CSC 180-01 Intelligent Systems (Fall 2019)

Mini-Project 2: Network Intrusion Detection

Due at 11 am, Friday, October 11, 2019

Demo Session: class time, Friday, October 11, 2019

1. Problem Formulation

Software to detect network intrusions protects a computer network from unauthorized users, including perhaps insiders. This project aims to build a **network intrusion detector**, a predictive model capable of distinguishing between bad connections, called intrusions or attacks, and good normal connections.

Model this problem as a BINARY classification problem. Use the following models to detect bad connections (intrusions). Compare the accuracy, recall, precision and F1-score of the following models. PLOT the confusion matrix for each model.

- Fully-Connected Neural Networks
- Convolutional Neural Networks (CNN)

For data preprocessing, we need to encode good connections as "0" and bad connections as "1". To achieve this, you may want to apply some functions to the label column. Check this out if you need hints:

https://chrisalbon.com/python/data_wrangling/pandas_apply_operations_to_dataframes/

Hint: For CNN, find a way to view each sample data as an image. Please refer to our lab tutorial on using CNN to handle any data other than images. You may use either Conv2D or Conv1D.

2. Dataset

Download link:
https://drive.google.com/open?id=1FKyIqsKP4NBOTKRRuVbJjEFxTTCwzlHy

This database contains a wide variety of intrusions simulated in a military network environment. A connection is a sequence of TCP packets starting and ending at some well-defined times, between which data flows to and from a source IP address to a target IP address under some well-defined protocol. **Each connection is labeled as either normal, or as one specific attack type.**

You can also find the data here: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html**.** We use the 10% subset in this project.

The datasets contain a "normal" label and a total of 22 attack types, which are as follows:

back,buffer_overflow,ftp_write,guess_passwd,imap,ipsweep,land,loadmodule,multihop,neptune,nmap,normal,perl,phf,pod,portsweep,rootkit,satan,smurf,spy,teardrop,warezclient,warezmaster.

Type of each feature in the dataset is as follows (see appendix A for details):

duration: continuous.
protocol_type: symbolic.
service: symbolic.
flag: symbolic.
src_bytes: continuous.
dst_bytes: continuous.
land: symbolic.
wrong_fragment: continuous.
urgent: continuous.
hot: continuous.
num_failed_logins: continuous.
logged_in: symbolic.
num_compromised: continuous.
root_shell: continuous.
su_attempted: continuous.
num_root: continuous.
num_file_creations: continuous.
num_shells: continuous.
num_access_files: continuous.
num_outbound_cmds: continuous.
is_host_login: symbolic.
is_guest_login: symbolic.
count: continuous.
srv_count: continuous.
serror_rate: continuous.
srv_serror_rate: continuous.
rerror_rate: continuous.
srv_rerror_rate: continuous.
same_srv_rate: continuous.
diff_srv_rate: continuous.
srv_diff_host_rate: continuous.
dst_host_count: continuous.

dst_host_srv_count: continuous.
dst_host_same_srv_rate: continuous.
dst_host_diff_srv_rate: continuous.
dst_host_same_src_port_rate: continuous.
dst_host_srv_diff_host_rate: continuous.
dst_host_serror_rate: continuous.
dst_host_srv_serror_rate: continuous.
dst_host_rerror_rate: continuous.
dst_host_srv_rerror_rate: continuous

3. Requirements

- Split data for training and test. Use training data to train your models and evaluate the model quality using test data
- Drop all the redundant records. This data set has **a big number of redundant records**. Redundant records in the train set will cause learning algorithms to be biased towards the more frequent records.
- Encode categorical features and normalize numeric features.
- You must use EarlyStopping when training neural networks using Tensorflow.
- Tuning the following hyperparameters when training neural networks using Tensorflow to see how they affect performance
    - **Activation:** relu, sigmoid, tanh
    - **Layers and neuron counts**
    - **Optimizer:** adam and sgd
    - **Kernel number and kernel size** (for CNN only)

4. Grading breakdown

You may feel this project is described with some certain degree of vagueness, which is left on purpose. In other words, **creativity is strongly encouraged**. Your grade for this project will be based on the soundness of your design, the novelty of your work, and the effort you put into the project.

Use the evaluation form on Canvas as a checklist to make sure your work meet all the requirements.

Implementation	**70 pts**
Your report	**20 pts**
In-class defense	**10 pts**

5. Teaming:

Students must work in teams with no more than 3 people. Think clearly about who will do what on the project. Normally people in the same group will receive the same grade. However, the instructor reserve the right to assign different grades to team members depending on their contributions. So you should choose partner carefully!

6. Deliverables:

(1) All your source code in Python Jupyter notebook.

(2) Your report in PDF format, with your name, your id, course title, assignment id, and due date on the first page. As for length, I would expect a report with more than one page. Your report should include the following sections (but not limited to):

- **Problem Statement**
- **Methodology**
- **Experimental Results and Analysis**
- **Task Division and Project Reflection**

In the section "**Task Division and Project Reflection**", describe the following:

- who is responsible for which part,
- challenges your group encountered and how you solved them
- and what you have learned from the project as a team.

10 pts will be deducted for missing the section of task division and project reflection.

All the files must be submitted **by team leader** on Canvas before

11 am, Friday, October 11, 2019

NO late submissions will be accepted.

7. In-class Demo:

Each team member must demo your work during the scheduled demo session. Each team have **three minutes** to demo your work in class. Failure to show up in defense session will result in **zero** point for the project. The following is how you should allocate your time:

- Model/code design (1 minute)
- Findings/results (1 minute)
- Task division, challenges encountered, and what you learned from the project (1 minutes)

8. Hints

- The provided CSV file has no column headers. So you may want to add column names using the following code after you load data into dataframe using pd.read_csv():

```
df.columns = [
    'duration',
    'protocol_type',
    'service',
    'flag',
    'src_bytes',
    'dst_bytes',
    'land',
    'wrong_fragment',
    'urgent',
    'hot',
    'num_failed_logins',
    'logged_in',
    'num_compromised',
    'root_shell',
    'su_attempted',
    'num_root',
    'num_file_creations',
    'num_shells',
    'num_access_files',
    'num_outbound_cmds',
    'is_host_login',
    'is_guest_login',
    'count',
    'srv_count',
    'serror_rate',
    'srv_serror_rate',
    'rerror_rate',
    'srv_rerror_rate',
```

```
    'same_srv_rate',
    'diff_srv_rate',
    'srv_diff_host_rate',
    'dst_host_count',
    'dst_host_srv_count',
    'dst_host_same_srv_rate',
    'dst_host_diff_srv_rate',
    'dst_host_same_src_port_rate',
    'dst_host_srv_diff_host_rate',
    'dst_host_serror_rate',
    'dst_host_srv_serror_rate',
    'dst_host_rerror_rate',
    'dst_host_srv_rerror_rate',
    'outcome'
]
```

9. Think beyond the Project

- Can you model this intrusion detection problem as **a multi-class classification problem** so that we can detect **the specific attack type for each connection**? How good the predictive model can be in this case?

- Among all the features, can you identify the important features (this is so called feature importance analysis) and train models only on those important features, e.g., top-10 features? What would be the benefits to do that?

- Can you create a more balanced dataset to train your model so that you model will not be biased to the more frequent classes/attach types?

- Read this published paper based on the exact dataset used in the project.

  https://www.ee.ryerson.ca/~bagheri/papers/cisda.pdf

- A grander challenge dataset for you to play with about IoT applications

  https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/bot_iot.php

Appendix A

A complete description of the features is given in the three tables below.

feature name	*description*	*type*
duration	length (number of seconds) of the connection	continuous
protocol_type	type of the protocol, e.g. tcp, udp, etc.	discrete
service	network service on the destination, e.g., http, telnet, etc.	discrete
src_bytes	number of data bytes from source to destination	continuous
dst_bytes	number of data bytes from destination to source	continuous
flag	normal or error status of the connection	discrete
land	1 if connection is from/to the same host/port; 0 otherwise	discrete
wrong_fragment	number of ``wrong'' fragments	continuous
urgent	number of urgent packets	continuous

Table 1: Basic features of individual TCP connections.

feature name	*description*	*type*
hot	number of ``hot'' indicators	continuous
num_failed_logins	number of failed login attempts	continuous
logged_in	1 if successfully logged in; 0 otherwise	discrete
num_compromised	number of ``compromised'' conditions	continuous
root_shell	1 if root shell is obtained; 0 otherwise	discrete
su_attempted	1 if ``su root'' command attempted; 0 otherwise	discrete
num_root	number of ``root'' accesses	continuous
num_file_creations	number of file creation operations	continuous
num_shells	number of shell prompts	continuous
num_access_files	number of operations on access control files	continuous
num_outbound_cmds	number of outbound commands in an ftp session	continuous
is_hot_login	1 if the login belongs to the ``hot'' list; 0 otherwise	discrete
is_guest_login	1 if the login is a ``guest''login; 0 otherwise	discrete

Table 2: Content features within a connection suggested by domain knowledge.

feature name	*description*	*type*
count	number of connections to the same host as the current connection in the past two seconds	continuous
	Note: The following features refer to these same-host connections.	
serror_rate	% of connections that have ``SYN'' errors	continuous
rerror_rate	% of connections that have ``REJ'' errors	continuous
same_srv_rate	% of connections to the same service	continuous
diff_srv_rate	% of connections to different services	continuous
srv_count	number of connections to the same service as the current connection in the past two seconds	continuous
	Note: The following features refer to these same-service connections.	
srv_serror_rate	% of connections that have ``SYN'' errors	continuous
srv_rerror_rate	% of connections that have ``REJ'' errors	continuous
srv_diff_host_rate	% of connections to different hosts	continuous

Table 3: Traffic features computed using a two-second time window.