University of Sussex

Artificial intelligence and adaptive systems

Adaptive systems

Neural network hyper-parameter optimization using simulated annealing

Author:
Andres Salazar

Lecturer:
Dr. Chris Johnson

March 6, 2024

Contents

List of Figures

List of Tables

Abstract

Disease prevention has been the focus in health-related research for years from cardiovascular disease prevention to early cancer detection. The aim is to develop new technologies that can process big quantity of data, interpretating features and extracting and creating correlation between characteristics. Tasks that even a doctor with years of experience in the area cannot always do with the precision of a machine learning model. That is why is key to develop machine learning models as Multi-Layer Perceptron (MLP) with the highest precision possible. To reach the best possible performance is important to do a proper hyperparameters tuning, this can be done by heuristic techniques or based on related work. However, finding the best hyperparameters configuration requires time and computer resources, and the hyperparameters set performance can vary depending on the data and the preprocessing applied. Therefore, simulated annealing (SA) as an optimization algorithm is proposed. The purpose of the experiment is to measure the SA algorithm performance as hyperparameter optimization for a MLP classifier in terms of computer complexity and F1-Score evaluation metric.

1 Introduction

Neural networks have been used extensively in tasks such as prediction and classification. Therefore, these models are used with a wide range of data, from power amplifiers behavioral modeling [1] to customer churn prediction [2]. Given the high differences between data sets, is not an easy task to find the best hyperparameters for each problem, the characteristics, pre-processing, and number of features are just a couple of the vast differences that can exist between data sets. That is why [1] and [2] use genetic algorithms to optimize the neural network hyperparameters and explore the solution space as much depth as possible. Moreover, the use of neural networks to detect, predict and classify diseases is key for modern medicine. [3] implements a genetic algorithm to optimize a deep residual neural network for acute lymphoblastic leukemia classification using microscopy images conclude that the accuracy reached was higher than the one presented in the literature. On the other hand, [4] proposes hyperparameter optimization of an artificial neural network and its structure with a genetic algorithm to breast cancer diagnosis/prediction. The results evidence that uses of this optimization algorithms improve the model performance considerably when compared to literature previous research. Diabetes affects 17% of the UK population and approximately one million people have undiagnosed type 2 diabetes, 40.000 children have diabetes, and more than 3000 children are diagnosed every year [5]. This is not just a problem in the UK, is a global problem where by 2012 around 346 million people [6]. This reality and the related work inspired the experimentation carried out in this report. Considering the question: Could the optimization algorithm simulated annealing improve considerably the performance and computer complexity of neural network classifier for diabetes prediction/classification? Finding the best hyperparameters for a machine learning model is fundamental to reach a high prediction/classification performance. That in this case is key to reach the highest accuracy possible to detect and classify diabetes as precise as possible. Because of the variability in data sets a Multi-Layer-Perceptron using diabetes Kaggle dataset is proposed to develop, implement, and evaluate the chosen optimization algorithm performance.

2 Methods

2.1 Proposed methodology

Sci-kit learn multi-layer perceptron classifier is proposed as a viable solution to the classification problem given the extended use in classification tasks [7] [8]. The optimization algorithm chosen to find the best hyperparameters was simulated annealing. Additionally, the fitness function was F1-score. Moreover, the multi-layer perceptron hyperparameters evolved were solver, activation function, hidden layer sizes, alpha and leaning rate. Figure 1 shows the workflow of the experiment.

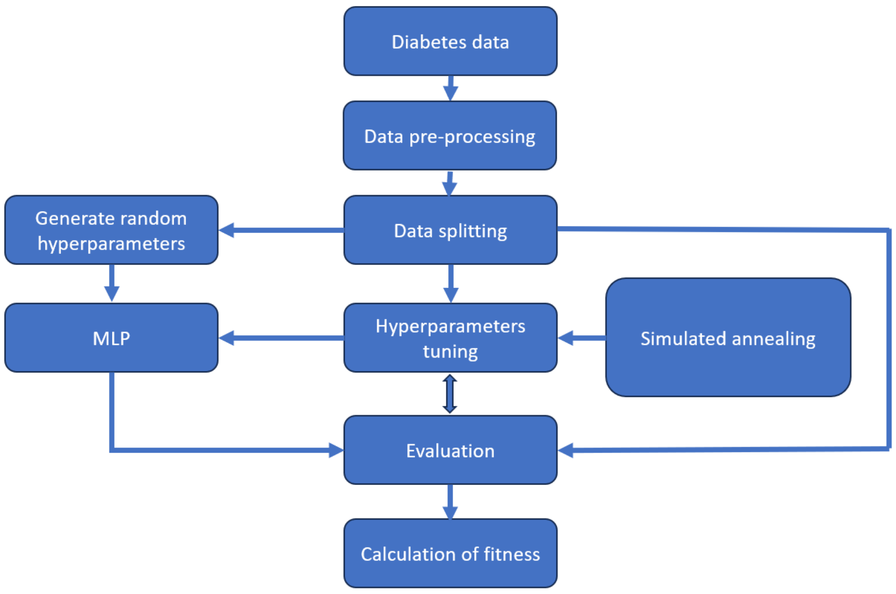


Figure 1: Experiment workflow.

Different values of cooling rate and maximum iterations were tested. A cooling rate of 0.99, 0.95 and 0.93 were evaluated. On the other hand, the maximum iterations tested were 1000, 500 and 300. Since in the first iterations the probability to accept a lower fitness is higher, and as what is wanted is to explore the solution space as deep as possible, testing the model with different generations and cooling rate may help to reach the highest fitness possible.

2.2 Diabetes data set

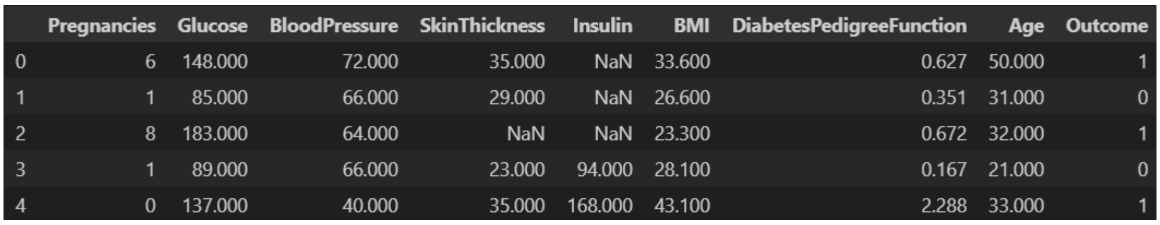
The diabetes data set used in the experiment is originally from the National Institute of Diabetes and Digestive and Kidney diseases. The main goal with this data is to predict if a patient has diabetes based on diagnostic measurements. The samples are from females at least 21 years old.

The features per patient are:

- Number of times pregnant
- Plasma glucose concentration

- Diastolic blood pressure
- Triceps skin fold thickness
- 2-Hour serum insulin
- Body mass index
- Diabetes pedigree function
- Age

The outcome is 0 for a positive diabetes diagnosis and 1 for a negative diagnosis. The data set is composed of 768 instances and extracted from Kaggle [9].



	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148.000	72.000	35.000	NaN	33.600	0.627	50.000	1
1	1	85.000	66.000	29.000	NaN	26.600	0.351	31.000	0
2	8	183.000	64.000	NaN	NaN	23.300	0.672	32.000	1
3	1	89.000	66.000	23.000	94.000	28.100	0.167	21.000	0
4	0	137.000	40.000	35.000	168.000	43.100	2.288	33.000	1

Figure 2: Diabetes data set example.

The data set does not present direct missing values as NaN. However, it has some values a of 0 in some numeric features that may indicate a missing value. Hence, the potential missing values were filled with the column's median. As the distribution is homogeneous, filling missing values with median or mean values is logical.

2.3 Simulated annealing

Simulated annealing was first introduced by Scott Kirkpatrick in 1982 and was extended and tested as an optimization algorithm in 1983 [10] based on the statistical mechanics of annealing in solids. [11] explain the process and provide an example: *"Consider how to coerce a solid into a low energy state. A low energy state usually means a highly ordered state, such as a crystal lattice; a relevant example here is the need to grow silicon in the form of highly ordered, defect-free crystal for use in semiconductor manufacturing. To accomplish this, the material is annealed: heated to a temperature that permits many atomic rearrangements, then cooled carefully, slowly, until the material freezes into a good crystal"*.

The annealing nature is described by the law of thermodynamics that states that at temperature t, the probability of an increase in energy of magnitude ΔE, is given by:

$$P(\Delta E) = exp(\frac{-\Delta E}{kt}) \tag{1}$$

Where k is a constant known as Boltzmann's constant.

The probability of changing to higher energy decreases with the size of the energy jump. What means that higher energy temperature makes larger energy changes more likely.

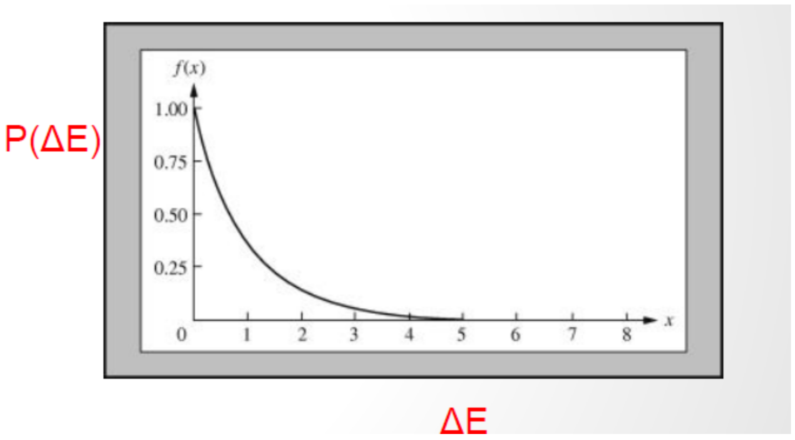


Figure 3: Simulated annealing graph [12].

Simulated annealing as optimization algorithm equals the energy with fitness what is interpreted as the degree to which particles jump around is controlled by the temperature, then this particles behavior works like a mutation rate [12].

The simulated annealing pseudo-code was implemented as below:

```
1. Mutate by a random amount, small changes are more likely.
    (a) Add a small extra probability to change the gene related to solver to 5%.
    (b) Add a small extra probability to change the gene related to activation to 5%.
2. Measure change in fitness given by $\Delta F = Fnew - Fold$.
    (a) If $\Delta F$ is greater than 0, the move is accepted.
    (b) If $\Delta F$ is less than 0 accept the move with a probability $P = exp(\frac{-\Delta F}{t})$.
    (c) End if statement.
3. the temperature $t$ by small increment $\Delta t$.
4. Go to 1.
```

As represented in the pseudocode, an extra probability was added to the mutation when the gene to modify is the one related to solver or activation function. Because a change in these specific hyperparameters leads to a big change in fitness and the changes in fitness do not have to be that relevant, also, those changes usually lead to significantly lower fitness. Furthermore, the change in numeric parameters as hidden layer sizes, alpha and learning rate is done in a range that is close to the actual value e.g. alpha = 0.005 would be change in a range between 0.003 and 0.007. Thus, avoid the aforementioned big changes in fitness.

Finally, to obtain a characteristic result, the algorithm was executed 10 times.

2.4 Fitness function

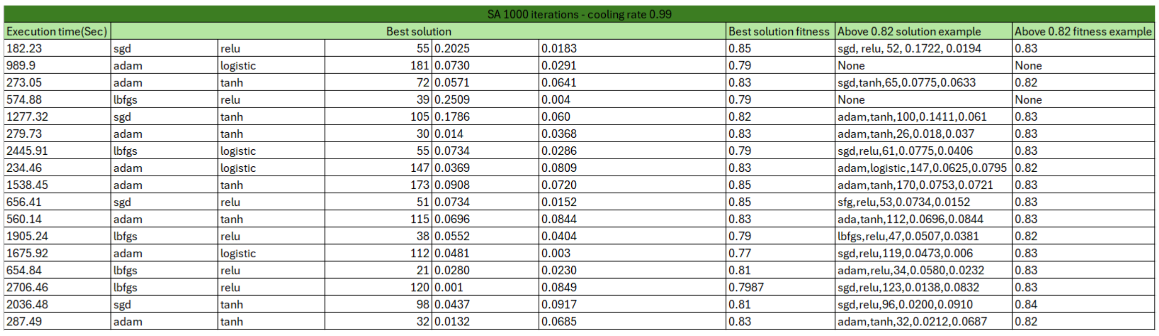
The fitness function used was the F1 score. The MLP was trained with random values in the training subset. Furthermore, F1 score was calculated using the *sklearn* Python library.

3 Results

3.1 Simulated annealing 1000 iterations

The simulated annealing algorithm offers the possibility to evolve a current random solution, that in this experiment is a MLP classifier. The hyperparameters are mutated given a probability that is getting smaller every iteration. If the new fitness is higher than the current one, the changes are accepted, if not the probability that accept negative changes in fitness is evaluated given the difference in fitness and the time.

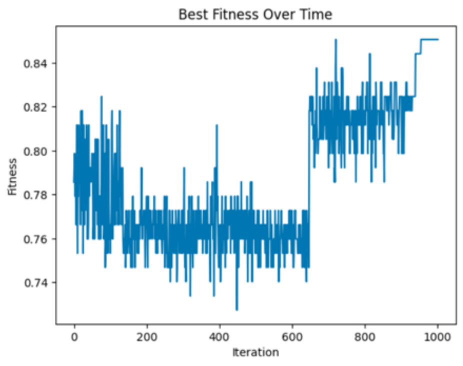
The algorithm was executed multiple times. In each scenario the algorithm was tested 10 times to obtain characteristic results and build strong solutions. Table 1 shows the results of using 1000 iterations and a cooling rate of 0,99. Additionally, some solutions were not the best solution overtime. An example of a solution that was greater or equal to 0.82 is shown in the table. This evidence shows that there are multiple viable solutions and that in some cases a change in the hyperparameters in the last iterations may lead to a lower performance, even if there is a small possibility.



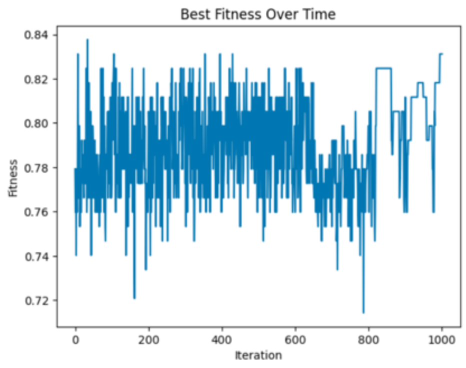
SA 1000 iterations - cooling rate 0.99								
Execution time(Sec)	Best solution					Best solution fitness	Above 0.82 solution example	Above 0.82 fitness example
182.23	sgd	relu	55	0.2025	0.0183	0.85	sgd, relu, 52, 0.1722, 0.0194	0.83
989.9	adam	logistic	181	0.0730	0.0291	0.79	None	None
273.05	adam	tanh	72	0.0571	0.0641	0.83	sgd,tanh,65,0.0775,0.0633	0.82
574.88	lbfgs	relu	39	0.2509	0.004	0.79	None	None
1277.32	sgd	tanh	105	0.1786	0.060	0.82	adam,tanh,100,0.1411,0.061	0.83
279.73	adam	tanh	30	0.014	0.0368	0.83	adam,tanh,26,0.018,0.037	0.83
2445.91	lbfgs	logistic	55	0.0734	0.0286	0.79	sgd,relu,61,0.0775,0.0406	0.83
234.46	adam	logistic	147	0.0369	0.0809	0.83	adam,logistic,147,0.0625,0.0795	0.82
1538.45	adam	tanh	173	0.0908	0.0720	0.85	adam,tanh,170,0.0753,0.0721	0.83
656.41	sgd	relu	51	0.0734	0.0152	0.85	sfg,relu,53,0.0734,0.0152	0.83
560.14	adam	tanh	115	0.0696	0.0844	0.83	ada,tanh,112,0.0696,0.0844	0.83
1905.24	lbfgs	relu	38	0.0552	0.0404	0.79	lbfgs,relu,47,0.0507,0.0381	0.82
1675.92	adam	logistic	112	0.0481	0.003	0.77	sgd,relu,119,0.0473,0.006	0.83
654.84	lbfgs	relu	21	0.0280	0.0230	0.81	adam,relu,34,0.0580,0.0232	0.83
2706.46	lbfgs	relu	120	0.001	0.0849	0.7987	sgd,relu,123,0.0138,0.0832	0.83
2036.48	sgd	tanh	98	0.0437	0.0917	0.81	sgd,relu,96,0.0200,0.0910	0.84
287.49	adam	tanh	32	0.0132	0.0685	0.83	adam,tanh,32,0.0212,0.0687	0.82

Table 1: SA 1000 iterations - Cooling rate 0,99.

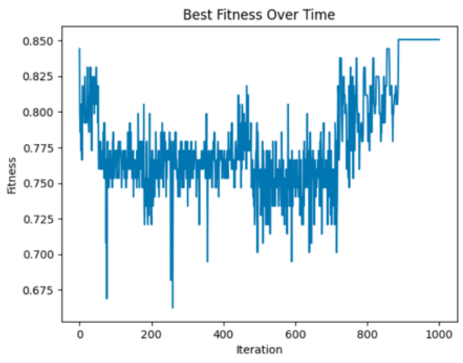
In figure 5 the aforementioned behavior is shown. The final solution has in many cases the best performance. However, this peak performance is reached in the 400-300 iterations. Multiple runs evidence that the algorithms have a constant best solution fitness. With no less than 0,79 in F1-Score metric.



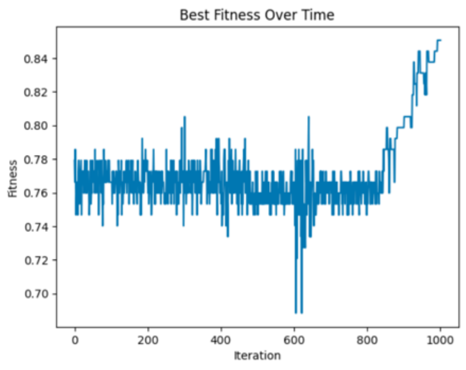
(a) SA 1000 Iterations - Cooling rate 0,99 - Fitness 0.85



(b) SA 1000 Iterations - Cooling rate 0,99 - Fitness 0.83



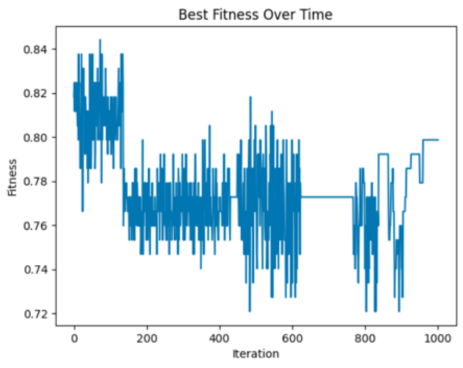
(c) SA 1000 Iterations - Cooling rate 0,99 - Fitness 0.85



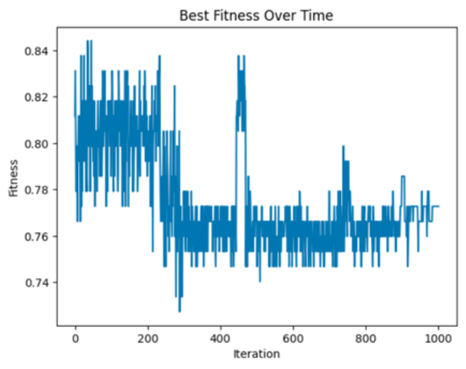
(d) SA 1000 Iterations - Cooling rate 0,99 - Fitness 0.85

Figure 4: Examples of SA 1000 Iterations - Cooling rate 0,99 performance.

Nevertheless, figure 5 exhibits that if the change in the first iterations leads to a low fitness. Is likely that the final fitness would be lower than the initial one. This can be improved by increasing the number of iterations.



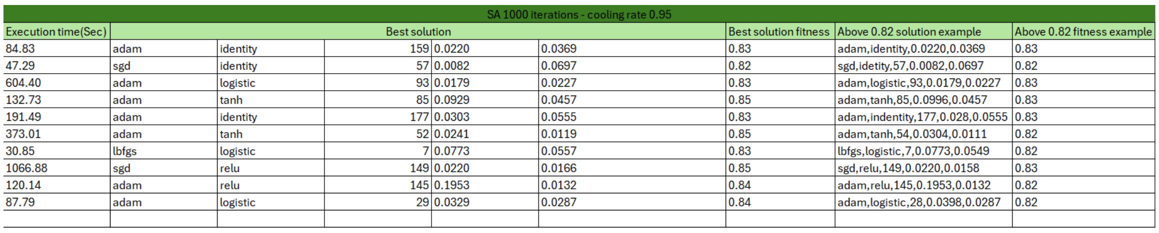
(a) SA 1000 Iterations - Cooling rate 0,99 - Fitness 0.79



(b) SA 1000 Iterations - Cooling rate 0,99 - Fitness 0.77

Figure 5: Examples of SA 1000 Iterations - Cooling rate 0,99 with low performance.

The results in table 2 shown the performance of the algorithm with 1000 iterations and a cooling rate of 0,95. The average execution time is lower. However, the average best solution fitness is higher than the previous configuration tested.

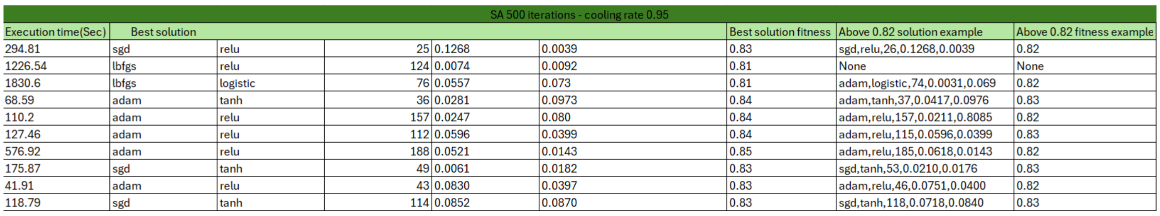


SA 1000 iterations - cooling rate 0.95								
Execution time(Sec)	Best solution					Best solution fitness	Above 0.82 solution example	Above 0.82 fitness example
84.83	adam	identity	159	0.0220	0.0369	0.83	adam,identity,0.0220,0.0369	0.83
47.29	sgd	identity	57	0.0082	0.0697	0.82	sgd,idetity,57,0.0082,0.0697	0.82
604.40	adam	logistic	93	0.0179	0.0227	0.83	adam,logistic,93,0.0179,0.0227	0.83
132.73	adam	tanh	85	0.0929	0.0457	0.85	adam,tanh,85,0.0996,0.0457	0.83
191.49	adam	identity	177	0.0303	0.0555	0.83	adam,indentity,177,0.028,0.0555	0.83
373.01	adam	tanh	52	0.0241	0.0119	0.85	adam,tanh,54,0.0304,0.0111	0.82
30.85	lbfgs	logistic	7	0.0773	0.0557	0.83	lbfgs,logistic,7,0.0773,0.0549	0.82
1066.88	sgd	relu	149	0.0220	0.0166	0.85	sgd,relu,149,0.0220,0.0158	0.83
120.14	adam	relu	145	0.1953	0.0132	0.84	adam,relu,145,0.1953,0.0132	0.82
87.79	adam	logistic	29	0.0329	0.0287	0.84	adam,logistic,28,0.0398,0.0287	0.82

Table 2: SA 1000 iterations - Cooling rate 0,95.

3.2 Simulated annealing 500 and 300 iterations

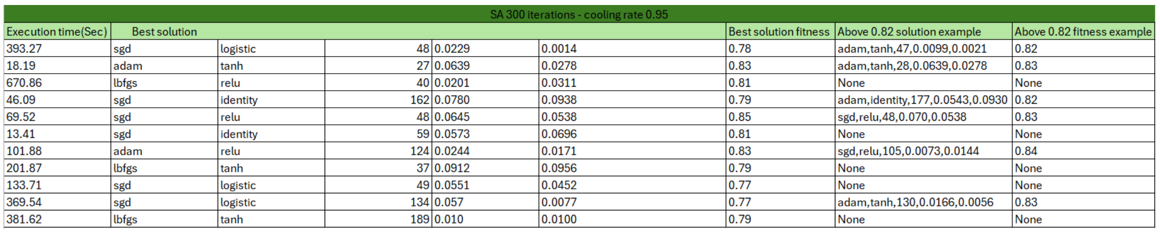
The test with 500 iterations and a cooling rate of 0,95 is shown in table 3. Some executions indicate that using adam as solver and relu as activation function lead to a high performance and low execution time. However, lbfgs presented the opposite behaviour with a lower fitness and higher time complexity.



SA 500 iterations - cooling rate 0.95								
Execution time(Sec)	Best solution					Best solution fitness	Above 0.82 solution example	Above 0.82 fitness example
294.81	sgd	relu	25	0.1268	0.0039	0.83	sgd,relu,26,0.1268,0.0039	0.82
1226.54	lbfgs	relu	124	0.0074	0.0092	0.81	None	None
1830.6	lbfgs	logistic	76	0.0557	0.073	0.81	adam,logistic,74,0.0031,0.069	0.82
68.59	adam	tanh	36	0.0281	0.0973	0.84	adam,tanh,37,0.0417,0.0976	0.83
110.2	adam	relu	157	0.0247	0.080	0.84	adam,relu,157,0.0211,0.8085	0.82
127.46	adam	relu	112	0.0596	0.0399	0.84	adam,relu,115,0.0596,0.0399	0.83
576.92	adam	relu	188	0.0521	0.0143	0.85	adam,relu,185,0.0618,0.0143	0.82
175.87	sgd	tanh	49	0.0061	0.0182	0.83	sgd,tanh,53,0.0210,0.0176	0.83
41.91	adam	relu	43	0.0830	0.0397	0.83	adam,relu,46,0.0751,0.0400	0.82
118.79	sgd	tanh	114	0.0852	0.0870	0.83	sgd,tanh,118,0.0718,0.0840	0.83

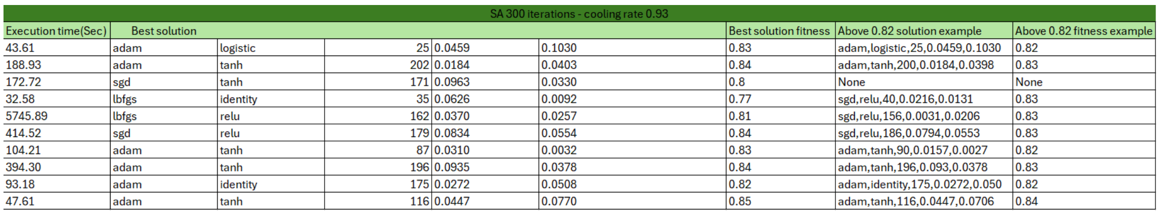
Table 3: SA 500 iterations - Cooling rate 0,95.

Additionally, with the objective to find the best solution in the less time possible and with the minimum iterations. It was tested the performance of the algorithm with 300 iterations and cooling rate of 0,95 and 0,93. Tables 4 and 5 document the results.



SA 300 iterations - cooling rate 0.95								
Execution time(Sec)	Best solution					Best solution fitness	Above 0.82 solution example	Above 0.82 fitness example
393.27	sgd	logistic	48	0.0229	0.0014	0.78	adam,tanh,47,0.0099,0.0021	0.82
18.19	adam	tanh	27	0.0639	0.0278	0.83	adam,tanh,28,0.0639,0.0278	0.83
670.86	lbfgs	relu	40	0.0201	0.0311	0.81	None	None
46.09	sgd	identity	162	0.0780	0.0938	0.79	adam,identity,177,0.0543,0.0930	0.82
69.52	sgd	relu	48	0.0645	0.0538	0.85	sgd,relu,48,0.070,0.0538	0.83
13.41	sgd	identity	59	0.0573	0.0696	0.81	None	None
101.88	adam	relu	124	0.0244	0.0171	0.83	sgd,relu,105,0.0073,0.0144	0.84
201.87	lbfgs	tanh	37	0.0912	0.0956	0.79	None	None
133.71	sgd	logistic	49	0.0551	0.0452	0.77	None	None
369.54	sgd	logistic	134	0.057	0.0077	0.77	adam,tanh,130,0.0166,0.0056	0.83
381.62	lbfgs	tanh	189	0.010	0.0100	0.79	None	None

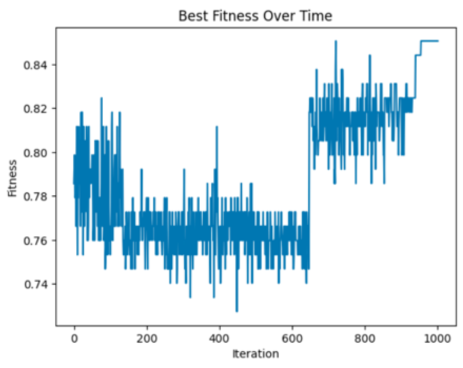
Table 4: SA 300 iterations - Cooling rate 0,95.



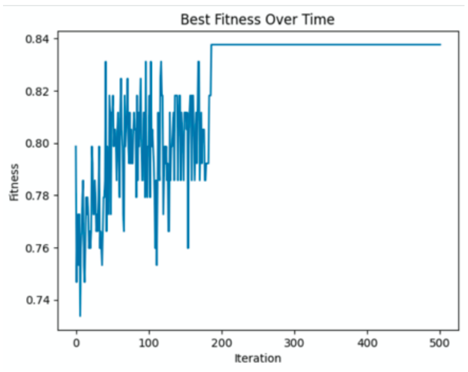
SA 300 iterations - cooling rate 0.93								
Execution time(Sec)	Best solution					Best solution fitness	Above 0.82 solution example	Above 0.82 fitness example
43.61	adam	logistic	25	0.0459	0.1030	0.83	adam,logistic,25,0.0459,0.1030	0.82
188.93	adam	tanh	202	0.0184	0.0403	0.84	adam,tanh,200,0.0184,0.0398	0.83
172.72	sgd	tanh	171	0.0963	0.0330	0.8	None	None
32.58	lbfgs	identity	35	0.0626	0.0092	0.77	sgd,relu,40,0.0216,0.0131	0.83
5745.89	lbfgs	relu	162	0.0370	0.0257	0.81	sgd,relu,156,0.0031,0.0206	0.83
414.52	sgd	relu	179	0.0834	0.0554	0.84	sgd,relu,186,0.0794,0.0553	0.83
104.21	adam	tanh	87	0.0310	0.0032	0.83	adam,tanh,90,0.0157,0.0027	0.82
394.30	adam	tanh	196	0.0935	0.0378	0.84	adam,tanh,196,0.093,0.0378	0.83
93.18	adam	identity	175	0.0272	0.0508	0.82	adam,identity,175,0.0272,0.050	0.82
47.61	adam	tanh	116	0.0447	0.0770	0.85	adam,tanh,116,0.0447,0.0706	0.84

Table 5: SA 300 iterations - Cooling rate 0,93.

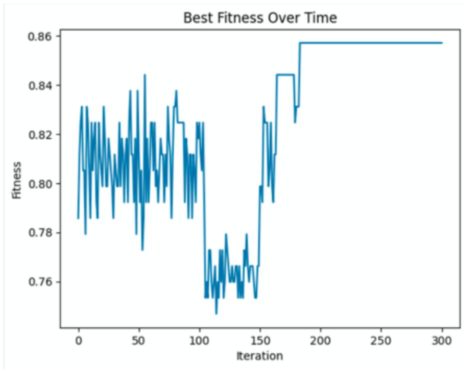
The results in previous tables 4 and 5 and in figure 6 verify that using less iterations with a lower cooling range lead to a lower average computer complexity. Nonetheless, it is possible that due to the limited iterations, the final solution fitness does not has enough chances to change and improve its fitness. Furthermore, in some cases a solution with more than 0.82 in fitness is never found. Moreover, this results suggest that with a higher number of iterations and a higher cooling rate a better solutions are found, what lead to previous test with 500 iterations and a cooling rate of 0,95. That in average has a better performance in fitness and time complexity.



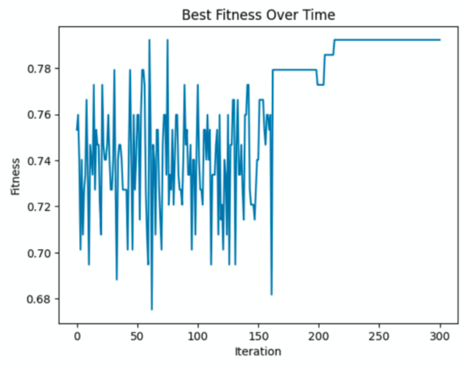
(a) SA 500 Iterations - Cooling rate 0,95 - Fitness 0.83



(b) SA 500 Iterations - Cooling rate 0,95 - Fitness 0.84



(c) SA 300 Iterations - Cooling rate 0,95 - Fitness 0.85



(d) SA 300 Iterations - Cooling rate 0,95 - Fitness 0.79

Figure 6: Examples of SA 500-300 Iterations - Cooling rate 0,95.

4 Discussion

Time complexity and machine learning models' performance are key factors when it comes to deciding whether the model is robust enough to be applied in real world problems. That is why the experiment's main goal was to test different configurations of maximum iterations and cooling rate to be used in simulated annealing algorithm that find the best solution in the less time possible. The evaluation metric that was used as fitness function was F1-Score. Where the best fitness was stored to be compared the best fitness during the process. Moreover, Fitness and hyperparameters set that were more than 0,82 were stored. The hyperparameters used were solver, activation function, hidden layer sizes, alpha and learning rate.

Results evidence that maximum number of iterations and cooling rate are key parameters in simulated annealing algorithm. MLP classifier hyperparameter optimization evidenced an increase in fitness. Since simulated annealing does not only accept positive changes in fitness but also negative ones, is possible to explore the solution space deeply and effectively. Results evidence that generally the algorithm finds a better hyperparameters set than the initial one, even if in the first hundred iterations the change in fitness is high, when the mutations probability become smaller, the algorithm finds its way to high fitness.

The best fitness and lower time complexity was found using 500 iterations and a cooling rate of 0.95. The average fitness was around 0.83 and the execution time in seconds was around 450 seconds. Also, in this case the final solution was most of the time the best solution over time. On the other hand, lbfgs solver usually led to high time complexity, even 4 or 5 times higher than the other ones and the solutions find with this solver is not significantly high to consider further testing.

To reproduce the experiment with other classification models and data sets, is important to understand the nature of the machine learning models and optimizations algorithms. Simulated annealing is effective to explore widely the solution space. However, the changes in fitness must be small and within a limited range near the current hyperparameter value. Therefore, the extra possibility was added in the case solver or activation functions were selected to be mutated, as a change in these hyperparameters lead to big changes in fitness.

5 Future work

Is important to go through the data structures used. Simulated annealing is a great option as an optimization algorithm. However, using more suitable data structures may help to reduce time complexity and find better solutions. Moreover, increasing the maximum iterations provides the opportunity to explore deeply the solution space and find better hyperparameters set. Likewise, cooling rate is a key value as represents the mutation rate that is going to decrease through iterations, try a value between 0,95 and 0,99 seems to be the best option. However, a lower value could be used with more generations as all positive changes are accepted but negative ones have a small probability to be accepted. Is recommended to test the performance of the algorithm with a fixed solver and evolve the other 4 proposed hyperparameters, this could lead to finding the best solution when using different classifiers and data sets.

References

[1] S. Wang, M. Roger, J. Sarrazin, and C. Lelandais-Perrault, "Hyperparameter optimization of two-hidden-layer neural networks for power amplifiers behavioral modeling using genetic algorithms," *IEEE microwave and wireless componets letters*, pp. 802–805, 2019. DOI: `10.1109/LMWC.2019.2950801`. [Online]. Available: `https://ieeexplore.ieee.org/abstract/document/8897013`.

[2] M. Fridrich, "Hyperparameter optimization of artificial neural network in customer churn prediction using genetic algorithm," *Brno University of Technology Faculty of Business and Management Department of Informatics*, vol. 11, no. 28, pp. 9–21, 2017. DOI: `https://doi.org/10.13164/trends.2017.28.9`. [Online]. Available: `https://ieeexplore.ieee.org/abstract/document/8897013`.

[3] L. F. Rodrigues, A. R. Backes, B. A. N. Travençolo, and G. M. B. de Oliveira, "Optimizing a deep residual neural network with genetic algorithm for acute lymphoblastic leukemia classification," *Journal of digital imaging*, vol. 35, pp. 623–637, 2022. DOI: `https://doi.org/10.1007/s10278-022-00600-3`. [Online]. Available: `https://trends.fbm.vutbr.cz/index.php/trends/article/view/385`.

[4] M. A. Bülbül, "Optimization of artificial neural network structure and hyperparameters in hybrid model by genetic algorithm: Ios–android application for breast cancer diagnosis/prediction," *The journal of super computing*, vol. 80, pp. 4533–4553, 2023. DOI: `https://doi.org/10.1007/s11227-023-05635-z`. [Online]. Available: `https://link.springer.com/article/10.1007/s11227-023-05635-z`.

[5] C. A. Whicher, S. O'Neill, and R. I. G. Holt, "Diabetes uk position statements diabetes in the uk: 2019," *The journal of super computing*, vol. 37, pp. 242–247, 2020. DOI: `https://doi.org/10.1111/dme.14225`. [Online]. Available: `https://onlinelibrary.wiley.com/doi/full/10.1111/dme.14225`.

[6] T. Scully, "Diabetes in numbers," *Nature*, vol. 485, S2–S3, 2012. DOI: `https://doi.org/10.1038/485S2a`. [Online]. Available: `https://www.nature.com/articles/485S2a`.

[7] B. Chaudhuri and U. Bhattacharya, "Efficient training and improved performance of multilayer perceptron in pattern classification," *Neurocomputing*, vol. 34, no. 1, pp. 11–27, 2000. DOI: `https://doi.org/10.1016/S0925-2312(00)00305-2`. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S0925231200003052`.

[8] L. Atlas, J. Connor, D. Park, *et al.*, "A performance comparison of trained multilayer perceptrons and trained classification trees," *ieee*, vol. 3, pp. 915–920, 1989. DOI: `10.1109/ICSMC.1989.71429`. [Online]. Available: `https://ieeexplore.ieee.org/abstract/document/71429`.

[9] V. Sigillito. "Diabetes dataset." (), [Online]. Available: `https://www.kaggle.com/datasets/mathchi/diabetes-data-set`. (accessed: 02.03.2024).

[10] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, pp. 671–680, 1983. DOI: `10.1126/science.220.4598.671`. [Online]. Available: `https://www.science.org/doi/abs/10.1126/science.220.4598.671`.

[11] R. A. Rutenbar, "Simula ted annealing algorithms: An overview," *IEEE Circuits and Devices Magazine*, vol. 220, pp. 19–26, 1989. DOI: `10.1109/101.17235`. [Online]. Available: `https://ieeexplore-ieee-org.sussex.idm.oclc.org/stamp/stamp.jsp?tp=&arnumber=17235&tag=1`.

[12] C. Johnson, “Other optimization methods,” *University of Sussex*, pp. 12–30, 2023.