Network intrusion detection

CS221 Artificial Intelligence Project Progress Report
Amir Ziai

Background
In this project I will focus on the task of network intrusion detection. Network operators are generally aware of common attack vectors that they defend against. For most networks the vast majority of traffic is legitimate. However new attack vectors are continually designed and attempted by bad actors which bypass detection and go unnoticed due to low volume. One strategy for finding such activity is to look for anomalous behavior.

I will start with exploring unsupervised machine learning methods for detecting anomalous behavior. I will then explore the possibility of improving on this methodology by incorporating a small set of labeled data. Labeling network activity can be very time and resource intensive. In many cases security analysts need to investigate anomalous behavior for days or even weeks and this process is not very scalable. For the final part of the project I will explore active learning to guide the analyst in focusing on the most uncertain data points in an attempt to improve the sampling efficiency.

Dataset
I will use the KDD Cup 1999 dataset with 4.8M network connections in a military network environment. Each record is labeled as either “normal” or one of 22 different types of intrusion. Examples of these intrusions include smurf, IP sweep, and teardrop. Each record has 41 features including duration, protocol, and number of bytes exchanged between the source and destination.

I will focus on a single attack type of IP sweep. The resulting dataset has 98,525 rows and represents a binary classification task. I will extend the experiments that I have conducted for this report to more attack types and consider other network security datasets in order to provide a more comprehensive benchmark for the proposed algorithm.

Input and output example
The following table depicts 3 rows of data (excluding the label):

duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...	dst_host_srv_count
0	tcp	http	SF	181	5450	0	0	0	0	...	9
0	tcp	http	SF	239	486	0	0	0	0	...	19
0	tcp	http	SF	235	1337	0	0	0	0	...	29

The objective of the detection system is to label each row as either “normal” or “anomalous” using an unsupervised machine learning approach.

Features and experimental setup

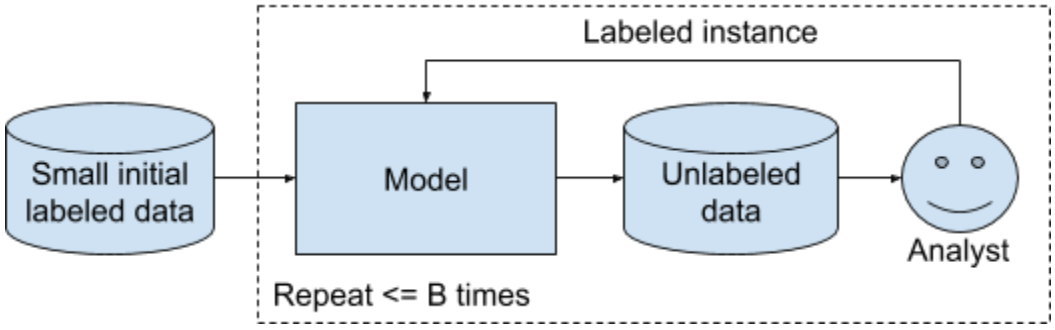
There are four categorical variables in this dataset that I have encoded using one-hot encoding. There are no missing values so imputation was not necessary. The resulting dataset consists of 82 features. The majority of the datasets (98.73%) are labeled normal and only 1.26% represent IP sweep (anomalous). After generating these features I partition the dataset into train (80%), validation (10%), and test (10%) sets. Models are evaluated on the validation set and the best model is evaluated against the test set to provide an unbiased estimate of the generalization performance.

Evaluation metric

Since labeled data is very hard to come by in this space I have decided to treat this problem as a semi-supervised learning one. Therefore the machine learning model receives a subset of the labeled data. I will use the F1 score to capture the trade-off between precision and recall. A model that is highly precise (does not produce false positives) is desirable as it won’t waste the analyst’s time but this usually comes at the cost of being too conservative and not catching anomalous activity that is indeed an intrusion.

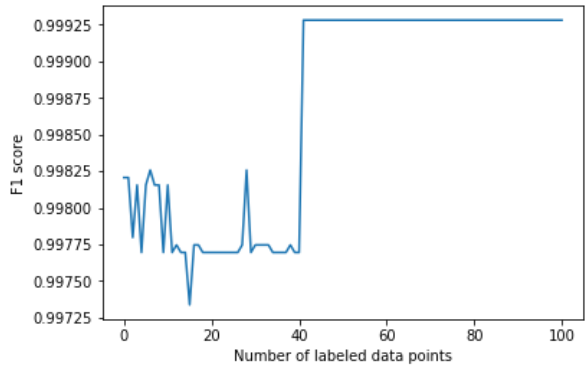
Algorithm

The proposed algorithm starts with training a supervised model on a small subset of the data. My initial assessment suggests that even a small dataset of labeled data points is superior to a purely unsupervised approach for this problem. Once the model is trained we can score the remainder of the data points in the training set and rank order them by uncertainty. The challenge with this part is to ensure that the the model is seeing both positive and negative examples. I’ll assume that we have a small database of these data points that we can tap into. I will then query the analyst for the most uncertain data point for collecting a label and retrain the model. I will repeat this process until I reach the desired F1 score on the validation set or I exhaust my budget B (maximum number of queries we can make to the analyst). The labeled data will be removed from the pool of unlabeled data and scoring will not happen on the next round on labeled data points.

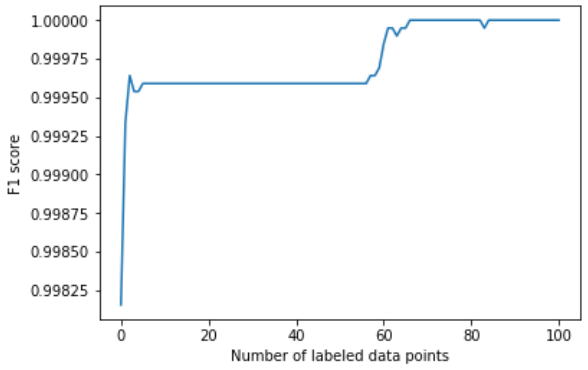


On the first run I'll present the 100 data points to my model for training. This model generates an F1 score of 0.9954 (45 false positives, 0 false negatives) which is better than random guessing or a majority classifier. Then I'll score all unlabeled data points and take the data point corresponding to the highest value of 1 - P(y=anomaly | x). Ties are broken randomly.

The following figure compares the F1 score against number of labeled data points:



(a) Random sampling from the pool



(b) Taking the most uncertain instance

The graph on the left randomly selects the next instance to train on whereas the one on the right follows the proposed methodology and selects the most uncertain instance. It's clear, at least in this limited example, that selection by uncertainty wins out.

Oracle and baseline

Due to the imbalanced nature of the dataset a classifier that always predicted the outcome to be "normal" would have a very high F1 score. On the development set this translate to an F1 score of 0.9942. I also trained an Isolation Forest [1] unsupervised algorithm using scikit-learn which resulted in an F1 score of 0.9901. In other words a purely unsupervised approach is worse than always predicting normal.

Next I used a random forest classifier (1,000 estimators) with a random sample of 100 data points to establish the baseline. I repeated training on a randomly chosen set of 100 data points 30 times and then subsequently scored on the validation set. The average F1 score in this scenario is 0.9982 which is better than the majority classifier. I will use this value as my benchmark. Since these numbers are very close to 1 I will also report the number of mismatches for added intuition. In this example the best model across the 30 runs yields only 7 false positives and no false negatives. In contrast the majority classifier would generate 113 false positives. Imagine that the cost of investigating a network intrusion is $10,000 and therefore we have just saved $1,060,000 by using a more accurate model.

For the oracle I trained a random forest classifier on the entire dataset. This model generates an F1 score of 0.9999 and only a single false positive on the development/validation set. This translates to a savings of $60,000.

As I mentioned earlier label collection can be expensive. Let's assume that each label costs $100 to acquire and therefore we have spent $10,000 to acquire 100 random data points for the baseline. The question now is how much of the $60,000 can we capture with the minimum possible number of labels to acquire which will eat into that amount. We have to pick the data points for labeling judiciously.

Experiments

The following table depicts the results of some preliminary experimentation for a few different algorithms:

Algorithm	**Budget % used**	**Query strategy**	**False positives**	**F1 score**	**Improvement over baseline**
Random forest	100%	Uncertainty	0	1.0000	0.58%
Random forest	100%	Random	14	0.9993	0.51%
Random forest	0%	-	36	0.9981	0.39%
Logistic regression	0%	-	66	0.9890	-0.52%
Logistic regression	100%	Uncertainty	12	0.9974	0.32%

The results suggest that for this problem we can drive the number of false positives to zero after 60 (see charter from Algorithm section) queries with uncertainty ranking. With further hyper-parameter tuning and exploration of other query selection strategy I may be able to lower this value.

Next steps

I want to expand this algorithm to incorporate more sampling strategies for selecting the next data point for labeling. Examples of these strategies include using margin and entropy. Also I want to explore methods such as label spreading and propagation that act in a more semi-supervised manner where the unlabeled data is used to create a create an affinity matrix on the normalized graph Laplacian. In other words the unlabeled data is used explicitly and not only for selecting the next instance to query. I want to explore more security-related datasets to present a comprehensive assessment of these methods. Finally I will explore RNNs for feeding log-level data and dispensing with feature engineering.

References

1- Liu, Fei Tony, Kai Ming Ting, and Zhi-Hua Zhou. "Isolation-based anomaly detection." ACM Transactions on Knowledge Discovery from Data (TKDD) 6.1 (2012): 3.