НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
Факультет прикладної математики
Кафедра прикладної математики

Звіт
із лабораторної роботи №3
з дисципліни «Системи нейронних мереж»
на тему
«Нейромережеве розпізнавання кібератак»

Виконав:
студент групи КМ-02
Пилипченко Б. О.

Керівник:
Терейковський І. А.

Київ—2023

Мета роботи:

Розробка програмного забезпечення для реалізації нейронної мережі PNN, призначеної для розпізнавання кібератак, сигнатури яких представлені у базах даних KDD-99.

Теоретичні відомості:

Роберт Каллан Основні концепції нейронних мереж ст. 152—164

Архітектура мережі

PNN мережа складається з трьох шарів:

1. Вхідний шар:

   Вхідний шар розподіляє характеристики вхідного зразка на шар зразків. Кожен нейрон вхідного шару пов'язаний із кожним нейроном шару зв'язків.

2. Шар зразків:

   Шар зразків має по одному нейрону для кожного зразка. Вагові коефіцієнти вхідних сигналів нейрону дорівнюють значенням характеристик відповідного зразка.

   Функція активації нейрону шару зразків:

$$O_j = \exp\left(\frac{-\sum(w_{ij} - x_i)^2}{\sigma^2}\right)$$

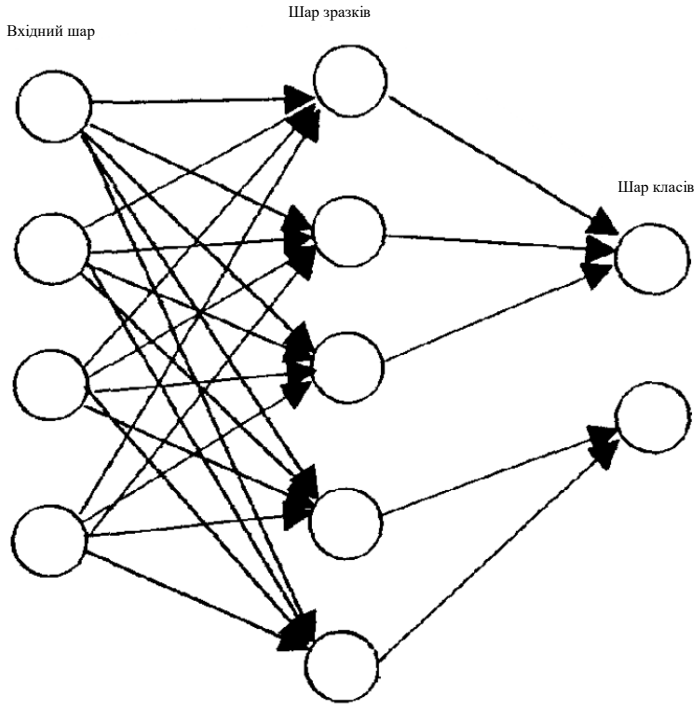
Oj—характеристика близькості вхідного зразка до j-того зразка класу;
xi—i-та характеристика вхідного зразка;
wij—i-та характеристика j– того зразка класу;
σ—параметр мережі.

3. Шар класів:

   Шар класів має по одному нейрону для кожного класу. Кожен нейрон шару класів має зв'язки лише з нейронами зв'язків, які належать до класу. Всі вагові коефіцієнти між шаром зразків та шаром класів рівні 1.

   Функція активації нейрону шару класів є сумою вхідних сигналів нейрону.

PNN мережа не потребує навчання, подібного до навчання мереж із оберненим поширенням сигналу. Всі параметри мережі PNN, крім значення сігми, визначаються безпосередньо навчальними даними:
Кількість нейронів вхідного шару = кількість характеристик зразків
Кількість нейронів шару зразків = кількість зразків
Кількість нейронів шару класів = кількість класів
Навчання мережі полягає у підборі оптимального значення сігми.
В роботі сігма підбиралась експериментально.



Огляд даних

В якості датасету було обрано KDD99. Остаточне тестування мереж здійснювалось на всьому датасеті (490020 зразків).

Початковий датасет містить 42 колонки (41 ознака + label):

```
>>> Column types >>>
 duration                int64
protocol_type           object
service                 object
flag                    object
src_bytes               int64
dst_bytes               int64
land                    int64
wrong_fragment          int64
urgent                  int64
hot                     int64
num_failed_logins       int64
logged_in               int64
lnum_compromised        int64
lroot_shell             int64
lsu_attempted           int64
lnum_root               int64
lnum_file_creations     int64
lnum_shells             int64
lnum_access_files       int64
lnum_outbound_cmds      int64
is_host_login           int64
is_guest_login          int64
count                   int64
srv_count               int64
serror_rate             float64
srv_serror_rate         float64
rerror_rate             float64
srv_rerror_rate         float64
same_srv_rate           float64
diff_srv_rate           float64
srv_diff_host_rate      float64
dst_host_count          int64
dst_host_srv_count      int64
dst_host_same_srv_rate  float64
dst_host_diff_srv_rate  float64
dst_host_same_src_port_rate  float64
dst_host_srv_diff_host_rate  float64
dst_host_serror_rate    float64
dst_host_srv_serror_rate  float64
dst_host_rerror_rate    float64
dst_host_srv_rerror_rate  float64
label                   object
```

Колонки lnum_outbound_cmds, is_host_login дорівнюють нулю для всіх зразків у датасеті:

```
Column: lnum_outbound_cmds
Unique Values Count: 1
Unique Values: [0]

Column: is_host_login
Unique Values Count: 1
Unique Values: [0]
```

тому було вирішено видалити ці колонки для всіх зразків. Остаточний датасет має 39 ознак.

В датасеті наявні ознаки, значеннями яких є рядкові літерали. Для таких ознак було застосовано LabelEncoder із бібліотеки sklearn. Таким чином нечислові ознаки було перетворено на числові, значення яких змінюється від 0 до n-1, де n—кількість унікальних значень ознаки.

Останнім кроком перед розбиттям даних на тренувальну та тестувальну вибірки є нормалізація значень ознак.

Розбиття даних

Параметри розбиття:

PNN_REFERENCE_COUNT_BY_CLASS	Набір пар «назва класу» - кількість прикладів класу pnn мережі відносно тренувальної вибірки
TEST_SIZE	Розмір тестової вибірки відносно вхідного (обробленого) датасету
REAL_DATA_COUNT_PER_CLASS	Максимальна кількість зразків класу в pnn мережі. Якщо кількість зразків класу перевищує параметр, то із всіх зразків класу обирається RE-AL_DATA_COUNT_PER_CLASS зразків випадковим чином.
RANDOM_STATE	Єдине значення random_state для всіх функцій, що потребують встановлення зерна.

Приклад набору параметрів:

```
PNN_REFERENCE_COUNT_BY_CLASS = {
   "smurf":          0.6,
   "neptune":        0.6,
   "normal":         0.6,
   "back":           1.0,
   "satan":          1.0,
   "ipsweep":        1.0,
   "portsweep":      1.0,
   "warezclient":    1.0,
   "teardrop":       1.0,
   "pod":            1.0,
   "nmap":           1.0,
   "guess_passwd":   1.0,
   "buffer_overflow": 1.0,
   "land":           1.0,
   "warezmaster":    1.0,
   "imap":           1.0,
   "rootkit":        1.0,
   "loadmodule":     1.0,
   "ftp_write":      1.0,
   "multihop":       1.0,
   "phf":            1.0,
   "perl":           1.0,
   "spy":            1.0
}
TEST_SIZE = 0.25
REAL_DATA_COUNT_PER_CLASS = 150
RANDOM_STATE = 1450
```

Розбиття обробленого датасету на тренувальну та навчальну вибірки виконується наступним чином:

1. Оброблений датасет розбивається на «елементарні» датасети—кожен елементарний датасет містить всі зразки одного класу, лише зразки одного класу.
2. Для кожного елементарного датасету: кроки 3 або 4
3. Якщо кількість записів у елементарному датасеті менша за REAL_DATA_COUNT_PER_CLASS—всі записи елементарного датасету заносяться у pnn мережу, навчальну вибірку, тестову вибірку.
4. Якщо кількість записів у елементарному датасеті більша за REAL_DATA_COUNT_PER_CLASS, тоді береться випадкова вибірка з елементарного датасету розміром REAL_DATA_COUNT_PER_CLASS. Отримана вибірка розбивається на тренувальну та тестову вибірки у співвідношенні TEST_SIZE. Далі із тренувальної вибірки обираються приклади, що будуть занесені у власне pnn мережу у співвідношенні, заданому PNN_REFERENCE_COUNT_BY_CLASS.
5. Внаслідок кроків 3, 4 отримуємо 3 набори даних для кожного класу: pnn набір, тренувальний набір, тестовий набір. Виконується об’єднання відповідних наборів у кінцеві pnn дані, тренувальні дані, тестові дані.

Навчання мережі

Тренувальні, тестові вибірки, текстові представлення pnn мереж: MLDL-lab3
Код скрипта-генератора даних, реалізації PNN: https://github.com/Bohdan628318ylypchenko/MLDL-Lab3.git

Параметри генерації PNN, тестової вибірки, валідаційної вибірки:

```
PNN_REFERENCE_COUNT_BY_CLASS = {
    "smurf":           0.6,
    "neptune":         0.6,
    "normal":          0.6,
    "back":           1.0,
    "satan":          1.0,
    "ipsweep":         1.0,
    "portsweep":       1.0,
    "warezclient":     1.0,
    "teardrop":        1.0,
    "pod":            1.0,
    "nmap":           1.0,
    "guess_passwd":    1.0,
    "buffer_overflow": 1.0,
    "land":           1.0,
    "warezmaster":     1.0,
    "imap":           1.0,
    "rootkit":         1.0,
    "loadmodule":      1.0,
    "ftp_write":       1.0,
    "multihop":        1.0,
    "phf":            1.0,
    "perl":           1.0,
    "spy":            1.0
}
TEST_SIZE = 0.25
REAL_DATA_COUNT_PER_CLASS = 2000
RANDOM_STATE_1 = 1450
RANDOM_STATE_2 = 860
```

Остаточні дані можна переглянути в репозиторії (файли pnn-2000.txt, train-2000.txt, test-2000.txt

Початкове значення s = 0.01

Запуск мережі на тренувальній вибірці:

```
t
Class: back-1600; eval: 1.0000000000000000000
Class: buffer_overflow-30; eval: 0.90000000000000002220
Class: ftp_write-8; eval: 1.0000000000000000000
Class: guess_passwd-53; eval: 1.0000000000000000000
Class: imap-12; eval: 1.0000000000000000000
Class: ipsweep-1247; eval: 1.0000000000000000000
Class: land-21; eval: 1.0000000000000000000
Class: loadmodule-9; eval: 1.0000000000000000000
Class: multihop-7; eval: 0.85714285714285709528
Class: neptune-960; eval: 0.99937500000000001332
Class: nmap-231; eval: 0.98701298701298700866
Class: normal-960; eval: 0.95750000000000001776
Class: perl-3; eval: 1.0000000000000000000
Class: phf-4; eval: 1.0000000000000000000
Class: pod-264; eval: 1.0000000000000000000
Class: portsweep-1040; eval: 0.99903846153846154188
Class: rootkit-10; eval: 1.0000000000000000000
Class: satan-1589; eval: 1.0000000000000000000
Class: smurf-960; eval: 1.0000000000000000000
Class: spy-2; eval: 1.0000000000000000000
Class: teardrop-979; eval: 1.0000000000000000000
Class: warezclient-1020; eval: 1.0000000000000000000
Class: warezmaster-20; eval: 1.0000000000000000000
zero count: 0
time: 3.097883
```

Запуск мережі на тестовій вибірці:

```
v
Class: back-1600; eval: 1.0000000000000000000
Class: buffer_overflow-30; eval: 0.90000000000000002220
Class: ftp_write-8; eval: 1.0000000000000000000
Class: guess_passwd-53; eval: 1.0000000000000000000
Class: imap-12; eval: 1.0000000000000000000
Class: ipsweep-1247; eval: 1.0000000000000000000
Class: land-21; eval: 1.0000000000000000000
Class: loadmodule-9; eval: 1.0000000000000000000
Class: multihop-7; eval: 0.85714285714285709528
Class: neptune-960; eval: 0.99750000000000005329
Class: nmap-231; eval: 0.98701298701298700866
Class: normal-960; eval: 0.89500000000000001776
Class: perl-3; eval: 1.0000000000000000000
Class: phf-4; eval: 1.0000000000000000000
Class: pod-264; eval: 1.0000000000000000000
Class: portsweep-1040; eval: 0.99903846153846154188
Class: rootkit-10; eval: 1.0000000000000000000
Class: satan-1589; eval: 1.0000000000000000000
Class: smurf-960; eval: 0.99750000000000005329
Class: spy-2; eval: 1.0000000000000000000
Class: teardrop-979; eval: 1.0000000000000000000
Class: warezclient-1020; eval: 1.0000000000000000000
Class: warezmaster-20; eval: 1.0000000000000000000
```

Загалом результати позитивні: для більшості класів мережа вірно класифікує усі приклади класу, наявні у вибірці. Найгірші показники мають класи normal (0.95 | 0.89) та multihop (0.85 | 0.85).
Особливо цікавим є клас multihop, оскільки, не зважаючи на те, що ВСІ приклади з початкового датасету наявні в мережі, мережа не дає правильної відповіді на деякі з прикладів цього класу.

Спробуємо запустити мережу на кожному прикладі класу multihop окремо:

```
Enter command:
r 0.00,0.50,0.28,0.90,0.00,0.00,0.00,0.00,0.00,0.07,0.00,1.00,0.00,0.00,0.00,0.00,0.04,0.00,0.00,1.00,0.00,0.00,0.00,0.00,0.00,0.00,1.00,0.00,0.00,1.00,0.00,0.00,0.01,0.00,0.00,0.00,0.00,0.04,0.00
back-1600 : 0.000000000000000000
buffer_overflow-30 : 0.000000000000000000
ftp_write-8 : 0.000000000000000000
guess_passwd-53 : 0.000000000000000000
imap-12 : 0.000000000000000000
ipsweep-1247 : 0.000000000000000000
land-21 : 0.000000000000000000
loadmodule-9 : 0.000000000000000000
multihop-7 : 0.482749206693218857
neptune-960 : 0.000000000000000000
nmap-231 : 0.000000000000000000
normal-960 : 0.000000000000000000
perl-3 : 0.000000000000000000
phf-4 : 0.000000000000000000
pod-264 : 0.000000000000000000
portsweep-1040 : 0.000000000000000000
rootkit-10 : 0.000000000000000000
satan-1589 : 0.000000000000000000
smurf-960 : 0.000000000000000000
spy-2 : 0.000000000000000000
teardrop-979 : 0.000000000000000000
warezclient-1020 : 0.000000000000000000
warezmaster-20 : 0.000000000000000000
```

```
Enter command:
r
0.00,0.50,0.28,0.90,0.00,0.00,0.00,0.00,0.00,0.03,0.00,1.00,0.00,0.00,0.00,0.00,0.04,0.00,0.00,1.00,0.00,0.00,0.00,0.00,0.00,0.00,1.00,0.00,0.00,1.00,0.01,0.01,0.01,0.00,0.00,0.00,0.00
,0.04,0.00
back-1600 : 0.000000000000000000
buffer_overflow-30 : 0.000000000000000000
ftp_write-8 : 0.000000000000000000
guess_passwd-53 : 0.000000000000000000
imap-12 : 0.000000000000000000
ipsweep-1247 : 0.000000000000000000
land-21 : 0.000000000000000000
loadmodule-9 : 0.000000000000000000
multihop-7 : 0.545100167113102940
neptune-960 : 0.000000000000000000
nmap-231 : 0.000000000000000000
normal-960 : 0.000000000000000000
perl-3 : 0.000000000000000000
phf-4 : 0.000000000000000000
pod-264 : 0.000000000000000000
portsweep-1040 : 0.000000000000000000
rootkit-10 : 0.000000000000000000
satan-1589 : 0.000000000000000000
smurf-960 : 0.000000000000000000
spy-2 : 0.000000000000000000
teardrop-979 : 0.000000000000000000
warezclient-1020 : 0.000000000000000000
warezmaster-20 : 0.000000000000000000
```

```
Enter command:
r
0.00,0.50,0.29,0.90,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,
1.00,0.00,0.00,0.00,0.00,1.00,0.00,1.00,0.00,0.00,0.00,0.00,0.00
back-1600 : 0.000000000000000000
buffer_overflow-30 : 0.000000000000000000
ftp_write-8 : 0.000000000000000000
guess_passwd-53 : 0.000000000000000000
imap-12 : 0.000000000000000000
ipsweep-1247 : 0.000000000000000000
land-21 : 0.000000000000000000
loadmodule-9 : 0.000000000000000000
multihop-7 : 0.645699336191089213
neptune-960 : 0.000000000000000000
nmap-231 : 0.000000000000000000
normal-960 : 0.000000000000000000
perl-3 : 0.000000000000000000
phf-4 : 0.000000000000000000
pod-264 : 0.000000000000000000
portsweep-1040 : 0.000000000000000000
rootkit-10 : 0.000000000000000000
satan-1589 : 0.000000000000000000
smurf-960 : 0.000000000000000000
spy-2 : 0.000000000000000000
teardrop-979 : 0.000000000000000000
warezclient-1020 : 0.000000000000000000
warezmaster-20 : 0.837178979793791633
```

```
r 0.00,0.50,0.29,0.90,0.00,0.09,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,1.00,0.00,0.00,0.01,0.01,1.00,0.00,1.00,0.00,0.00,0.00,0.00,0.00
back-1600 : 0.000000000000000000
buffer_overflow-30 : 0.000000000000000000
ftp_write-8 : 0.000000000000000000
guess_passwd-53 : 0.000000000000000000
imap-12 : 0.000000000000000000
ipsweep-1247 : 0.000000000000000000
land-21 : 0.000000000000000000
loadmodule-9 : 0.000000000000000000
multihop-7 : 0.795459086582364638
neptune-960 : 0.000000000000000000
nmap-231 : 0.000000000000000000
normal-960 : 0.000000000000000000
perl-3 : 0.000000000000000000
phf-4 : 0.000000000000000000
pod-264 : 0.000000000000000000
portsweep-1040 : 0.000000000000000000
rootkit-10 : 0.000000000000000000
satan-1589 : 0.000000000000000000
smurf-960 : 0.000000000000000000
spy-2 : 0.000000000000000000
teardrop-979 : 0.000000000000000000
warezclient-1020 : 0.000000000000000000
warezmaster-20 : 0.000000000000000000
```

```
r
0.00,0.50,0.29,0.90,0.00,0.19,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,1.00,0.00,1.00,0.01,0.01,1.00,0.00,1.00,0.00,0
.00,0.00,0.00,0.00
back-1600 : 0.000000000000000000
buffer_overflow-30 : 0.000000000000000000
ftp_write-8 : 0.000000000000000000
guess_passwd-53 : 0.000000000000000000
imap-12 : 0.000000000000000000
ipsweep-1247 : 0.000000000000000000
land-21 : 0.000000000000000000
loadmodule-9 : 0.000000000000000000
multihop-7 : 0.716074647101321404
neptune-960 : 0.000000000000000000
nmap-231 : 0.000000000000000000
normal-960 : 0.000000000000000000
perl-3 : 0.000000000000000000
phf-4 : 0.000000000000000000
pod-264 : 0.000000000000000000
portsweep-1040 : 0.000000000000000000
rootkit-10 : 0.000000000000000000
satan-1589 : 0.000000000000000000
smurf-960 : 0.000000000000000000
spy-2 : 0.000000000000000000
teardrop-979 : 0.000000000000000000
warezclient-1020 : 0.000000000000000000
warezmaster-20 : 0.000000000000000000
```

```
r
0.00,0.50,0.86,0.90,0.00,0.00,0.00,0.00,0.00,0.10,0.00,1.00,0.02,1.00,0.00,0.04,0.14,1.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,1.
00,0.00,0.00,0.00,0.00,1.00,0.00,1.00,0.00,0.00,0.00,0.00,0.00
back-1600 : 0.000000000000000000
buffer_overflow-30 : 0.000000000000000000
ftp_write-8 : 0.000000000000000000
guess_passwd-53 : 0.000000000000000000
imap-12 : 0.000000000000000000
ipsweep-1247 : 0.000000000000000000
land-21 : 0.000000000000000000
loadmodule-9 : 0.000000000000000000
multihop-7 : 0.414702410466252780
neptune-960 : 0.000000000000000000
nmap-231 : 0.000000000000000000
normal-960 : 0.000000000000000000
perl-3 : 0.000000000000000000
phf-4 : 0.000000000000000000
pod-264 : 0.000000000000000000
portsweep-1040 : 0.000000000000000000
rootkit-10 : 0.000000000000000000
satan-1589 : 0.000000000000000000
smurf-960 : 0.000000000000000000
spy-2 : 0.000000000000000000
teardrop-979 : 0.000000000000000000
warezclient-1020 : 0.000000000000000000
warezmaster-20 : 0.000000000000000000
```

```
r
0.01,0.50,0.86,0.90,0.00,0.00,0.00,0.00,0.00,0.50,0.00,1.00,0.04,1.00,0.00,0.05,0.14,0.50,0.25,0.00,0.00,0.00,0.00,0.00,0.00,0.00,1.00,0.00,0.00,0.00,0.00,1.00,0.00,1.00,0.00,0.00,0.00,0.00,0.00
back-1600 : 0.000000000000000000
buffer_overflow-30 : 0.000000000000000000
ftp_write-8 : 0.000000000000000000
guess_passwd-53 : 0.000000000000000000
imap-12 : 0.000000000000000000
ipsweep-1247 : 0.000000000000000000
land-21 : 0.000000000000000000
loadmodule-9 : 0.000000000000000000
multihop-7 : 0.345086233247635987
neptune-960 : 0.000000000000000000
nmap-231 : 0.000000000000000000
normal-960 : 0.000000000000000000
perl-3 : 0.000000000000000000
phf-4 : 0.000000000000000000
pod-264 : 0.000000000000000000
portsweep-1040 : 0.000000000000000000
rootkit-10 : 0.000000000000000000
satan-1589 : 0.000000000000000000
smurf-960 : 0.000000000000000000
spy-2 : 0.000000000000000000
teardrop-979 : 0.000000000000000000
warezclient-1020 : 0.000000000000000000
warezmaster-20 : 0.000000000000000000
```

Дійсно, мережа класифікує приклад з id = 53129 як warezmaster, не multihop.
Спробуємо «перенавчити» мережу, встановивши сігму рівній 0.001.

Результат роботи мережі, s = 0.001

Тренувальна вибірка:

t
Class: back-1600; eval: 1.0000000000000000000
Class: buffer_overflow-30; eval: 0.93333333333333334814
Class: ftp_write-8; eval: 1.0000000000000000000
Class: guess_passwd-53; eval: 1.0000000000000000000
Class: imap-12; eval: 1.0000000000000000000
Class: ipsweep-1247; eval: 1.0000000000000000000
Class: land-21; eval: 1.0000000000000000000
Class: loadmodule-9; eval: 1.0000000000000000000
Class: multihop-7; eval: 1.0000000000000000000
Class: neptune-960; eval: 0.95437499999999997335
Class: nmap-231; eval: 1.0000000000000000000
Class: normal-960; eval: 0.78812499999999996447
Class: perl-3; eval: 1.0000000000000000000
Class: phf-4; eval: 1.0000000000000000000
Class: pod-264; eval: 1.0000000000000000000
Class: portsweep-1040; eval: 0.99903846153846154188
Class: rootkit-10; eval: 1.0000000000000000000
Class: satan-1589; eval: 1.0000000000000000000
Class: smurf-960; eval: 1.0000000000000000000
Class: spy-2; eval: 1.0000000000000000000
Class: teardrop-979; eval: 1.0000000000000000000
Class: warezclient-1020; eval: 1.0000000000000000000
Class: warezmaster-20; eval: 1.0000000000000000000
zero count: 0
time: 4.042179

Тестова вибірка:

v
Class: back-1600; eval: 1.0000000000000000000
Class: buffer_overflow-30; eval: 0.93333333333333334814
Class: ftp_write-8; eval: 1.0000000000000000000
Class: guess_passwd-53; eval: 1.0000000000000000000
Class: imap-12; eval: 1.0000000000000000000
Class: ipsweep-1247; eval: 1.0000000000000000000
Class: land-21; eval: 1.0000000000000000000
Class: loadmodule-9; eval: 1.0000000000000000000
Class: multihop-7; eval: 1.0000000000000000000
Class: neptune-960; eval: 0.87500000000000000000
Class: nmap-231; eval: 1.0000000000000000000
Class: normal-960; eval: 0.48249999999999998446
Class: perl-3; eval: 1.0000000000000000000
Class: phf-4; eval: 1.0000000000000000000
Class: pod-264; eval: 1.0000000000000000000
Class: portsweep-1040; eval: 0.99903846153846154188
Class: rootkit-10; eval: 1.0000000000000000000
Class: satan-1589; eval: 1.0000000000000000000
Class: smurf-960; eval: 0.99750000000000005329
Class: spy-2; eval: 1.0000000000000000000
Class: teardrop-979; eval: 1.0000000000000000000
Class: warezclient-1020; eval: 1.0000000000000000000
Class: warezmaster-20; eval: 1.0000000000000000000
zero count: 0
time: 2.273892

Мережа змогла перенавчитись, для обох вибірок правильно класифікує всі приклади multihop. При цьому результати розпізнавання інших класів очікувано погіршились (клас normal (s = 0.01) t: 0.9575; v: 0.895 проти (s = 0.001) t: 0.7881; v: 0.4824 | клас neptune (s = 0.01) t: 0.9993; v: 0.9975 проти (s = 0.001) t: 0.9543; v: 0.8750).
Враховуючи кількість зразків класу normal (97277 у початковому датасеті), не має сенсу розпізнавати всі приклади multihop за рахунок 0.4824 правильних відповідей для класу normal.

Оберемо сігму рівній 0.1

Результат роботи мережі, s = 0.1

Тренувальна вибірка:

t
Class: back-1600; eval: 0.99875000000000002665
Class: buffer_overflow-30; eval: 0.76666666666666671848
Class: ftp_write-8; eval: 0.87500000000000000000
Class: guess_passwd-53; eval: 1.0000000000000000000
Class: imap-12; eval: 1.0000000000000000000
Class: ipsweep-1247; eval: 0.99438652766639934466
Class: land-21; eval: 1.0000000000000000000
Class: loadmodule-9; eval: 1.0000000000000000000
Class: multihop-7; eval: 0.85714285714285709528
Class: neptune-960; eval: 1.0000000000000000000
Class: nmap-231; eval: 0.98701298701298700866
Class: normal-960; eval: 0.80500000000000004885
Class: perl-3; eval: 1.0000000000000000000
Class: phf-4; eval: 1.0000000000000000000
Class: pod-264; eval: 1.0000000000000000000
Class: portsweep-1040; eval: 0.99903846153846154188
Class: rootkit-10; eval: 0.80000000000000004441
Class: satan-1589; eval: 1.0000000000000000000
Class: smurf-960; eval: 1.0000000000000000000
Class: spy-2; eval: 1.0000000000000000000
Class: teardrop-979; eval: 1.0000000000000000000
Class: warezclient-1020; eval: 1.0000000000000000000
Class: warezmaster-20; eval: 1.0000000000000000000
zero count: 0
time: 2.449062

Тестова вибірка:

v
Class: back-1600; eval: 1.0000000000000000000
Class: buffer_overflow-30; eval: 0.76666666666666671848
Class: ftp_write-8; eval: 0.87500000000000000000
Class: guess_passwd-53; eval: 1.0000000000000000000
Class: imap-12; eval: 1.0000000000000000000
Class: ipsweep-1247; eval: 0.99438652766639934466
Class: land-21; eval: 1.0000000000000000000
Class: loadmodule-9; eval: 1.0000000000000000000
Class: multihop-7; eval: 0.85714285714285709528
Class: neptune-960; eval: 1.0000000000000000000
Class: nmap-231; eval: 0.98701298701298700866
Class: normal-960; eval: 0.76249999999999995559
Class: perl-3; eval: 1.0000000000000000000
Class: phf-4; eval: 1.0000000000000000000
Class: pod-264; eval: 1.0000000000000000000
Class: portsweep-1040; eval: 0.99903846153846154188
Class: rootkit-10; eval: 0.80000000000000004441
Class: satan-1589; eval: 1.0000000000000000000
Class: smurf-960; eval: 1.0000000000000000000
Class: spy-2; eval: 1.0000000000000000000
Class: teardrop-979; eval: 1.0000000000000000000
Class: warezclient-1020; eval: 1.0000000000000000000
Class: warezmaster-20; eval: 1.0000000000000000000
zero count: 0
time: 1.756200

Мережа є більш точною при s = 0.1 аніж при s = 0.001. При цьому результати для s = 0.1 гірші за результати для s = 0.01 (наприклад клас buffer_overflow (s = 0.1) t: 0.7666; v: 0.7666 проти (s = 0.01) t: 0.9000; v: 0.9000).
Отже оптимальна сігма знаходиться в межах (0.01; 0.1).

Протестуємо мережу для значень сігми (0.015, 0.025, 0.035, 0.045):

s = 0.015

t
Class: back-1600; eval: 1.0000000000000000000
Class: buffer_overflow-30; eval: 0.90000000000000002220
Class: ftp_write-8; eval: 0.87500000000000000000
Class: guess_passwd-53; eval: 1.00000000000000000000
Class: imap-12; eval: 1.00000000000000000000
Class: ipsweep-1247; eval: 1.00000000000000000000
Class: land-21; eval: 1.00000000000000000000
Class: loadmodule-9; eval: 1.00000000000000000000
Class: multihop-7; eval: 0.85714285714285709528
Class: neptune-960; eval: 0.99937500000000001332
Class: nmap-231; eval: 0.98701298701298700866
Class: normal-960; eval: 0.96937499999999998668
Class: perl-3; eval: 1.00000000000000000000
Class: phf-4; eval: 1.00000000000000000000
Class: pod-264; eval: 1.00000000000000000000
Class: portsweep-1040; eval: 0.99903846153846154188
Class: rootkit-10; eval: 0.90000000000000002220
Class: satan-1589; eval: 1.00000000000000000000
Class: smurf-960; eval: 1.00000000000000000000
Class: spy-2; eval: 1.00000000000000000000
Class: teardrop-979; eval: 1.00000000000000000000
Class: warezclient-1020; eval: 1.00000000000000000000
Class: warezmaster-20; eval: 1.00000000000000000000
zero count: 0
time: 3.141179

v
Class: back-1600; eval: 1.00000000000000000000
Class: buffer_overflow-30; eval: 0.90000000000000002220
Class: ftp_write-8; eval: 0.87500000000000000000
Class: guess_passwd-53; eval: 1.00000000000000000000
Class: imap-12; eval: 1.00000000000000000000
Class: ipsweep-1247; eval: 1.00000000000000000000
Class: land-21; eval: 1.00000000000000000000
Class: loadmodule-9; eval: 1.00000000000000000000
Class: multihop-7; eval: 0.85714285714285709528
Class: neptune-960; eval: 1.00000000000000000000
Class: nmap-231; eval: 0.98701298701298700866
Class: normal-960; eval: 0.92500000000000004441
Class: perl-3; eval: 1.00000000000000000000
Class: phf-4; eval: 1.00000000000000000000
Class: pod-264; eval: 1.00000000000000000000
Class: portsweep-1040; eval: 0.99903846153846154188
Class: rootkit-10; eval: 0.90000000000000002220
Class: satan-1589; eval: 1.00000000000000000000
Class: smurf-960; eval: 1.00000000000000000000
Class: spy-2; eval: 1.00000000000000000000
Class: teardrop-979; eval: 1.00000000000000000000
Class: warezclient-1020; eval: 1.00000000000000000000
Class: warezmaster-20; eval: 1.00000000000000000000
zero count: 0
time: 2.272826

s = 0.025

t
Class: back-1600; eval: 0.99937500000000001332
Class: buffer_overflow-30; eval: 0.90000000000000002220
Class: ftp_write-8; eval: 0.87500000000000000000
Class: guess_passwd-53; eval: 1.00000000000000000000
Class: imap-12; eval: 1.00000000000000000000
Class: ipsweep-1247; eval: 0.99919807538091420795
Class: land-21; eval: 1.00000000000000000000
Class: loadmodule-9; eval: 1.00000000000000000000
Class: multihop-7; eval: 0.85714285714285709528
Class: neptune-960; eval: 1.00000000000000000000
Class: nmap-231; eval: 0.97835497835497831076
Class: normal-960; eval: 0.97624999999999995115
Class: perl-3; eval: 1.00000000000000000000
Class: phf-4; eval: 1.00000000000000000000
Class: pod-264; eval: 1.00000000000000000000
Class: portsweep-1040; eval: 0.99903846153846154188
Class: rootkit-10; eval: 0.90000000000000002220
Class: satan-1589; eval: 1.00000000000000000000
Class: smurf-960; eval: 1.00000000000000000000
Class: spy-2; eval: 1.00000000000000000000
Class: teardrop-979; eval: 1.00000000000000000000
Class: warezclient-1020; eval: 1.00000000000000000000
Class: warezmaster-20; eval: 1.00000000000000000000
zero count: 0
time: 3.096189

v
Class: back-1600; eval: 1.00000000000000000000
Class: buffer_overflow-30; eval: 0.90000000000000002220
Class: ftp_write-8; eval: 0.87500000000000000000
Class: guess_passwd-53; eval: 1.00000000000000000000
Class: imap-12; eval: 1.00000000000000000000
Class: ipsweep-1247; eval: 0.99919807538091420795
Class: land-21; eval: 1.00000000000000000000
Class: loadmodule-9; eval: 1.00000000000000000000
Class: multihop-7; eval: 0.85714285714285709528
Class: neptune-960; eval: 1.00000000000000000000
Class: nmap-231; eval: 0.97835497835497831076
Class: normal-960; eval: 0.93500000000000005329
Class: perl-3; eval: 1.00000000000000000000
Class: phf-4; eval: 1.00000000000000000000
Class: pod-264; eval: 1.00000000000000000000
Class: portsweep-1040; eval: 0.99903846153846154188
Class: rootkit-10; eval: 0.90000000000000002220
Class: satan-1589; eval: 1.00000000000000000000
Class: smurf-960; eval: 1.00000000000000000000
Class: spy-2; eval: 1.00000000000000000000
Class: teardrop-979; eval: 1.00000000000000000000
Class: warezclient-1020; eval: 1.00000000000000000000
Class: warezmaster-20; eval: 1.00000000000000000000
zero count: 0
time: 2.230893

s = 0.035

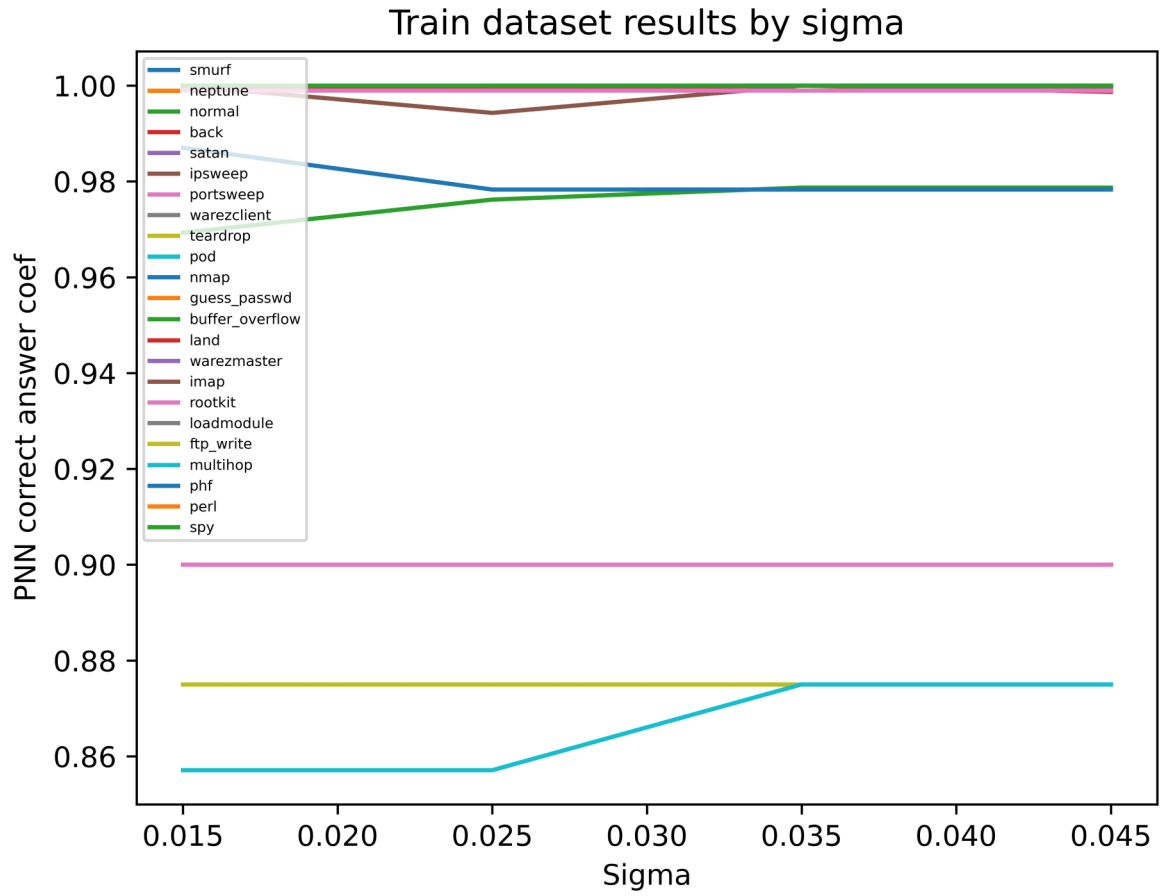
t
Class: back-1600; eval: 1.0000000000000000000
Class: buffer_overflow-30; eval: 0.90000000000000002220
Class: ftp_write-8; eval: 0.87500000000000000000
Class: guess_passwd-53; eval: 1.00000000000000000000
Class: imap-12; eval: 1.00000000000000000000
Class: ipsweep-1247; eval: 0.99759422614274262386
Class: land-21; eval: 1.00000000000000000000
Class: loadmodule-9; eval: 1.00000000000000000000
Class: multihop-7; eval: 0.85714285714285709528
Class: neptune-960; eval: 1.00000000000000000000
Class: nmap-231; eval: 0.97835497835497831076
Class: normal-960; eval: 0.97875000000000000888
Class: perl-3; eval: 1.00000000000000000000
Class: phf-4; eval: 1.00000000000000000000
Class: pod-264; eval: 1.00000000000000000000
Class: portsweep-1040; eval: 0.99903846153846154188
Class: rootkit-10; eval: 0.90000000000000002220
Class: satan-1589; eval: 1.00000000000000000000
Class: smurf-960; eval: 1.00000000000000000000
Class: spy-2; eval: 1.00000000000000000000
Class: teardrop-979; eval: 1.00000000000000000000
Class: warezclient-1020; eval: 1.00000000000000000000
Class: warezmaster-20; eval: 1.00000000000000000000
zero count: 0
time: 3.171035

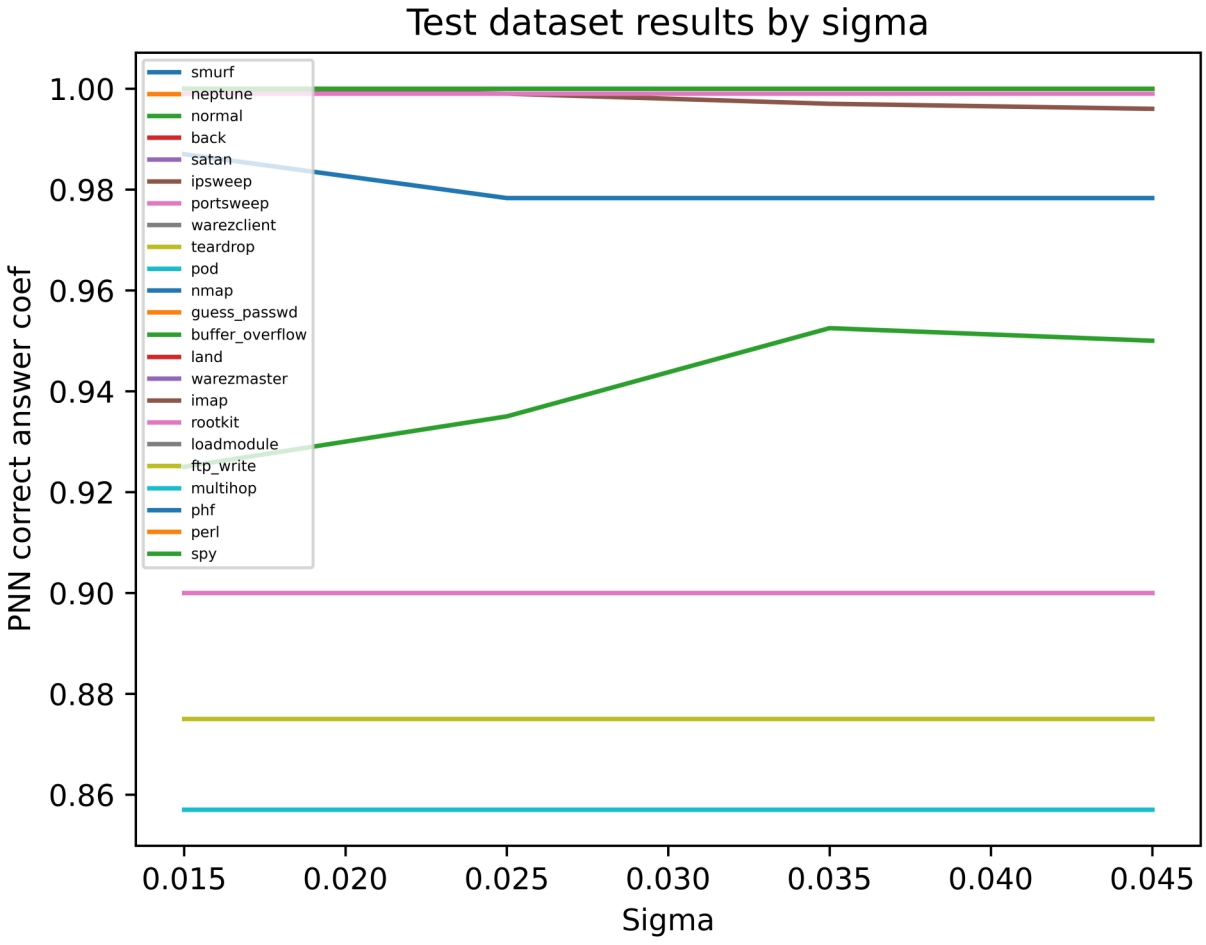
v
Class: back-1600; eval: 1.0000000000000000000
Class: buffer_overflow-30; eval: 0.90000000000000002220
Class: ftp_write-8; eval: 0.87500000000000000000
Class: guess_passwd-53; eval: 1.00000000000000000000
Class: imap-12; eval: 1.00000000000000000000
Class: ipsweep-1247; eval: 0.99759422614274262386
Class: land-21; eval: 1.00000000000000000000
Class: loadmodule-9; eval: 1.00000000000000000000
Class: multihop-7; eval: 0.85714285714285709528
Class: neptune-960; eval: 1.00000000000000000000
Class: nmap-231; eval: 0.97835497835497831076
Class: normal-960; eval: 0.95250000000000001332
Class: perl-3; eval: 1.00000000000000000000
Class: phf-4; eval: 1.00000000000000000000
Class: pod-264; eval: 1.00000000000000000000
Class: portsweep-1040; eval: 0.99903846153846154188
Class: rootkit-10; eval: 0.90000000000000002220
Class: satan-1589; eval: 1.00000000000000000000
Class: smurf-960; eval: 1.00000000000000000000
Class: spy-2; eval: 1.00000000000000000000
Class: teardrop-979; eval: 1.00000000000000000000
Class: warezclient-1020; eval: 1.00000000000000000000
Class: warezmaster-20; eval: 1.00000000000000000000
zero count: 0
time: 2.225466

s = 0.045

t
Class: back-1600; eval: 0.99875000000000002665
Class: buffer_overflow-30; eval: 0.90000000000000002220
Class: ftp_write-8; eval: 0.87500000000000000000
Class: guess_passwd-53; eval: 1.00000000000000000000
Class: imap-12; eval: 1.00000000000000000000
Class: ipsweep-1247; eval: 0.99679230152365672080
Class: land-21; eval: 1.00000000000000000000
Class: loadmodule-9; eval: 1.00000000000000000000
Class: multihop-7; eval: 0.85714285714285709528
Class: neptune-960; eval: 1.00000000000000000000
Class: nmap-231; eval: 0.97835497835497831076
Class: normal-960; eval: 0.97812500000000002220
Class: perl-3; eval: 1.00000000000000000000
Class: phf-4; eval: 1.00000000000000000000
Class: pod-264; eval: 1.00000000000000000000
Class: portsweep-1040; eval: 0.99903846153846154188
Class: rootkit-10; eval: 0.90000000000000002220
Class: satan-1589; eval: 1.00000000000000000000
Class: smurf-960; eval: 1.00000000000000000000
Class: spy-2; eval: 1.00000000000000000000
Class: teardrop-979; eval: 1.00000000000000000000
Class: warezclient-1020; eval: 1.00000000000000000000
Class: warezmaster-20; eval: 1.00000000000000000000
zero count: 0
time: 3.217059

v
Class: back-1600; eval: 1.0000000000000000000
Class: buffer_overflow-30; eval: 0.90000000000000002220
Class: ftp_write-8; eval: 0.87500000000000000000
Class: guess_passwd-53; eval: 1.00000000000000000000
Class: imap-12; eval: 1.00000000000000000000
Class: ipsweep-1247; eval: 0.99679230152365672080
Class: land-21; eval: 1.00000000000000000000
Class: loadmodule-9; eval: 1.00000000000000000000
Class: multihop-7; eval: 0.85714285714285709528
Class: neptune-960; eval: 1.00000000000000000000
Class: nmap-231; eval: 0.97835497835497831076
Class: normal-960; eval: 0.94999999999999995559
Class: perl-3; eval: 1.00000000000000000000
Class: phf-4; eval: 1.00000000000000000000
Class: pod-264; eval: 1.00000000000000000000
Class: portsweep-1040; eval: 0.99903846153846154188
Class: rootkit-10; eval: 0.90000000000000002220
Class: satan-1589; eval: 1.00000000000000000000
Class: smurf-960; eval: 1.00000000000000000000
Class: spy-2; eval: 1.00000000000000000000
Class: teardrop-979; eval: 1.00000000000000000000
Class: warezclient-1020; eval: 1.00000000000000000000
Class: warezmaster-20; eval: 1.00000000000000000000
zero count: 0
time: 2.195528


Train dataset results by sigma
smurf
neptune
normal
back
satan
ipsweep
portsweep
warezclient
teardrop
pod
nmap
guess_passwd
buffer_overflow
land
warezmaster
imap
rootkit
loadmodule
ftp_write
multihop
phf
perl
spy
PNN correct answer coef
1.00
0.98
0.96
0.94
0.92
0.90
0.88
0.86
0.015
0.020
0.025
0.030
0.035
0.040
0.045
Sigma


Test dataset results by sigma
smurf
neptune
normal
back
satan
ipsweep
portsweep
warezclient
teardrop
pod
nmap
guess_passwd
buffer_overflow
land
warezmaster
imap
rootkit
loadmodule
ftp_write
multihop
phf
perl
spy
PNN correct answer coef
1.00
0.98
0.96
0.94
0.92
0.90
0.88
0.86
0.015
0.020
0.025
0.030
0.035
0.040
0.045
Sigma

Виходячи з наведених графіків, в якості оптимального значення сігми було обрано 0.035.
Протестуємо мережу на всьому датасеті:

v
Class: back-1600; eval: 1.0000000000000000000
Class: buffer_overflow-30; eval: 0.90000000000000002220
Class: ftp_write-8; eval: 0.87500000000000000000
Class: guess_passwd-53; eval: 1.00000000000000000000
Class: imap-12; eval: 1.00000000000000000000
Class: ipsweep-1247; eval: 0.99759422614274262386
Class: land-21; eval: 1.00000000000000000000
Class: loadmodule-9; eval: 1.00000000000000000000
Class: multihop-7; eval: 0.85714285714285709528
Class: neptune-960; eval: 0.99978544976259553501
Class: nmap-231; eval: 0.97835497835497831076
Class: normal-960; eval: 0.95845883405121456988
Class: perl-3; eval: 1.00000000000000000000
Class: phf-4; eval: 1.00000000000000000000
Class: pod-264; eval: 1.00000000000000000000
Class: portsweep-1040; eval: 0.99903846153846154188
Class: rootkit-10; eval: 0.90000000000000002220
Class: satan-1589; eval: 1.00000000000000000000
Class: smurf-960; eval: 0.99919868941201606116
Class: spy-2; eval: 1.00000000000000000000
Class: teardrop-979; eval: 1.00000000000000000000
Class: warezclient-1020; eval: 1.00000000000000000000
Class: warezmaster-20; eval: 1.00000000000000000000
zero count: 0
time: 143.127316

На всіх класах, крім ftp_write, multihop, buffer_overflow, rootkit маємо точність, більшу за 95%. На інших класах точність є не меншою за 85%.
Окремо варто виділити той факт, що тестування мережі, загальна кількість зразків в якій дорівнює 11029, на датасеті розміром 494020 зразків, зайняло всього лиш ~143 секунди.

В оригінальному датасеті клас normal містить 97277 зразків. Клас normal відповідає нормальному трафіку. В наведеній нейронній мережі наявно лише 960 зразків класу normal. Значення характеристик зразків, що належать класу normal, можуть сильно варіюватись, оскільки характер «нормального» трафіку визначається лише діями користувачів. Збільшимо кількість зразків класу normal до 3600 (за рахунок зміни значення REAL_DATA_COUNT_PER_CLASS, зміна цього параметра також впливає на кількість прикладів у pnn для інших класів).

Параметри генерації даних:
```
PNN_REFERENCE_COUNT_BY_CLASS = {
    "smurf":          0.6,
    "neptune":        0.6,
    "normal":         0.6,
    "back":           1.0,
    "satan":          1.0,
    "ipsweep":        1.0,
    "portsweep":      1.0,
    "warezclient":    1.0,
    "teardrop":       1.0,
    "pod":            1.0,
    "nmap":           1.0,
    "guess_passwd":   1.0,
    "buffer_overflow": 1.0,
    "land":           1.0,
    "warezmaster":    1.0,
    "imap":           1.0,
    "rootkit":        1.0,
    "loadmodule":     1.0,
    "ftp_write":      1.0,
    "multihop":       1.0,
    "phf":            1.0,
    "perl":           1.0,
    "spy":            1.0
}
TEST_SIZE = 0.25
REAL_DATA_COUNT_PER_CLASS = 8000
RANDOM_STATE_1 = 1450
```

v
Class: back-2203; eval: 0.99954607353608715403
Class: buffer_overflow-30; eval: 0.90000000000000002220
Class: ftp_write-8; eval: 0.87500000000000000000
Class: guess_passwd-53; eval: 1.00000000000000000000
Class: imap-12; eval: 1.00000000000000000000
Class: ipsweep-1247; eval: 0.99759422614274262386
Class: land-21; eval: 1.00000000000000000000
Class: loadmodule-9; eval: 1.00000000000000000000
Class: multihop-7; eval: 0.85714285714285709528
Class: neptune-3600; eval: 0.99977612149140404618
Class: nmap-231; eval: 0.97835497835497831076
Class: normal-3600; eval: 0.98229797382731787181
Class: perl-3; eval: 1.00000000000000000000
Class: phf-4; eval: 1.00000000000000000000
Class: pod-264; eval: 1.00000000000000000000
Class: portsweep-1040; eval: 0.99903846153846154188
Class: rootkit-10; eval: 0.90000000000000002220
Class: satan-1589; eval: 0.99937067337948393142
Class: smurf-3600; eval: 0.99949784536486341313
Class: spy-2; eval: 1.00000000000000000000
Class: teardrop-979; eval: 1.00000000000000000000
Class: warezclient-1020; eval: 1.00000000000000000000
Class: warezmaster-20; eval: 1.00000000000000000000
zero count: 0
time: 240.441858

Спостерігаємо очікуване покращення результату для класу normal:
0.98229797382731787181 (3600 зразків normal) проти 0.95845883405121456988 (960 зразків normal).

Текстове представлення остаточної мережі—файл pnn-8000.txt у репозиторії.

Варто зауважити, що мережа помилково розпізнає деякі приклади класів малої розмірності (buffer_overflow, ftp_write, multihop, rootkit). В роботі було показано що показники розпізнавання таких класів можна покращити, значно зменшивши сігму. Але в такому випадку значно погіршуються показники «основних» класів.

Лог процесу навчання—файл report-session.txt у репозиторії.

Окремо хотілось виділити швидкодію мережі:
Розпізнавання 494020 зразків мережею, що містить 11029 зразків, займає 143 секунди. Розпізнавання одного зразка— ~1.5 мілісекунд.
Розпізнавання 494020 зразків мережею, що містить 19552 зразків, займає 240 секунди. Розпізнавання одного зразка— ~2.8 мілісекунд.

Тестування здійснювалось на процесорі Intel Core I5-8250U (1.6GHz—3.4GHz, 4 cores / 8 threads).

Таким чином використання мови C та технології OpenMP для реалізації мережі є виправданим.

p.s.

Як виявилось, за посиланням Kddcup99 - Dataset - DataHub - Frictionless Data—лише вибірка розміром 10% із оригінального датасету KDD99. Оригінальний датасет Kdd99 можна знайти за посиланням: KDD Cup 1999 Data (uci.edu). Цей датасет містить 4898431 зразків.
Кількість зразків кожного класу в датасеті:

```
>>> Class | Count >>>
label
smurf              2807886
neptune            1072017
normal              972781
satan                15892
ipsweep              12481
portsweep            10413
nmap                  2316
back                  2203
warezclient           1020
teardrop               979
pod                    264
guess_passwd            53
buffer_overflow         30
land                    21
warezmaster             20
imap                    12
rootkit                 10
loadmodule               9
ftp_write                8
multihop                 7
phf                      4
perl                     3
spy                      2
Name: count, dtype: int64
```

Протестуємо остаточну мережу (представлення—файл pnn-8000.txt у репозиторії) на всіх ~4.8M прикладів. Результат:

```
v
Class: back-2203; eval: 0.99773036768043577016
Class: buffer_overflow-30; eval: 0.86666666666666669627
Class: ftp_write-8; eval: 0.87500000000000000000
Class: guess_passwd-53; eval: 1.00000000000000000000
Class: imap-12; eval: 1.00000000000000000000
Class: ipsweep-1247; eval: 0.99679512859546515191
Class: land-21; eval: 1.00000000000000000000
Class: loadmodule-9; eval: 1.00000000000000000000
Class: multihop-7; eval: 0.71428571428571430157
Class: neptune-3600; eval: 0.99978171987944219889
Class: nmap-231; eval: 0.96243523316062173922
Class: normal-3600; eval: 0.98279880055223123314
Class: perl-3; eval: 1.00000000000000000000
Class: phf-4; eval: 1.00000000000000000000
Class: pod-264; eval: 1.00000000000000000000
Class: portsweep-1040; eval: 0.99654278305963694962
Class: rootkit-10; eval: 0.90000000000000002220
Class: satan-1589; eval: 0.99496602063931538495
Class: smurf-3600; eval: 0.99951743055095543244
Class: spy-2; eval: 1.00000000000000000000
Class: teardrop-979; eval: 1.00000000000000000000
Class: warezclient-1020; eval: 0.85882352941176465233
Class: warezmaster-20; eval: 1.00000000000000000000
zero count: 0
time: 2349.750440
```

Точність розпізнавання прикладів мережею є задовільна: для більшості класів точність є більшою за 95%. Винятком є класи ftp_write, multihop, warezclient.
В якості подальших покращень можна запропонувати незначно збільшити кількість зразків класу warezclient в мережі. Враховуючи, що всі приклади класів ftp_write та multihop наявні в мережі (7 та 8 прикладів відповідно), покращити результати розпізнавання цих класів можна лише зменшивши сігму—але тоді значно погіршаться показники інших класів.

Окремо варто відмітити, що мережа розміром 19552 зразків розпізнала ~4.8M прикладів за 2349.750440 / 60 = ~39.16 хвилин.

Selected files

7 printable files

pnnlib\pch.c
pnnlib\pch.h
pnnlib\pnn_alloc_check.h
pnnlib\pnn_core.c
pnnlib\pnn_io.c
pnnlib\pnn_memory.c
pnnlib\pnn.h

pnnlib\pch.c

```c
// pch.c: source file corresponding to the pre-compiled header

#include "pch.h"

// When you are using pre-compiled headers, this source file is necessary for compilation to succeed.

```

pnnlib\pch.h

```c
// pch.h: This is a precompiled header file.
// Files listed below are compiled only once, improving build performance for future builds.
// This also affects IntelliSense performance, including code completion and many code browsing features.
// However, files listed here are ALL re-compiled if any one of them is updated between builds.
// Do not add files here that you will be updating frequently as this negates the performance advantage.

#ifndef PCH_H
#define PCH_H

#include "pnn.h"
#include "pnn_alloc_check.h"

#endif //PCH_H

```

pnnlib\pnn_alloc_check.h

```c
#pragma once

#include <stdio.h>
#include <stdlib.h>

/// <summary>
/// Macros to check for memory allocation fail.
/// </summary>
#define ALLOC_ERR_MSG "Error: memory allocation failed.\n"
```

```
#define pnn_fail_alloc_check(p)\
{\
    if (p == NULL)\
    {\
        fprintf(stderr, ALLOC_ERR_MSG);\
        abort();\
    }\
}


```

pnnlib\pnn_core.c

```
#include "pch.h"

#include "pnn.h"

#include "pnn_alloc_check.h"

#include <stdlib.h>
#include <math.h>
#include <omp.h>
#include <string.h>

#define MAX_SINGLE_THREAD 250

static double act(int property_count, pnn_reference * reference, double * input, double sigma);
static int index_of_largest_prediction(int n, pnn_prediction * prediction_arr);
static int are_class_names_equal(char * class_name1, char * class_name2);

pnn_prediction * pnn_predict(pnn_data * net, double * input)
{
    pnn_prediction * prediction_arr = (pnn_prediction *)malloc(net->class_count * sizeof(pnn_prediction));
    pnn_fail_alloc_check(prediction_arr);

    for (int i = 0; i < net->class_count; i++)
    {
        pnn_class * current_class = net->pnn_class_arr[i];

        prediction_arr[i].class_name = current_class->class_name;
        prediction_arr[i].prediction = 0;

        for (int j = 0; j < current_class->reference_count; j++)
        {
            prediction_arr[i].prediction += act(net->property_count,
                                                current_class->reference_arr[j], input, net->sigma);
        }
    }

    return prediction_arr;
}

```

```
40  pnn_evaluation * pnn_evaluate(pnn_data * net, pnn_data * data)
41  {
42      pnn_evaluation * evaluation_arr = (pnn_evaluation *)malloc(data->class_count *
    sizeof(pnn_evaluation));
43      pnn_fail_alloc_check(evaluation_arr);
44
45      for (int i = 0; i < data->class_count; i++)
46      {
47          pnn_class * current_class = data->pnn_class_arr[i];
48          pnn_evaluation * current_evaluation = &(evaluation_arr[i]);
49
50          current_evaluation->class_name = current_class->class_name;
51          current_evaluation->accuracy = 0;
52
53          if (current_class->reference_count > MAX_SINGLE_THREAD)
54          {
55              double acc = 0;
56              int j;
57              #pragma omp parallel for reduction(+:acc)
58              for (j = 0; j < current_class->reference_count; j++)
59              {
60                  pnn_reference * current_reference = current_class->reference_arr[j];
61
62                  pnn_prediction * prediction_arr = pnn_predict(net, current_reference->
    reference);
63                  int k = index_of_largest_prediction(net->class_count, prediction_arr);
64
65                  if (are_class_names_equal(current_class->class_name, prediction_arr[k]
    .class_name) == 0)
66                      acc += 1.0;
67
68                  free(prediction_arr);
69              }
70
71              current_evaluation->accuracy = acc / (double)(current_class->reference_count);
72          }
73          else
74          {
75              for (int j = 0; j < current_class->reference_count; j++)
76              {
77                  pnn_reference * current_reference = current_class->reference_arr[j];
78
79                  pnn_prediction * prediction_arr = pnn_predict(net, current_reference->
    reference);
80                  int k = index_of_largest_prediction(net->class_count, prediction_arr);
81
82                  if (are_class_names_equal(current_class->class_name, prediction_arr[k]
    .class_name) == 0)
83                      current_evaluation->accuracy += 1.0;
84
85                  free(prediction_arr);
86              }
87
88              current_evaluation->accuracy /= (double)(current_class->reference_count);
89          }
90      }
91
```

```
92      return evaluation_arr;
93  }
94
95  static double act(int property_count, pnn_reference * reference, double * input, double
    sigma)
96  {
97      double acc = 0, d;
98      for (int i = 0; i < property_count; i++)
99      {
100         d = reference->reference[i] - input[i];
101         acc += d * d;
102     }
103     acc /= -(sigma * sigma);
104
105     return exp(acc);
106 }
107
108 static int index_of_largest_prediction(int n, pnn_prediction * prediction_arr)
109 {
110     int i = 0;
111     for (int j = 0; j < n; j++)
112     {
113         if (prediction_arr[i].prediction < prediction_arr[j].prediction)
114             i = j;
115     }
116     return i;
117 }
118
119 static int are_class_names_equal(char * class_name1, char * class_name2)
120 {
121     char * dash1 = strchr(class_name1, '-');
122     char * dash2 = strchr(class_name2, '-');
123
124     size_t n = dash1 - class_name1;
125     if (n != dash2 - class_name2)
126         return 1;
127
128     return strncmp(class_name1, class_name2, n);
129 }
130
```

pnnlib\pnn_io.c

```
1   #include "pch.h"
2
3   #include "pnn.h"
4
5   #include "pnn_alloc_check.h"
6
7   #include <stdio.h>
8   #include <stdlib.h>
9
10  pnn_data * pnn_data_load(FILE * f)
11  {
12      int property_count;
```

```
    int total_reference_count;
    int class_count;

    fread(&property_count, sizeof(int), 1, f);
    fread(&total_reference_count, sizeof(int), 1, f);
    fread(&class_count, sizeof(int), 1, f);

    pnn_class ** pnn_class_arr = (pnn_class **)malloc(class_count * sizeof(pnn_class *));
    pnn_fail_alloc_check(pnn_class_arr);
    for (struct
         {
             int i;
             size_t class_name_len; char * class_name;
             int reference_count;
         }
         state_class = { .i = 0, .class_name_len = 0, .class_name = NULL, .reference_count = 0 };
         state_class.i < class_count;
         state_class.i++)
    {
        fread(&(state_class.class_name_len), sizeof(size_t), 1, f);

        state_class.class_name = (char *)malloc((state_class.class_name_len + 1) * sizeof(char));
        pnn_fail_alloc_check(state_class.class_name);
        fread(state_class.class_name, sizeof(char), state_class.class_name_len, f);
        state_class.class_name[state_class.class_name_len] = '\0';

        fread(&(state_class.reference_count), sizeof(int), 1, f);

        pnn_reference ** pnn_reference_arr =
            (pnn_reference **)malloc(state_class.reference_count * sizeof(pnn_reference *));
        pnn_fail_alloc_check(pnn_reference_arr);
        for (struct
             {
                 int j;
                 int reference_id; int property_count;
                 double * reference;
             }
             state_reference = { .j = 0, .reference_id = 0, .property_count = 0, .reference = NULL};
             state_reference.j < state_class.reference_count;
             state_reference.j++)
        {
            fread(&(state_reference.reference_id), sizeof(int), 1, f);
            fread(&(state_reference.property_count), sizeof(int), 1, f);

            state_reference.reference = (double *)malloc(state_reference.property_count * sizeof(double));
            pnn_fail_alloc_check(state_reference.reference);
            fread(state_reference.reference, sizeof(double), state_reference.property_count, f);

            pnn_reference_arr[state_reference.j] = pnn_reference_create(state_reference.reference_id, state_reference.property_count,
state_reference.reference);
        }
```

```
64
65          pnn_class_arr[state_class.i] = pnn_class_create(state_class.class_name,
    state_class.reference_count, pnn_reference_arr);
66      }
67
68      pnn_data * data = pnn_data_create(DEFAULT_SIGMA, property_count, total_reference_count,
    class_count,
69                                        pnn_class_arr);
70
71      return data;
72  }
73
74  void pnn_data_fprint(pnn_data * data, FILE * f)
75  {
76      fprintf(f, "sigma = %lf; property_count = %d; total_reference_count = %d; class_count =
    %d\n",
77              data->sigma, data->property_count, data->total_reference_count, data->
    class_count);
78
79      for (struct
80           {
81               int i;
82               pnn_class * current_class;
83           }
84           state_class = { .i = 0, .current_class = NULL };
85           state_class.i < data->class_count; state_class.i++)
86      {
87          state_class.current_class = data->pnn_class_arr[state_class.i];
88
89          fprintf(f, "    class_name = %s; reference_count = %d\n",
90                  state_class.current_class->class_name, state_class.current_class->
    reference_count);
91
92          for (struct
93               {
94                  int j;
95                  pnn_reference * current_reference;
96               }
97               state_reference = { .j = 0, .current_reference = NULL };
98               state_reference.j < state_class.current_class->reference_count;
    state_reference.j++)
99          {
100             state_reference.current_reference = state_class.current_class->
    reference_arr[state_reference.j];
101
102             fprintf(f, "        id = %d; property_count = %d; ",
103                     state_reference.current_reference->id, state_reference.current_reference-
    >property_count);
104
105             for (int k = 0; k < state_reference.current_reference->property_count - 1; k++)
106             {
107                 fprintf(f, "%.2lf,", state_reference.current_reference->reference[k]);
108             }
109             fprintf(f, "%.2lf\n", state_reference.current_reference->
    reference[state_reference.current_reference->property_count - 1]);
110         }
111     }
112 }
```

113

pnnlib\pnn_memory.c

```c
#include "pch.h"

#include "pnn.h"

#include "pnn_alloc_check.h"

#include <stdlib.h>

pnn_reference * pnn_reference_create(int id, int property_count, double * reference)
{
    pnn_reference * obj = (pnn_reference *)malloc(sizeof(pnn_reference));
    pnn_fail_alloc_check(obj);

    obj->id = id;
    obj->property_count = property_count;
    obj->reference = reference;

    return obj;
}

pnn_reference * pnn_reference_free(pnn_reference * obj)
{
    free(obj->reference);
    free(obj);

    return NULL;
}

pnn_class * pnn_class_create(char * class_name, int reference_count, pnn_reference **
reference_arr)
{
    pnn_class * obj = (pnn_class *)malloc(sizeof(pnn_class));
    pnn_fail_alloc_check(obj);

    obj->class_name = class_name;
    obj->reference_count = reference_count;
    obj->reference_arr = reference_arr;

    return obj;
}

pnn_class * pnn_class_free(pnn_class * obj)
{
    free(obj->class_name);
    for (int i = 0; i < obj->reference_count; i++)
        pnn_reference_free(obj->reference_arr[i]);
    free(obj);

    return NULL;
}

```

```
pnn_data * pnn_data_create(double sigma, int property_count, int total_reference_count, int class_count,
                           pnn_class ** pnn_class_arr)
{
    pnn_data * obj = (pnn_data *)malloc(sizeof(pnn_data));
    pnn_fail_alloc_check(obj);

    obj->sigma = sigma;
    obj->property_count = property_count;
    obj->total_reference_count = total_reference_count;
    obj->class_count = class_count;
    obj->pnn_class_arr = pnn_class_arr;

    return obj;
}

pnn_data * pnn_data_free(pnn_data * obj)
{
    for (int i = 0; i < obj->class_count; i++)
        pnn_class_free(obj->pnn_class_arr[i]);
    free(obj);

    return NULL;
}

```

pnnlib\pnn.h

```
#pragma once

#include <stdio.h>

#define DEFAULT_SIGMA 0.01
#define EPSILON 0.001

typedef struct pnn_reference
{
    int id;
    int property_count;
    double * reference;
} pnn_reference;

typedef struct pnn_class
{
    char * class_name;
    int reference_count;
    struct pnn_reference ** reference_arr;
} pnn_class;

typedef struct pnn_data
{
    double sigma;
    int property_count;
    int total_reference_count;
    int class_count;
```

```
28      struct pnn_class ** pnn_class_arr;
29  }pnn_data;
30
31  pnn_reference * pnn_reference_create(int id, int property_count, double * reference);
32  pnn_reference * pnn_reference_free(pnn_reference * obj);
33
34  pnn_class * pnn_class_create(char * class_name, int reference_count, pnn_reference **
    reference_arr);
35  pnn_class * pnn_class_free(pnn_class * obj);
36
37  pnn_data * pnn_data_create(double sigma, int property_count, int total_reference_count, int
    class_count,
38                             pnn_class ** pnn_class_arr);
39  pnn_data * pnn_data_free(pnn_data * obj);
40
41  pnn_data * pnn_data_load(FILE * f);
42  void pnn_data_fprint(pnn_data * data, FILE * f);
43
44  typedef struct
45  {
46      char * class_name;
47      double prediction;
48  }
49  pnn_prediction;
50
51  typedef struct
52  {
53      char * class_name;
54      double accuracy;
55  }
56  pnn_evaluation;
57
58  pnn_prediction * pnn_predict(pnn_data * data, double * input);
59  pnn_evaluation * pnn_evaluate(pnn_data * net, pnn_data * data);
60
```

data-processing.py

```python
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from shutil import rmtree
from os.path import exists, isdir
from os import makedirs

DATA_NAME = "original_data.csv"
OUT_DATA_PREFIX = "/data"
PNN_OUTPUT_DIR = "./out-pnn"
PNN_REFERENCE_OUTPUT_DIR = PNN_OUTPUT_DIR + OUT_DATA_PREFIX
TRAIN_OUTPUT_DIR = "./out-train"
TRAIN_REFERENCE_OUTPUT_DIR = TRAIN_OUTPUT_DIR + OUT_DATA_PREFIX
TEST_OUTPUT_DIR = "./out-test"
TEST_REFERENCE_OUTPUT_DIR = TEST_OUTPUT_DIR + OUT_DATA_PREFIX

PNN_REFERENCE_COUNT_BY_CLASS = {
    "smurf":           0.6,
    "neptune":         0.6,
    "normal":          0.6,
    "back":            1.0,
    "satan":           1.0,
    "ipsweep":         1.0,
    "portsweep":       1.0,
    "warezclient":     1.0,
    "teardrop":        1.0,
    "pod":             1.0,
    "nmap":            1.0,
    "guess_passwd":    1.0,
    "buffer_overflow": 1.0,
    "land":            1.0,
    "warezmaster":     1.0,
    "imap":            1.0,
    "rootkit":         1.0,
    "loadmodule":      1.0,
    "ftp_write":       1.0,
    "multihop":        1.0,
    "phf":             1.0,
    "perl":            1.0,
    "spy":             1.0
}
TEST_SIZE = 0.25
REAL_DATA_COUNT_PER_CLASS = 8000
RANDOM_STATE = 1450

# Read data
data = pd.read_csv(DATA_NAME)

# Show column info
print(">>> Original data >>>")
for column in data.columns:
    unique_values_count = data[column].nunique()
```

```
    unique_values = data[column].unique()
    print(f"Column: {column}")
    print(f"Unique Values Count: {unique_values_count}")
    print(f"Unique Values: {unique_values}")
    print()
print(">>> Column types >>>\n", data.dtypes, "\n")

# Show unique classes and their count
print(">>> Class | Count >>>")
print(data["label"].value_counts(), "\n")

# Drop lnum_outbound_cmds and is_host_login columns since values are same for all rows
data = data.drop(columns = ["lnum_outbound_cmds", "is_host_login"])

# Convert protocol_type, service, flag columns to numeric
nonnumeric_columns = ["protocol_type", "service", "flag"]
label_encoder = LabelEncoder()
for column in nonnumeric_columns:
    data[column] = label_encoder.fit_transform(data[column])

# normalize data
for column_name in data.columns:
    if column_name != 'label':
        column = data[column_name]
        min_val = column.min()
        max_val = column.max()
        data[column_name] = (column - min_val) / (max_val - min_val)

# Show column info
print(">>> Data: removed useless columns, all numeric >>>")
for column in data.columns:
    unique_values_count = data[column].nunique()
    unique_values = data[column].unique()
    print(f"Column: {column}")
    print(f"Unique Values Count: {unique_values_count}")
    print(f"Unique Values: {unique_values}")
    print()
print(">>> Column types >>>\n", data.dtypes, "\n")

# Show unique classes and their count
print(">>> Class | Count >>>")
print(data["label"].value_counts(), "\n")

# Delete old run data and recreate out directory tree
if exists(PNN_OUTPUT_DIR):
    rmtree(PNN_OUTPUT_DIR)
makedirs(PNN_REFERENCE_OUTPUT_DIR)
if exists(TRAIN_OUTPUT_DIR):
    rmtree(TRAIN_OUTPUT_DIR)
makedirs(TRAIN_REFERENCE_OUTPUT_DIR)
if exists(TEST_OUTPUT_DIR):
    rmtree(TEST_OUTPUT_DIR)
makedirs(TEST_REFERENCE_OUTPUT_DIR)

# data export
property_count = len(data.columns.tolist()) - 1
```

```
data.insert(0, "property_count", property_count)
rows_by_class = dict(tuple(data.groupby("label")))
count_by_class = { "pnn": 0, "train" : 0, "test": 0 }
for k,k_data in rows_by_class.items():
    k_data.drop(columns = ["label"], inplace = True)
    if k_data.shape[0] > REAL_DATA_COUNT_PER_CLASS:
        # select real count from k_data
        k_data_real = k_data.sample(n = REAL_DATA_COUNT_PER_CLASS, random_state = RANDOM_STATE)

        # split
        k_data_train, k_data_test = train_test_split(k_data_real, test_size = TEST_SIZE, random_state = RANDOM_STATE)

        # take pnn data from train data
        k_data_pnn = k_data_train.sample(frac = PNN_REFERENCE_COUNT_BY_CLASS[k], random_state = RANDOM_STATE)

        # save data
        count_by_class["pnn"] += k_data_pnn.shape[0]
        k_data_pnn.to_csv(PNN_REFERENCE_OUTPUT_DIR + "/" + k + "-" + str(k_data_pnn.shape[0]) + ".csv",
                          header = False, index = True, lineterminator = ',')
        count_by_class["train"] += k_data_train.shape[0]
        k_data_train.to_csv(TRAIN_REFERENCE_OUTPUT_DIR + "/" + k + "-" + str(k_data_train.shape[0]) + ".csv",
                           header = False, index = True, lineterminator = ',')
        count_by_class["test"] += k_data_test.shape[0]
        k_data_test.to_csv(TEST_REFERENCE_OUTPUT_DIR + "/" + k + "-" + str(k_data_test.shape[0]) + ".csv",
                           header = False, index = True, lineterminator = ',')
    else:
        # save whole data as pnn, train, test
        count_by_class["pnn"] += k_data.shape[0]
        k_data.to_csv(PNN_REFERENCE_OUTPUT_DIR + "/" + k + "-" + str(k_data.shape[0]) + ".csv",
                          header = False, index = True, lineterminator = ',')
        count_by_class["train"] += k_data.shape[0]
        k_data.to_csv(TRAIN_REFERENCE_OUTPUT_DIR + "/" + k + "-" + str(k_data.shape[0]) + ".csv",
                           header = False, index = True, lineterminator = ',')
        count_by_class["test"] += k_data.shape[0]
        k_data.to_csv(TEST_REFERENCE_OUTPUT_DIR + "/" + k + "-" + str(k_data.shape[0]) + ".csv",
                           header = False, index = True, lineterminator = ',')

# save metadata
with open(PNN_OUTPUT_DIR + "/info.csv", "w", newline = "\n") as info:
    info.write(",".join([str(property_count),
                         str(count_by_class["pnn"]),
                         str(len(data["label"].unique()))]))
with open(TRAIN_OUTPUT_DIR + "/info.csv", "w", newline = "\n") as info:
    info.write(",".join([str(property_count),
                         str(count_by_class["train"]),
                         str(len(data["label"].unique()))]))
with open(TEST_OUTPUT_DIR + "/info.csv", "w", newline = "\n") as info:
    info.write(",".join([str(property_count),
                         str(count_by_class["test"]),
```

```
159                     str(len(data["label"].unique()))])
```