Network Intrusion Detection

Amir Ziai (amirziai@stanford.edu)

Motivation

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.

Problem

Investigating anomalous behavior requires significant **time and resources**. Collecting a large amount of **labeled examples** for training supervised models is both prohibitively **expensive** and subject to obsoletion as new attacks surface. A purely unsupervised methodology is ideal, however research has shown that even a very small amount of labeled examples can significantly improve the quality of anomaly detection. A methodology that **minimizes** the **number of required labels** while **maximizing** the **quality of detection** is desirable. False positives in this context result in wasted effort or blockage of legitimate traffic and false negatives translate to undetected attacks.

Dataset

I have used the **KDD Cup 1999** dataset which consists of about **500k records** representing network connections in a military environment. Each record is either "normal" or one of 22 different types of intrusion such as smurf, IP sweep, and teardrop. Out of these 22 only 10 have at least 100 occurrences, the rest were removed for this project. Each record has **41 features** including duration, protocol, and bytes exchanged. Prevalence of attack types varies substantially with **smurf** being the most pervasive at about **50%** of total records and **Nmap** at less than **0.01%** of total records.

Approach

Input and output

The objective is to **classify** each network connection as either "normal" or "attack". A snippet of a sample record looks as follows:

```
{
   "protocol_type": "tcp",
   "service": "http",
   "src_bytes": 181,
   ...
}
```

Evaluation metric

The **F1 score** is used for quantifying the quality of detection which captures the trade-off between precision and recall.

Baseline

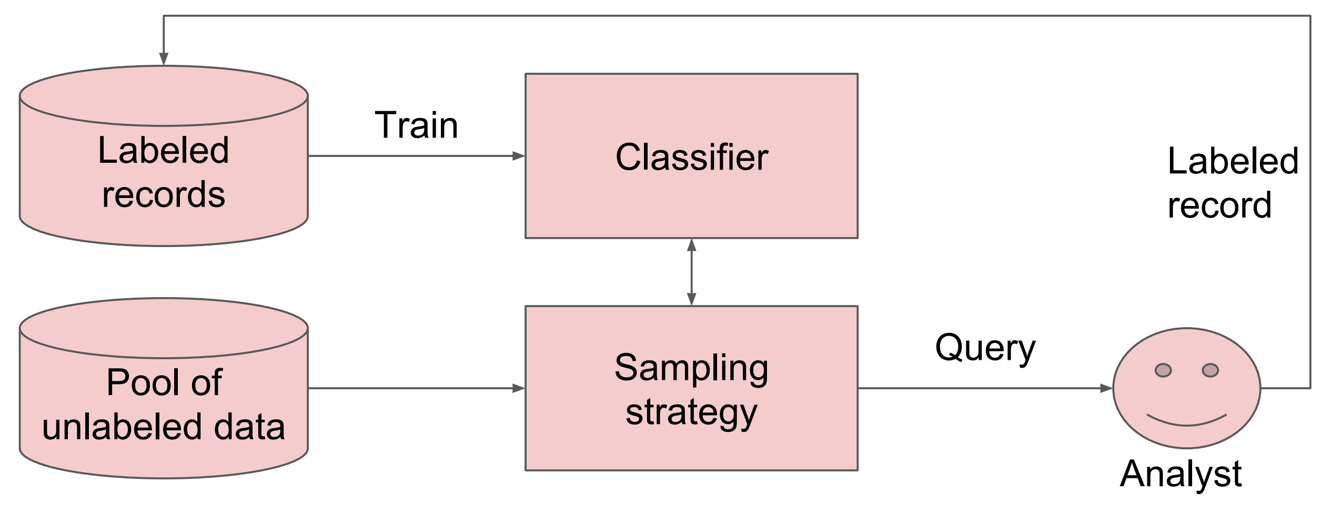
I trained an **isolation forest** (unsupervised) for each attack type and achieved an F1 score of **0.34±0.26**.

Oracle

I used the entire dataset for each attack type to train a random forest classifier with 100 estimators which achieved an F1 score of **1.00±0.01**. This is however prohibitively expensive (average of 130k training samples).

Analyst-in-the-loop Machine Learning

The proposed approach starts with training a classifier on a small random subset of the data (i.e. 1,000 samples) and then continually queries a security analyst for the next record to label. There's a maximum budget of 100 queries.



Active Learning

Processing pipeline

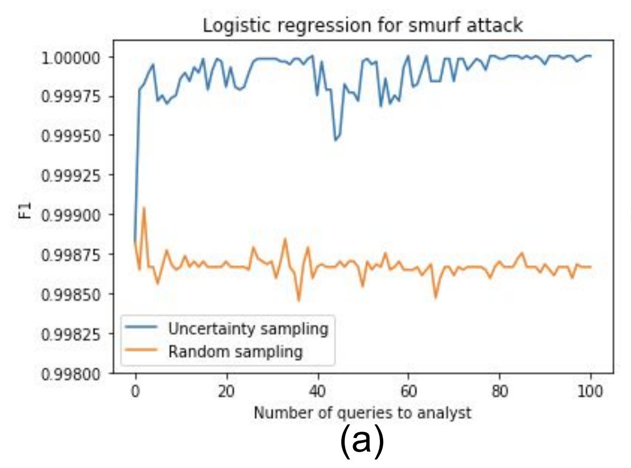
I generated 10 separate datasets consisting of normal traffic and each of the attack vectors. This way we can study the proposed algorithm over 10 different attack vectors with varying prevalence and ease of detection. Each dataset is then **split** into train, development, and test partitions with **80%**, **10%**, and **10%** proportions. All algorithms are trained on the train set and evaluated on the development set. The winning strategy is tested on the test set to generate an unbiased estimate of generalization. Categorical features are one-hot encoded and missing values are filled with zero.

Comparison of learners and sampling strategies

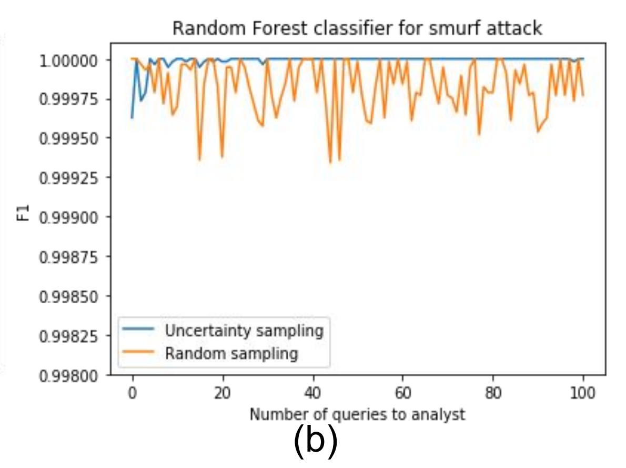
The following table depicts the results for using Logistic Regression (LR) and Random Forest (RF) with either random sampling, uncertainty sampling (sort by classifier score and take the least certain), or entropy sampling (highest entropy among scores):

Algorithm	Sampling strategy	F1 initial	F1 after 10	F1 after 50	F1 after 100	Train time (s)	Query time (s)
LR	Random		0.76±0.32	0.79±0.31	0.86±0.17		0.09±0.08
LR	Uncertainty	0.76±0.32	0.83±0.26	0.85±0.31	0.88±0.20	**0.05±0.01**	0.1±0.08
LR	Entropy		0.83±0.26	0.85±0.31	0.88±0.20		**0.08±0.08**
RF	Random		0.91±0.12	0.84±0.31	0.95±0.07		0.09±0.07
RF	Uncertainty	**0.90±0.14**	0.98±0.03	**0.99±0.03**	**0.99±0.03**	0.11±0.00	0.16±0.06
RF	Entropy		**0.98±0.04**	0.98±0.03	**0.99±0.03**		0.12±0.08

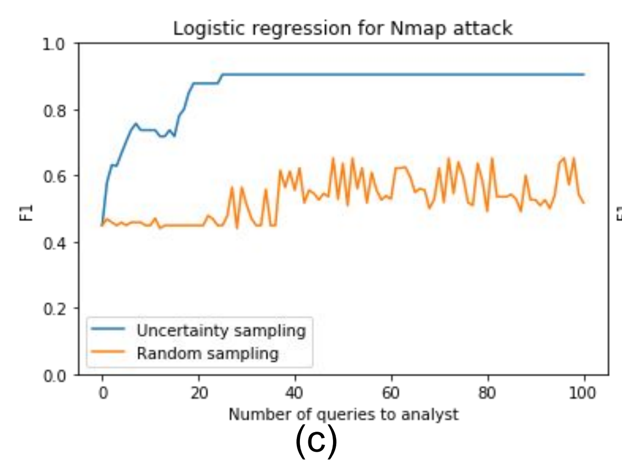
Random forest has the higher average initial F1 score at the cost of higher train and query time. Both entropy and uncertainty sampling significantly outperform random sampling which underlines the importance of **judicious sampling**. The following figures (a) and (b) illustrate how quickly uncertainty sampling can reach high quality results for the smurf attack with high prevalence as well as Nmap with much lower prevalence in (c) and (d).



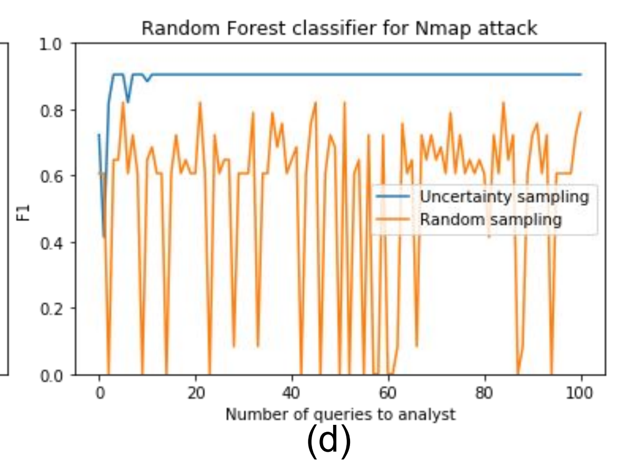
(a)



(b)



(c)



(d)

Future work

Using an **ensemble** analyst-in-the-loop system that combines supervised, unsupervised, and semi-supervised methods for security threat detection is an unexplored area which I am planning to pursue next. Also to make this analysis more comprehensive I plan on including more security datasets in the benchmark suite, more supervised learners (including RNNs), and semi-supervised methods such as label propagation and GANs.