НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра прикладної математики

Звіт

лабораторної роботи №3

з дисципліни «[СИСТЕМИ ГЛИБИННОГО НАВЧАННЯ](https://classroom.google.com/c/NjIxMzk4NjAyNjI4)»

на тему

«Нейромережеве розпізнавання  
кібератак»

|  |  |
| --- | --- |
| Виконав:  Студент групи КМ-03  Мітченко А.Д. | Перевірив:  доцент  Терейковський І. А. |
|  |  |
|  |  |
|  |  |

Київ — 2023

ЗМІСТ

[Теоретичні відомості 3](#_Toc152086486)

[Probabilistic neural networks 3](#_Toc152086487)

[Атака типу Proxy Host File 5](#_Toc152086488)

[Умова завдання: 5](#_Toc152086489)

[Скріншоти та опис програми 6](#_Toc152086490)

[ДОДАТОК А ТЕКСТ ПРОГРАМИ 9](#_Toc152086491)

[Завдання : 9](#_Toc152086492)

# Теоретичні відомості

## Probabilistic neural networks

Probabilistic neural networks  (PNN) — це тип штучної нейронної мережі прямого зв’язку, яка тісно пов’язана з оцінкою щільності ядра (KDE) через вікно Парзена, яке асимптотично наближається до оптимальної мінімізації ризику Байєса. Ця техніка широко використовується для оцінки умовної щільності класу (також відомої як ймовірність) у завданнях машинного навчання, таких як контрольоване навчання.

Изображение выглядит как диаграмма, линия, круг

Автоматически созданное описание

Рис. 1 Структура NN

Изображение выглядит как Шрифт, линия, диаграмма, рукописный текст

Автоматически созданное описание

Рис. 2 Parzen window density estimation

Изображение выглядит как текст, рукописный текст, доска, Шрифт

Автоматически созданное описание

Рис. 3 Формула яка розраховує вхідний сигнал для нейронів шару додавання.

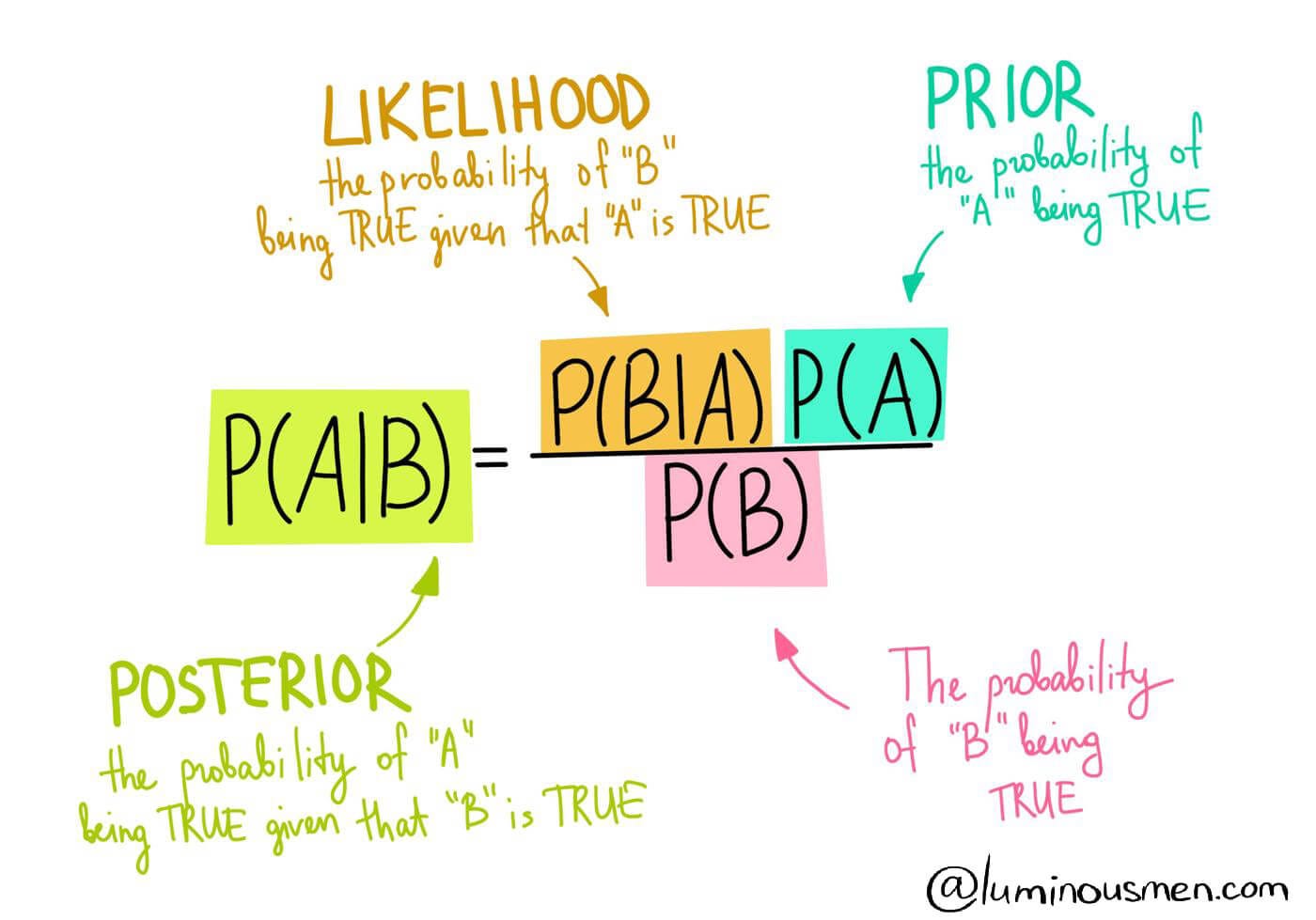


Рис 4. Теорема Байєса.

Нейронна мережа, яку представив Шпехт, складається з чотирьох рівнів:

1. Вхідний рівень: особливості точок даних (або спостережень)
2. Шар шаблону: розрахунок PDF умовного класу
3. Рівень підсумовування: підсумовування шаблонів між класами
4. Вихідний рівень: Перевірка гіпотези з максимальною апостериорною ймовірністю (MAP)

PNN мають кілька переваг, включаючи їх здатність обробляти дані великої розмірності, легкість навчання (особливо для малих і середніх наборів даних) і їх притаманний імовірнісний характер, який може забезпечити оцінки невизначеності для класифікації. Але є і обмеження. PNN можуть не підходити для надзвичайно великих наборів даних через вимоги до пам’яті та обчислень. Вони також можуть бути чутливими до вибору функції ядра та пропускної здатності у вікні Parzen.

Загалом PNN є корисним інструментом для завдань класифікації шаблонів, але їх придатність залежить від конкретної проблеми та набору даних. Вони використовувалися в різних програмах, зокрема для розпізнавання символів, медичної діагностики та контролю якості.

## Атака типу Proxy Host File

Для нашої лабораторної роботи потрібно розуміти що таке "PHF" або "Proxy Host File" атака. Це один з типів атак на веб-додатки, який використовується для отримання несанкціонованого доступу до веб-сервера через уразливості в обробці HTTP-запитів.

Атака PHF використовує вразливість в обробці команд, які передаються в HTTP-запитах. Зазвичай вразливість пов'язана з можливістю використання веб-сервера як проксі-сервера для виконання деяких команд або взаємодії з внутрішніми ресурсами сервера.

Щоб запобігти атакам типу PHF та подібним загрозам, важливо ретельно налаштовувати та оновлювати веб-сервери, використовувати механізми аутентифікації та авторизації, а також регулярно проводити аудит безпеки коду веб-додатків. Також слід використовувати файрволи та інші засоби безпеки для виявлення та блокування подібних атак.

Зазначте, будь ласка, якщо у вас є конкретні питання або якась конкретна аспект, про який ви хочете дізнатися більше.

## Умова завдання:

Згідно до варіанту 10. Розпізнавання мережевої кібератаки типу phf на базі PNN. Усі обчислення були проведенні у Юпітер ноутбуку за допомогою процесору Intel Core i5 11го покоління.

# Скріншоти та опис програми

В данній роботі я обрав набір данних KDD-99 з ресурсу [Kaggle](https://www.kaggle.com/datasets)

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, число, Параллельный

Автоматически созданное описание

Подивимось на наш таргет.

Изображение выглядит как текст, программное обеспечение, число, дисплей

Автоматически созданное описание

Як бачимо в нас всього 4 записи

Наступним кроком нам потрібно замінити категорійні колонки на числові, оскільки модель може сприймати лише числа.

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, число, программное обеспечение

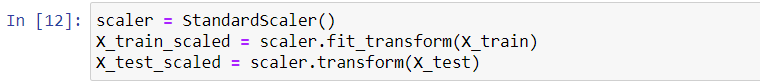
Автоматически созданное описание

Як бачимо, в нас тепер тільки цифри.

Далі ділимо вибірку на навчальні та тестувальні дані і створюємо модель.

Изображение выглядит как текст, Шрифт, линия, снимок экрана

Автоматически созданное описание



Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Після обрахунків ми бачимо що операція тренування відбувалася 0.1212 сек. Точність моделі вийшла 0.39.

# ДОДАТОК А ТЕКСТ ПРОГРАМИ

## Завдання :

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import accuracy\_score, classification\_report

from sklearn.preprocessing import LabelEncoder

import time

Datafile = "kddcup99\_csv.csv"

df = pd.read\_csv(Datafile,sep=",")

df['label'].value\_counts()

data = df

per\_label=(data["label"].value\_counts()/len(data["label"])\*100)

per\_label

label\_encoder = LabelEncoder()

for col in ['service', 'flag', 'protocol\_type']:

data[col] = label\_encoder.fit\_transform(data[col])

X = data.drop('label', axis=1)

y = data['label']

class\_to\_perceive = 'phf'

y\_binary = (y == class\_to\_perceive).astype(int)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y\_binary, test\_size=0.2, random\_state=42)

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

class PNN:

def \_\_init\_\_(self, std):

self.std = std

self.class\_means = None

self.class\_probs = None

def gauss\_kernel(self, x, mean):

return np.exp(-0.5 \* np.sum(((x - mean) / self.std) \*\* 2))

def train(self, X, y):

self.class\_means = [np.mean(X[y == 1], axis=0), np.mean(X[y == 0], axis=0)]

self.class\_probs = [len(X[y == 1]) / len(X), len(X[y == 0]) / len(X)]

def predict\_instance(self, x):

class\_scores = []

for mean, prob in zip(self.class\_means, self.class\_probs):

class\_score = prob \* self.gauss\_kernel(x, mean)

class\_scores.append(class\_score)

return np.argmax(class\_scores)

def predict(self, X):

predictions = [self.predict\_instance(x) for x in X]

return np.array(predictions)

start\_time = time.time()

pnn = PNN(std=0.1)

pnn.train(X\_train\_scaled, y\_train)

end\_time = time.time()

predictions = pnn.predict(X\_test\_scaled)

accuracy = accuracy\_score(y\_test, predictions)

print(f"Accuracy: {accuracy:.2f}")

print("Classification Report:")

print(classification\_report(y\_test, predictions))

training\_duration = end\_time - start\_time

print(f"Training Duration: {training\_duration:} seconds")