Cyber attacs recognition with PNN

Daniel Paliura

09 01 2021

Introduction

Cyber security is much more important nowadays, as we live in digital age. Different intrusions can take place on each computer. It can bring slight or harmful effects. Some defending program have to recognize specific attack action to protect machine. Data set **NLS-KDD** contains such connection signatures with expertly known classes of actions, whether it normal connection or specified attack.

Purpose

To create PNN classifier based on KDD set, that available to recognize intrusions with same signature. To analyze PNN result and offer probable model improvements.

Data set

Short description of NLS-KDD. What is KDD?

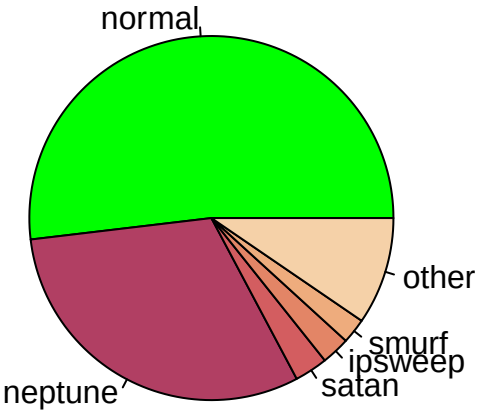
NLS-KDD contains 41 features for connection signature and one feature that describes expertly known class of connection. Class is name of cyber attack if connection is intrusion and "normal" if connection is safe. NLS-KDD divided into train-test subsets.

```
## Whole NLS-KDD has 148516 records.
## Train set - 125973(84.8211640496647%)records.
## Test set - 22543(15.1788359503353%)records.
```

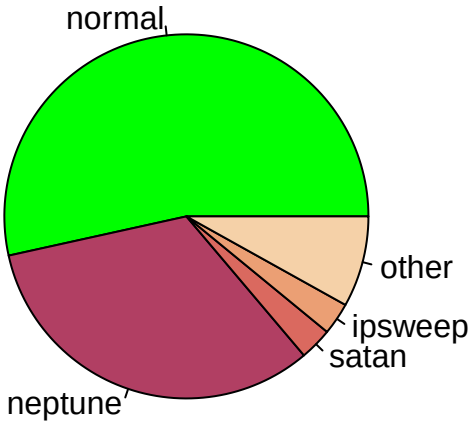
Summary of KDD in Addition 1:

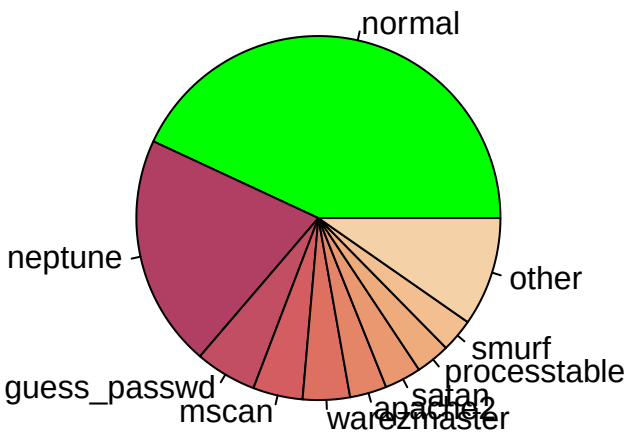
Pie charts describe how classes distributed in KDD, train and test sets:

Connection types in hole NLS KDD



Connection types in train set





About model

I decided to try use probability neural network (PNN) as classifier for attacks recognition goal. PNN trains with teacher and returns name of the most probable class for according connection signature. It has one disadvantage: each trained example is associated with one neuron in it's pattern layer, so PNN is unstable for large data sets.

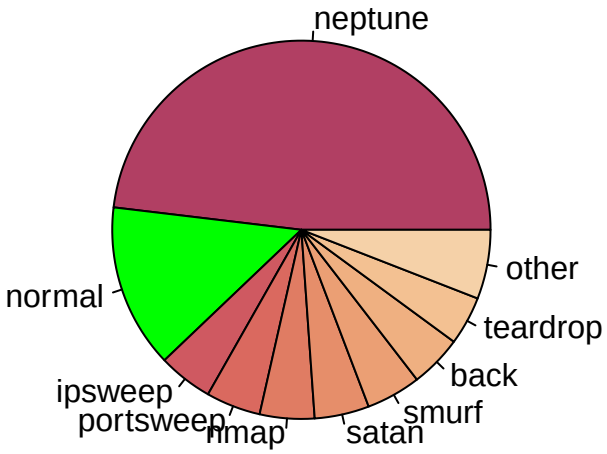
As implementation of PNN I used my model from other project. It was written on **Python 3** language with library **Numpy**. As I tried it in one previous work, so I refer to existing report of mentioned work to acquaint with structure of my PNN (p. 7, paragraph 'PNN solution'). Also mentioned report contains description of encoding and normalization of data.

I manipulated with train data set to reduce it's size so that PNN works faster rather than it worked on full set. I reduced set next way:

- if number of records for some class less than 1000, then all records included.
- if number of records greater than 1000, then included maximum number from quarter of its' amount and 1000.
- amount of normal records - 3000
- reduced set was shuffled.

After such manipulations:

```
## Train set has 21417 records (17.001% from full train set)
```



As we can see, there are few 'normals' and much more 'neptune'. There is no need to get very large amount of normals and still reasonable to get more different examples of attacks. But it makes sense to reduce number of neptunes.

Speed of recognition for such PNN was about 1 record per second. My computer has processor Intel Core i7-6500U CPU, 2.50 GHz 2.59 GHz. But I wasn't apply multiprocessing or threads to paralleling computations, so only single processing unit was charged.

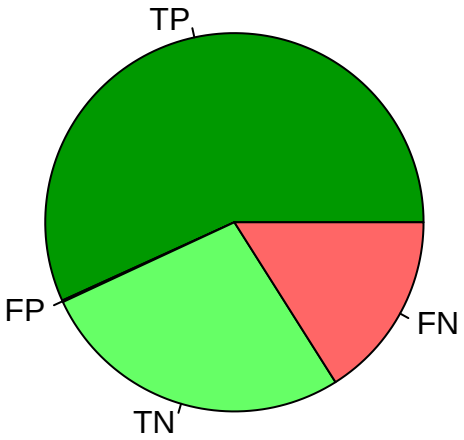
Recognition results

To estimate accuracy of model I will view

- ACC - accuracy itself.
- TP, FP, TN, FN - true-positive, false-positive, true-negative, false-negative. All types of attack abstracted into class 'attack' so that different attacks are equal. Attack presence will be recognized as 'positive' value (because purpose is to recognize attacks).
- ACC_bin - binary accuracy for mentioned abstraction.
- Precision and recall.
- F1 - F1-measure.
- Partial accuracies for each type of attacks.

Accuracy measures:

```
## ACC = 0.4385397
## ACC_bin = 0.8383977
```




```
## Precision is 0.9975844
## Recall is 0.779944
## F1 = 0.8754402
```

Result impress at first time but let's dig in deep. Accuracy for all classes is less than 0.5, which is slightly bad. I expected that issue to be due to confusing number of attack types. So I abstracted all attacks into single class to see how model differs normals and attacks. On pie diagram we can see that less than quarter of records recognized incorrectly. But there is some problem: to much false-negatives. I meant that many (about third part) normal connections recognized as attacks, so this model seems to be barking on many normal connections. I guess 3000 of normals is to low number. I should try something like 50% of normals with 50% of attacks.

Precision and recall have values close to 1. That's good, but still they don't include false-negatives, which appeared to be important.

F1-measure also close to 1, which is good too, but it inherits mentioned problem.

Partial accuracies for each type of attacks - parts of correctly recognized attacks of each type.

```
##               type   accuracy amount
## 5          neptune 0.33304703   4657
## 9     guess_passwd 0.01380991   1231
## 1            mscan 0.00000000    996
## 2      warezmaster 0.45444915    944
## 8          apache2 0.00000000    737
## 11           satan 0.58095238    735
```

```
## 3     processtable 0.00000000    685
## 10           smurf 1.00000000    665
## 12            back 0.97214485    359
## 7         snmpguess 0.00000000    331
## 13           saint 0.00000000    319
## 6         mailbomb 0.00000000    293
## 19   snmpgetattack 0.00000000    178
## 18       portsweep 0.96178344    157
## 21         ipsweep 0.98581560    141
## 15      httptunnel 0.00000000    133
## 16            nmap 0.00000000     73
## 17             pod 0.87804878     41
## 28 buffer_overflow 0.10000000     20
## 24        multihop 0.00000000     18
## 27           named 0.00000000     17
## 31              ps 0.00000000     15
## 14        sendmail 0.00000000     14
## 22           xterm 0.00000000     13
## 25         rootkit 0.00000000     13
## 20        teardrop 1.00000000     12
## 29           xlock 0.00000000      9
## 4             land 1.00000000      7
## 32          xsnoop 0.00000000      4
## 35       ftp_write 0.00000000      3
## 23            perl 1.00000000      2
## 26             phf 0.50000000      2
## 30        udpstorm 0.00000000      2
## 33      loadmodule 0.00000000      2
## 34       sqlattack 0.00000000      2
## 37            worm 0.00000000      2
## 36            imap 0.00000000      1
```

There many attacks are not recognized at all and some recognized badly. Let's see attacks with accuracy less than 0.8 to try found out some patterns.

```
##               type   accuracy amount
## 5          neptune 0.33304703   4657
## 9     guess_passwd 0.01380991   1231
## 1            mscan 0.00000000    996
## 2      warezmaster 0.45444915    944
## 8          apache2 0.00000000    737
## 11           satan 0.58095238    735
## 3     processtable 0.00000000    685
## 7        snmpguess 0.00000000    331
## 13           saint 0.00000000    319
## 6         mailbomb 0.00000000    293
## 19   snmpgetattack 0.00000000    178
## 15      httptunnel 0.00000000    133
## 16            nmap 0.00000000     73
## 28 buffer_overflow 0.10000000     20
## 24        multihop 0.00000000     18
## 27           named 0.00000000     17
## 31              ps 0.00000000     15
## 14        sendmail 0.00000000     14
## 22           xterm 0.00000000     13
```

```
## 25         rootkit 0.00000000     13
## 29           xlock 0.00000000      9
## 32          xsnoop 0.00000000      4
## 35       ftp_write 0.00000000      3
## 26             phf 0.50000000      2
## 30        udpstorm 0.00000000      2
## 33      loadmodule 0.00000000      2
## 34       sqlattack 0.00000000      2
## 37            worm 0.00000000      2
## 36            imap 0.00000000      1
```

We can pick out 3 groups of attacks:

- Badly recognized attacks with not small amount (>100). neptune, guess_passwd, warezmaster and satan
- Not recognized attacks with not small amount. mscan, apache2, processtable, snmpguess, saint, mailbomb, mailbomb, snmpgetattack, httptunnel
- Not recognized or badly recognized attacks with small amount (<100). All the rest

Model improvements

We can form 4 heuristic factors from mentioned attacks groups and group of attacks with good accuracy, which was excluded. This factors have to be named. I propose to replace single PNN with **PNNs tree**.

At first we have to factorize data-set so that it have factors including some types of alike attacks. It can be more scientific way to do so, for example, we can try to find tendencies for all attack types to be often mistaken with some particular type and add these types into single group. Such mistakes occur when two types has slightly similar signatures, so they can be taken into factor.

After groups ready, we have to train PNNs. **First-level PNN** will recognize whether connection normal or attack. It must be trained on about equal number of normals and attacks. I propose to get something like 500 normals and 1500 attack of each type. And attacks amounts have to be not less than 30 if available. If connection is attack, than record transfers to second-level PNN. **Second-level PNN** recognizes which group of attack types refers to current attack. It should have about 2000 neurons in pattern layer. In case we have 4 groups it would be nice to take 500 records from each group into train set. Each signature must be associated with it's group name. Also train set must contain each type of attacks. After recognition of group record transfers to according third-level PNN. **Third-level PNN** recognizes which specific attack takes place for given record. In the PNNs tree there are equal number of third-level PNNs to number of groups. Each third-level PNN should be trained on set of size from 1000 to 5000, depending on group size. Train sets for each PNN must have balanced amounts of records of each attack type in according group.

It would be nice if almost all neurons in all PNNs are different. So such model will have 6 trained PNNs for 4 groups from example. Also such PNNs should have much less summary amount of neurons in pattern layers. I guess we can heavily reduce computations on this basis. It might increase accuracy, speed up recognition, but probably will take some more memory.

Addition 1 – summary of KDD

```
##     duration       protocol_type     service           flag
##  Min.   :    0.0   icmp:  9334   http    :48191   SF     :89819
##  1st Qu.:    0.0   tcp :121569   private :26627   S0     :36864
##  Median :    0.0   udp : 17613   domain_u: 9937   REJ    :15083
##  Mean   :  276.8                 smtp    : 8247   RSTR   : 3090
##  3rd Qu.:    0.0                 ftp_data: 7711   RSTO   : 2335
##  Max.   :57715.0                 other   : 5197   S1     :  386
##                                  (Other) :42606   (Other):  939
##    src_bytes           dst_bytes              land           wrong_fragment
```

```
##  Min.   :0.000e+00   Min.   :0.000e+00   Min.   :0.0000000   Min.   :0.00000
##  1st Qu.:0.000e+00   1st Qu.:0.000e+00   1st Qu.:0.0000000   1st Qu.:0.00000
##  Median :4.400e+01   Median :0.000e+00   Median :0.0000000   Median :0.00000
##  Mean   :4.023e+04   Mean   :1.709e+04   Mean   :0.0002155   Mean   :0.02052
##  3rd Qu.:2.780e+02   3rd Qu.:5.710e+02   3rd Qu.:0.0000000   3rd Qu.:0.00000
##  Max.   :1.380e+09   Max.   :1.310e+09   Max.   :1.0000000   Max.   :3.00000
##
##      urgent              hot           num_failed_logins    logged_in
##  Min.   :0.000000   Min.   :  0.0000   Min.   :0.000000   Min.   :0.0000
##  1st Qu.:0.000000   1st Qu.:  0.0000   1st Qu.:0.000000   1st Qu.:0.0000
##  Median :0.000000   Median :  0.0000   Median :0.000000   Median :0.0000
##  Mean   :0.000202   Mean   :  0.1894   Mean   :0.004323   Mean   :0.4028
##  3rd Qu.:0.000000   3rd Qu.:  0.0000   3rd Qu.:0.000000   3rd Qu.:1.0000
##  Max.   :3.000000   Max.   :101.0000   Max.   :5.000000   Max.   :1.0000
##
##  num_compromised      root_shell        su_attempted          num_root
##  Min.   :   0.000   Min.   :0.000000   Min.   :0.0000000   Min.   :   0.000
##  1st Qu.:   0.000   1st Qu.:0.000000   1st Qu.:0.0000000   1st Qu.:   0.000
##  Median :   0.000   Median :0.000000   Median :0.0000000   Median :   0.000
##  Mean   :   0.255   Mean   :0.001508   Mean   :0.0009763   Mean   :   0.274
##  3rd Qu.:   0.000   3rd Qu.:0.000000   3rd Qu.:0.0000000   3rd Qu.:   0.000
##  Max.   :7479.000   Max.   :1.000000   Max.   :2.0000000   Max.   :7468.000
##
##  num_file_creations    num_shells       num_access_files   num_outbound_cmds
##  Min.   :  0.00000   Min.   :0.000000   Min.   :0.000000   Min.   :0
##  1st Qu.:  0.00000   1st Qu.:0.000000   1st Qu.:0.000000   1st Qu.:0
##  Median :  0.00000   Median :0.000000   Median :0.000000   Median :0
##  Mean   :  0.01207   Mean   :0.000525   Mean   :0.004013   Mean   :0
##  3rd Qu.:  0.00000   3rd Qu.:0.000000   3rd Qu.:0.000000   3rd Qu.:0
##  Max.   :100.00000   Max.   :5.000000   Max.   :9.000000   Max.   :0
##
##  is_host_login      is_guest_login        count          srv_count
##  Min.   :0.00e+00   Min.   :0.00000   Min.   :  0.00   Min.   :  0.00
##  1st Qu.:0.00e+00   1st Qu.:0.00000   1st Qu.:  2.00   1st Qu.:  2.00
##  Median :0.00e+00   Median :0.00000   Median : 13.00   Median :  7.00
##  Mean   :8.08e-05   Mean   :0.01231   Mean   : 83.34   Mean   : 28.25
##  3rd Qu.:0.00e+00   3rd Qu.:0.00000   3rd Qu.:141.00   3rd Qu.: 17.00
##  Max.   :1.00e+00   Max.   :1.00000   Max.   :511.00   Max.   :511.00
##
##   serror_rate     srv_serror_rate   rerror_rate     srv_rerror_rate
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000
##  Median :0.0000   Median :0.0000   Median :0.0000   Median :0.0000
##  Mean   :0.2569   Mean   :0.2553   Mean   :0.1379   Mean   :0.1385
##  3rd Qu.:0.8500   3rd Qu.:0.9100   3rd Qu.:0.0000   3rd Qu.:0.0000
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
##
##  same_srv_rate   diff_srv_rate     srv_diff_host_rate dst_host_count
##  Min.   :0.000   Min.   :0.00000   Min.   :0.00000    Min.   :  0.0
##  1st Qu.:0.100   1st Qu.:0.00000   1st Qu.:0.00000    1st Qu.: 87.0
##  Median :1.000   Median :0.00000   Median :0.00000    Median :255.0
##  Mean   :0.673   Mean   :0.06776   Mean   :0.09744    Mean   :183.9
##  3rd Qu.:1.000   3rd Qu.:0.06000   3rd Qu.:0.00000    3rd Qu.:255.0
##  Max.   :1.000   Max.   :1.00000   Max.   :1.00000    Max.   :255.0
```

```
##
##  dst_host_srv_count dst_host_same_srv_rate dst_host_diff_srv_rate
##  Min.   :  0.0      Min.   :0.0000         Min.   :0.0000
##  1st Qu.: 11.0      1st Qu.:0.0500         1st Qu.:0.0000
##  Median : 72.0      Median :0.6000         Median :0.0200
##  Mean   :119.5      Mean   :0.5345         Mean   :0.0841
##  3rd Qu.:255.0      3rd Qu.:1.0000         3rd Qu.:0.0700
##  Max.   :255.0      Max.   :1.0000         Max.   :1.0000
##
##  dst_host_same_src_port_rate dst_host_srv_diff_host_rate dst_host_serror_rate
##  Min.   :0.0000              Min.   :0.00000             Min.   :0.0000
##  1st Qu.:0.0000              1st Qu.:0.00000             1st Qu.:0.0000
##  Median :0.0000              Median :0.00000             Median :0.0000
##  Mean   :0.1459              Mean   :0.03058             Mean   :0.2561
##  3rd Qu.:0.0500              3rd Qu.:0.01000             3rd Qu.:0.6000
##  Max.   :1.0000              Max.   :1.00000             Max.   :1.0000
##
##  dst_host_srv_serror_rate dst_host_rerror_rate dst_host_srv_rerror_rate
##  Min.   :0.0000           Min.   :0.0000       Min.   :0.0000
##  1st Qu.:0.0000           1st Qu.:0.0000       1st Qu.:0.0000
##  Median :0.0000           Median :0.0000       Median :0.0000
##  Mean   :0.2513           Mean   :0.1362       Mean   :0.1364
##  3rd Qu.:0.5000           3rd Qu.:0.0000       3rd Qu.:0.0000
##  Max.   :1.0000           Max.   :1.0000       Max.   :1.0000
##
##        class
##  normal   :77053
##  neptune  :45871
##  satan    : 4368
##  ipsweep  : 3740
##  smurf    : 3311
##  portsweep: 3088
##  (Other)  :11085
```