네트워크 침입 탐지 성능 향상을 위한 최적화 연구

이지원, 김형태, 김경백

전남대학교

maum-@hanmail.net, akkain0513@gmail.com, kyungbaekkim@jnu.ac.kr

Study of Optimization for improving the Performance of Network Intrusion Detection

Jiwon Lee, Hyungtae Kim, Kyungbaek Kim

Chonnam National University.

요 약

우리의 생활이 네트워크 환경과 밀접해지면서 네트워크 위협이 증가하고 있다. 이에 이상 탐지기반 침입 탐지 연구가 활발히 이루어지고 있으며, 네트워크 침입 탐지 데이터셋과 이를 활용한 기계학습 모델들이 주목받고 있다. 현재 탐지율 향상을 위해 다양한 기계학습 모델들이 사용되고 있지만, 연구 대부분이 다양한 환경을 고려하기 보다는 주어진 한가지 데이터셋에 성능을 향상시키기 위해 모델의 변형 및 고도화에 집중되어 있다. 이러한 모델들은 기존 사용하던 데이터셋과 다른 데이터셋을 사용하면 기존의 성능을 끌어내지 못하였다. 본 논문에서는 최소한의 기능의 순수한 모델을 만들고, 각 모델들의 모델 성능을 끌어내기 위해 하이퍼 파라미터 최적화를 이용하여 각 데이터셋 마다 성능을 측정하였다. 이를 이용하여 각 환경에서 유연한 네트워크 탐지 모델 개발에 도움을 주고자 한다.

Ⅰ. 서 론

현재 사회에서 네트워크 환경은 우리 일상에 더욱더 가까워지고 있다. 그와 동시에 다양한 네트워크 환경에 노출되며, 예상치 못한 사이버 공격이 증가하고 있다. 현재 기계학습 연구가 발달함으로써 기계학습을 도입한 네트워크 침입 탐지 연구가 활발히 이루어 지고, 그에 필요한 다양한 환경의 네트워크 데이터셋 자료 또 한 공개되었다. 이에 다양한 데이터셋에 적용 가능한 범용적인 모델이 필요하지만, 많은 연구들이 다른 환경의 데이터셋은 고려하지 않고, 한가지 데이터셋에 최적화된 앙상블 모델을 개발하는 등 성능 향상을 위해 모델 고도화에 몰두되고 있다[1, 2]. 데이터셋에 불균형을 해소하기 위해 오버샘플링을 적용한 연구도 존재한다[3,4]. 랜덤 오버샘플링을 적용하여 KNN, LSTM 등 다양한 모델로 실험을 한 연구에서는 기계학습 모델에서는 기본값과 딥러닝에서는 임의로 생성한 모델을 사용하여 적용하였다[5]. 우리는 하이퍼 파리미터 설정만으로 충분히 높은 성능을 기대하며, 실험 구성을 위해 간단한 모델을 만들었다. 또 한 최적의 하이퍼 파라미터를 적용하기 위해 Bayesian optimization를 사용했으며, NSL-KDD, UNSW-NB15데이터셋의 사용하여 결과를 내놓았다.

Ⅱ. 관련연구

2.1 데이터셋

우리는 NSL-KDD, UNSW-NB15 데이터셋을 선택했다. 두 데이터의 특징으로는 다양한 공격 방법들이 포함되어 있고, 피처의 개수가 비슷하고 일부 공통된 피처도 존재하지만, 기계학습 모델마다 성능이 다르게 측정되었다. 또 한 다른 데이터와 달리 학습 데이터셋, 테스트 데이터셋이 나뉘어서 제공되기 때문에 좀 더 객관적인 측정이 가능하다는 점이다. 이런 특징을 가지고 있어 활발하게 연구들이 이루어져 있다. [1] 같은 경우에는 Decision Tree모델을 이용한 앙상블 모델을 만들어 높은 정확도를 보이고 있다. Decision Tree 모델 이외에도 딥러닝 모델을 포함한 4가지의 이상의 모델들을 결합한 앙상블 모델에서도 85% 내외에 정확도를 보여주고 있다[2]. Chuanlong Yin의 연구에서는 LSTM의 하이퍼 파라미터 중에 노드의 개수에 대한 성능을 보여주고 있다[11]. UNSW-NB15 데이터셋에 Autoencoder와 SVM을 결합한 모델로 측정한 정확도는 89%, J48로 측정한 결과로는 88%로 의사결정나무 모델도 높은 정확도를 보인다[6,7]. 또 한 데이터셋에 불균형을 해결하기 위해 데이터셋에 랜덤 오버샘플링을 적용한 후 Conv1D모델을 측정한 결과에도 비슷한 정확도를 보여주고 있다[3].

표 1 관련 연구 정확도

Method	Dataset	Result
Decision Tree Bagging Ensemble[1]	NSL-KDD	Accurancy = 85.81%
Ansemble model[2]	NSL-KDD	Accurancy =85.2%
LSTM[6]	NSL-KDD	Accurancy = 83.28%
Autoencoder & SVM [7]	UNSW-NB15	Accurancy = 89.134%
Decision Tree(J48) [8]	UNSW-NB15	Accurancy = 88.3%
Conv1D[5]	UNSW-NB15	Accurancy = 89.4%

2.1 기계학습 모델

K-Nearest Neighbors(KNN)은 입력 후에 학습 데이터들을 k개의 이웃 무리로 구분 짓고, 이웃 중에서 근접한 이웃끼리 거리를 맞춰서 대입하는 것이다. 게으른 학습 방법의 하나로써 테스트 속도가 느리지만, 특정 부류를 구분하는 작업에서 효율이 좋다. 주 파라미터는 K값으로써 이웃들의 수를 설정할 수 있다.

Logistic Regression(LR)는 선형 회귀 모델과 같이 입력된 데이터 특성의 가중치 합을 계산한다. 하지만 선형 회귀와 다르게 입력 데이터에 대한 값이 아닌, 그것에 값이 특정 영역에 속할 확률을 계산하는데 사용된다. 그리고 그 모델의 강도를 조절하는데 역수인 C를 사용하여, C값이 클수록 모델의 규제를 줄일 수 있다.

Support Vector Machine(SVM)는 학습 데이터들을 구분 짓기 위

한 경계를 찾는 것이 목적으로, 경계와 데이터 사이에 마진이라는 여백 공간을 최대로 하기 위해, 마진과 가장 가까운 값인 서포트 벡터를 기준으로 마진을 최대화 하는 경계를 찾는 것이 목적이다.

의사결정나무(DT) 모델은 학습 데이터로부터 특징을 추출하여 트리 형태를 만드는 알고리즘으로써, 다양한 데이터셋에 활용이 가능한 다재다능의 알고리즘이다. 아래에 있는 모델들의 부모격인 알고리즘으로써, 다양한 파생모델들이 존재한다. 주요 파라미터는 max_depth, leaf이다.

Random Forest(RF)는 훈련 데이터에 중복을 허용하지 않는 pasting(페이스팅) 방법을 사용하여 만든 결정 트리의 앙상블이다. 결정나무 모델은 bagging처럼 중복을 허용하여 최고의 특징을 뽑아내기 위해 초점이 맞춰져 있다. 반면 랜덤 포레스트는 무작위로 다양한 특징들을 뽑아낸 후, 그 특징들에서 최적의 해를 뽑아낸다. 그렇기에 편향된 데이터를 분산시켜 과적합을 피할 수 있다.

eXtreme Gradient Boosting(XGB)은 Gradient Boosting Machine(GBM)의 하나의 방법으로 약한 학습기를 결합하여 이전 값의 오차를 결합 나가는 앙상블 모델이다. GBM 보다 좀더 속도 측면에서 개선되었지만, 여전히 속도는 다른 모델에 비해서 느린 편이다. 과접합 문제를 해결하기 위한 여러 가지 기능들이 내장되어 다양한 하이퍼 파라미터를 적용할 수 있다. 많은 하이퍼 파라미터 때문에 다른 검색 방법보다 Bayesian optimization이 하이퍼 파라미터를 찾는데 우수한 성능을 보였다[12].

Light Gradient Boosting Machine(LGBM)은 XGB의 개량 모델로 속도가 XGB보다 빠르다. 일반적으로 의사결정나무 기반 모델들은 균형 트리 분할이지만 LGBM은 리프 중심 트리 분할방식이여 트리 모양이 비대칭인 특징을 가진다. 트리를 생성할 때 최대 손실값을 가진 리프 노드를 지속적으로 분할 하면서 깊은 트리를 만드는 구조이기에 적은 데이터 사용 시 과적합 발생이 쉽다는 단점이 있다.

Deep Neural Network(DNN)은 인공신경망 모델에서 은닉층을 늘려서 사용하는 모델로써, 은닉층이 2개 이상인 학습 모델이다. 딥러닝 중에 기초 모델로써 이를 응용한 여러 모델들이 파생되었다.

Long Short-Term Memory models(LSTM)은 순환신경망 모델로써 Recurrent Neural Networks(RNN)에서의 장기 메모리 문제를 해결하기 위해 만들었다. Gated Recurrent Units (GRU)는 LSTM에서 좀 더 간소화된 모델로 LSTM과 유사한 점이 있어서 두 모델의 성능을 비교한 연구들이 많다.

2.3 최적화 방법

현재 나오는 기계학습 모델 중에는 모델에 사용자가 직접 설정하는 변수를 지정할 수 있으며 이 개념을 하이퍼 파라미터라고 한다. 기계학습 모델 중에서 의사결정나무 모델 같은 경우 트리의 깊이와 리프 노드의 최대 개수등 이 있으며, 딥러닝 모델에서는 epoch 값이나 옵티마이저 설정 등이 있다.

이러한 하이퍼 파라미터를 이용한 최적화 탐색 방법으로는 직접 값을 지정하여 넣는 Manual Search 방법과 모델 하이퍼 파라미터에 넣을 수 있는 변수를 여러 개 집어넣어서 우수한 모델을 찾는 Grid serach 가 대표적이다. 반대로 말 그대로 하이퍼 파라미터를 랜덤하게 집어넣고 그중에서 우수한 값을 뽑아낼 수 있는 Random search 방법이 있으며, Grid Search 보다 좋은 성능을 도출했다[9]. 하지만 위의 방법들은 좋지 않을 확률이 상대적으로 높거나 성능을 확인하기 위한 시간이 너무 오래 걸린다.

그렇기에 우리는 새로운 해결 방안으로 주목되고 있는 Bayesian optimization을 적용하였다[10]. Bayesian optimization은 어느 입력값을 받는 미지의 목적 함수를 상정 후, 그 함숫값을 최대로 만드는 최적의 해를 찾는 것으로 목적을 두어 좋은 파라미터를 찾는 것이다. 현재 대표적인 툴로 Hyperopt가 있다[11].

III. 실험

우리는 머신러닝 모델에서는 KNN, SVM, Logistic regression과 의사결정나무 계열에서는 DecisionTree, RandomForest, LGBM, XGB 모델 그리고 딥러닝 계열의 모델로는 DNN, LSTM, GNU를 사용하였다. 사용한 데이터는 KDD-NSL, UNSW-NB15를 사용했으며 전처리 과정에서 공격 분류는 normal, unomal로 이진 분류하였고, 스트링 기반의 피처들은 라벨 인코딩을 사용하였다.

표 2. default 구성

Model	Configuration	
KNN	default	
SVM	default	
LR	default	
Model	Configuration	
DT	default	
RF	default	
XGB	default	
LGBM	default	
Model	Configuration	compile
DNN	Dense(f*4)-Dropout(0.2)- Dense(f*2)-Dropout(0.2)- Dense(f)-Dropout(0.2)- Dense(1)	optimizer=adam epoch=10
LSTM	LSTM(f)- Dense(1)	optimizer=adam epoch=10
GRU	GRU(f)- Dense(1)	optimizer=adam epoch=10

표 3. 하이퍼 파라미터 검색 조건

Model	Configuration	
KNN	n_neighbors=(1-20), weights=(uniform, distance)	
SVM	C=(0.001-100), max_iter=(50-500)	
LR	C=(0.001-100), max_iter=(50-500), solver =(newton-cg, lbfgs, liblinear, sag, saga)	
Model	Configuration	
DT	max_depth=(3-18), min_samples_leaf=(1-10) min_samples_split=(2-10), criterion=(gini, entropy)	
RF	max_depth=(3-18), min_samples_leaf=(1-10) min_samples_split=(2-10), criterion=(gini, entropy) n_estimators=(50-200)	
XGB	eta=(0.01-0.5), min_child_weight=(1-10), max_depth=(3-18), subsample=(0.1-1), gamma=(0.1-10)	
LGBM	learning_rate=(0.01-0.5), min_child_weight=(1-10), max_depth=(3-18), colsample_bytree=(0.1-1)	
Model	Configuration	compile
DNN	Dense(10-320)-Dropout(0.1-0.3)- Dense(10-320)-Dropout(0.1-0.3)- Dense(10-320)-Dropout(0.1-0.3)- Dense(1)	optimizer= (adam, nadam) epoch=(10-30)
LSTM	LSTM(10-320)- Dense(1)	optimizer= (adam, nadam) epoch=(10-30)
GRU	GRU(10-320)- Dense(1)	optimizer= (adam, nadam) epoch=(10-30)

우리는 모델의 default값과 Bayesian optimization를 사용한 모델의 성능

을 비교하기 위해 각 모델을 default와 optimization으로 구성하였다. 기계학습 모델과 의사결정나무 모델에서는 최소한의 하이퍼 파라미터를 적용한 default 모델과 Hyperopt 툴을 적용한 하이퍼 파라미터 구한 모델을 준비하였다. 딥러닝 default 모델은 표 2처럼 DNN은 3개의 은닉층에 각 피쳐의 개수의 4배, 2배, 1배수의 노드와 과접합 방지를 위해 Dropout비율을 0.2로 설정하였으며, LSTM, GRU에는 한 개의 층으로 노드 개수는 피쳐의 개수로 설정하였다. compile 부분에는 epoch는 10, optimzier은 adam으로 설정하였다. 딥러닝 optimization모델의 경우에는 표 3과 같이 각 노드 개수와 Dropout 함수의 범위 epoch 값과 optimizer을 adam, nadam 중에 선택하는 옵션을 Hyperopt 툴에 적용하여 찾은 하이퍼 파라미터를 적용했다.

이후 KNN, LR XGB, LGBM 같은 모델에서는 난수 생성기를 사용하지 않거나, 일정하게 값이 나오는 모델들은 1회, 나머지 모델들은 각 10회 실험 진행하여 최댓값, 최솟값, 평균을 구하였다.

표 4. KDD-NSL Accurancy

Model	default			optimization		
	max	min	mean	max	min	mean
KNN	0.7720			0.7681		
SVM	0.7690	0.6425	0.7124	0.7760	0.6372	0.7297
LR	0.7021			0.7025		
DT	0.7902	0.7667	0.7785	0.7918	0.7699	0.7773
RP	0.7898	0.7657	0.7721	0.7827	0.7607	0.7731
LGBM	0.7935			0.7931		
XGB	0.7940			0.7933		
DNN	0.8071	0.7332	0.7591	0.7716	0.7350	0.7493
LSTM	0.8111	0.6964	0.7680	0.8006	0.7567	0.7808
GRU	0.8007	0.7520	0.7745	0.8389	0.7776	0.8037

표 5. UNSW-NB15 Accurancy

Model	default			optimization		
	max	min	mean	max	min	mean
KNN	0.8099			0.7819		
SVM	0.8049	0.5577	0.6654	0.8398	0.4682	0.6995
LR	0.743466			0.7434		
DT	0.8968	0.8941	0.8954	0.9029	0.9014	0.9020
RP	0.9030	0.9012	0.9018	0.9037	0.9016	0.9025
LGBM	0.9091			0.9172		
XGB	0.9137			0.9089		
DNN	0.7998	0.7316	0.7741	0.8101	0.7268	0.7921
LSTM	0.9050	0.8648	0.8775	0.9015	0.8791	0.8897
GRU	0.8914	0.8730	0.8833	0.8983	0.8337	0.88255

실험 결과로 대체적으로 default 상태로 돌린것보다 optimization를 설정한 값이 대체적으로 높았다. 특히 딥러닝에서는 대부분의 모델들이 default 모델 보다 optimization 모델이 1-2프로 높은 성능을 보여줬다. 반면 성능에 큰 차이가 없는 모델들도 존재했다. 의사결정나무 계열의 모델들에도 다양한 하이퍼 파라미터가 존재하기에 이를 적용한 otimization 성능에서 딥러닝 모델만큼의 효율을 보여줄 것을 기대했으나, 성능 향상에 큰 영향을 주지 않았다.

IV. 결론

본 논문에서는 최적화를 적용하여 각 모델별로 성능을 측정하였다. 의사결정나무 모델이 자원 효율까지 고려할 경우 딥러닝 모델보다 우위에 있었지만, 딥러닝의 막대한 하이퍼 파라미터 적용 범위에서 일부만 수정하여도 성능 개선의 여지를 보여주었다. 현재 연구에서는 다른 연구보다 딥러닝 연구에 비해 간결하지만, 이러한 기본적인 모델에 하이퍼 파라미터 최적화를 적용하여 성능을 어느 정도 끌어낼 수 있는지 알아보았다.

ACKNOWLEDGMENT

본 연구는 과학기술정보통신부 및 정보통신기획평가원의 대학ICT연구센터육성지원사업의 연구결과로 수행되었음 (IITP-2021-2016-0-00314*)

참 고 문 헌

[1]Poulmanogo Illy, Georges Kaddoum et al., "Securing Fog-to-Things Environment Using Intrusion Detection System Based On Ensemble Learning," IEEE Wireless Communications and Networking Conference, 15- 18 April 2019

[2]Xianwei Gao, Chun Shan et al., "An Adaptive Ensemble Machine Learning Model for Intrusion Detection," IEEE Access, Volume 7, 82512 - 8252, June 2019.

[3]Yun-Gyung Cheong, Kinam Park, Hyunjoo Kim, Jonghyun Kim and angwon Hyun, "Machine learning based intrusion detection systems for class imbalanced datasets," Journal of Korea Institute of Information Security & Cryptology, vol. 27, no. 6, Dec. 2017

[4]Abhishek Divekar, Meet Parekh et al., "Benchmarking datasets for Anomaly-based Network Intrusion Detection: KDD CUP 99 alternatives," in Proc. of 2018 IEEE 3rd International Conference on Computing, mmunication and Security, pp. 1-8, Kathmandu, Nepal, Oct. 2018.

[5]Jong-Hwa Leew, Jiwon Bang et al., "Experimental Comparison of Network Intrusion Detection Models Solving Imbalanced Data Problem, " KNOM Review '20-02 Vol.23 No.02, 2020.

[6]Chuanlong Yin, Yuefei Zhu, Jinlong Fei and Xinzheng He, "A deep learning approach for intrusion detection using recurrent neural networks," Journal of IEEE Access, vol. 5, pp. 21954-21961, Oct. 2017.

[7]A. F. A. Khan, A. Gumaei and A.Hussain, "A novel two-stage deep learning model for efficient network intrusion detection," IEEE Access, vol. 7, pp. 30373 - 30385, 2019.

[8]Hebatallah Mostafa Anwer, Mohamed Farouk and Ayman A. Abdel-Hamid, "A framework for efficient network anomaly intrusion detection with features selection," in 2018 9th International Conference on Information and Communication Systems (ICICS), pp. 157 - 162, IEEE, 2018.

[9]J. Bergstra, and Y. Bengio, "Random search for hyper-parameter optimization," Journal of Machine Learning Research, Vol. 13, pp. 281-305, February 2012.

[10]J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," Advances in neural information processing systems, pp. 2951-2959, 2012.

[11]James Bergstra, Dan Yamins, David D. Cox, "Hyperopt: A Python Library for Optimizing the Hyperparameters of Machine Learning Algorithms, "PROC. OF THE 12th PYTHON IN SCIENCE CONF. (SCIPY 2013)

[12]Sayan Putatunda , Kiran Rama, "A Comparative Analysis of Hyperopt as Against Other Approaches for Hyper-Parameter Optimization of XGBoost," SPML '18: Proceedings of the 2018 International Conference on Signal Processing and Machine, pp 6 - 10, November 2018.