic search based evolutionary strategies(ESs) have more potential to explore the wider domain, regarded as global optimisation method. Among them, CMA-ES is one of the most successful ESs, with most solid mathematical foundation and parallelable in nature. Recently machine learning scientists begin to use CMA-ES for tuning hyperparamters[10], reinforcement learning[13]. There are also some efforts to reduce time complexity of CMA-ES [8] for larger scale problem. But its unawareness of differentiability is its limit for the interested problem in this project.

In order to fully leverage the information of gradient, I propose a framework that injects a local optimiser into the global optimiser. Local optimiser can be line search and Adam, or other gradient descent methods but I haven't checked. The global optimiser is CMA-ES. The local optimiser serves as moving each candidate to a nearest local minima, whereas CMA-ES serves as generating next generation of candidates from local-optimised candidates in a larger scale in domain.

In general, the more common properties we know about the objective function, the better we can improve the optimiser. Some papers point out some insights about there are some common properties for high dimensional non-convex optimisation problem, such as the possible exponentially increasing saddle points[4, 7], rare appearance of local minimas[4, 6]. In this paper, I haven't pay much attention on it, but it's certainly an important direction for future work.

The chapter 2 details the algorithm of new optimiser and tested benchmark functions. The chapter 3 reports the experiments in 2D case and high dimensional case. The final chapter is conclusion.

Chapter 2

METHODS

In general, various techniques for general non-convex problems always involve compromises. Local optimisation methods, like gradient descent, provide no information about distance to global optimum. Global optimisation method has worst-case complexity growing exponentially with problem size. To compare different optimisers, four objective functions are proposed as benchmark functions.

2.1 OPTIMISER

2.1.1 CMA-ES

CMA-ES stands for covariance matrix adaptation evolution strategy. In an evolution strategy, new candidate solutions are sampled according to a multivariate normal distribution in $\mathbb{R}^n$. Recombination amounts to selecting a new mean value for the distribution. Mutation amounts to adding a random vector, a perturbation with zero mean. Pairwise dependencies between the variables in the distribution are represented by a covariance matrix. The covariance matrix adaptation (CMA) is a method to update the covariance matrix of this distribution. This is particularly useful if the function f is ill-conditioned.

It is used as the outer optimiser to be combined into my proposed method and it is also one baseline for my experiments. The algorithm detail of CMA-ES is covered in 3rd part of this chapter.

2.1.2 GRADIENT DESCENT

Gradient descent is a first-order iterative optimization algorithm for finding a local minimum of a differentiable function. Two typical methods are used to be combined with CMA-ES and as counterparts for comparing with my proposed optimiser: line search and adam.

Adam is one of the variant of SGD with the help of momentum and adaptive learning rate. There are many literatures[9] about it so I leave it out in my report.

I used inexact, backtracking-Armijo line search as below in this project. The stopping criteria can be maximum number of outer while loop iteration and/or minimum magnitude of gradient norm.

When the algorithm stops, the output can be either local minima, saddle point(on my chosen benchmark functions, this is rare case) or random candidate without convergence within maximum iteration number.

Algorithm 1 Backtracking-Armijo Line Search

Require: $\alpha_{init} > 0, \beta \in (0,1), \quad \vec{x}_{init}, \quad$ objective function $f()$, derivative $\nabla f()$
1: $i \leftarrow 0, \alpha \leftarrow \alpha_{init}, \quad \vec{x}^{(0)} \leftarrow \vec{x}_{init}, \vec{g}^{(0)} \leftarrow \nabla f(\vec{x}^{(0)})$
2: **while** stopping criteria not met **do**
3: $\quad i = i + 1$
4: $\quad$ **while** $f(\vec{x}_{(i)} + \alpha\vec{g}^{(i)}) > f(\vec{x}^{(i)}) + \beta\alpha\vec{g}^{(i)} \cdot (-\vec{g}^{(i)})$ **do**
5: $\quad\quad \alpha \leftarrow \alpha/2$
6: $\quad$ **end while**
7: $\quad \vec{x}^{(i+1)} \leftarrow \vec{x}^{(i)} + \alpha\vec{g}^{(i)}$
8: $\quad \vec{g}^{(i+1)} \leftarrow \nabla f(\vec{x}^{(i+1)})$
9: **end while**
10: Output $\vec{x}^{(i)}$

2.1.3 MY PROPOSED OPTIMISER

The new group of optimiser follows a unified framework that a gradient-based local optimiser is injected as an inner optimiser into CMA-ES, a more global optimiser.

The framework works as following: in each iteration of CMA-ES, each candidate solution sampled from multivariate normal distribution are provided as initial solution for gradient method to optimise; therefore, the original CMA-ES will be equivalent to a composite CMA-ES with $do\_nothing(f, \nabla f, x_{init})$ inner optimiser.

In this project, line search and Adam are chosen as inner optimiser. In addition, it is worth to point out that inner optimiser can even be another CMA-ES with smaller covariance matrix.

Refer to wiki[3] for the detail of algorithm from line 10th to line 14th.

2.2 BENCHMARK OBJECTIVE FUNCTION

Four objective functions are chosen as benchmarks. Three of them are referred from website[2]. One is modified by myself for testing assumption. The derivative of function is calculated by myself and rearranged in a compact way below. Also of them has following common properties:

- They have derivative function in closed-form which require comparable complexity to evaluate as original function.
- They have one known global minimum and many local minimums.
- The minimum is 0, i.e. function value is non-negative.

Algorithm 2 CMA-ES with inner gradient descent optimiser

Require: $\vec{\mu}_{init}, \sigma_{init} > 0$, objective function $f()$, derivative $\nabla f()$

$i \leftarrow 0, \vec{\mu}^{(0)} \leftarrow \vec{\mu}_{init}, \sigma^{(0)} \leftarrow \sigma_{init}, \quad C^{(0)} \leftarrow I, p_\sigma^{(0)} \leftarrow 0, p_c^{(0)} \leftarrow 0$
while stopping criteria not met **do**
  $i \leftarrow i + 1$
  for $j = 1, 2, ..., \lambda$ **do**
    $\vec{x}_{j(before)}^{(i)} \leftarrow$ sample_multivariate_normal(mean=$\vec{\mu}^{(i)}$, covariance_matrix=$\sigma^{(i)} C^{(i)}$)
    $\vec{x}_j^{(i)} \leftarrow$ inner_optimiser($f()$, $\nabla f()$, $\vec{x}_{j(before)}^{(i)}$)
    $y_j^{(i)} \leftarrow f(\vec{x}_j^{(i)})$
  end for
  $\vec{x}_{1,...,\lambda}^{(i)} \leftarrow \vec{x}_{s(1),...,s(\lambda)}^{(i)}$ with $s(k) \leftarrow argsort(y_{1,..,\lambda}^{(i)}, k)$ (sort candidates)
  $\vec{\mu}^{(i+1)} \leftarrow$ update_mean($\vec{x}_1^{(i)}, ..., \vec{x}_\lambda^{(i)}$) (move mean to better candidates)
  $p_\sigma^{(i+1)} \leftarrow$ update_ps($p_\sigma^{(i)}, \frac{\vec{\mu}^{(i+1)} - \vec{\mu}^{(i)}}{\sigma^{(i)} \sqrt{C^{(i)}}}$) (update isotropic evolution path)
  $p_c^{(i+1)} \leftarrow$ update_pc($p_c^{(i)}, \frac{\vec{\mu}^{(i+1)} - \vec{\mu}^{(i)}}{\sigma^{(i)}}, p_\sigma^{(i+1)}$) (update anisotropic evolution path)
  $C^{(i+1)} \leftarrow$ update_C($C^{(i)}, p_c^{(i+1)}, \frac{\vec{x}_1^{(i)} - \vec{\mu}^{(i)}}{\sigma^{(i)}}, \frac{\vec{x}_\lambda^{(i)} - \vec{\mu}^{(i)}}{\sigma^{(i)}}$) (update covariance matrix)
  $\sigma^{(i+1)} \leftarrow$ update_sigma($\sigma^{(i)}, \|p_\sigma^{(i+1)}\|$) (update step-size using isotropic path length)
end while
Output $\vec{x}_1^{(i)}$ or $\vec{\mu}^{(i)}$

For each benchmark function, I report its function, derivative function, global minimum and its position.

The equation 2.2.1 and the figure 2.2.1 are Ackley function, which is well defined from one to any high dimension. Local minimum are distributed at point with integer attributes.

$$
\begin{gathered}
f(x) = -20 \exp\left(-\frac{1}{5}\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right) - \exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos\left(2\pi x_i\right)\right) + 20 + e, \\
f'(x) = \left(..., g\left(x_j\right), ...\right)^d, \\
g(x_j) = 4\frac{x_j}{d\sqrt{\frac{1}{2}\sum_{i=1}^{2} x_i^2}} \exp\left(-\frac{1}{5}\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right) + \frac{2\pi}{d}\sin(2\pi x_j)\exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos\left(2\pi x_i\right)\right), \\
f^*(x^*) = 0, x^* = (0)^d
\end{gathered}
\tag{2.2.1}
$$

The equation 2.2.2 and the figure 2.2.2 are Bukin function, only defined in two dimension. It has continuous local minimum along a ridge. The global minimum is also located on the ridge but slightly smaller than the other local minimums.

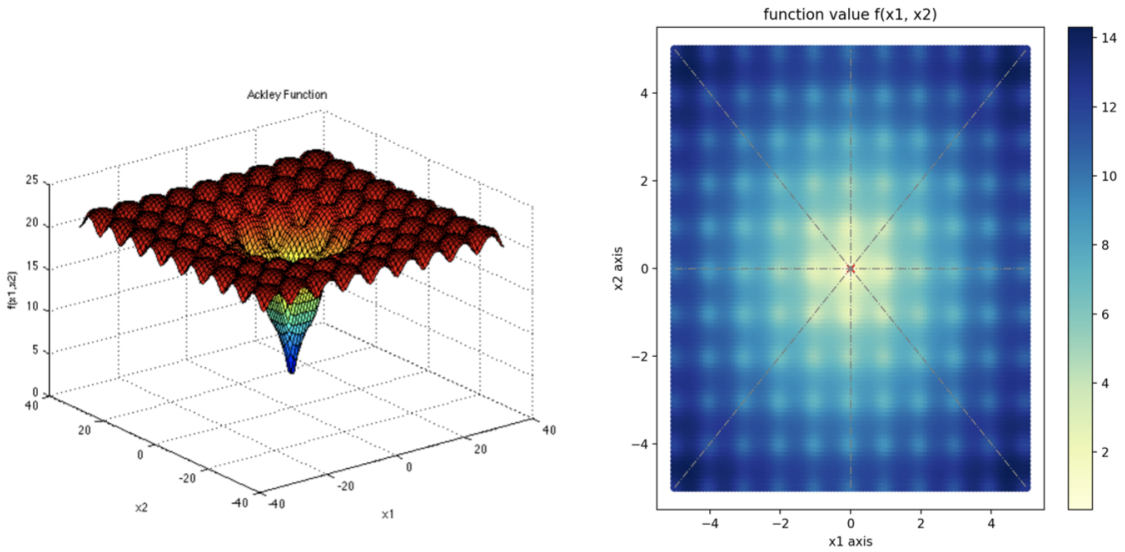

Figure 2.2.1: 3D visualisation and 2D function value contour of 2D Ackely function

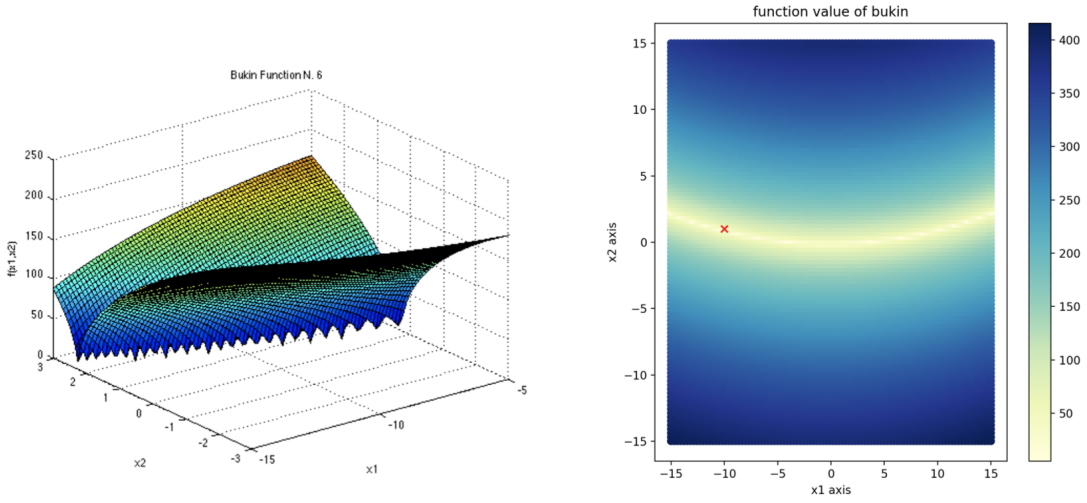

Figure 2.2.2: 3D visualisation and 2D function value contour of Bukin function

$$f(x) = 100\sqrt{|x_2 - 0.01x_1^2|} + 0.01\,|x_1 + 10|$$

$$f'(x) = \left(-x_1 \frac{1}{\sqrt{|x_2 - 0.01x_1^2|}} sgn(x_2 - 0.01x_1^2) + 0.01 sgn(x_1 + 10),\ \ \frac{50}{\sqrt{|x_2 - 0.01x_1^2|}} sgn(x_2 - 0.01x_1^2) \right)$$

$$f^*(x^*) = 0, x^* = (-10, 1) \tag{2.2.2}$$

The equation 2.2.3 and the figure 2.2.3 are Eggholder function, only defined in 2D case. Its function value fluctuate violently and even infinitely when far away from origin. To obtain one global minimum, I define the restricted domain as $[-550, 550] \times [-550, 500]$, function values outside domain are a constant bigger than any function value inside domain.

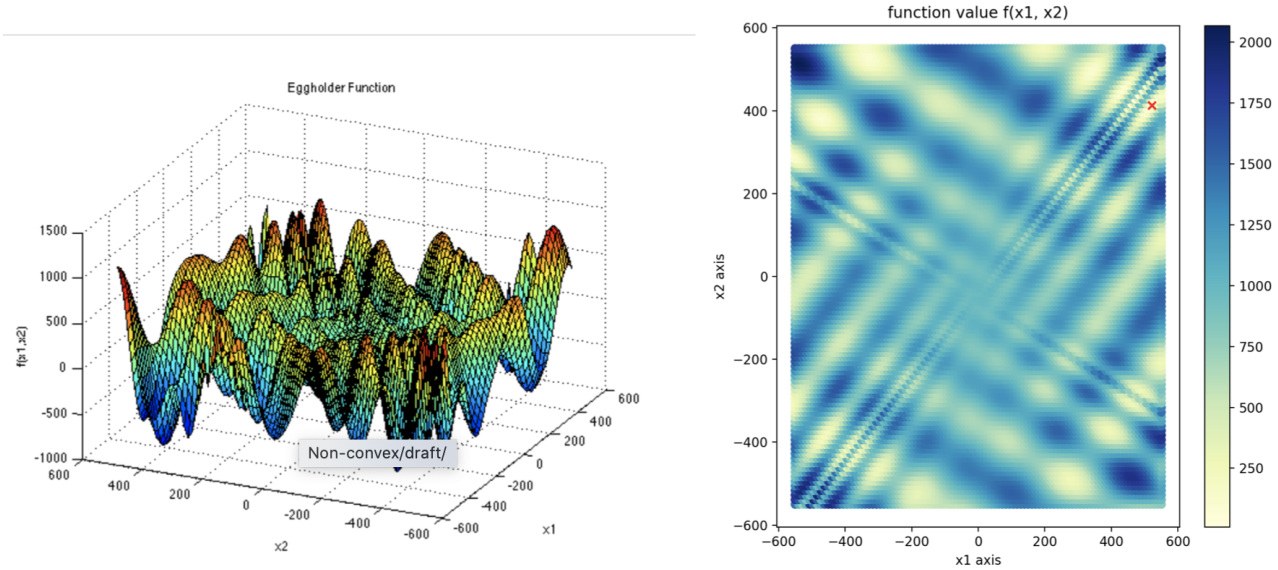

Figure 2.2.3: 3D visualisation and 2D function value contour of Eggholder function

$$
\begin{gathered}
f(x) = -(x_2+47)\sin\left(\sqrt{\left|\frac{x_1}{2}+(x_2+47)\right|}\right) - x_1\sin\left(\sqrt{|x_1-(x_2+47)|}\right) + 959.6407, \\
|x_1|, |x_2| \leq 512 \\
f'(x) = \left(-\sin\left(\sqrt{|x_2'|}\right) - \frac{1}{2}(x_2+47)g(x_1') - x_1 g(x_2'),\ \ -\sin\left(\sqrt{|x_1'|}\right) - (x_2+47)g(x_1') + x_1 g(x_2')\right) \\
g(x_j') = \cos\left(\sqrt{|x_j'|}\right)\frac{1}{2\sqrt{|x_j'|}} sgn(x_j') \\
x_1' = \frac{x_1}{2} + (x_2+47), \\
x_2' = x_1 - (x_2+47) \\
f^*(x^*) = 0, x^* = (512, 404.2319)
\end{gathered}
\tag{2.2.3}
$$

The equation 2.2.4 and the figure 2.2.4 are tuned Ackley function, a custom function tuned for checking my assumption. We firstly define a property that is enjoyed by original Ackley function and Bukin function:

Definition 2.2.1. weak convex relationship If we connect the local minimum by interpolation, the function become convex, then we call this as weak convex relationship.

With tunned Ackley function, I destruct this property for checking if the high performance of new optimiser rely on this. The tuned Ackley function is defined in domain 2d case with domain $[-22, 22] \times [-22, 22]$. The function value outside domain is a constant larger than any value inside domain.

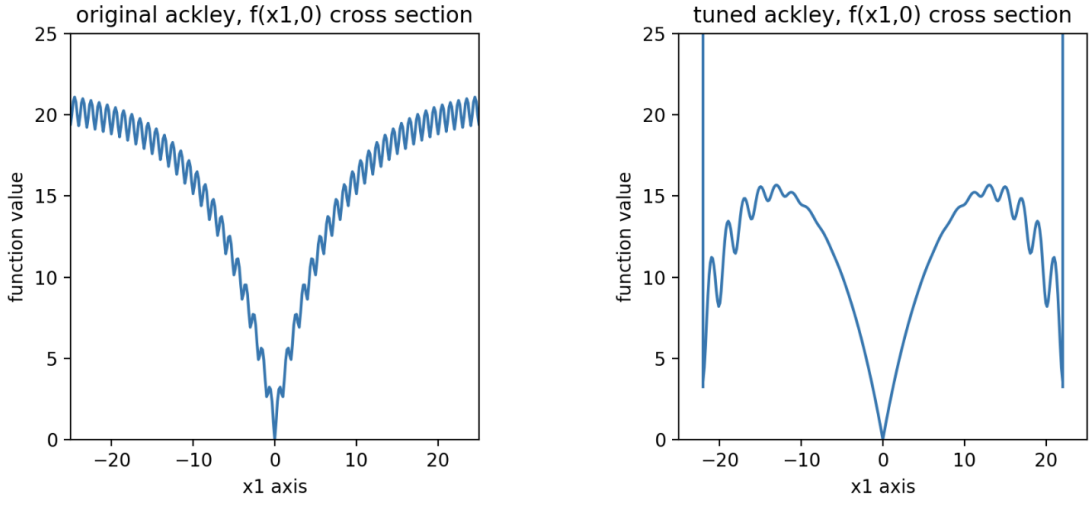

Figure 2.2.4: cross section of 2D Orginal Ackley and Tuned Ackley function

$$
\begin{aligned}
f(x) = -20\exp\left(-\frac{1}{5}\sqrt{\frac{1}{d}\sum_{i=1}^{d}x_i^2}\right) - \frac{1}{10}\left(-\frac{1}{5}\sqrt{\frac{1}{d}\sum_{i=1}^{d}x_i^2}\right)^4 \exp\left(\frac{1}{2d}\sum_{i=1}^{d}\cos\left(\pi x_i\right)\right) + 20 \\
f'(x) = \left(g\left(x_1\right),\ \ g\left(x_2\right)\right), \\
g(x_j) = 4\frac{x_j}{d\sqrt{\frac{1}{d}\sum_{i=1}^{d}x_i^2}}\exp\left(-\frac{1}{5}\sqrt{\frac{1}{d}\sum_{i=1}^{d}x_i^2}\right) \\
+\frac{\pi}{20d}\left(-\frac{1}{5}\sqrt{\frac{1}{d}\sum_{i=1}^{d}x_i^2}\right)^4 \sin(\pi x_j)\exp\left(\frac{1}{2d}\sum_{i=1}^{d}\cos\left(\pi x_i\right)\right) \\
-\frac{4}{6250d^2}\exp\left(\frac{1}{2d}\sum_{i=1}^{d}\cos\left(\pi x_i\right)\right)\left(\sum_{i=1}^{d}x_i^2\right)x_j \\
f^*(x^*) = e, x^* = (0,0)
\end{aligned}
\tag{2.2.4}
$$

Chapter 3

EXPERIMENTS

Optimisers are compared from two aspects: performance and efficiency. Performance is measured by loss and probability to find global minimum. Loss is the function value evaluated on output solution of optimiser. For a single time of optimisation, if the loss is less than 0.1 or distance to global solution is less than 0,1, then this is considered as finding the global minimum. Efficiency is measured by wall time and number of function evaluation. Wall time is the complete running time of optimiser to output a solution after enough evaluations, given a initial guess of solution. Number of evaluations measures the total number of evaluations of function and derivative function during this process.

3.1 DETAILED LOW DIMENSIONAL CASE

In this section, I present the experiments that show the performance and efficiency of three optimisers if we sample initial solution randomly from the domain of four benchmark functions.

3.1.1 PERFORMANCE

The domain Ackley is symmetrical along $x_1 - axis$, $x_2 - axis$, line $x_2 = x_1$ and line $x_2 = -x_1$. This means we can just sample initial candidate from one eighth of domain, a triangle-like region, to test optimisers.

In the region $0 < x_1 < x_2 < 200$, I continue to divide it into many small squares with 10 by 10 size. In each small square, I sample 10 points according to uniform distribution and color this square with the averaged loss (i.e. function value after optimisation).

From figure[], it clearly show the local optimiser Adam performs worst, CMA-ES is better, whereas CMA-ES-line-search obtains nearly the minimum function value zero after optimisation.

For Bukin function, the sample zone is $[-15, 15] \times [-5, 5]$ with small squaure of 1 by 1 size.

These two benchmark function satisfy the property of weak convex relationship defined in 2.2.1. It explains in some extent the almost 100 percent of probability to find global minimum.

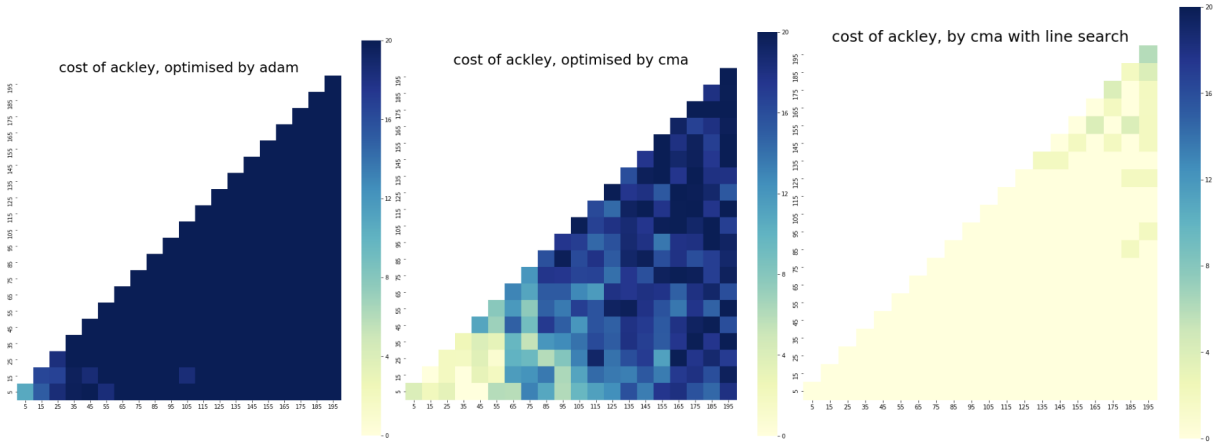

Figure 3.1.1: The function value(loss) of Ackley after optimisation by Adam, CMA, CMA-line-search. From left to right optimiser, the average loss is 20.03, 15.06, **0.24** and the average probability to find global minimum is 0.05%, 21.71%, **98.81%**.

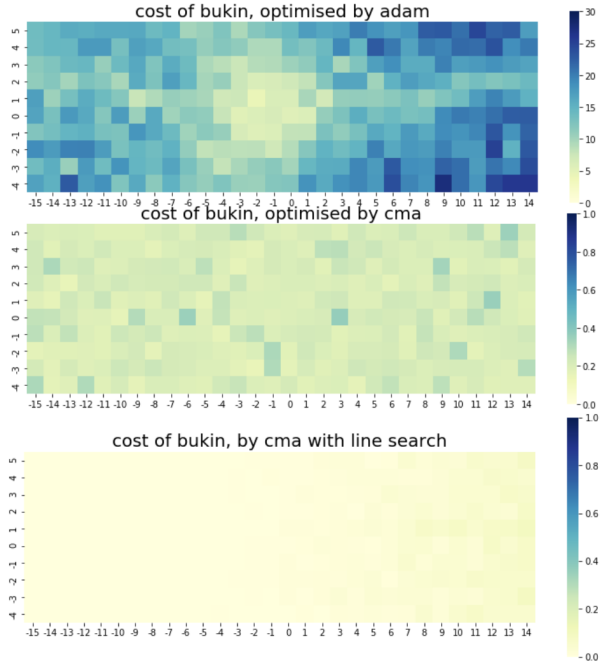

Figure 3.1.2: The function value(loss) of Bukin after optimisation by Adam, CMA, CMA-line-search. From top to bottom optimiser, the average loss is 14.64, 0.22, **0.017** and the average probability to find global minimum is 0.43%, 0.7%, **95.6%**.

The later two benchmark functions, Eggholder and tuned Ackley, does not have this proporty.

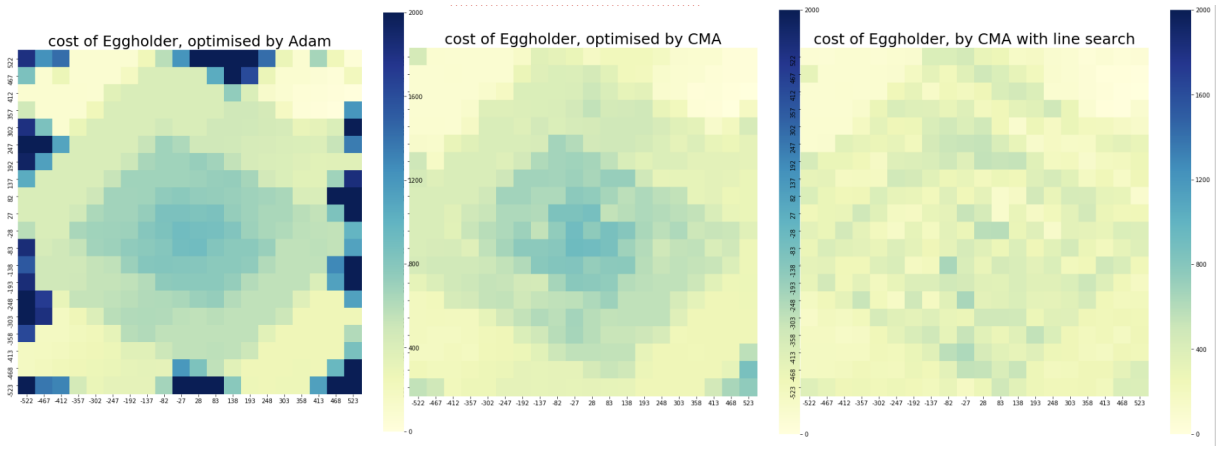

Figure 3.1.3: The function value(loss) of Eggholder after optimisation by Adam, CMA, CMA-line-search. From left to right optimiser, the average loss is 627, 394, **289** and the average probability to find global minimum is 0, 1.3%, **5.6%**.

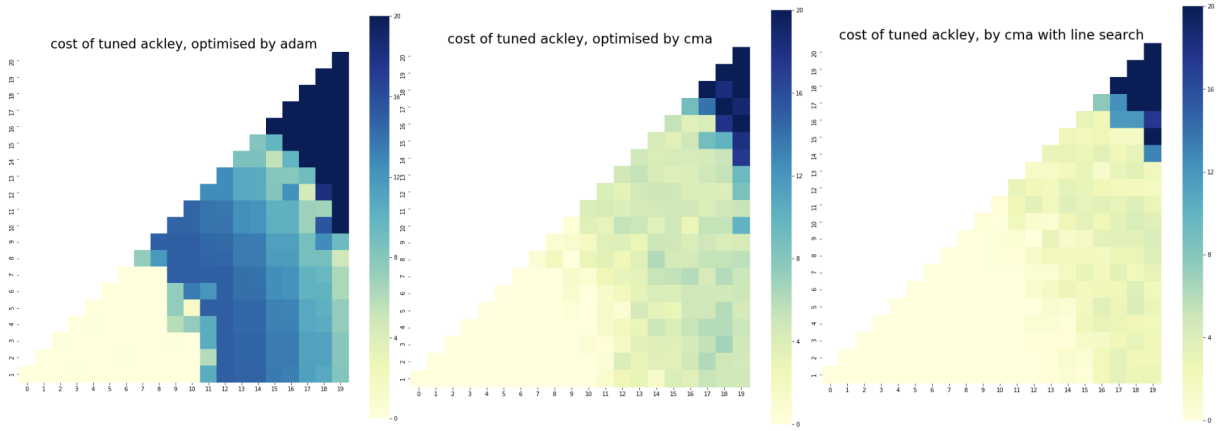

Figure 3.1.4: The function value(loss) of tuned Ackley after optimisation by Adam, CMA, CMA-line-search. From left to right optimiser, the average loss is 13.83, 4.49, **3.09** and the average probability to find global minimum is 25.1%, 43.3%, **62.5%**.

The performance of CMA-ES-line-search is not as good as on the former two benchmark functions, but still much better than the other two optimiser on a fixed benchmark function. It is worth to note that in case of tuned Ackley function, CMA-line search still outperform original CMA. This partly show the success of global-local optimiser framework does not totally rely on weak convex relationship.

3.1.2 EFFICIENCY

The figure 3.1.5 shows the time and number of evaluations recorded from same set of experiments as previous section.

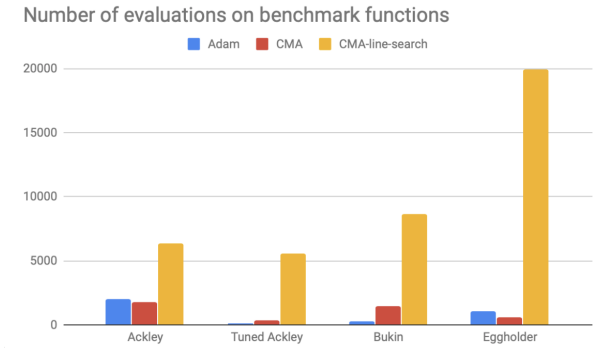

Figure 3.1.5: compare efficiency in terms of number of evaluations

objective	optimizer	prob to global minima	loss	# of evaluations	time
ackley	*do nothing*	*0*	*21.55*	*1*	*0.01s*
	adam	0.	19.98	**0.14k**	0.9s
	line search	0.	19.98	39.9k	**0.12s**
	cma	0.	19.82	1.46k	12.6s
	cma-line-search	**0.41**	**11.73**	70.9k	199s
	cma-adam	0.19	16.27	156.4k	818s
eggholder	*do nothing*	*0*	*940.7*	*1*	*0.019s*
	adam	0.	715	1.6k	15.6s
	line search	0.	464	**0.24k**	**1.3s**
	cma	0.	436	0.27k	4.9s
	cma-line-search	**0.136**	**341**	45k	280s
	cma-adam	0.012	428	0.28k	8.1s
tuned ackley	*do nothing*	*0*	*21.55*	*1*	*0.012s*
	adam	0.2.	15.1	**0.12k**	**0.8s**
	line search	0.27	13.3	0.31k	0.9s
	cma	0.47	4.0	0.46k	4.7s
	cma-line-search	**0.62**	3.6	26.1k	75s
	cma-adam	0.54	**2.9**	14.5k	128s
bukin	*do nothing*	*0*	*260*	*1*	*0.007s*
	adam	0.	11.36	7.2k	21.7s
	line search	0.46	0.12	**2.1k**	**11.8s**
	cma	0.	0.99	2.4k	12.7s
	cma-line-search	**0.97**	**4e-5**	266k	263s
	cma-adam	0.	0.89	22.4k	64s

Table 3.1.1: summary of performance and efficiency for one baseline: *do_nothing*() and five optimisers on four benchmark functions in 2D case

The CMA-line-search performs better than Adam and original CMA but need much more number of evaluations functions. The time per evaluation is about at magnitude of 10^{-6} for three optimisers on a machine with MacOS, 2.3 GHz Dual-Core Intel Core i5, without GPU.

The table 3.1.2 summaries both performance and efficiency in 2D case from another smaller scale set of experiments. There are clear patterns that composite CMA-ES always outperform in loss and probability to find global minimum, whereas gradient method always use less time and number of evaluations.

3.2 SCALABILITY WITH INCREASING DIMENSION

After inspecting the behaviours of different optimisers in 2D case, this set of experiments focus on the change of loss and time with respect to dimension.

The experiment setup is as following: five optimisers are tested on 10, 20, ..., 200 dimensional Ackley function; each time, an initial solution is uniformly sampled from $[0, 10]^d, d = 10, 20, ..., 200$.

The figure 3.2.1 shows loss keeps nearly zero whereas time increases exponentially with respect to dimension. This result can also be confirmed from other literature[8] that mentions the time complexity of gradient descent is $\mathcal{O}(n)$, whereas that of CMA-ES is $\mathcal{O}(n^2)$

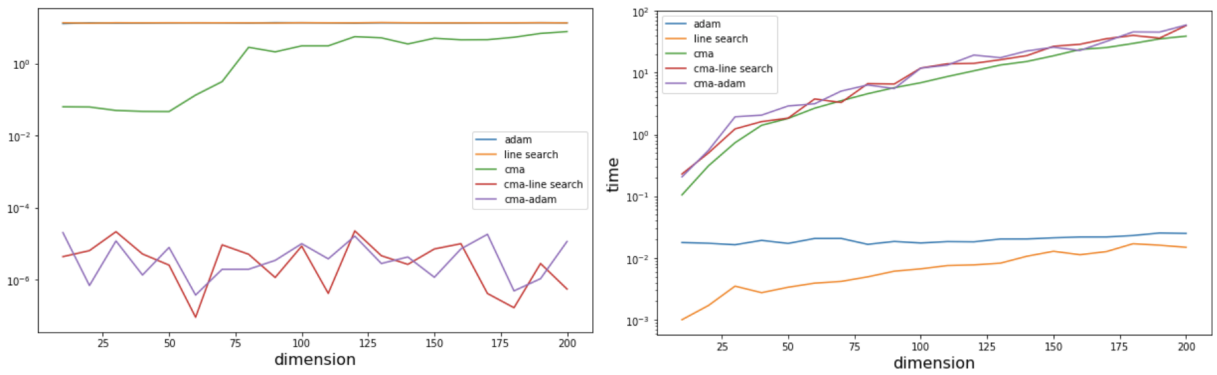

Figure 3.2.1: loss and time with increasing dimension, in log scale

Chapter 4

CONCLUSION

CMA-ES with line search/Adam has big advantages over Adam, original CMA-ES, line search in terms of : More convergence on low dimensional benchmark functions; Keep effective on high dimensional benchmark functions.

And the most suitable application of new composite optimiser is for objective function satisfying: Differentiable and efficient to compute gradient; Low dimensional parameter space($<$ 500 Dims); Multi-modal function with many local minimum.

There is still some unsolved limitations and therefore become the possible extension direction for this project. For example, how to mitigate curse of dimensionality? One possible solution is to use early stop of CMA-ES, and rely more on local gradient based optimiser at later period of optimisation. Another concern is that high dimensional problems of practical interest often has exponential number of saddle points. This kind of saddle point is more like large areas of plain area with little gradient rather than a single point. Traditional gradient method need long time of running to escape from this area(if possible), but what does this influence my proposed composite CMA optimiser is relative unknown.

Bibliography

[1] Yoshua Bengio and MONTREAL CA. "Rmsprop and equilibrated adaptive learning rates for nonconvex optimization". In: *corr abs/1502.04390* (2015).

[2] Derek Bingham. *Virtual library of simulation experiments : test function and dataset-Optimization Test Problems*. URL: `https://www.sfu.ca/~ssurjano/optimization.html`.

[3] Wikipedia community. *Covariance matrix adaptation evolution strategy*. URL: `https://en.wikipedia.org/wiki/CMA-ES`.

[4] Yann N Dauphin et al. "Identifying and attacking the saddle point problem in high-dimensional non-convex optimization". In: *Advances in neural information processing systems*. 2014, pp. 2933–2941.

[5] Timothy Dozat. "Incorporating nesterov momentum into adam". In: (2016).

[6] Rong Ge, Jason D Lee, and Tengyu Ma. "Matrix completion has no spurious local minimum". In: *Advances in Neural Information Processing Systems*. 2016, pp. 2973–2981.

[7] Rong Ge et al. "Escaping from saddle points—online stochastic gradient for tensor decomposition". In: *Conference on Learning Theory*. 2015, pp. 797–842.

[8] Nikolaus Hansen, Sibylle D Müller, and Petros Koumoutsakos. "Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES)". In: *Evolutionary computation* 11.1 (2003), pp. 1–18.

[9] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[10] Ilya Loshchilov and Frank Hutter. *CMA-ES for Hyperparameter Optimization of Deep Neural Networks*. 2016. arXiv: `1604.07269 [cs.NE]`.

[11] Jeong Joon Park et al. "Deepsdf: Learning continuous signed distance functions for shape representation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 165–174.

[12] Edoardo Remelli et al. *MeshSDF: Differentiable Iso-Surface Extraction*. 2020. arXiv: `2006.03997 [cs.CV]`.

[13] Tim Salimans et al. "Evolution strategies as a scalable alternative to reinforcement learning". In: *arXiv preprint arXiv:1703.03864* (2017).