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

Звіт
із лабораторної роботи №3
з дисципліни «Системи глибинного навчання»
на тему: “Нейромережеве розпізнавання кібератак”
Варіант 7 – Розпізнавання мережевої кібератаки типу guess_passwd на базі PNN

Виконав:
студент групи КМ-01
Іваник Ю. П.

Викладач:
Професор кафедри ПМА
Терейковський І. А.

Київ — 2023

Зміст

Теоретичні відомості

Імовірнісна нейронна мережа (PNN) є прямопрогонною нейронною мережею, яка широко використовується в завданнях класифікації та визначення шаблонів. У PNN батьківська функція розподілу ймовірностей для кожного класу наближається за допомогою методу ядерних оцінок густини ймовірності з використанням Гаусових ядер.

Операції в PNN організовані у багатошарову прямопрогонну мережу з чотирма рівнями:

1. Вхідний рівень: Кожен нейрон цього рівня відображає змінну прогнозування.
2. Рівень шаблонів: Цей рівень включає один нейрон для кожного випадку в навчальному наборі даних.
3. Рівень додавання: В PNN присутній один нейрон шаблону для кожної категорії цільової змінної.
4. Вихідний рівень: Вихідний рівень порівнює зважені голоси для кожної цільової категорії, накопичені в рівні шаблонів, і використовує найвищий голос для прогнозування цільової категорії.

PNN часто використовують у завданнях класифікації, вони виникли як результат поєднання концепцій Байєсової мережі та статистичного алгоритму.

Основна частина

За індивідуальним варіантом завдання треба розробити нейронну мережу на базі PNN, яка буде розпізнавати мережеві кібератаки типу ‘guess_passwd’.

Для цього нам знадобиться датасет NSL-KDD, який містить дані про мережеві кібератаки та типи цих атак. Датасет було взято з інтернет ресурсу Kaggle.

Датасет містить такі назви колонок:

duration	***num_root***	***srv_diff_host_rate***
protocol_type	***num_file_creations***	***dst_host_count***
service	***num_shells***	***dst_host_srv_count***
flag	***num_access_files***	***dst_host_same_srv_rate***
src_bytes	***num_outbound_cmds***	***dst_host_diff_srv_rate***
dst_bytes	***is_host_login***	***dst_host_same_src_port_rate***
land	***is_guest_login***	***dst_host_srv_diff_host_rate***
wrong_fragment	***count***	***dst_host_serror_rate***
urgent	***srv_count***	***dst_host_srv_serror_rate***
hot	***serror_rate***	***dst_host_rerror_rate***
num_failed_logins	***srv_serror_rate***	***dst_host_srv_rerror_rate***
logged_in	***rerror_rate***	***attack***
num_compromised	***srv_rerror_rate***	***level***
root_shell	***same_srv_rate***	
su_attempted	***diff_srv_rate***	

Мітками типів атак є колонка 'attack'.

Завантажимо тренувальний та тестовий датасети. Об’єднаємо їх в один, залишимо лише ті записи, які мають атаку guess_passwd або normal. 75 % даних будуть навчальними, а 25 % тестовими.

```
Розмір навчального набору: 58752
Розмір тестового набору: 19585
```

Перевіримо скільки записів із типом атаки `guess_passwd`, скільки із `normal`.

```
Кількість записів із 'guess_passwd' у навчальному наборі : 963
Кількість записів із 'guess_passwd' у тестовому наборі   : 321
Кількість записів із 'normal'       у навчальному наборі : 57789
Кількість записів із 'normal'       у тестовому наборі   : 19264
```

Перекодуємо категорійні дані в числові

Подивимось чи співпадають колонки після автоматичного перекодування

```
Колонок у 'filtered_train_data': 78
Колонок у 'filtered_test_data':  76
Кількість спільних колонок: 75

Унікальні колонки в filtered_train_data:
{'flag_RSTOS0', 'flag_SH', 'service_link'}

Унікальні колонки в filtered_test_data:
{'service_remote_job'}
```

Колонки не співпадають. Подивимось скільки даних, які утворюють ці колонки

```
Кількість записів із 'link'       у навчальному наборі: 1
Кількість записів із 'SH'         у навчальному наборі: 2
Кількість записів із 'RSTOS0'     у навчальному наборі: 1
Кількість записів із 'remote_job' у тестовому наборі  : 1
```

Бачимо, що таких даних дуже мало порівняно з величиною датасету, тому просто видалимо ці записи.

Перекодовуємо знову

```
Колонок у 'filtered_train_data': 75
Колонок у 'filtered_test_data':  75
Кількість спільних колонок:      75
```

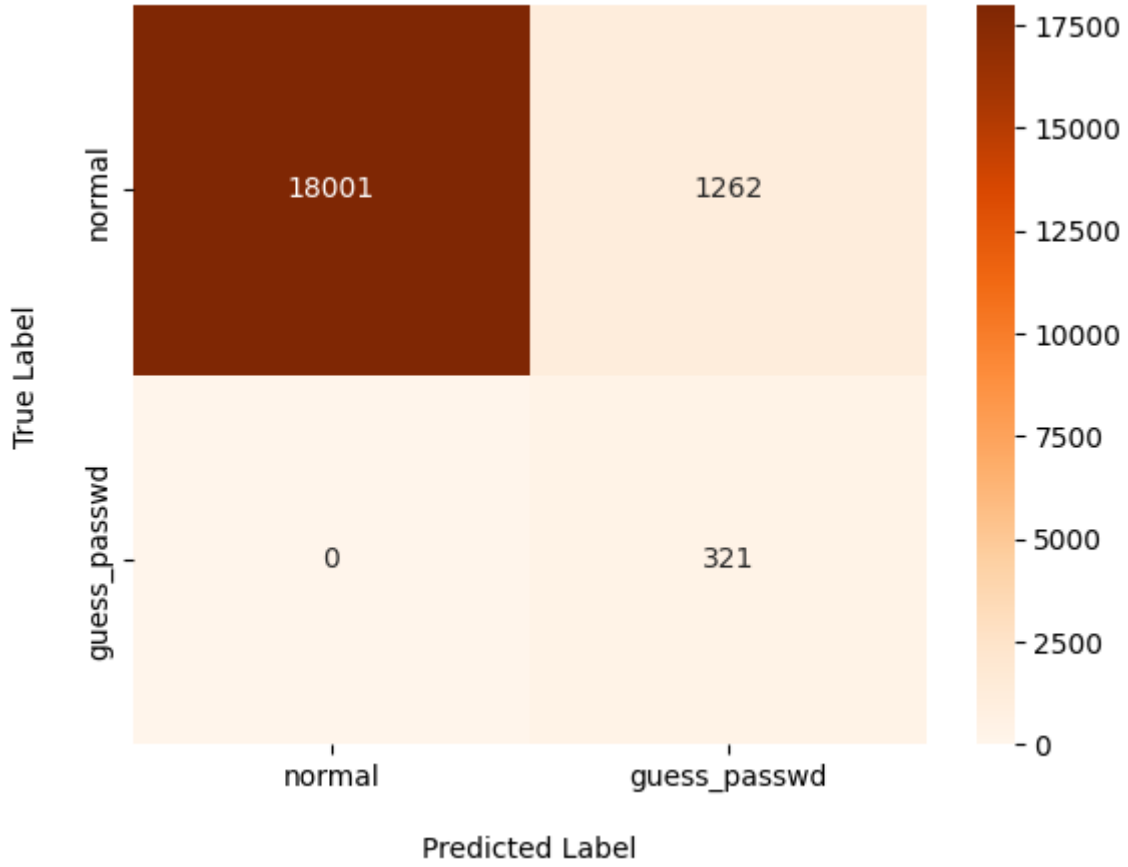
Нормалізовуємо дані, для передачі їх у PNN. Ініціалізуємо мережу.

Навчаємо PNN на навчальному наборі даних із параметром sigma = 1.0

Тестуємо мережу та подивимось метрики

```
Accuracy (Точність) : 93.55596 %
Precision (Точність): 20.27795 %
Recall (Повнота)    : 100.0 %
F1 Score (F-міра)   : 33.71849 %
```

Із цих метрик варто звернути увагу на повноту. Мережа виявила усі атаки, які дійсно були атаками, точність не настільки висока, тому намалюємо матрицю помилок та подивимось у скількох випадках модель помилилась



PNN помилилась у 1262 випадках. Але правильно класифікувала 321 атаку, вважаю, що цей результат є хорошим, оскільки основною задачею мережі є передбачати атаки, а вона передбачила усі ‘реальні’ атаки.

Перелік посилань

1. Руденко О.Г. Штучні нейронні мережі. Навч. посіб. / О. Г. Руденко, Є. В. Бодянський.
2. NSL-KDD - https://www.kaggle.com/datasets/hassan06/nslkdd/data?select=KDDTrain%2B_20Percent.txt

Додаток А – Код програми

```
# КМ-01, Іваник Юрій, Лаб 3

### Імпортуємо бібліотеки

import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.preprocessing import MinMaxScaler

from sklearn.model_selection import train_test_split

from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix

### Завантажимо датасет NSL-KDD

train_data = pd.read_csv('KDDTrain+.txt')

test_data = pd.read_csv('KDDTest+.txt')

### Даємо назви колонкам у датафреймі

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','attack','level'])

train_data.columns = columns

test_data.columns = columns

Об'єднуємо датасети -> Залишаємо лише `guess_passwd` та `normal` -> 75 % - навчання, 25 % тест

Поєднання двох датафреймів

combined_data = pd.concat([train_data, test_data], axis=0)

Розділення за типом атаки

attack_data = combined_data[combined_data['attack'] == 'guess_passwd']

normal_data = combined_data[combined_data['attack'] == 'normal']

Розбиття на навчальний та тестовий набір

train_attack, test_attack = train_test_split(attack_data, test_size=0.25, random_state=42)

train_normal, test_normal = train_test_split(normal_data, test_size=0.25, random_state=42)

Об'єднання навчальних та тестових наборів

train_data = pd.concat([train_attack, train_normal], axis=0)

test_data = pd.concat([test_attack, test_normal], axis=0)

Виведення розмірів навчального та тестового наборів

print(f'Розмір навчального набору: {train_data.shape[0]}')

```
print(f'Розмір тестового набору:  {test_data.shape[0]}')
### Перевіримо кількість записів які містять тип атаки 'guess_passwd' або 'normal'
count_train_guess_pass = (train_data['attack'] == 'guess_passwd').sum()
count_test_guess_pass = (test_data['attack'] == 'guess_passwd').sum()
count_train_normal = (train_data['attack'] == 'normal').sum()
count_test_normal = (test_data['attack'] == 'normal').sum()

print(f"Кількість записів із 'guess_passwd' у навчальному наборі : {count_train_guess_pass}")
print(f"Кількість записів із 'guess_passwd' у тестовому наборі : {count_test_guess_pass}")
print(f"Кількість записів із 'normal' у навчальному наборі : {count_train_normal}")
print(f"Кількість записів із 'normal' у тестовому наборі : {count_test_normal}")
### Перекодуємо категорійні значення
# Використання one-hot encoding для категорійних змінних
filtered_train_data = pd.get_dummies(train_data, drop_first=True)
filtered_test_data  = pd.get_dummies(test_data,  drop_first=True)
### Перевіримо чи співпадають назви колонок та їх кількість
Це важливо оскільки ми передаємо ці значення в нейронну мережу
print(f'Колонок у \'filtered_train_data\': {filtered_train_data.shape[1]}')
print(f'Колонок у \'filtered_test_data\':  {filtered_test_data.shape[1]}')

# Отримання назв колонок для обох датафреймів
columns_train = set(filtered_train_data.columns)
columns_test = set(filtered_test_data.columns)

# Знаходження спільних назв колонок
common_columns = columns_train.intersection(columns_test)
```

```
# Виведення назв та кількості спільних колонок
print('Кількість спільних колонок:', len(common_columns))

# Знаходження унікальних колонок у кожному датафреймі
unique_columns_train = columns_train.difference(columns_test)
unique_columns_test = columns_test.difference(columns_train)

# Виведення назв унікальних колонок у кожному датафреймі
print(f'\nУнікальні колонки в filtered_train_data: \n{unique_columns_train}')
print(f'\nУнікальні колонки в filtered_test_data: \n{unique_columns_test}')
### Подивимось наскільки багато цих даних і чи можемо ми їх позбутися
service_link = (train_data['service'] == 'link').sum()
flag_SH = (train_data['flag'] == 'SH').sum()
flag_RSTOS0 = (train_data['flag'] == 'RSTOS0').sum()

service_remote_job = (test_data['service'] == 'remote_job').sum()


print(f"Кількість записів із 'link'       у навчальному наборі: {service_link}")
print(f"Кількість записів із 'SH'         у навчальному наборі: {flag_SH}")
print(f"Кількість записів із 'RSTOS0'     у навчальному наборі: {flag_RSTOS0}")
print(f"Кількість записів із 'remote_job' у тестовому наборі: {service_remote_job}")
### Видаляємо дані
```

Тренувальний датасет містить більше 58 тис. даних, а тестовий датасет 19 тис.

Тому видалення такої кількості даних не повинно сильно вплинути на результати досліджень

```
train_data = train_data[(train_data['service'] != 'link') &
```

```
                (train_data['flag'] != 'SH') &
                (train_data['flag'] != 'RSTOS0')]

test_data = test_data[(test_data['service'] != 'remote_job')]
### Перевіримо чи зараз співпадають назви та кількість колонок
# Розділення 'attack' від інших змінних
attacks_train = train_data[['attack']]
attacks_test = test_data[['attack']]

# Видалення 'attack' з оригінального набору даних
train_data = train_data.drop(columns=['attack'])
test_data = test_data.drop(columns=['attack'])

# Перекодування за допомогою pd.get_dummies() для інших змінних
filtered_train_data = pd.get_dummies(train_data, drop_first=True)
filtered_test_data = pd.get_dummies(test_data,  drop_first=True)

# Додавання нового стовпця 'attack_guess_passwd'
filtered_train_data['attack_guess_passwd'] = attacks_train['attack'].map(lambda x: 1.0 if x == 'guess_passwd' else 0.0)
filtered_test_data['attack_guess_passwd'] = attacks_test['attack'].map(lambda x: 1.0 if x == 'guess_passwd' else 0.0)
print(f'Колонок у \'filtered_train_data\': {filtered_train_data.shape[1]}')
print(f'Колонок у \'filtered_test_data\':  {filtered_test_data.shape[1]}')

# Отримання назв колонок для обох датафреймів
columns_train = set(filtered_train_data.columns)
columns_test = set(filtered_test_data.columns)
```

```
# Знаходження спільних назв колонок
common_columns = columns_train.intersection(columns_test)

# Виведення назв та кількості спільних колонок
print(f'Кількість спільних колонок: \t {len(common_columns)}')
### Для коректної роботи НМ треба нормалізувати дані
scaler = MinMaxScaler()

new_columns = filtered_train_data.columns

filtered_train_data[new_columns] = scaler.fit_transform(filtered_train_data[new_columns])
filtered_test_data[new_columns] = scaler.transform(filtered_test_data[new_columns])
### Створюємо клас ймовірнісної нейронної мережі (PNN)
class PNN:
    def __init__(self, input_size, output_size):
        self.input_size = input_size
        self.output_size = output_size
        self.mean_vectors = None
        self.sigma = None
        self.weights = None


    def train(self, X, y, sigma=1.0):
        self.mean_vectors = []
        self.sigma = sigma

        # Обчислюємо середні вектори для кожного класу
```

```
        for class_label in range(self.output_size):
            class_samples = X[y == class_label]
            mean_vector = np.mean(class_samples, axis=0)
            self.mean_vectors.append(mean_vector)

        self.mean_vectors = np.array(self.mean_vectors)

        # Обчислюємо ваги для кожного класу
        self.weights = np.ones(self.output_size) / self.output_size



    def predict(self, X):
        predictions = []

        for sample in X:
            probabilities = []

            # Розраховуємо ймовірності для кожного класу
            for class_label in range(self.output_size):
                mean_vector = self.mean_vectors[class_label]
                sample = sample.astype(float)
                activation = np.exp(-0.5 * np.sum((sample - mean_vector) ** 2) / (self.sigma
** 2))
                probability = activation * self.weights[class_label]
                probabilities.append(probability)

            # Визначаємо клас з найвищою ймовірністю
            predicted_class = np.argmax(probabilities)
```

```
            predictions.append(predicted_class)

        return np.array(predictions)
### Розділимо датафрейми `filtered_train_data` та `filtered_test_data` на X_train, X_test, y_train, y_test
X_train = filtered_train_data.drop(columns=['attack_guess_passwd'])
y_train = filtered_train_data['attack_guess_passwd']

X_test = filtered_test_data.drop(columns=['attack_guess_passwd'])
y_test = filtered_test_data['attack_guess_passwd']
### Ініціалізуємо модель PNN
input_size = train_data.shape[1]
output_size = 2  # розмірність вихідного шару (два класи: guess_passwd та normal)

pnn = PNN(input_size, output_size)
### Навчання PNN
pnn.train(X_train, y_train, sigma=1.0)
### Використання PNN
predictions = pnn.predict(X_test.values)
### Знайдемо метрики, щоб зрозуміти наскільки хороших результатів було досягнуто
y_true = y_test
accuracy = accuracy_score(y_true, predictions)
precision = precision_score(y_true, predictions)
recall = recall_score(y_true, predictions)
f1 = f1_score(y_true, predictions)

print(f"Accuracy (Точність) : {round(accuracy * 100, 5)} %")
print(f"Precision (Точність): {round(precision * 100, 5)} %")
```

```
print(f"Recall (Повнота)    : {round(recall * 100, 5)} %")
print(f"F1 Score (F-міра)   : {round(f1 * 100, 5)} %")
### Побудуємо теплову карту з результатами дослідження
cm = confusion_matrix(y_true, predictions)
sns.heatmap(cm, annot=True, fmt="d", cmap="Oranges",
        xticklabels=['normal', 'guess_passwd'], yticklabels=['normal', 'guess_passwd'])
plt.xlabel("\nPredicted Label")
plt.ylabel("True Label\n")
plt.show()
```