IoT Intrusion Detection By kNN:

Multi-Objective Optimization

William Eng

Oakland University

Course Number: CSI 5130

Dr. Tianle Ma

12/2/2024

GitHub

Abstract

Multi-objective optimization of kNN machine learning model of intrusion detection for IoT devices and networks. The model was trained using the UNSW-NB15 dataset. Multi-objective optimization algorithms were from the MEALPY Python Open-Source library. Those algorithms were used to optimize the number of nearest neighbors and F1-score. There is a slight improvement to the F1-score and accuracy.

Introduction

We are increasingly living in an interconnected world. The need to protect our networks and traditional computer systems is well established. However, Internet of Things (IoT) and Operational Technology networks and devices have often been overlooked. With more of these devices having network capability, they are now easy targets of opportunity. A recent Joint Cybersecurity Advisory press release from the NSA, FBI and other international allies detailed a PRC botnet of compromised IoT devices with other devices such as SOHO routers and firewalls (Federal Bureau of Investigation et al., 2024).

A key use case for AI in IoT networks and devices is intrusion detection. Based on this course, several of the use cases were the ability to identify objects and classify them. This is essential for intrusion detection. Currently, much of the market is transitioning to AI assisted intrusion detection to protect networks and Extended Detection and Response solutions to protect endpoints.

Metaheuristic algorithms are designed to optimize machine learning model parameters. For the purposes of this project, I will be using k-Nearest Neighbors (kNN). This machine learning algorithm

There are several options to optimize machine learning models. This includes improving classification accuracy, detection rate/recall, precision, false alarm/error rate/ specificity, number of features, response time, memory usage, category detection. These options are in competition with each other such as the number of features and the

Existing implementations include Particle Swarm Optimization (PSO), Whale Optimization Algorithm (WOA), Non-dominated Sorting Genetic Algorithm, Binary Gravitational Search Algorithm, Binary Grey Wolf Optimization, etc. Current work in the field revolves around either hybridizing the algorithms or the feature selection (Salem et al., 2024, 22).

Related Works

Initial inspiration for this paper comes from the survey paper *Multi-objective optimization algorithms for intrusion detection in IoT networks; A systematic review* (Sharma et al., 2024). From there, I started reading about the different multi-objective optimization algorithms referenced in this paper.

In *An Effective Feature Selection Model Using Hybrid Metaheuristic Algorithms for IoT Intrusion Detection*, the authors hybridize Gorilla Troop Optimization and Bird Swarm Algorithm to create a unique algorithm. From the paper, they list their population parameter at 30 (Kareem et al., 2022, 14).

MEALPY is an open-source library for meta-heuristic algorithms. They have the algorithms separated in subpackages by biology, evolutionary, human, math, music, physics, swarm, and system. Thieu also took the liberty of rating the different modules. There is also the ability to run a multi-objective metaheuristic algorithms and assigning weights to objectives (Thieu & Mirjalili, 2023).

Data

For this project, I am using the UNSW-NB15 dataset from the Intelligent Security Group at UNSW Canberra. The data is network capture packets that were created in a simulated network environment. This dataset has 9 types of attacks. They are labelled: Generic, Exploits, Fuzzers, DoS, Reconnaissance, Analysis, Backdoor, Shellcode. The control is aptly labelled as

“Normal”. Generic attacks are brute force attacks (Moustafa & Slay, 2015, 5). They included both the training and testing datasets in CSV format. There are 175,341 and 82,332 records respectively. This will be useful when providing the kNN model. (Moustafa & Slay, 2021)

Attack Categories	**Training Dataset Total Count**	**Testing Dataset Total Count**
Analysis	2000	37000
Backdoor	1746	18871
DoS	12264	11132
Exploits	33393	6062
Fuzzers	18184	4089
Generic	40000	3496
Normal	56000	677
Reconnaissance	10491	583
Shellcode	1133	378
Worms	130	44
Grand Total	**175341**	**82332**

Table 1. UNSW-NB15 Dataset Breakdown

Data was preprocessed before feeding into the kNN model. As seen from the training dataset, it is skewed to Normal compared to attacks. It is also skewed to brute force attacks.   In an attempt to balance the attack categories and the available data, I excluded attacks labelled Analysis, Backdoor, Shellcode, and Worms. These attacks had less than 10,000 records.

Initially, for the more prominent labels (Generic, Exploits, Fuzzers, DoS, Reconnaissance), I kept around 10,000 records of each. All normal records were included. Roughly, the number of normal records is equal to the number of attacks in the subset. The subset I am using for training ended up with 106950 records total.

Ultimately, I created another smaller subset so each attack would have about 2,000 records and about 10,000 normal records. This was due to computational time constraints of running each iteration of the multi-object optimization algorithm.

Methods

I initially selected kNN as it was listed several times in the original survey paper (Sharma et al., 2024, 4). kNN is simple compared to its more complicated rivals such as Convolutional Neural Network. Since the model is relatively simple, an optimization algorithm should demonstrate proof of concept.

For the multi-objective algorithms, I selected PSO and WOA as they were also present in the original survey paper. Both algorithms appear to be relatively mature compared to algorithms like Gorilla Troop Optimization. They were both supported in MEALPY as well.

Experiments

Experiments were run in Google Colab. Essential libraries include pandas, sklearn, numpy, MEALPY. Pandas is used to load our training and test datasets. Sklearn from scikit-learn was used instead of PyTorch as I am not using tensor (scikit-learn, 2024). Numpy was used for vector calculations. MEALPY was used for objective optimization algorithms.

Runtime type	Python 3
Hardware accelerator	CPU

Table 2. Google Colab Settings

Initial k Nearest Neighbor	5
Population	30
Epoch	3
Filter Out	Analysis, Backdoor, Shellcode, Worms
Optimize For	k, f1

Table 3. Parameters

Unfortunately, due to time constraints, epoch was set to 3. Initial implementation of a single objective optimization was nearly 2 and ½ hours for a single epoch. Future experiments should be done with 100 epochs.

Key metrics that I want to track are the cyber attack accuracy, time to detect, and type of cyberattack detected. Accuracy is important for any response to an attack. The time is important because the goal is to minimize risk. The longer the attack can go undetected, the greater the risk. The third metric is dependent on the available data. If there is enough data to run through multiple types of cyberattacks, I would like to compare the AI's effectiveness against each type of cyberattack.

Baseline kNN Large Dataset

The kNN model using a training dataset of 106950 records was generated in 3 mins and 39 seconds. At our baseline, we are starting at roughly 77% accuracy of determining that there is an attack. . However, differentiating the attack types other than a normal record are subpar. The best performing attack categories were normal and generic with F1-scores of 0.74 and 0.66. Generic attacks were most commonly mistaken for a fuzzer attack. Computational time. Below are the results for the baseline kNN:

	Precision	**Recall**	**F1-score**	**Support**
DoS	0.26	0.46	0.34	4089
Exploits	0.64	0.42	0.51	11132
Fuzzers	0.13	0.36	0.19	6062
Generic	1.00	0.50	0.66	18871
Normal	0.74	0.74	0.74	37000
Reconnaissance	0.44	0.38	0.41	3496
Accuracy			0.58	80650
Macro avg	0.54	0.48	0.48	80650
Weighted avg	0.71	0.58	0.62	80650

Table 4. Filtered Attack Category Classification Report

	DoS	**Exploits**	**Fuzzers**	**Generic**	**Normal**	**Reconnaissance**
DoS	1897	484	536	2	975	195
Exploits	2876	4686	980	4	2072	514

Fuzzers	926	84	2192	4	2739	117
Generic	934	814	5374	9359	2275	115
Normal	356	981	7457	0	27463	743
Reconnaissance	189	294	322	0	1347	1344

Table 5. Filtered Attack Category Confusion Matrix

	Precision	Recall	F1-score	Support
Normal	0.75	0.74	0.74	37000
Attack	0.78	0.79	0.79	43650
Accuracy			0.77	80650
Macro avg	0.76	0.76	0.76	80650
Weighted avg	0.77	0.77	0.77	80650

Table 6. Filtered Attack Label Classification Report

	Normal	Attack
Normal	27289	9711
Attack	9144	34506

Table 7. Filtered Attack Label Confusion Matrix

	Precision	Recall	F1-score	Support
Normal	0.74	0.74	0.74	37000
Attack	0.79	0.79	0.79	45332

Accuracy			0.77	82332
Macro avg	0.76	0.76	0.76	82332
Weighted avg	0.77	0.77	0.77	82332

Table 8. All Attack Label Classification Report

Baseline kNN Small Dataset

The baseline kNN performed slightly worse overall and
At our baseline, we are starting at roughly 77% accuracy of determining that there is an attack. However, differentiating the attack types other than a normal record are subpar. Generic attacks were most commonly mistaken for a fuzzer attack. Below are the results for the baseline kNN:

	Precision	Recall	F1-score	Support
DoS	0.19	0.47	0.27	4089
Exploits	0.62	0.48	0.54	11132
Fuzzers	0.17	0.66	0.27	6062
Generic	0.99	0.52	0.68	18871
Normal	0.88	0.48	0.62	37000
Reconnaissance	0.24	0.52	0.33	3496
Accuracy			0.51	80650
Macro avg	0.51	0.52	0.45	80650
Weighted avg	0.71	0.51	0.57	80650

Table 9. Filtered Attack Category Classification Report

	DoS	Exploits	Fuzzers	Generic	Normal	Reconnaissance
DoS	1933	634	813	5	440	264
Exploits	2625	5361	1685	10	630	821

Fuzzers	981	78	4030	8	402	563
Generic	2426	297	5699	9768	386	295
Normal	1989	1975	11167	89	17870	3910
Reconnaissance	222	249	594	0	606	1825

Table 10. Filtered Attack Category Confusion Matrix

	Precision	Recall	F1-score	Support
Normal	0.88	0.48	0.62	37000
Attack	0.68	0.95	0.79	43650
Accuracy			0.77	82332
Macro avg	0.76	0.76	0.76	82332
Weighted avg	0.77	0.77	0.77	82332

Table 11. Filtered Attack Label Classification Report

	Normal	Attack
Normal	27291	9709
Attack	9448	35884

Table 12. Filtered Attack Label Confusion Matrix

Particle Swarm Optimization (Small Dataset)

The Particle Swarm Optimization determined the k-value of 23.18. I rounded this value to 23. This did improve the F1-scores for attack categories by at least 0.02. For attack labels, it gave similar results. This is most likely due to the low epoch value of 3. Below are the results of the kNN model using a k-value of 23.

	Baseline F1-score	PSO F1-score
DoS	0.27	0.32
Exploits	0.54	0.55
Fuzzers	0.27	0.28
Generic	0.68	0.71
Normal	0.62	0.64
Reconnaissance	0.33	0.40
Accuracy	0.51	0.53
Macro avg	0.45	0.48
Weighted avg	0.57	0.59

Table 13. Comparison of Attack Category Baseline to PSO

	Baseline F1-score	PSO F1-score
Normal	0.62	0.65
Attack	0.79	0.82
Accuracy	0.77	0.76
Macro avg	0.76	0.73
Weighted avg	0.77	0.74

Table 14. Comparison of Attack Label Baseline to PSO

	Precision	Recall	F1-score	Support
DoS	0.23	0.55	0.32	4089
Exploits	0.61	0.50	0.55	11132
Fuzzers	0.17	0.72	0.28	6062
Generic	0.97	0.56	0.71	18871
Normal	0.97	0.48	0.64	37000

Reconnaissance	0.30	0.59	0.40	3496
Accuracy			0.53	80650
Macro avg	0.54	0.57	0.48	80650
Weighted avg	0.80	0.53	0.59	80650

Table 13. Filtered Attack Category Classification Report

	DoS	**Exploits**	**Fuzzers**	**Generic**	**Normal**	**Reconnaissance**
DoS	2269	656	898	8	31	227
Exploits	2495	5593	1931	16	149	948
Fuzzers	1016	98	4371	19	123	435
Generic	2854	305	4802	10541	164	205
Normal	988	2261	12677	255	17832	2987
Reconnaissance	300	329	793	1	13	2060

Table 14. Filtered Attack Category Confusion Matrix

	Precision	**Recall**	**F1-score**	**Support**
Normal	0.98	0.48	0.65	37000
Attack	0.69	0.99	0.82	43650
Accuracy			0.76	82332
Macro avg	0.84	0.74	0.73	82332
Weighted avg	0.82	0.76	0.74	82332

Table 15. Filtered Attack Label Classification Report

	Normal	**Attack**
Normal	17831	19169
Attack	384	44948

Table 16. Filtered Attack Label Confusion Matrix

Whale Optimization Algorithm (Small Dataset)

The WOA determined the k-value of 22.52. I rounded this value to 23. This did improve the F1-score by . Below are the results of the kNN model using a k-value of 23. It is identical to the PSO results.

	Precision	Recall	F1-score	Support
DoS	0.23	0.55	0.32	4089
Exploits	0.61	0.50	0.55	11132
Fuzzers	0.17	0.72	0.28	6062
Generic	0.97	0.56	0.71	18871
Normal	0.97	0.48	0.64	37000
Reconnaissance	0.30	0.59	0.40	3496
Accuracy			0.53	80650
Macro avg	0.54	0.57	0.48	80650
Weighted avg	0.80	0.53	0.59	80650

Table 13. Filtered Attack Category Classification Report

	DoS	Exploits	Fuzzers	Generic	Normal	Reconnaissance
DoS	2269	656	898	8	31	227
Exploits	2495	5593	1931	16	149	948
Fuzzers	1016	98	4371	19	123	435
Generic	2854	305	4802	10541	164	205
Normal	988	2261	12677	255	17832	2987
Reconnaissance	300	329	793	1	13	2060

Table 14. Filtered Attack Category Confusion Matrix

	Precision	Recall	F1-score	Support
Normal	0.98	0.48	0.65	37000

Attack	0.69	0.99	0.82	43650
Accuracy			0.76	82332
Macro avg	0.84	0.74	0.73	82332
Weighted avg	0.82	0.76	0.74	82332

Table 11. Filtered Attack Label Classification Report

	Normal	**Attack**
Normal	17831	19169
Attack	384	44948

Table 12. Filtered Attack Label Confusion Matrix

Conclusion

I was able to implement multi-objective optimization of the kNN machine learning model. . The model was trained using the UNSW-NB15 dataset. Multi-objective optimization algorithms were from the MEALPY Python Open-Source library. Those algorithms were used to optimize the number of nearest neighbors and F1-score. There is a slight improvement to the F1-score and accuracy. Due to the low epoch value of 3, for compute time considerations, the improvements were not as high impact to bring accuracy to at least 0.80. For future iterations of the project, a full 100 epochs should be used.

In the future, this project should consider expanding from 2 objectives to 3 objectives to introduce further complexity. There is another dataset from Intelligent Security Group at UNSW Canberra called TON_IoT. This dataset details specific IoT devices.

References

Federal Bureau of Investigation, Cyber National Mission Force, & National Security Agency. (2024, September 18). *People's Republic of China-Linked Actors Compromise Routers and IoT Devices for Botnet Operations*. Department of Defense. Retrieved November 3, 2024, from https://media.defense.gov/2024/Sep/18/2003547016/-1/-1/0/CSA-PRC-LINKED-ACTORS-BOTNET.PDF

Kareem, S. S., Mostafa, R. R., Hashim, F. A., & El-Bakry, H. M. (2022, February 11). An Effective Feature Selection Model Using Hybrid Metaheuristic Algorithms for IoT Intrusion Detection. *Sensors*, *22*(4), 1396. https://www.mdpi.com/1424-8220/22/4/1396

Mastafa, N. (2016). The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data se. *Information Security Journal: A Global Perspective*, 1-14. https://www.tandfonline.com/doi/abs/10.1080/19393555.2015.1125974

Moustafa, N., Creech, G., & Slay, J. (2017). Big Data Analytics for Intrusion Detection System: Statistical Decision-Making Using Finite Dirichlet Mixture Models. In I. Palomares Carrascosa, H. K. Kalutarage, & Y. Huang (Eds.), *Data Analytics and Decision Support for Cybersecurity: Trends, Methodologies and Applications*. Springer International Publishing.

Moustafa, N., & Slay, J. (2015). UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). *2015 Military Communications and Information Systems Conference (MilCIS)*. 10.1109/MilCIS.2015.7348942

Moustafa, N., & Slay, J. (2021, June 2). *The UNSW-NB15 Dataset*. The UNSW-NB15 Dataset | UNSW Research. Retrieved November 3, 2024, from https://research.unsw.edu.au/projects/unsw-nb15-dataset

Moustafa, N., Slay, J., & Creech, G. (2019, December 01). Novel Geometric Area Analysis Technique for Anomaly Detection Using Trapezoidal Area Estimation on Large-Scale Networks. *IEEE Transactions on Big Data*, *5*(4), 481-494. https://ieeexplore.ieee.org/abstract/document/7948715

Salem, A. H., Azzam, S. M., Emam, O. E., & Abohany, A. A. (2024). Advancing cybersecurity: a comprehensive review of AI-driven detection techniques. *Journal of Big Data*, *11*(105). https://journalofbigdata.springeropen.com/articles/10.1186/s40537-024-00957-y

Sarhan, M., Layeghy, S., Moustafa, N., & Portmann, M. (2020, November 18). NetFlow Datasets for Machine Learning-based Network Intrusion Detection Systems. *BDTA 2020*, 1-16. https://arxiv.org/abs/2011.09144

scikit-learn. (2024, September). *scikit-learn*. scikit-learn: Machine Learning in Python. https://scikit-learn.org/stable/

Sharma, S., Kumar, V., & Dutta, K. (2024). Multi-objective optimization algorithms for intrusion detection in IoT networks: A systematic review. *Internet of Things and Cyber-Physical Systems*, *4*, 258-267. https://doi.org/10.1016/j.iotcps.2024.01.003

Thieu, N. V., & Mirjalili, S. (2023). MEALPY: An open-source library for latest meta-heuristic algorithms in Python. *Journal of Systems Architecture*, *139*(2023). https://www.sciencedirect.com/science/article/abs/pii/S1383762123000504?via%3Dihub