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-03
Шаповалов Г. Г.

Викладач:
Професор кафедри ПМА
Терейковський І. А.

Київ — 2023

Зміст

Теоретичні відомості

Ймовірнісна нейронна мережа (PNN) - це прямопрогонна нейронна мережа, яка широко використовується у задачах класифікації та розпізнавання шаблонів. В алгоритмі PNN батьківська функція розподілу ймовірностей (PDF) кожного класу наближається за допомогою методу ядерних оцінок густини ймовірності, Гаусових функцій.

Операції в PNN організовані в багатошарову прямопрогонну мережу з чотирма шарами:

1. Вхідний шар: Кожен нейрон вхідного шару представляє змінну прогнозування.
2. Шар шаблонів: Цей шар містить один нейрон для кожного випадку в навчальному наборі даних.
3. Шар додавання: Для PNN є один нейрон шаблону для кожної категорії цільової змінної.
4. Вихідний шар: Вихідний шар порівнює зважені голоси для кожної цільової категорії, накопичені в шарі шаблонів, і використовує найбільший голос для прогнозування цільової категорії.

PNN часто використовуються в задачах класифікації. Вони були виведені з Байєсової мережі та статистичного алгоритму.

Основна частина

На основі датасету NSL-KDD [2] треба навчити та протестувати нейронну мережу PNN.

Назви колонок у датасеті:

```
'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'
```

Колонка - 'attack' відповідає за вид атаки. Якщо normal то атака не підтвердилась.

Оскільки стандартне розподілення на тестові та тренувальні дані має дуже велику різницю в балансі класів:

Кількість 'guess_passwd' у 'train_data': 53

Кількість 'guess_passwd' у 'test_data' : 1231

Кількість 'normal' у 'train_data': 67342

Кількість 'normal' у 'test_data' : 9711

Тому зробимо спільний датасет та розподілимо його так, щоб тренувальна вибірка була 80% від усього датасету. Тепер:

Кількість 'guess_passwd' у 'train_data': 1027

Кількість 'guess_passwd' у 'test_data' : 257

Кількість 'normal' у 'train_data': 61642

Кількість 'normal' у 'test_data' : 15411

Перекодування колонок із категорійних у числові відбувається за допомогою pandas.get_dummies()

Після кодування з’являються зайві стовпці, тому перевіримо як багато даних, які створюють ці стовпці:

Кількість 'link' у 'train_data': 1

Кількість 'shell' у 'train_data': 4

Кількість 'SH' у 'train_data': 2

Кількість 'RSTOS0' у 'train_data': 1

Кількість 'remote_job' у 'test_data' : 1

Як бачимо, цих даних не багато, порівняно з об’ємами датасету, тому видалимо їх. Тепер кількість колонок у навчальному та тестову датасеті однакова:

Колонок у 'filtered_train_data': 74

Колонок у 'filtered_test_data': 74

Кількість спільних колонок: 74

Створюємо нейронну мережу PNN у якої на виході буде 2 нейрони (атака, не атака). Навчаємо на тренувальних даних. Параметр сігма = 0.1

Тестуємо мережу та дивимось на метрики, щоб зрозуміти наскільки добре вона працює:

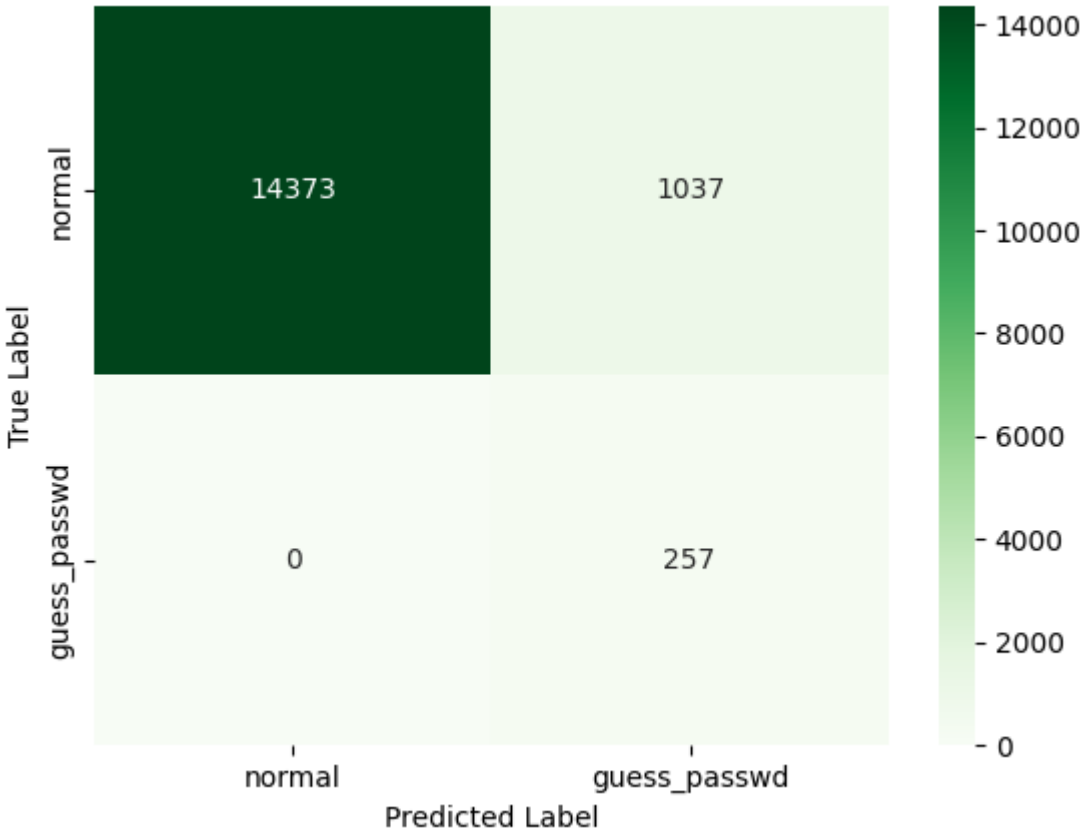
Accuracy (Точність) : 93.38099 %

Precision (Точність): 19.8609 %

Recall (Повнота) : 100.0 %

F1 Score (F-міра) : 33.13991 %

Як бачимо, повнота = 100%, тобто всі атаки типу `guess_passwd` були розпізнані. Побудуємо теплову карту для наглядності.



Як бачимо, PNN помилилась у 1037 випадках, сказавши, що була атака, але атаки не було. Результат непоганий, але його можна покращити кращим підбором параметра сігма, збільшенням навчального набору даних та детальному вивченні кожної колонки в датасеті (можливо є колонки, які не впливають на результат або навпаки ‘заплутують’ модель)

Результат відпрацювання програми:

```
КМ-03 | Шаповалов Г. Г. | Лаб 3 | Варіант 24 (7 - 'guess_passwd')

Розмір навчального набору: 62669
Розмір тестового набору: 15668
Кількість 'guess_passwd' у 'train_data': 1027
Кількість 'guess_passwd' у 'test_data' : 257
Кількість 'normal'       у 'train_data': 61642
Кількість 'normal'       у 'test_data' : 15411


Перекодуємо стовпці в числовий формат
Колонок у 'filtered_train_data': 78
Колонок у 'filtered_test_data':  75
Кількість спільних колонок: 74
Унікальні колонки в filtered_train_data:    {'flag_SH', 'flag_RSTOS0', 'service_shell', 'service_link'}
Унікальні колонки в filtered_test_data:     {'service_remote_job'}
Отже, назви та кількість не співпадають


Перевіримо як багато даних із певими атрибутами
Кількість 'link'       у 'train_data': 1
Кількість 'shell'      у 'train_data': 4
Кількість 'SH'         у 'train_data': 2
Кількість 'RSTOS0'     у 'train_data': 1
Кількість 'remote_job' у 'test_data' : 1


Цих даних не багато, можемо видалити їх
Колонок у 'filtered_train_data': 74
Колонок у 'filtered_test_data':  74
Кількість спільних колонок: 74
Отже назви колонок у датафреймах повністю однакові

*Навчаємо PNN*

*Тестуємо PNN*

Подивимось метрики, які відображають результат тестування PNN
Accuracy (Точність) : 93.38099 %
Precision (Точність): 19.8609 %
Recall (Повнота)    : 100.0 %
F1 Score (F-міра)   : 33.13991 %
```

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

```
import time
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score, confusion_matrix
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler


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)


def check_guess_pass_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 'guess_passwd' у 'train_data': {count_train_guess_pass}")
    print(f"Кількість 'guess_passwd' у 'test_data' : {count_test_guess_pass}")
    print(f"Кількість 'normal'       у 'train_data': {count_train_normal}")
    print(f"Кількість 'normal'       у 'test_data' : {count_test_normal}")


if __name__ == '__main__':
    print('\n\nКМ-03 | Шаповалов Г. Г. | Лаб 3 | Варіант 24 (7 -
\'guess_passwd\')')

    start_timer = time.time()
    train_data = pd.read_csv('KDDTrain+.txt')
    test_data = pd.read_csv('KDDTest+.txt')
    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_sr
v_diff_host_rate','dst_host_serror_rate',
                'dst_host_srv_serror_rate','dst_host_rerror_rate','dst_host_srv_rer
ror_rate','attack','level'])
    train_data.columns = columns
    test_data.columns = columns
```

```
    # 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=0.2,
random_state=42)
    train_normal, test_normal = train_test_split(normal_data, test_size=0.2,
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'\nРозмір 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'
    check_guess_pass_normal()

    ### Прибираємо з датасету всі записи, які не містять 'normal' або guess_passwd
    train_data = train_data[(train_data['attack'] == 'guess_passwd') |
(train_data['attack'] == 'normal')]
    test_data  = test_data[ (test_data['attack']  == 'guess_passwd') |
(test_data['attack']  == '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'\n\nПерекодуємо стовпці в числовий формат')
    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'Унікальні колонки в filtered_train_data:    {unique_columns_train}')
    print(f'Унікальні колонки в filtered_test_data:     {unique_columns_test}')
    print(f'Отже, назви та кількість не співпадають')

    ### Подивимось наскільки багато цих даних і чи можемо ми їх позбутися
    a = (train_data['service'] == 'link').sum()
    b = (train_data['service'] == 'shell').sum()
    c = (train_data['flag'] == 'SH').sum()
    d = (train_data['flag'] == 'RSTOS0').sum()
    e = (test_data['service'] == 'remote_job').sum()

    print(f'\n\nПеревіримо як багато даних із певими атрибутами')
    print(f"Кількість 'link'       у 'train_data': {a}")
    print(f"Кількість 'shell'      у 'train_data': {b}")
    print(f"Кількість 'SH'         у 'train_data': {c}")
    print(f"Кількість 'RSTOS0'     у 'train_data': {d}")
    print(f"Кількість 'remote_job' у 'test_data' : {e}")

    ### Видаляємо дані
    # Тренувальний датасет містить більше 67 тис. даних, а тестовий датасет 9
тис.
    # Тому видалення такої кількості даних повинно сильно вплинути на результати
досліджень
    # Видалення записів з умовами у train_data
    print(f'\n\nЦих даних не багато, можемо видалити їх')
    train_data = train_data[(train_data['service'] != 'link') &
                            (train_data['service'] != 'shell') &
                            (train_data['flag'] != 'SH') &
                            (train_data['flag'] != 'RSTOS0')]

    # Видалення записів з умовами у test_data
    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('Кількість спільних колонок:', len(common_columns))
    print('Отже назви колонок у датафреймах повністю однакові')

    ### 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])

    ### 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']

    print(f'\n*Навчаємо PNN*')
    ### 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=0.1)

    ### Використання PNN
    print(f'\n*Тестуємо 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'\nПодивимось метрики, які відображають результат тестування PNN')
    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="Greens",
                xticklabels=['normal', 'guess_passwd'], yticklabels=['normal',
'guess_passwd'])
    plt.title('Теплова карта прогнозування кібератаки типу \'guess_passwd\'\n')
    plt.xlabel("Predicted Label")
    plt.ylabel("True Label")
    plt.show()
```