Candidate Number: 237707
Module: 817G5

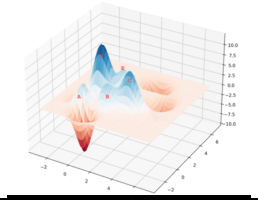
Comparison of Optimization Techniques: Gradient Ascent and Genetic Algorithm

1. Introduction

Spanning from real-world problems to highly complex mathematical functions and machine learning, optimization algorithms are techniques widely employed in minimization/maximization of sophisticated objective functions. Although gradient-based optimizers are widely regarded as the benchmark technique [1], it is without saying that genetic algorithms have manifested compelling results grasping of the interests of the research community [3]. As a starting point, this report aims to review key characteristics as well as various implementations for both the gradient ascent algorithm and genetic algorithm. Subsequently, further experimentation will be directed towards finding the best implementation of each algorithm against the objective function. Although one can anticipate trade-offs in response various approaches in which the optimizers are implemented, our main focus will be the computational performance as well as accuracy of the solution. Finally, we aim to establish how each algorithm behaves in the face of complexity and compare their overall computational performance as well as convergence.

2. Methods

2.1.1 Choice of Objective Functions

Complex Landscape Objective Function	
$z = 4(1-x)^{2} * e^{-x^2-(y+1)^2} - 15\left(\frac{x}{5-x^3-y^5}\right) * \left(e^{-x^2-y^2}\right) - \left(\frac{e^{-(x+1)^2-y^2}}{3}\right) - (2(x-3)^7 - 0.3(y-4)^5 + (y-3)^9) * (e^{-(x-3)^2-(y-3)^2})$	
	The complex landscape consists of 5 hills including 4 local maximums located on hills A, B, C and E. It also has a global maximum located on plane D where $\boldsymbol{x = 0}$ and $\boldsymbol{y = 1.585}$ with a maximum height ($\boldsymbol{z}$) of approximately ***12.235***. Both $\boldsymbol{x}$ and $\boldsymbol{y}$ preclude boundaries between -3 and 7.

2.2.1 Gradient Based Algorithm

Widely employed in optimizing neural networks, the gradient ascent algorithm is a mathematically guided optimizer that requires the computation of derivatives. Although limited to only reaching local maximum in non-convex problems [1], among numerous variants to this algorithm include the stochastic gradient ascent (SGA) algorithm. Instead of focusing on the entire dataset, SGA randomly selects samples from the dataset enabling the realization of the global maxima [3]. Our simulation will include the following procedures: *[1.]* Identify slope by computing the gradient of the objective function. *[2.]* Compute the partial derivative for $\boldsymbol{x}$ and $\boldsymbol{y}$ in the objective function plugging in a random value within the pre-defined boundaries. *[3.]* Calculate the step size: $\boldsymbol{step\ size\ =\ gradient\ *\ learning\ rate}$ *[4.]* Calculate new parameters: $\boldsymbol{new\ parameters\ =\ old\ parameters + step\ size}$. *[5.]* Repeat steps 2-4 for the number of epochs set.

Although the SGA is regarded to perform adequately in reaching global maxima, defining a suitable learning rate can play key role in improving the convergence. In essence, a large learning rate may lead to the overshooting of the global maxima [2]. In contrast, despite an increase in accuracy a small learning rate can heavily increase the computational cost. Respectively, studies suggest that a larger learning rate at initial stages followed by smaller learning rate can improve the quality of the solution [4]. Therefore, our experiment will firstly attempt to identify various strategies to reach the slope leading to global maxima using a larger learning rate. Finally, we will aim to reduce the learning rate in an according manner to increase the accuracy and efficiency of our solution.

2.2.2 Genetic Algorithm

Regarded as a meta-heuristic search algorithm founded upon theories of evolution and natural selection, the genetic algorithm (GA) follows a discrete and non-linear approach that is not mathematically dictated [5]. Our simulation will include the following procedures: [1] A random set

chromosome (***x*** and ***y*** in the objective function) are generated to initialize the population. [2] Lengths of population will be calculated expressing a measurement of performance in respect to reproductive capacities. [3] A proportion of the population will be selected based on the ranking of the performance. [4] Crossover occurs where random chromosomes are then combined to create new offspring. [5] New chromosomes are then multiplied within a randomly selected range representing mutation.

Interactions among parameters such as mutation rate, crossover rate and population size have a direct influence on the results and can be of vital importance to a successful search [5]. Respectively, GA's design can be limited to the approach taken with these parameters. Studies focusing on large population sizes suggested increased probabilities of containing an individual representing the optimal solution [6]. Other approaches found small population sizes to be effective with higher mutation rates. Although crossover rates was seen to be more effective with larger population sizes, it was also found that mutation and crossover yielded better results when used hand in hand as opposed to just one used exclusively. Therefore, our experiments include large population sizes (combined with small mutation) and smaller population sizes (combined with larger mutation). Finally, we aim to determine a threshold population size that is both computationally inexpensive and suitable enough to provide adequate search space to reach global maxima and will followed with further experimentations using both uniform and one-point crossovers.

3. Results and Discussions

3.1.1 Stochastic Gradient Ascent Algorithm (SGA)

Fig 1-

Learning Rates	1	0.5	0.1	0.075	0.04	0.03
Max Height	0.109	0.422	11.312	11.427	12.235	12.166

average results displayed after running SGA 50 times to minimize the randomness in the algorithm

The data above suggests that global maximum cannot be reached with large learning rates of 1 and 0.5 [fig 1]. However, when learning rate is decreased to 0.1 and 0.075 we get close to 90% of the global maximum making them suitable large learning rates to start and later decrease. On the other hand, global maximum is reached with learning rates of 0.04 and 0.03. Based in results in fig 1, large learning rates of 0.075 (green), 0.1 (blue) and 0.125 (orange) [fig 2] were selected in our modified SGA. It is noted that higher decrease of the learning rate leads to higher calculation time as well as lower accuracies in global maximum. This may be due to the fact that the starting point is randomly selected at the local minimum and learning rate becomes too small and can't make it to global maximum. Whilst all 3 learning rates appear to reach an optimal computational performance with multiplier of 0.85-0.95, learning rate of 0.125 (orange) has the best performance when multiplier is 0.975. Thus, it is able to run the modified SGA 1000 times in 9.53 seconds making it 5.9 times faster than the unmodified SGA which takes 56.55 seconds.

Fig 2-

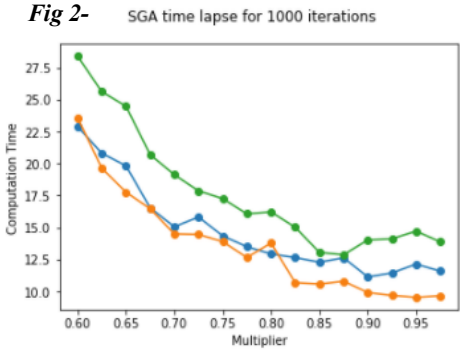


Fig 3-

3.1.2 Genetic Algorithm (GA)

Population Size	Crossover	Mutation	Height		Generations		Accuracy	
20	**0.9**	**0.5**	12.235	100%	18.6	8.9	100%	100%
30	**0.8**	**0.45**	12.235	100%	36.4	9.1	100%	100%
50	**0.8**	**0.4**	12.235	100%	87.4	51.8	100%	100%
80	**0.6**	**0.1**	11.13	100%	333.9	127.2	71%	90%
100	**0.5**	**0.1**	11.30	100%	335.95	134.69	69%	89%

average results displayed after running GA 100 times to minimize the randomness associated with the algorithm

Our experiment with small population sizes of 20, 30 and 50 were able to reach the global maximum with 100% accuracy over 100 iterations for both one-point (yellow) and uniform crossover (blue) [fig 3]. Although both GA's performed best with smaller populations, one-point crossover provided a better solution with smallest populations of 20 and 30. On the other hand, both GA algorithms saw a significant decline in the average performance with larger population sizes (80 and 100). However, from trials that were able to reach the global maximum with the highest population of 100 also completed the solution with an average of 3.8 generation (uniform crossover) and 2.8 (one-point crossover). This can relate our hypothesis that larger populations have higher chance of including the individuals with optimal chromosomes from the start. Nevertheless, the solution provided by a populations size of 20 was by far the most reliable and computationally efficient solution for both GA's. Finally, GA with uniform crossover yielded superior computational performance completing 1000 iterations in 149.9 seconds as oppose to the GA with one-point at 172.3 second (with populations size of 20).

3.2 Discussions and Concluding Remarks

Fig 4-

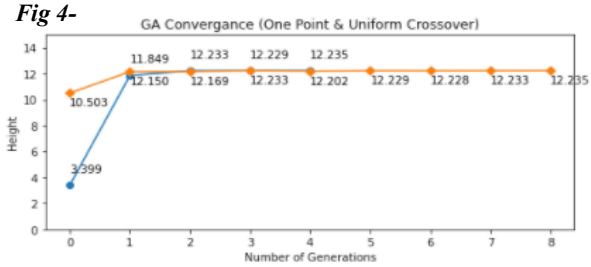
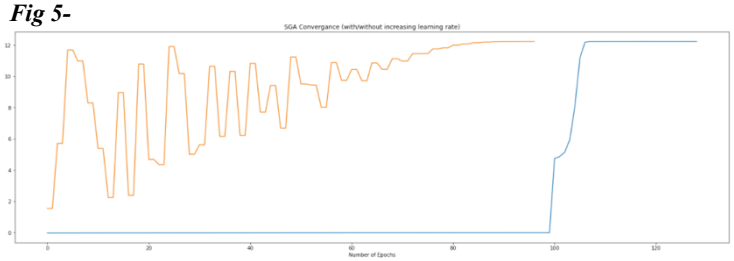


Fig 5-



Although convergence in SGA is improved when starting with a large learning rates that decreased (orange) [fig 5], GA was able to converge to close to global maxima only after 1 generation [fig 4]. Moreover, one-point crossover (blue) took smaller numbers of generations to reach global maximum than uniform crossover (orange) [fig 4]. Nevertheless, both GA algorithms were merely comparable to the computational performance achieved in our SGA solution completing 1000 iterations at only 8.8 seconds. Overall much of the hypothesis surrounding vital parameters GA and SGA were validated in this experiment and we were not able to achieve global maxima from our implementations but also improve convergence (orange plots for GA and SGA) [fig 4 & 5] using different techniques. Although, the quality of coding may have led to slight decrease in GA's performance computationally, SGA was approximately 17 times faster in reaching global maxima which in essence explains why it is widely considered as the benchmark technique for optimization problems.

Reference List (APA Style)

[1] Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

[2] Haji, S. H., & Abdulazeez, A. M. (2021). Comparison of Optimization Techniques Based on Gradient Descent Algorithm: A Review. *PalArch's Journal of Archaeology of Egypt/Egyptology*, *18*(4), 2715-2743.

[3] Kaoudi, Z., Quiané-Ruiz, J. A., Thirumuruganathan, S., Chawla, S., & Agrawal, D. (2017, May). A cost-based optimizer for gradient descent optimization. In *Proceedings of the 2017 ACM International Conference on Management of Data* (pp. 977-992).

[4] Zhao, H., Liu, F., Zhang, H., & Liang, Z. (2019). Research on a learning rate with energy index in deep learning. *Neural Networks*, *110*, 225-231.

[5] Hassanat, A., Almohammadi, K., Alkafaween, E., Abunawas, E., Hammouri, A., & Prasath, V. B. (2019). Choosing mutation and crossover ratios for genetic algorithms—a review with a new dynamic approach. *Information*, *10*(12), 390.

[6] Muhlenbein, Heinz, and Dirk Schlierkamp-Voosen. "Optimal interaction of mutation and crossover in the breeder genetic algorithm." *Proceedings of the Fifth International Conference on Genetic Algorithms*. 1993.

[7] Rylander, S. G. B., & Gotshall, B. (2002). Optimal population size and the genetic algorithm. *Population*, *100*(400), 900.