CS7641 – Machine Learning

Assignment 2

Jingyao Zhu

jzhu398@gatech.edu

1 Overview

This report contains two parts. three algorithms including randomized hill climbing, simulated annealing and genetic algorithm are applied to twitch game dataset in HW1 to find the weights for neural networks models in the first part. Then, the second part is to apply four different algorithms - randomized hill climbing, simulated annealing, genetic algorithm and MIMIC to solve three optimization problems.

As shown in the HW1, the dataset is coming from Stanford snap data in Spring 2018, which is about 168,114 twitch gamers' information. It has two separate CSV files involving 'edges.csv' and 'features.csv'. In the 'edges.csv', 6,797,557 edges have been linked. In the 'features.csv', it contains 9 features such as 'views', 'mature', 'lifetime', etc. 7 of 9 features in features.csv will be used to train ML models. Then edges in edges.csv will be used to count mutual friends, which will be used in the ML models. Overall, 8 features will be used to predict and dead accounts will be used as a tag "churn". In order to balance two classes, each class will only randomly select 5000 records. The whole original dataset contains 10,000 records. (https://snap.stanford.edu/data/twitch_gamers.html)

2 Classification Problems Analysis

i. Neural Network with gradient descent

1. parameter tuning

Basic neural networks with gradient descent is to update the weights to help find the global/local optima point. The structure is input layer, hidden layer, and output layer. Figure 1 and 2 shows the training and testing accuracy for different hidden layers. Overall, 55% training and testing accuracies are the highest scores with layer 3. In figure 3, training time with layer 3 is the quickest. 9 and 12 hidden layers takes the much longer time to train the model.

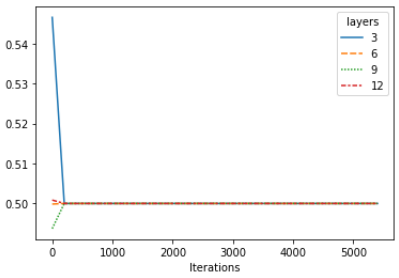

Figure 1:training accuracy with different layers

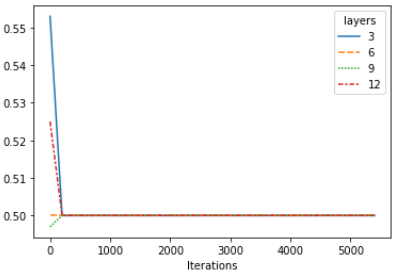

Figure 2: testing accuracy with different layers

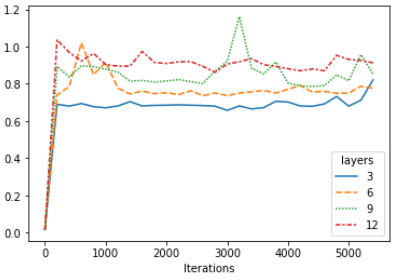

Figure 3: running time with different layers

2. model results

Overall, the neural network model with 3 hidden layers provide 55% model accuracy, which will be used as a baseline to compare.

ii. Neural Networks with simulated annealing

1. parameter tuning

Simulated annealing is a global optimization algorithm. It is quite similar to randomized hill climbing. They all are working well in non-linear functions because it used randomness in search process. Like randomized hill climbing, it changed one situation each time but it accepts some worse situation, which could make it search for global optima. The appropriate use of this algorithm is to apply in unimodal optimization problems. Hyperparameters such as temperature and cooling could also help find global optima.

As shown in Figure 4 and 5, there are two key points. First, when more data is provided, simulated annealing alogirthm could provide more accurate results. Second, layer 6 generally performance better than any other number of the layers when interations go up in training accuracy comparison and layer 6 is much stable.

In figure 6, ArithDecay temperatures algorithm with different decay pace such as 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1 shows that 0.1 decay in temperatures algorithm could provide much accurate results – 88.5%. In figure 8, model has been applied best parameters. Then layer 6 and 0.1 decay pace is the earliest one to converge to stable state. With decay pace 0.1 value and all other best parameters, figure 7 shows that layer 6 has compitable running time compared with any other layers. Thus, 0.1 will be used as a decay pace for simulated annealing. Simulated annealing algorithms need to set a proper temperature. The smaller the temperature is to set, the fastest converge and highest accuracy model will perform.

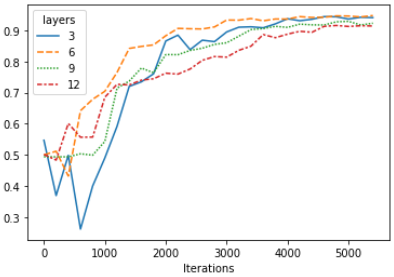

Figure 4:training accuracy with different layers

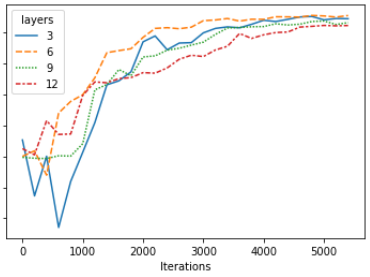

Figure 5: testing accuracy with different layers

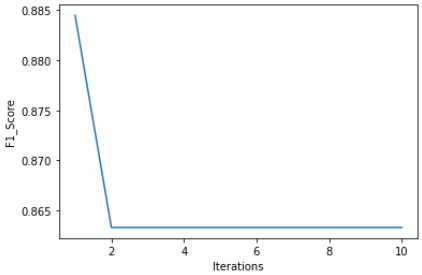

Figure 6: ArithDecay with different iterations

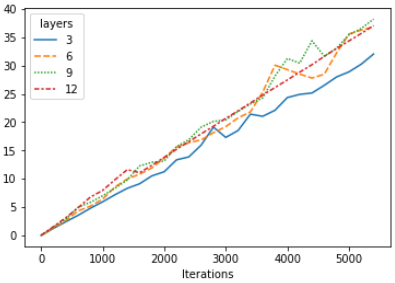

Figure 7: running time with different layers

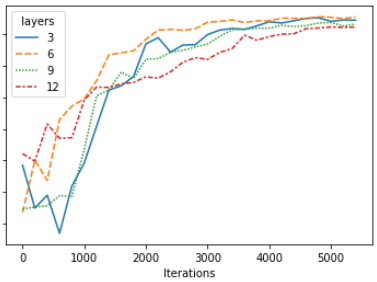

Figure 8: F1 score with different layers

2. model results

Genreally speaking, model performs well after around 4000 times iteration. Then less iterations becomes the key since it means less training time. 0.01 temperature with 0.1 decay pace wins. It helps hit 95.5% accuracy in training and testing within 27s. Overall, the simulated annealing provides a better result compared with the results from basic neural networks. But it is slightly worse than randomized hill climbing.

iii. Neural Networks with Randomized Hill Climbing

1. parameter tuning

Randomized hilling climbing is an optimization algorithm. It is working well in non-linear functions because it used randomness in search process. It has its limitation that it will work in local optima search. The appropriate use of this algorithm is to apply in unimodal optimization problems.

As shown in Figure 9, there are three key points. First, when more data is provided, randomized hill climbing alogirthm could provide more accurate results. In other words, It is hard for an alogirthm to even find local optimal with limited numbers of iterations. Second, layer 6 generally performance better than any other number of the layers when interations go up in training accuracy comparison. Third, when data is sufficient enough, factor layer does not affect much on the accuracy of the training set. The same logic could also gain from the Figure 10, which is the testing accuracy graph. Compared Figure 10 and Figure 13, different measurement may affect parameters selection when data is limited. In Figure 12, more hidden layers added to the neural networks, the slower the running time wil be. In Figure 11, whenever the number of restart points is greater than 1, F1 score won't change.

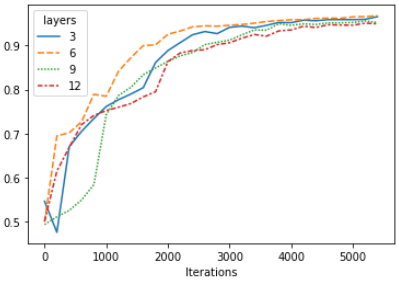

Figure 9:training accuracy with different layers

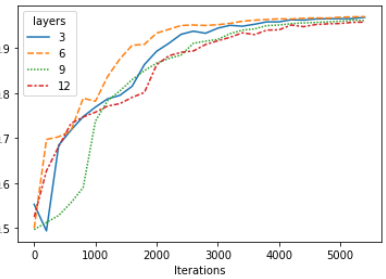

Figure 10: testing accuracy with different layers.

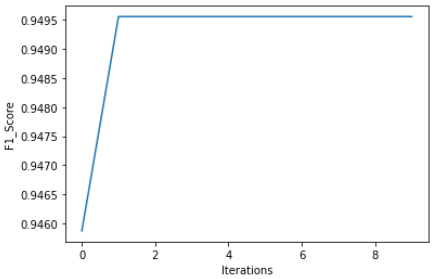

Figure 11: different restart with F1 score

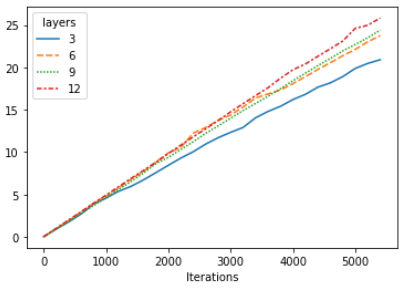

Figure 12: running time with different layers

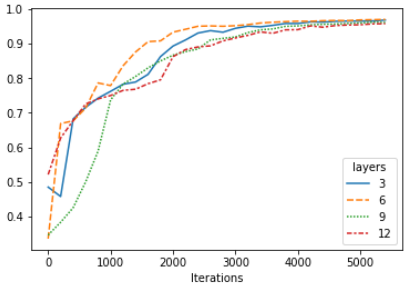

Figure 13: F1 score with different layers

2. model results

The range [2000, 3000] in training iterations is the best literatons for twitch gaming dataset because different measurements of the training and testing accuracy remains at 96% and running time is around 12s. Although, layers 3 has lowest running time, layers 6 provides more accurate result with competitable running time. Overall, the randomized hill climbing provides a better result compared with the results from basic neural networks.

iv. Neural Networks with genetic algorithm

1. parameter tuning

Genetic algorithm is an global search optimization algorithm. In genetic algorithm, it will be randomly picked in each iteration to train and heavier weights will be assign to the correct predictions. Since it is inspired by the biological theory of evolution, a new generation will be combined with two individuals. Thus, population and mute percentage are two hyperparameters here to be tuned. In figure 14, 15, and 18, they are showing that layer 3 and iterations 300 are best parameters to build a high performance model. In figure 17, layer 3 has second lower running time with 300 iterations. The interesting point here is that layer 12 with more iterations does not add more running time. With different set of population, 90 has the highest training and testing accuracy.

After determined 90 as population, let's keep all best parameters in the model and tune mute probability. In Figure 19, 0.25 has its advantange compared to any other probability.

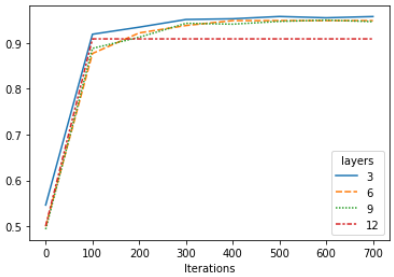

Figure 14:training accuracy with different layers

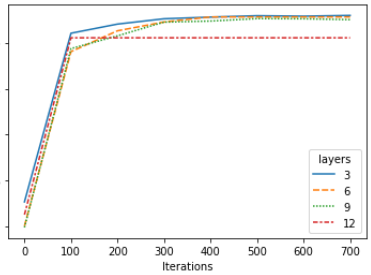

Figure 15: testing accuracy with different layers.

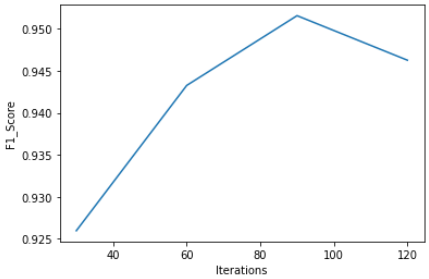

Figure 16: different population with F1 score

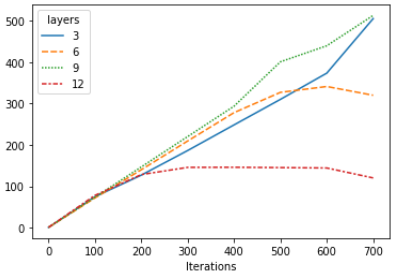

Figure 17:running time with different layers

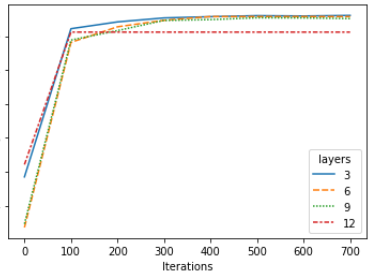

Figure 18: F1 score with different layers

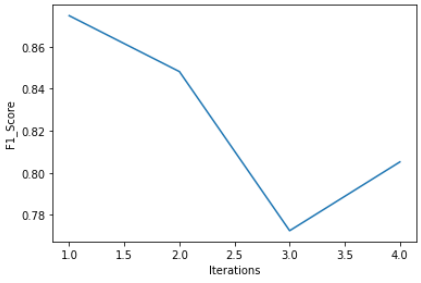

Figure 19: different Probability of a mutation with F1 score

2. model results

To sum up, layer 3 has second lower running time and best accuracy 96.3% and similar F1 score. 90 population and 0.25 probability mutation are best configurations to genetic algorithms. Compared with the basic neural network, it much better. Compared with the current best randomized hill climbing, it has super close performance.

v. Summary

The reason why three optimization algorithms perform better is that each of the three algorithms has its own fully researched knowledge base and tries to find the local/global optima. Genetic algorithm and randomized hill climbing have super close performances. I think the

reason is that genetic algorithm needs to tune parameters and hyperparameters, which means that it has high potiential to improve. randomized hill climbing wins because it provides competiable fitness score and running time is the fastest.

3 Optimization Problems Analysis

i. Continuous Peaks

1. Description

Continuous Peaks is a simple optimization problem that contains multiple local optima and eventually finds the global optima. In MLROSE_HIIVE, t_pct is 0.1 and it is suitable for use discrete-state with max_val 2. The continuous Peaks optimization problem is quite common in the world since it could help find the best number from a time of period. There are many good examples to apply this algorithm such as finding the best growth rate from different months of product usage for the new users.

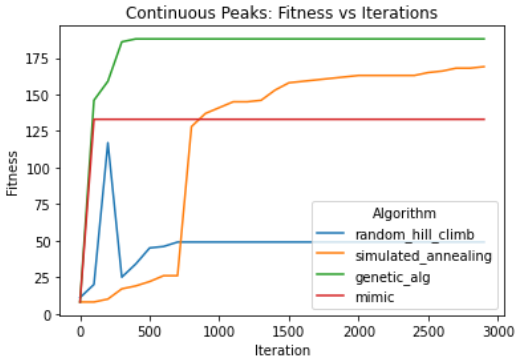

Figure 1: four algorithms with fitness of continuous peaks.

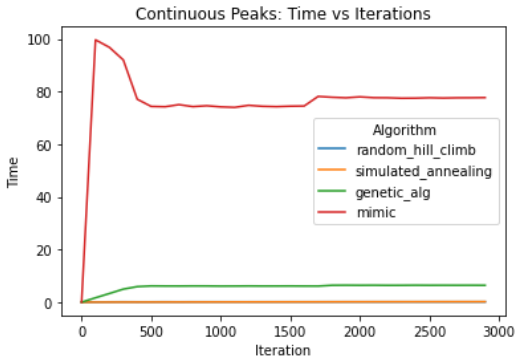

Figure 2: four algorithms with time of continuous peaks

2. optimization result

As shown in figure 1, genetic algorithm provides the highest fitness score. In other words, it can find global optima correctly compared with other three algorithms. It also provides the second lowest running time to find the global optima. MIMIC and Simulated Annealing perform okay just considering fitness score. Random hill climbing performs the worst in fitness score. In figure 2, the simulated annealing performs the best in running time and MIMIC performs worst in time running to find the global optima as usual.

ii. Queen

1. Description

Queen is a N-Queens optimization problem that is derived from The eight queens puzzle problem. The problem is to find how to place n non-attacking queens on an n×n chessboard. In MLROSE_HIIVE, it is suitable for use discrete-state optimization problems. The Queen optimization problem is quite common in the world since it could help find a solution how to place n Queens without any attack. There are many good examples to apply this algorithm such as placing and functioning different candidates parallelly in a company.

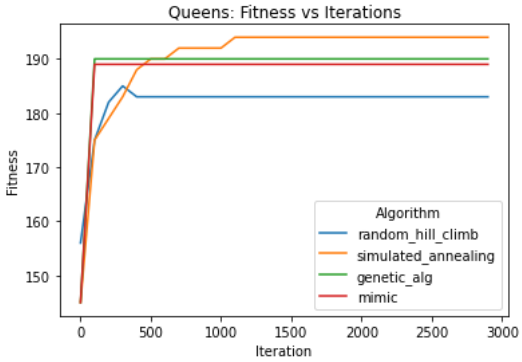

Figure 3: four algorithms with fitness of queen.

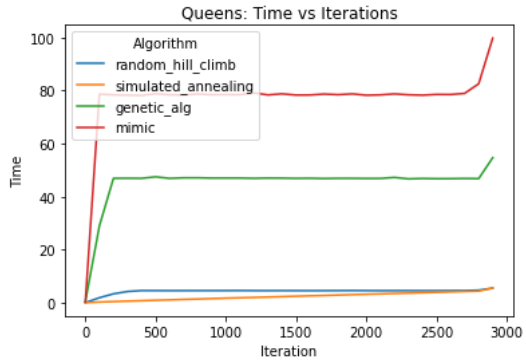

Figure 4: four algorithms with time of queen

2. Optimazation result

As shown in figure 3, Simulated Annealing provides the highest fitness score. In other words, it can find all solutions correctly compared with other three algorithms. It also provides the lowest running time to find all possible solutions. MIMIC and genetic algorithm perform okay just considering fitness score. Random hill climbing performs the worst in fitness score. In figure 4, the simulated annealing performs the best in running time and MIMIC performs worst in time running to find the global optima as usual.

iii. One Max

1. Description

The OneMax is a simple optimization question that is to find a way to maximize the sum of its digits given the length of binary string. In MLROSE_HIIVE, it is suitable for use discrete-state optimization problems. The Queen optimization problem is quite common use in learning the genetic frameworks. There are many good examples to apply in genetic study.

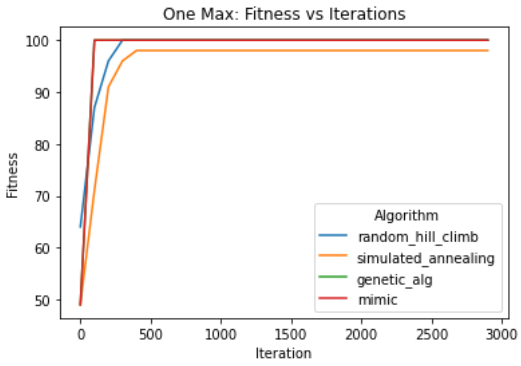

Figure 5: four algorithms with fitness of OneMax.

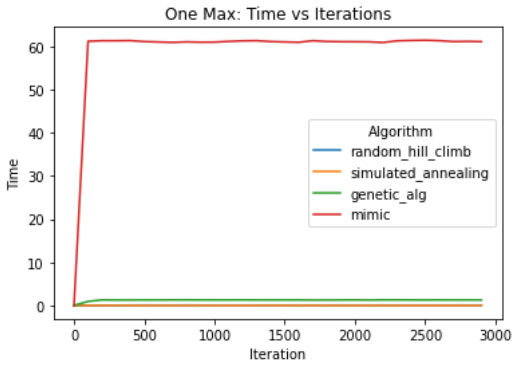

Figure 6: four algorithms with time of OneMax

2. Optimzation results

As shown in figure 6, Random hill climbing, MIMIC and genetic algorithm except Simulated Annealing perform super close considered fitness score. In figure 7, the simulated annealing performs the best in running time and MIMIC performs worst in time running to find the global optima as usual. But it is hard to say which one is the best. Then I limit the data iterations with less than 100. From figure 6, it is easy to tell that MIMIC wins because even though it has worst running time, its fitness score hits 100 within 100 iterations (far more better than any others).

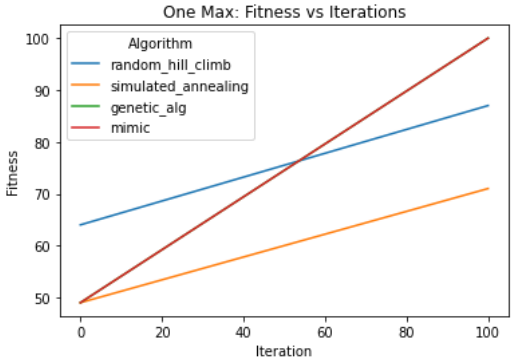

Figure 6: four algorithms with fitness of OneMax

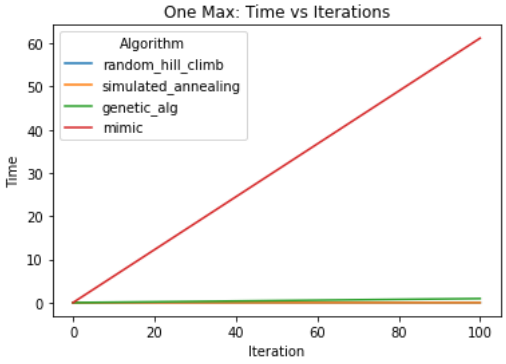

Figure 7: four algorithms with time of OneMax

iv. Compare

In Figure 8 and 9, from a running time aspect, randomized hill climbing, genetic algorithm, and simulated annealing are faster to learn compared with MIMIC. From a fitness score aspect, four algorithms are comptitable. But randomized hill climbing and simulated annealing are easy to implenment but hard to improve performance after tunning. MIMIC, and genetic algorithm are hard to implenment but easy to improve after tunning. For iteration persepective, MIMIC performs good with less iteration but other three have high volatility when iteration is small.

Algorithm	Problem	
genetic_alg	Continuous Peaks	188.0
	One Max	100.0
	Queens	190.0
mimic	Continuous Peaks	133.0
	One Max	100.0
	Queens	189.0
random_hill_climb	Continuous Peaks	117.0
	One Max	100.0
	Queens	185.0
simulated_annealing	Continuous Peaks	169.0
	One Max	98.0
	Queens	194.0

Figure 8: four algorithms with max fitness

Algorithm	Problem	
genetic_alg	Continuous Peaks	6.451427
	One Max	1.303160
	Queens	54.662958
mimic	Continuous Peaks	99.702667
	One Max	61.436756
	Queens	99.762370
random_hill_climb	Continuous Peaks	0.134890
	One Max	0.079285
	Queens	5.491245
simulated_annealing	Continuous Peaks	0.191175
	One Max	0.010751
	Queens	5.519350

Figure 9: four algorithms with max time

In Figure 10 and 15, compare different problems with different algorithms in average running time and average fitness score. The results inherence the previous conclusion.

Algorithm	Problem	
genetic_alg	Queens	188.500000
mimic	Queens	187.533333
random_hill_climb	Queens	181.866667
simulated_annealing	Queens	190.133333

Figure 10: four algorithms with avg fitness

Algorithm	Problem	
genetic_alg	Queens	45.032903
mimic	Queens	76.739012
random_hill_climb	Queens	4.286476
simulated_annealing	Queens	2.364027

Figure 11: four algorithms with avg time

Algorithm	Problem	
genetic_alg	Continuous Peaks	179.566667
mimic	Continuous Peaks	128.833333
random_hill_climb	Continuous Peaks	47.500000
simulated_annealing	Continuous Peaks	119.400000

Figure 12: four algorithms with avg fitness

Algorithm	Problem	
genetic_alg	Continuous Peaks	5.757229
mimic	Continuous Peaks	75.655351
random_hill_climb	Continuous Peaks	0.117826
simulated_annealing	Continuous Peaks	0.099497

Figure 13: four algorithms with avg time

Algorithm	Problem	
genetic_alg	One Max	98.300000
mimic	One Max	98.300000
random_hill_climb	One Max	98.233333
simulated_annealing	One Max	95.166667

Figure 14: four algorithms with avg fitness

Algorithm	Problem	
genetic_alg	One Max	1.226261
mimic	One Max	59.130900
random_hill_climb	One Max	0.065995
simulated_annealing	One Max	0.008070

Figure 15: four algorithms with avg time