機械学習を用いた不正アクセス検知

大阪府立岸和田高等学校
梶原　大進

要旨

今日、あまり情報関連の話題に関心がない人々の耳にも『サイバー攻撃』や『サイバーテロ』、『情報漏えい』などの言葉が入るようになってきている。日々増加、進化を遂げるサイバー攻撃に対応し、大切な情報を守り抜くためには既存の攻撃だけでなく未知の攻撃に対する迅速な対応が必要となる。

そこで本稿では既存の攻撃だけでなく、新しい攻撃にも対応するために機械学習(機械学習については後述)により攻撃手法を学習した分類器を用いることにした。作成した分類器を用いた結果、72%という確率で未知の攻撃に対応することができた。この数字はよい結果とはけして言えないが、本実験を通じて機械学習の手法を学び、データセットの大切さを実感することができ、また様々な今後の課題も見つかった。

機械学習

機械学習とは大量のデータをもとにデータを分類したり、結果を予測したりする技術。大別すると、教師あり学習、教師なし学習、がある。教師あり学習とはあらかじめ入力データと出力データをセットであたえ、正しい出力ができるように学習する手法である。教師なし学習は出力データのみを与え、そこからデータ同士の関係を見つけ出す手法である。本稿では教師あり学習を用いた。

また機械学習には大別すると分類、会期、クラスタリング、次元削減の手法がある。本稿では分類に区分けされるサポートベクターマシンを用いることにした。サポートベクターマシンとは学習用のデータから各データ点との距離が最大になるように空間を区切る超平面を見つけデータを分類するという手法である。この手法を用いた理由としては、サポートベクターマシンは非常に高い精度で分類を行えることで知られているからである。

1. 序論

近年、新型のコンピュータウイルスや不正アクセスの増加に伴い、不正侵入検知システム(以下IDS)や不正侵入防御システム(以下 IPS)の開発が盛んにおこなわれている。基本的な仕組みとしては、あらかじめ既知の攻撃パターンが登録されているシグネチャを持っており、このシグネチャと HTTP リクエストを照らし合わせることで侵入の検知、防御を行う。しかし、IDS や IPS ではシグネチャに記述されている、既知の手法を用いた攻撃しか検知することができない。そこで本稿では未知の攻撃にも対応するために、不正アクセス時のパケットデータを学習させた機械学習の分類器を用いて不正アクセスの検出を行えないかを検討することにした。そこで我々は機械学習ライブラリである scikit-learn を用いて分類器を作成し、自作したデータセットを学習させて、未知の攻撃に対応できるかどうか実験を行った。

2. 方法

2.1. 環境

今回用いた環境、ソフトウェアを表 1 に示す。

表 1

OS	Windows10(64bit) Kalilinux2.0(攻撃用) CentOs7.0(被攻撃用)

CPU	Intel(R) Core(TM)i7-2670QM CPU @2.20GHz
メモリ	8GB
利用ソフト	Python3.4 VMWare Wireshark armitage
利用 python ライブラリ	Scikit-learn0.17

なお、kalilinux2.0、centos7.0 は 1 台のコンピュータ上で複数の OS を作動させることができる仮想化ソフトウェアの一つである VMWare を用いて 1 台のコンピュータで動作させた。

2.2. データセットの作成

様々な web サーバに対して通常の HTTP アクセスが発生するネットワーク環境において正常な HTTP リクエストを抽出する。次にいくつかの攻撃パターンで web サーバに対して攻撃を行い、その時の HTTP リクエストを抽出した。そして 2 つのデータを 1 つの csv ファイルとしてまとめた。なお、パケットの抽出にはネットワーク上を流れるパケットの取得、解析が行えるソフトウェアである wire shark を用いた。また抽出した特徴量の一覧を表 2 に、データセットの一部を表 3 に示す。なお抽出する特徴量は下記サイトを参考にした。

もりたこ@mrtc0「機械学習入門以前」(2014) <http://www.slideshare.net/mrtc0/machine-learning-44100565>(参照 2017-6-27)

表 2

パケット長
送信元ポート番号
送信先ポート番号
User Agent
Content Type
Content Length
Method
Payload

表 3

Length	Source Port	Destination Port	Request Method	…
559	80	45687	-	…
238	58405	80	GET	…
164	80	58405	-	…
602	47016	80	POST	…
164	80	47016	-	…
・ ・ ・	・ ・ ・	・ ・ ・	・ ・ ・	・ ・ ・

2.3. データセットの調整

表 2 を見てもわかる通り、今回集めたデータは各特徴量の基準がバラバラである。そこでデータの標準化を行う。本稿では各特徴量の平均を 0 に分散を 1 になるように調整した。また、データの特徴量が多すぎると未知のデータに対する汎用性が低下してしまう。そこで攻撃の検知と関連のある特徴量を見つけるために主成分分析を行い、あまり重要でない特徴量を次元削減を行うことによって省いた。

Scikitlearn0.17 では標準化は StandScaler 関数を、次元削減には PCA 関数を用いた。コードは図 1 に示す。

```
# データの標準化
scaler = preprocessing.StandardScaler()
scaler.fit(features)
features = scaler.transform(features)

# データの次元削減
pca = decomposition.PCA(n_components = 2)
pca.fit(features)
features = pca.transform(features)
```

図 1

2.4. 学習

2.4.1. グリッドサーチ

前述したとおり、学習にはサポートベクターマシン（以下 SVM）を用いた。SVM では分類器を

作成するにあたって許容できる誤検知の量や利用するカーネル関数などいくつかのパラメータを設定しなければならない。そこで最適なパラメータを見つけるためにグリッドサーチを行った。グリッドサーチとは、機械学習の分類器のパラメータの最適値を与えられた探索範囲内から自動で探す方法。本稿では scikitlearn0.17 の GridSearchSV を用い、探索範囲は C：1~10, gamma：1~0.0001 とした。

2.4.2. k-交差検定法

グリッドサーチでのパラメータの評価には、分類器がどれだけ未知のデータに対応できるかを検証するために k-交差検定法を用いた。K-交差検定法とは、学習に用いるデータを K 個に分割し、そのうちの 1 つをテストデータに、残りの K－1 個のデータを学習データとして学習と評価を行う。これを K 回繰り返し、それぞれの正解率の平均をとり分類器の評価を行うという手法。本稿では scikilearn0.17 の StratifieldKFold を利用した。

2.4.4. svm の適用

scikitlearn0.17 の SVC 関数を用いて学習を行う。カーネル関数は RFB カーネルを用いた。パラメータ C とγはの GridSearchCV でグリッドサーチを行い最適な値を求める。結果の評価には StratifieldKFold を用いて交差検定を適用する。コードは最終頁図 2 に示す。

3. 結果

グリッドサーチで最も正解率が高かったパラメータの値とその時の結果を表 4 に示す。

表 4

パラメータ	C = 7, σ = 1
正解率	0.975054229935

4. 実験

2 で作成した分類器を用いて学習に使用したデータセットに含まれていない攻撃手法で攻撃を行った。不正アクセス時のパケット 604 件、通常時のパケット 876 件の計 1,480 件中、72%が正しく分類された。その結果を表 5 に示す。なお評価指数には正解率、適合率、再現率、F 値を用いた。

正解率

予測した値で正解のものの割合。これが高いだけでは、けして精度が高いとは言えないので、ほかの指標とあわせて使われることが多。

適合率

正だと判断されたもののうち実際に正解だったものの割合。実際は負であるものを正であると予測される割合を低く抑えたいときに用いる。

再現率

実際に正であるもののうち正だと予測されたものの割合。実際は正であるものを負であると予測される割合を低く抑えたいときに用いる。

F 値

適合率と再現率は一方が高くなると他方が低くなるというトレードオフの関係にあるので、この 2 つの指標の調和平均を取ったもの。

表 5

正解率	適合率	再現率	F 値
0.7236	0.72	0.72	0.72

5. 考察

結果より、今回作成した分類器は正解率が 72%と低いだけでなく、分類の内訳をみてみると、誤検知率もやや高かった。そのため、非常に精度が低く、あまり実用的ではない。

その理由として、どの特徴量を抽出するのかを十分に考えていないことや、通常の HTTP アクセス時のパケットデータが少ないことなどがあげられる。また機械学習に対する理解度も低いこともあげられる。

SVM のより最適なパラメータが、グリッドサ

ーチを行った範囲外に存在した可能性もある。さらに SVM 以外の手法の方が今回の場合はより高い精度で分類できたかもしれない。

また、機械学習を用いて解決したい問題によって最適な評価指数は異なる。そのため、本稿ではどのような評価指数が最適であるのかも考える必要があった。

6. 結論

本稿では不正アクセスの検知に機械学習によって攻撃手法を学習させた分類器を用いることで未知の攻撃に対応し、その検知率を上げることを目標とした。作成した分類器はけして精度が高いとは言えなかったが、機械学習を用いることで未知の攻撃にもある程度対応できることがわかった。また、機械学習におけるデータセットの重要性も実感することができた。

今後の課題としては学習データの特徴量の見直し、通常時のパケットデータを増やすなどの学習データの質の向上、より広い範囲でグリッドサーチを行う、SVM 以外の分類方法も用いるなどの最適な分類器・パラメータの追求、さらに、機械学習、HTTP 通信に対する更なる理解などがあげられる。

参考文献

・Willi Richert・Luis Pedro Coelho(2014)『実践 機械学習システム』斎藤康毅訳, オライリージャパン

・もりたこ@mrtc0「機械学習入門以前」(2014), <http://www.slideshare.net/mrtc0/machine-learning-44100565>(参照 2017-6-27)

・大曽根圭輔「いまさら聞けない機械学習の評価指数」(2016),<http://data.gunosy.io/entry/2016/08/05/115345>(参照 2017-6-27)

・「scikit-learn User Guide」,<http://scikit-learn.org/0.17/user_guide.html>(参照 2017-6-27)

```
#  Created by 梶原大進 on 2017/03/28.
#  Copyright © 2017年 梶原大進. All rights reserved.
#


import numpy as np
import pandas as pd
from sklearn import svm, cross_validation, decomposition, preprocessing
from sklearn.grid_search import GridSearchCV

np.set_printoptions(threshold = np.inf)

data = pd.read_csv("ronbun-v4.csv", header=None)
flags = data[1]
features = data[[2,3,4,5,6,7,8,9]]

scaler = preprocessing.StandardScaler()
scaler.fit(features)
features = scaler.transform(features)

pca = decomposition.PCA(n_components = 2)
pca.fit(features)
features = pca.transform(features)

param_grid = {"C":[1,2,3,4,5,6,7,8,9,10], "gamma":[1,0.1,0.01,0.001,0.0001], "kernel":["rbf"]}
svm = svm.SVC()
cv = cross_validation.StratifiedKFold(flags, n_folds = 5, shuffle = True)
clf = GridSearchCV(svm, param_grid, cv = cv)
clf.fit(features, flags)
print(clf.grid_scores_)
print(clf.best_estimator_)
```

図 2