КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет комп'ютерних наук та кібернетики
Кафедра системного аналізу та теорії прийняття рішень

Кваліфікаційна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Системний аналіз»
за спеціальністю 124 Системний аналіз
на тему:

Розробка системи інтелектуального аналізу процесів поширення інфекційних захворювань з урахуванням життєвого циклу населення

Виконала студентка 4-го курсу
Ірина Вергунова _____

Науковий керівник:
кандидат фіз.-мат. наук
Юлія Шевчук _____

Засвідчую, що в цій роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Роботу розглянуто й допущено до захисту на засіданні кафедри системного аналізу та теорії прийняття рішень
«07» червня 2022 р., протокол №10

Завідувач кафедри
Олександр НАКОНЕЧНИЙ _____

Київ — 2022

ЗМІСТ

ВСТУП 4

Розділ 1. Імітаційні моделі поширення інфекційних захворювань 7

1.1. Огляд моделей поширення інфекційних захворювань 7

1.2. Аналіз розв'язків епідеміологічної моделі, яка враховує життєвий цикл населення 9

1.2.1. Умови додатності розв'язків епідеміологічної моделі, яка враховує життєвий цикл населення 10

1.2.2. Знаходження точних розв'язків епідеміологічної моделі, яка враховує життєвий цикл населення 11

1.3. Оцінювання в SIR моделях з врахуванням життєвого циклу населення 14

1.3.1. Алгоритм знаходження оптимальних оцінок параметрів моделі поширення інфекційних захворювань 16

1.4. Результати оцінювання параметрів SIRM-моделі поширення COVID-2019 в Японії 19

Розділ 2. Огляд нейронних мереж 23

2.1. Поняття та переваги нейронних мереж 23

2.2. Модель нейрона 25

2.3. Функції активації 27

2.4. Класифікація нейронних мереж та їх властивості 35

2.5. Топології нейронних мереж 39

2.6. Типи навчання нейронних мереж 44

Розділ 3. Використання нейронних мереж для побудови прогнозних оцінок динаміки процесу поширення інфекційних захворювань 50

3.1. Використання нейронних мереж для прогнозування COVID-19 50

3.2. Найуживаніші стратегії машинного навчання для прогнозування COVID-19 52
3.2.1. Узагальнений огляд 52
3.3. Розробка прототипу інтелектуального аналізу процесів поширення інфекційних захворювань з урахуванням життєвого циклу населення на мові Python 56
3.3.1. Технології для створення нейронних мереж на базі Python 56
3.3.2. Архітектура моделі LSTM для прогнозу розповсюдження COVID-19 58
3.3.3. Застосування та порівняння LSTM та SIRM-LSTM моделей для прогнозу захворювання 60

ВИСНОВКИ 65

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 67

ДОДАТОК 1. Програмний код нейронної мережі для прогнозування динаміки 75

ДОДАТОК 2. Код програмної реалізації знаходження оцінок параметрів SIRM-моделі 79

ВСТУП

Спалах COVID-19 на початку 2020 року призвів до різкого падіння якості життя людей у країнах, де хвороба набула значного поширення. Уряди багатьох країн зосередились на розробці превентивних засобів. Методами пригнічення епідемії є введення карантинних заходів, складання та оцінка плану проведення вакцинації, що дозволяє знизити рівень захворюваності та смертності.

Моделювання поширення інфекційних захворювань є інструментом, що дозволяє передбачати нові спалахи захворювання і оцінити стратегію боротьби з ним на основі вивчення механізму епідемії. Потрібно розуміти динаміку розповсюдження хвороби для правильного прогнозування кількості осіб, що захворіють та будуть потребувати постійного медичного нагляду. Це дасть змогу попередити колапс медичної системи. Також важливо розуміти, які засоби допомагають, а які навпаки шкодять, наприклад, бізнесу.

Традиційно вважають Kermack W.O. і McKendrick A.G. одними з перших науковців, які можливості використання моделей поширення інфекційних захворювань у вигляді систем диференціальних рівнянь до аналізу реальних епідемій та пандемій. В подальшому цим прикладним задачам приділяли увагу та свої наукові роботи багато науковців. Але практичне використання таких класичних математичних підходів показує недостатню точність у деяких випадках, адже в такому великому світі кількість показників, що можуть впливати на поширення захворювання, дуже сильно зросла порівняно із минулим століттям. Велику аудиторію здобувають методи машинного та глибинного навчання. Хоча все ще триває активний розвиток даної галузі інформатики та математики, але вже практичне використання в деяких країнах показує доволі високі результати. Особливу увагу приділяють розробці гібридних методів прогнозування поширення вірусних захворювань, таким чином намагаючись досягти ще кращої точності прогнозів об'єднавши надбання минулого і цього століття. У випадку деяких захворювань важливо брати до уваги збільшення групи сприйнятливих до захворювання. Тому постає необхідність у включенні народжених та померлих. Отримуємо модель SIRM. В ході дослідження даної системи диференційних рівнянь постає питання оцінки невідомих

параметрів моделі, а також аналіз та знаходження аналітичного розв'язку. Таким чином, дана робота пропонує програмну реалізацію гарантованих оцінок SIRM моделі, знаходження аналітичних розв'язків із стаціонарними та нестаціонарними параметрами за умови незмінності групи, а також використання методів машинного навчання як окремий засіб прогнозування, так і додатковий до вже використаних математичних алгоритмів на базі мови програмування Python, а саме бібліотеки для нейронних мереж Keras на прикладі країни Японії.

Мета роботи:

Провести аналіз літератури із застосуванням моделей типу SIR для отримання прогнозу розповсюдження коронавірусної хвороби.

Провести огляд літератури із знаходження аналітичного розв'язку SIR моделей.

Запропонувати аналітичний розв'язок при стаціонарних та нестаціонарних параметрах при умові незмінності групи.

Провести аналіз літератури із застосування різних нейронних мереж для отримання прогнозу розповсюдження коронавірусної хвороби.

Зробити порівняльний аналіз запропонованих методів.

Розробити програмну реалізацію для гібридного методу прогнозу розповсюдження вірусу на основі гарантованих оцінок та методу машинного навчання.

Об'єкт дослідження: Системи нелінійних звичайних диференціальних та різницевих рівнянь, що моделюють процес поширення інфекційного захворювання, та методи машинного навчання для прогнозування вірусних хвороб.

Предмет дослідження: Аналітичні розв'язки SIRM моделі з урахуванням життєвого циклу населення та гібридний метод на основі знайдених оцінок параметрів моделі та нейронної мережі.

Структура та обсяг роботи. Кваліфікаційна робота складається зі вступу, трьох розділів, висновку, списку використанної літератури. Список використаної літератури включає в себе 75 джерела. Робота виконана на 74 сторінках друкованого тексту.

У першому розділі розглянуто різні моделі поширення інфекційних захворювань типу SIR та можливості їх застосування, наведені умови додатності розв'язків моделі, що використовується для моделювання епідемій

вірусів та враховує життєвий цикл населення досліджуваної групи, алгоритм знаходження точних розв'язків моделі при заданій умові.

Другий розділ присвячений аналізу методів використання засобів машинного навчання на основі практичних застосувань, розгляду поняття, склад, переваги та різні класифікації нейронних мереж, дослідженню різні методи навчання нейронних мереж.

У третьому розділі проведено аналіз різних моделей для прогнозування коронавірусної хвороби за допомогою глибинного навчання, запропоновано та реалізовано архітектуру системи інтелектуального аналізу на базі LSTM моделі, що навчається із вчителем, та гібридного методу SIRM-LSTM із гарантованими оцінками параметрів епідеміологічної моделі.

Результати дослідження були представлені на наукових міжнародних конференціях:

- Shevchuk, I. Vergunova, I. "Analysis of the SIRM-model with birth and morality rates."Abstracts of XXXVI International Conference "Problems of Decision Making under Uncertainties"(PDMU-2021): 99—100.
- Вергунова І.В. Оцінка в нелінійних моделях поширення інфекційних захворювань (на прикладі пандемії COVID-19 у Чеській Республіці), Матеріали XIX Міжнародної науково-практичної конференції «Шевченківська весна—2021», С. 32—33.
- Вергунова І.В, Шевчук Ю.М Інтелектуальна система моделювання процесу поширення інфекційних захворювань, Матеріали VIII Всеукраїнської науково-практичної конференції студентів, аспірантів та молодих вчених ОБ'ЄДНАНІ НАУКОЮ: перспективи міждисциплінарних досліджень, 2021, С. 76—78.
- Вергунова І. В., Вертелко М. О, Шевчук Ю.М Аналіз SIR моделі поширення інфекційних захворювань з врахуванням життєвого циклу населення, Матеріали XX Міжнародної наукової-практичної конференції "Шевченківська весна — 2022 С. 34—35.

Розділ 1. Імітаційні моделі поширення інфекційних захворювань

Розвиток суспільства, медичної сфери, вдосконалення санітарних умов, винахід антибіотиків та запровадження програм масової вакцинації формували можливість подальшого повного ліквідування інфекційних хвороб. Питання виникнення та появи даних захворювань перебувало під пильною увагою епідеміологів протягом багатьох років. Прогнозування епідемій дає можливість прийняти своєчасні рішення із забезпечення якомога безпечнішого сценарію розповсюдження вірусу, що з часом приведе до повного його стримання.

Засобами пригнічення епідемії є введення карантинних заходів, складення та оцінка плану проведення вакцинації, що дозволяє знизити рівень захворюваності та смертності. Моделювання поширення інфекційних захворювань є інструментом, що дозволяє передбачати нові спалахи захворювання і оцінити стратегію боротьби з ним на основі вивчення механізму епідемії.

Найбільш широко використовуваним засобом для даного моделювання є компартментні моделі (або просто детерміновані), в яких популяції діляться на компартменти (групи, наприклад, інфіковані та ті, що одужали). Можливий перехід від однієї групи до іншої відповідно до деякого темпу переходу. Швидкості переходу індивідів з однієї групи в іншу математично виражаються похідними, а весь процес описується системою диференціальних рівнянь.

1.1 Огляд моделей поширення інфекційних захворювань

Однією із базових моделей поширення інфекційних захворювань в групі осіб є модель SIR (модель Кермака-МакКендріка [1]) — система із трьох диференціальних рівнянь, які відображають динаміку кількості сприйнятливих до захворювання ($S(t), t \in (0,T)$), кількості інфікованих ($I(t)$,

$t \in (0,T)$) та кількості одужавших осіб ($R(t)$, $t \in (0,T)$):

$$\begin{cases} \dot{S}(t) = -\alpha(t)S(t)I(t), \\ \dot{I}(t) = \alpha(t)S(t)I(t) - \beta(t)I(t), \ t \in (0,T), \\ \dot{R}(t) = \beta(t)I(t) \end{cases} \tag{1}$$

тут $\alpha(t)$, $t \in (0,T)$ — параметр інтенсивності інфікування; $\beta(t)$, $t \in (0,T)$ — параметр інтенсивності одужання.

Ця модель стала базовою для багатьох подальших досліджень задач моделювання процесів поширення інфекційних захворювань, наприклад, [2]– [4]. У ній передбачається, що швидкості зараження і одужання суттєво перевищують темпи народжуваності і смертності, тому ці останні фактори в моделі ігноруються.

Модель SIR та її модифікаії в загальному випадку є системами нелінійних диференціальних рівнянь й тому не мають узагальненого точного розв'язку. Але якщо виконується умова:

$$\dot{S}(t) + \dot{I}(t) + \dot{R}(t) = 0, t \in (0,T), \tag{2}$$

то аналітичними методами можна отримати точний розв'язок для таких систем з стаціонарними параметрами, що продемонстровано, наприклад, в роботах [8] та [9].

У випадку деяких захворювань важливо брати до уваги збільшення групи сприйнятливих до захворювання. Тому постає необхідність у включенні народжених та померлих. Отримуємо модель SIR з включення народжуваності та смертності (наприклад, [5]):

$$\begin{cases} \dot{S}(t) = -\alpha(t)S(t)I(T) - \mu(N - S(t)), \\ \dot{I}(t) = \alpha(t)S(t)I(t) - \beta(t)I(t) - \mu, \quad t \in (0,T), \\ \dot{R}(t) = \beta(t)I(t) - \mu R(t), \end{cases} \tag{3}$$

де μ — постійний коєфіцієнт, що описує темпи народжуваності та смертності, які вважаються однаковими для всіх груп, а N — кількість населення.

Можна отримати модель SIS із SIR, якщо брати до уваги, що особи, які одужали, не отримують ніякого імунітету до захворювання, тому переходять до групи сприйнятливих:

$$\begin{cases} \dot{S}(t) = -\alpha(t)S(t)I(T) - \mu(N - S(t)) + \beta(t)I(t), \\ \dot{I}(t) = \alpha(t)S(t)I(t) - \beta(t)I(t) - \mu, \end{cases} t \in (0,T), \tag{4}$$

У цій моделі залишаємо лише два рівняння для сприйнятливих та інфікованих, причому порівняно із попередньою моделлю, додаємо тих, хто видужав, до групи сприятливих (наприклад, [6]).

Наступна модель SIRS (наприклад, [7]) передбачена для ситуації короткострокового імунітету. Тобто одужалі згодом повертаються до групи сприйнятливих. Тоді модель має вигляд системи нелінійних диференціальних рівнянь:

$$\begin{cases} \dot{S}(t) = -\alpha(t)S(t)I(T) - \mu(N - S(t)) + \sigma R(t), \\ \dot{I}(t) = \alpha(t)S(t)I(t) - \beta(t)I(t) - \mu, \qquad t \in (0,T), \\ \dot{R}(t) = \beta(t)I(t) - \mu R(t) - \sigma R(t), \end{cases} \tag{5}$$

де σ — середня швидкість втрати імунітету.

Якщо до базової моделі (1) додати компоненти, які відповідають за процеси народжуваності та смертності, то одержимо систему нелінійних диференціальних рівнянь:

$$\begin{cases} \dot{S}(t) = -\alpha(t)S(t)I(t) - \mu_1(t)S(t) + \gamma(t), \\ \dot{I}(t) = \alpha(t)S(t)I(t) - \beta(t)I(t) - \mu_2(t)I(t), \ t \in (0,T), \\ \dot{R}(t) = \beta(t)I(t) - \mu_3(t)R(t), \end{cases} \tag{6}$$

яка є SIR моделлю з врахуванням життєвого циклу населення, де $\mu_1(t)$, $\mu_2(t)$, $\mu_3(t)$, $t \in (0,T)$ — параметри рівня смертності у відповідних групах; $\gamma(t)$, $t \in (0,T)$ — параметр народжуваності.

1.2 Аналіз розв'язків епідеміологічної моделі, яка враховує життєвий цикл населення

Дослідимо далі детальніше розв'язки системи (6):

- виведемо умови додатності розв'язків цієї системи (6);
- знайдемо точні розв'язки для випадку (6) із нестаціонарними параметрами за умови, що справедлива рівність (2), та її окремого випадку з стаціонарними параметрами.

1.2.1 Умови додатності розв'язків епідеміологічної моделі, яка враховує життєвий цикл населення

Для системи (6) можна вивести умови, за яких її розв'язки будуть не від'ємні. Це є доволі важливим результатом для подальшого аналізу моделі, оскільки предметна область не передбачає від'ємної кількості осіб, які є інфікованими чи одужавшими.

Теорема 1. Нехай для задачі Коші

$$\begin{cases} \dot{S}(t) = -\alpha(t)S(t)I(t) - \mu_1(t)S(t) + \gamma(t), S(0) = S_0 \\ \dot{I}(t) = \alpha(t)S(t)I(t) - \beta(t)I(t) - \mu_2(t)I(t), I(0) = I_0 \quad t \in (0,T), \\ \dot{R}(t) = \beta(t)I(t) - \mu_3(t)R(t), R(0) = R_0, \end{cases} \tag{7}$$

виконуються нерівності $S_0 \geqslant 0$, $I_0 \geqslant 0$, $R_0 \geqslant 0$, а функції $\gamma(t)$ та $\beta(t)$, $t \in (0,T)$ — неперервні невід'ємні функції на відрізку $[0,T]$ функції, тоді у системи (7) існують невід'ємні розв'язки.

Доведення. Розглянемо окремо кожне із рівнянь системи (7). Тоді рівняння динаміки кількості сприйнятливих до захворювання осіб можна записати у вигляді:

$$\dot{S}(t) = (-\alpha(t)I(t) - \mu_1(t))S(t) + \gamma(t), t \in (0,T). \tag{8}$$

Оскільки функція $\gamma(t)$, $t \in (0,T)$ — неперервна невід'ємна функція на відрізку $[0,T]$ функція, то для виразу (8) справедлива нерівність:

$$\dot{S}(t) = (-\alpha(t)I(t) - \mu_1(t))S(t) + \gamma(t) \geqslant (-\alpha(t)I(t) - \mu_1(t))S(t), t \in (0,T). \tag{9}$$

Застосувавши формулу Коші до виразу (9), одержимо наступні нерівності:

$$S(t) \geqslant S_0 exp\left\{-\int_0^T (\alpha(\tau)I(\tau) - \mu_1(\tau))d\tau\right\} \geqslant 0, t \in (0,T).$$

Проаналізуємо далі диференціальне рівняння для функції $I(t)$, $t \in (0,T)$:

$$\dot{I}(t) = (\alpha(t)S(t) - \beta(t) - \mu_2(t))I(t), t \in (0,T). \tag{10}$$

За формулою Коші отримаємо розв'язок рівняння (10) та його оцінку:

$$I(t) = I(0) exp\left\{\int_0^T (\alpha(\tau)S(\tau) - \beta(\tau) - \mu_2(t))d\tau\right\} \geqslant 0, t \in (0,T). \tag{11}$$

Диференціальне рівняння для $R(t)$, $t \in (0,T)$ є лінійним неоднорідним рівнянням за $R(t)$:

$$\dot{R}(t) = \beta(t)I(t) - \mu_3(t)R(t), t \in (0,T), \tag{12}$$

але в силу того, що $\beta(t)$, $t \in (0,T)$ — неперервна невід'ємна функція на відрізку $[0,T]$ функція, а з виразу (11) отримуємо нерівність $I(t) \geqslant 0$, $t \in (0,T)$, для (12) буде справедлива оцінка:

$$\dot{R}(t) = \beta(t)I(t) - \mu_3(t)R(t) \geqslant -\mu_3(t)R(t), t \in (0,T). \tag{13}$$

Тоді, застосувавши до (13) формулу Коші, одержимо нерівність для функції $R(t)$, $t \in (0,T)$:

$$R(t) \geqslant R(0)exp\left\{-\int_0^T \mu_3(\tau)d\tau\right\} \geqslant 0, t \in (0,T),$$

що і треба було довести.

1.2.2 Знаходження точних розв'язків епідеміологічної моделі, яка враховує життєвий цикл населення

Теорема 2. Нехай для задачі Коші (7) виконується умова:

$$N = S(t) + I(t) + R(t), t \in (0,T). \tag{14}$$

Тоді розв'язки системи будуть обчислюватись за формулами:

$$S(t) = \frac{I_0 k_1(t)}{\exp^{\int_0^t k_3(s)ds} - I_0 \int_0^t \exp^{\int_\tau^t k_3(s)ds} k_1(\tau)\alpha(\tau)d\tau} + k_2(t), t \in (0,T),$$

$$I(t) = \frac{I_0}{\exp^{\int_0^t k_3(s)ds} - I_0 \int_0^t \exp^{\int_\tau^t k_3(s)ds} k_1(\tau)\alpha(\tau)d\tau}, t \in (0,T),$$

$$R(t) = N - \frac{I_0(k_1(t)+1)}{\exp^{\int_0^t k_3(s)ds} - I_0 \int_0^t \exp^{\int_\tau^t k_3(s)ds} k_1(\tau)\alpha(\tau)d\tau} - k_2(t), t \in (0,T),$$

де

$$k_1(t) = \frac{\mu_3(t) - \mu_2(t)}{\mu_1(t) - \mu_3(t)}, t \in (0,T),$$

$$k_2(t) = \frac{\gamma(t) - \mu_3(t)N}{\mu_1(t) - \mu_3(t)}, t \in (0,T),$$

$$k_3(t) = \beta(t) + \mu_2(t) - k_2(t)\alpha(t), t \in (0,T).$$

Доведення Продиференціюємо вираз (14), тоді отримаємо рівність:

$$\dot{S(t)} + \dot{I(t)} + \dot{R(t)} = 0, t \in (0,T). \tag{15}$$

З виразів (14) та (15) одержимо:

$$\dot{R(t)} = -\dot{S(t)} - \dot{I(t)}, t \in (0,T). \tag{16}$$

$$R(t) = N - S(t) - I(t), t \in (0,T). \tag{17}$$

Додамо диференціальні рівняння для функцій $S(t)$ та $I(t)$, $t \in (0,T)$ з задачі Коші (7):

$$\dot{S}(t) + \dot{I}(t) = -\beta(t)I(t) - \mu_1(t)S(t) - \mu_2(t)I(t) + \gamma(t), t \in (0,T), \tag{18}$$

У силу рівності (16) вираз (18) можна представити в формі:

$$\dot{R}(t) = \mu_1(t)S(t) + \mu_2(t)I(t) - \gamma(t), t \in (0,T). \tag{19}$$

Також в диференціальне рівняння для функції $R(t)$, $t \in (0,T)$ із системи (7) підставимо вираз (17):

$$\dot{R}(t) = \beta(t)I(t) - \mu_3(t)(N - S(t) - I(t)), t \in (0,t). \tag{20}$$

Тоді вирази (19) і (20) можна прирівняти:

$$(\beta(t) + \mu_2(t))I(t) + \mu_1(t)S(t) - \gamma(t) =$$
$$= (\beta(t) + \mu_3(t))I(t) + \mu_3(t)S(t) - \mu_3(t)N, t \in (0,T). \tag{21}$$

Виразимо з рівності (21) $S(t)$, $t \in (0,T)$:

$$S(t) = \frac{(\mu_3(t) - \mu_2(t))I(t) - \mu_3(t)N + \gamma(t)}{\mu_1(t) - \mu_3(t)} = k_1(t)I(t) + k_2(t), t \in (0,T). \quad (22)$$

Підставимо вираз (22) у диференціальне рівняння для $I(t)$, $t \in (0,T)$ з системи (15), тоді отримаємо задачу Коші:

$$\dot{I}(t) = (\alpha(t)(k_1(t)I(t) + k_2(t)) - \beta(t) - \mu_2(t))I(t), t \in (0,t), I(0) = I_0. \quad (23)$$

Поділимо ліву та праву частини рівняння Ріккаті з (23) на $I^2(t)$, $t \in (0,T)$:

$$\frac{1}{I^2(t)}\dot{I}(t) = k_1(t)\alpha(t) + (k_2(t)\alpha(t) - \beta(t) - \mu_2(t))\frac{1}{I(t)}, t \in (0,T). \quad (24)$$

Уведемо до розгляду функцію $z(t) = \frac{1}{I(t)}$, $t \in (0,T)$, тоді

$$\dot{z}(t) = k_3(t)z(t) - k_1(t)\alpha(t), t \in (0,T), z(0) = \frac{1}{I_0}. \quad (25)$$

Застосуємо формулу Коші до (25):

$$z(t) = \frac{1}{I_0}\exp^{\int_0^t k_3(s)ds} - \int_0^t \exp^{\int_\tau^t k_3(s)ds} k_1(\tau)\alpha(\tau)d\tau, t \in (0,T).$$

Тоді отримаємо вираз для функції кількості інфікованих осіб:

$$I(t) = \frac{I_0}{\exp^{\int_0^t k_3(s)ds} - I_0 \int_0^t \exp^{\int_\tau^t k_3(s)ds} k_1(\tau)\alpha(\tau)d\tau}, t \in (0,T).$$

Знайдемо вираз, за яким буде обчислюватись кількість сприйнятливих до захворювання осіб:

$$S(t) = \frac{I_0 k_1(t)}{\exp^{\int_0^t k_3(s)ds} - I_0 \int_0^t \exp^{\int_\tau^t k_3(s)ds} k_1(\tau)\alpha(\tau)d\tau} + k_2(t), t \in (0,T).$$

Кількість одужавших осіб буде обчислюватись за формулою:

$$R(t) = N - \frac{I_0(k_1(t) + 1)}{\exp^{\int_0^t k_3(s)ds} - I_0 \int_0^t \exp^{\int_\tau^t k_3(s)ds} k_1(\tau)\alpha(\tau)d\tau} - k_2(t), t \in (0,T).$$

Доведення завершено.

Наслідок. Нехай виконуються умови Теореми 2 та параметри задачі (7) є стаціонарними, тобто $\alpha(t) = \alpha$, $\beta(t) = \beta$, $\gamma(t) = \gamma$, $\mu_1(t) = \mu_1$, $\mu_2(t) = \mu_2$, $\mu_3(t) = \mu_3$, $t \in (0,T)$, тоді справедливі наступну перетворення:

$$I(t) = \frac{I_0}{\exp^{\int_0^t k_3 ds} - I_0 \int_0^t \exp^{\int_\tau^t k_3 ds} k_1 \alpha d\tau} = \frac{I_0}{\exp^{k_3 t} - \alpha k_1 I_0 \int_0^t \exp^{k_3(t-\tau)} d\tau} =$$

$$= \frac{I_0}{\exp^{k_3 t} - \alpha k_1 I_0 \exp^{k_3 t} \int_0^t \exp^{-k_3 \tau} d\tau} = \frac{I_0}{\exp^{k_3 t} + \alpha k_1 I_0 \exp^{k_3 t} \frac{1}{k_3} \exp^{-k_3 \tau} |_0^t} =$$

$$= \frac{I_0 k_3}{k_3 \exp^{k_3 t} + \alpha k_1 I_0 \exp^{k_3 t}(\exp^{-k_3 t} - 1)} = \frac{I_0 k_3}{(k_3 - \alpha k_1 I_0) \exp^{k_3 t} + \alpha k_1 I_0}, t \in (0,T),$$

де

$$k_1 = \frac{\mu_3 - \mu_2}{\mu_1 - \mu_3}, k_2 = \frac{\gamma - \mu_3 N}{\mu_1 - \mu_3}, k_3 = \beta + \mu_2 - k_2 \alpha.$$

Тоді справедливі вирази для наступних функцій:

$$S(t) = \frac{I_0 k_1 k_3}{(k_3 - \alpha k_1 I_0) \exp^{k_3 t} + \alpha k_1 I_0} + k_2, t \in (0,T),$$

$$R(t) = N - \frac{I_0 k_3 (k_1(t) + 1)}{(k_3 - \alpha k_1 I_0) \exp^{k_3 t} + \alpha k_1 I_0} - k_2, t \in (0,T).$$

1.3 Оцінювання в SIR моделях з врахуванням життєвого циклу населення

Важливою задачею при аналізі процесів поширення інфекційних захворювань є побудова прогнозів динаміки кількості захворівших та одужавших. Здебільшого в літературі розв'язання цієї задачі проводять з використанням методології аналізу часових рядів чи засобів імітаційного моделювання ([10]— [16]).

Не менш важливою задачею є оцінювання параметрів моделі адже це також дає дані для створення представлення про перебіг процесу поширення

інфекційного захворювання, а також подальшого створення прогнозів. Цьому питанню присвячені роботи, в яких наведені результати оцінювання стаціонарних параметрів SIR моделі, а також динаміки процесу поширення інфекційного захворювання ([17]— [23]).

Теоретичні аспекти алгоритмів побудови оцінок для моделей динамічних систем доволі грунтовно розглянуті, наприклад, в роботах [24] — [33].

Далі розглянемо детальніше проблему побудови оцінок параметрів моделі (7). Для цього будемо досліджувати відповідну їй систему різницевих рівнянь:

$$\begin{cases} S(t_{k+1}) = S(t_k) - \alpha(t_k)S(t_k)I(t_k) - \mu_1(t_k)S(t_k) + \\ \qquad +\gamma(t_k), S(0) = S_0 > 0, \\ I(t_{k+1}) = I(t_k) + \alpha(t_k)S(t_k)I(t_k) - \beta(t_k)I(t_k) - \\ \qquad -\mu_2(t_k)I(t_k), I(0) = I_0 \geqslant 0, \qquad k = \overline{1,m}. \\ R(t_{k+1}) = R(t_k) + \beta(t_k)I(t_k) - \\ \qquad -\mu_3(t_k)R(t_k), R(0) = R_0 \geqslant 0, \\ M(t_{k+1}) = M(t_k) + \mu_2(t_k)I(t_k), M(0) = M_0 \geqslant 0, \end{cases} \tag{26}$$

Систему (26) можна представити у матричному вигляді:

$$\begin{pmatrix} S(t_{k+1}) \\ I(t_{k+1}) \\ R(t_{k+1}) \\ M(t_{k+1}) \end{pmatrix} = \begin{pmatrix} -S(t_k)I(t_k) & 0 & 0 \\ S(t_k)I(t_k) & -I(t_k) & -I(t_k) \\ 0 & I(t_k) & 0 \\ 0 & 0 & I(t_k) \end{pmatrix} \times \begin{pmatrix} \alpha(t_k) \\ \beta(t_k) \\ \mu_2(t_k) \end{pmatrix} +$$

$$+ \begin{pmatrix} (1-\mu_1(t_k))S(t_k) + \gamma(t_k) \\ I(t_k) \\ (1-\mu_3(t_k))R(t_k) \\ M(t_k) \end{pmatrix}, k = \overline{1,m+1}, \begin{pmatrix} S(0) \\ I(0) \\ R(0) \\ M(0) \end{pmatrix}. \tag{27}$$

У загальному випадку можна припустити, що $\mu_1(t_k) = \mu_3(t_k)$, $\gamma(t_k)$, $k \in \overline{1,m}$ відомі функції, а функції $\alpha(t_k), \beta(t_k), \mu_2(t_k)$, $k \in \overline{1,m}$ є невідомими, тоді систему (27) можна представити у вигляді:

$$x(k+1) = f(k,x(k))a_k + g(k,x(k)), k = \overline{1,m}, \tag{28}$$

де

$$x_1(k) = S(t_k), x_2(k) = I(t_k), x_3(k) = R(t_k), x_4(k) = M(t_k), k = \overline{1,m+1},$$

$$x(k) = \begin{pmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \\ x_4(k) \end{pmatrix}, x(1) = \begin{pmatrix} S(0) \\ I(0) \\ R(0) \\ M(0) \end{pmatrix}, a_k = \begin{pmatrix} \alpha(t_k) \\ \beta(t_k) \\ \mu_2(t_k) \end{pmatrix}, k = \overline{1,m},$$

$$f(k,x(k)) = \begin{pmatrix} -S(t_k)I(t_k) & 0 & 0 \\ S(t_k)I(t_k) & -I(t_k) & -I(t_k) \\ 0 & I(t_k) & 0 \\ 0 & 0 & I(t_k) \end{pmatrix}, k = \overline{1,m},$$

$$g(k,x(k)) = \begin{pmatrix} (1-\mu_1(t_k))S(t_k) + \gamma(t_k) \\ I(t_k) \\ (1-\mu_3(t_k))R(t_k) \\ M(t_k) \end{pmatrix}, k = \overline{1,m}.$$

1.3.1 Алгоритм знаходження оптимальних оцінок параметрів моделі поширення інфекційних захворювань

Розглянемо далі алгоритм для знаходження оцінок параметрів різницевих рівнянь, запропонований в роботі [35].

Нехай спостерігаються вектори $x(k) \in R^4$, $k = \overline{1, m+1}$ при невідомих параметрах $a_k \in R^3$, $k = \overline{1, m}$, які є розв'язками різницевих рівнянь:

$$x(k+1) = f(k,x(k))a_k + g(k,x(k)) + \eta_k, k = \overline{1,m},$$

де $f(k,x(k)$, $k = \overline{1, m}$ — задані матриці розмірності 4×3, $g(k,x(k)) \in R^4$, $k = \overline{1, m}$ — відомі вектори, η_k, $k = \overline{1, m}$ — невідомі вектори завад. Відомо, що $\Delta_+ a_k \in U_k \subseteq R^3$, $k = \overline{1, m-1}$, $\Delta_+ a_k = a_{k+1} - a_k$, $k = \overline{1, m-1}$. Припускаємо, що $\eta_k \in V_k \subseteq R^3$, $k = \overline{1, m}$.

Апостеріорна множина має вигляд:

$$G_a = \{a : (x(k+1) - f(k,x(k))a_k - g(k,x(k))) \in V_k, k = \overline{1,m},$$

$$\Delta_+ a_j = a_{j+1} - a_j, j = \overline{1, m-1}\},$$

де $a = (a_1, \ldots, a_m)$.

Означення 1. Матриці $\bar{a} = (\bar{a}_1, \ldots, \bar{a}_m) \in G_a$ назвемо *апостеріорними оцінками* матриці a.

При обмежених множинах U_j, $j = \overline{1,m-1}$ та V_k, $k = \overline{1,m}$, множину G_a можна представити у вигляді:

$$G_a = \left\{ a : \sum_{k=1}^{m} q_{2k} |x(k+1) - f(k,x(k))a_k - g(k,x(k))|^2 + \right.$$

$$\left. + \sum_{k=1}^{m-1} q_{1k} |a_{k+1} - a_k|^2 \leqslant \varphi \right\}, \tag{29}$$

де q_{1j}, $j = \overline{1,m-1}$ та q_{2k}, $k = \overline{1,m}$, φ – відомі скалярні величини.

Для множини G_a також справедливе представлення:

$$G_a = G_{a_1} \times \cdots \times G_{a_m}, a_k \in G_{a_k} \subseteq R^3, k = \overline{1,m}.$$

Уведемо функцію вигляду:

$$\Phi(a) = \sum_{k=1}^{m} q_{2k} |y(k) - f_k a_k|^2 + \sum_{k=1}^{m-1} q_{1k} |a_{k+1} - a_k|^2, \tag{30}$$

де

$$y(k) = x(k+1) - g(k,x(k)), f_k = f(k,x(k)), k = \overline{1,m}.$$

Означення 2. Матрицю $\hat{a} = (\hat{a}_1, \ldots, \hat{a}_m)$, яка знаходиться з умови $\hat{a} \in Arg \min\limits_{a \in G_a} \Phi(a)$, назвемо *оптимальною оцінкою* за функцією $\Phi(a)$.

Уведемо матрицю $u = (u_1, \ldots, u_{m-1})$, $u_j = a_{j+1} - a_j$, $j = \overline{1,m-1}$, $a_1 \in G_{a_1}$ і позначимо через G_1 множину:

$$G_1 = \left\{ (u,a_1) : \sum_{k=1}^{m} q_{2k} |y(k) - f_k a_k|^2 + \sum_{k=1}^{m-1} q_{1k} |u_k|^2 \leqslant \varphi \right\}.$$

Множину G_1 можна також представити у вигляді:

$$G_1 = U_{(1)} \times \cdots \times U_{(m-1)} \times G_{a_1},$$

$$u_i \in U_{(i)} \subseteq R^3, i = \overline{1,m-1}.$$

Задача знаходження оптимальної за функцією $\Phi(a)$ оцінки $\hat{a}$ еквівалентна задачі знаходження $(\hat{u}, \hat{a}_1) \in Arg \min\limits_{(u,a1) \in G_1} I(u,a_1)$, де

$$I(u,a_1) = \sum_{k=1}^{m} q_{2k} |y(k) - f_k a_k|^2 + \sum_{k=1}^{m-1} q_{1k} |u_k|^2. \tag{31}$$

Задачу знаходження оптимальної оцінки $\hat{a}$ на основі спостережень $x(k)$, $k = \overline{1,m+1}$ розв'яжемо шляхом реалізації наступної багатокрокової процедури. На j-у кроці ($j = \overline{1,m-1}$) відбувається пошук оцінки $\hat{a}_{j+1}$ на основі оцінки $\hat{a}_j$ та спостережень $x(j)$ та $x(j+1)$ з використанням фільтра Каллмана Бюсі.

Для спрощення, припускаємо $a_1 = 0$. Покладемо $\hat{a}_1 = 0$ та позначимо $I(u,\hat{a}_1) = I(u)$.

Теорема 2 [35]. Оптимальна за функцією $\Phi(a)$ оцінка $\hat{a}$ знаходиться за формулами:

$$\hat{a}_{k+1} = \hat{a}_k + F_k(y(k) - f_k\hat{a}_k), \hat{a}_1 = 0, k = \overline{1,m-1}, \tag{32}$$

де

$$F_k = P_k f_k^*(f_k p_k f_k^* + (q_{2k})^{-1}E)^{-1}, k = \overline{1,m-1},$$

$$P_{k+1} = (E - F_k f_k)P_k(E - F_k f_k)^* + (q_{1k})^{-1}E +$$

$$+(q_{2k})^{-1}F_k F_k^*, P_1 = 0, k = \overline{1,m-2},$$

а позначка * — знак транспонування.

Похибка алгоритму обчислюється за формулою:

$$\sigma_k = \max_{a_k \in G_{a_k}} \|\hat{a}_k - a_k\| = \lambda_{max}^{1/2}(H_k A^{-1} H_k^*)(\psi - \Phi(\hat{a}))^{1/2}, k = \overline{1,m}, \tag{33}$$

де $\lambda_{max}(A)$ — максимальне власне число матриці A; H_k — оператор проектування, для якого виконуються умови:

$$H_k a = a_k, k = \overline{1,m}; \tag{34}$$

а матриця $A = \{A_{ij}\}_{i,j=1}^m$ — трьохдіагональна матриця з елементами:

$$A_{k,k-1} = -q_{1,k-1}, A_{k,k} = q_{1,k-1} + q_{1,k} + q_{2k} f_k^* f_k, k = \overline{2,m-1},$$

$$A_{k,k+1} = -q_{1k}, k = \overline{2,m-1}, A_{11} = q_{11} + q_{21} f_1^* f_1, A_{12} = -q_{11},$$

$$A_{m,m-1} = -q_{m,m-1}, A_{m,m} = q_{m,m-1} + q_{2m} f_m^* f_m.$$

1.4 Результати оцінювання параметрів SIRM-моделі поширення COVID-2019 в Японії

Одним із завдань даної наукової роботи було розробка програмного продукту для моделювання процесів поширення інфекційних захворювань та оцінювання в них, зокрема оцінювання невідомих параметрів.

Для проведення моделювання було обрано Японію. Аналізувався часовий проміжок з 31 грудня 2020 р. по 1 квітня 2021 р., досліджувалася динаміка поширення COVID-19 в країні.

Дані для проведення моделювання отримані з [36].

У якості базової моделі для імітаційного моделювання була обрана модель (7) із постійними параметрами $\mu_1(t) = \mu_3(t) = \mu = const$ та $\gamma(t) = \gamma = const$, $t \in (0,T)$.

Припускається, що в досліджуваній країні параметри рівня смертності в групах сприйнятливих до інфекційного захворювання та одужавших осіб під час пандемії є подібними до параметру рівня смертності в доепідеміологічний період, і тому різницею між ними можна знехтувати. Тоді μ можна обчислити за формулою:

$$\mu = \frac{\text{Кількість загиблих в 2019 р.}}{365 \times \text{Населення країни в 2018 р.}}. \tag{35}$$

Подібним чином, нехтуючи різницею між параметром народжуваності в країні в доепідеміологічний період та під час поширення інфекційного захворювання, можна сформулювати і вираз для параметру γ:

$$\gamma = \frac{\text{Кількість народженних в 2019 р.} \times \text{Населення країни в 2019 р.}}{365 \times \text{Населення країни в 2018 р.}}. \tag{36}$$

Значення параметрів (35) та (36) можна обчислити на основі статистичних даних про населення країни, які знаходяться у відкритому доступі:

$$\mu_{\text{Японія}} \approx 0{,}000029; \gamma_{\text{Японія}} \approx 2520{,}026.$$

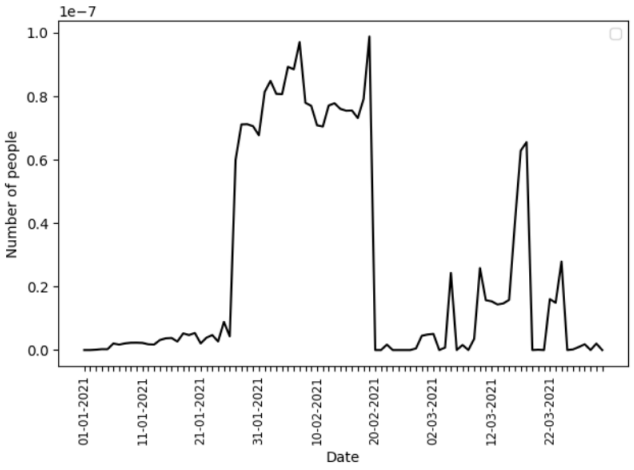

Рисунок 1 — Оптимальна оцінка параметру $\alpha(t_k)$, $k = \overline{1,92}$ в Японії з 31 грудня 2020 р. по 1 квітня 2021 р., де прямою лінією представлено $\hat{\alpha}(t_k)$, $k = \overline{1,92}$.

Підставивши їх у систему (26), одержимо базову модель для проведення імітаційного моделювання:

$$\begin{cases} S(t_{k+1}) = S(t_k) - \alpha(t_k)S(t_k)I(t_k) - 2{,}915S(t_k) + \\ \qquad +2520{,}026, S(0) = S_0 > 0, \\ I(t_{k+1}) = I(t_k) + \alpha(t_k)S(t_k)I(t_k) - \beta(t_k)I(t_k) - \\ \qquad -\mu_2(t_k)I(t_k), I(0) = I_0 \geqslant 0, \qquad k = \overline{1,92}., \\ \qquad R(t_{k+1}) = R(t_k) + \beta(t_k)I(t_k) - \\ \qquad -0{,}000029R(t_k), R(0) = R_0 \geqslant 0, \\ M(t_{k+1}) = M(t_k) + \mu_2(t_k)I(t_k), M(0) = M_0 \geqslant 0, \end{cases} \tag{37}$$

де $t_1 =$ 31 грудня 2020 р., $t_{m+1} =$ 1 квітня 2021 р., $\Delta t = t_{k+1} - t_k =$1 день, $k = \overline{1,92}$.

За формулою (32) знайдемо оптимальні оцінки невідомих параметрів $\alpha(t_k)$, $\beta(t_k)$, $\mu_2(t_k)$, $k = \overline{1,92}$ (Рис. 1 — 3).

На Рис. 4 наведено графіки похибки отриманих оптимальних оцінок невідомих параметрів моделі, обчислені за формулами (33) для різних випадків:

- $\varphi_1 = 1.1783e{+}06$ — випадок, коли всі спостереження $S(t_k)$, $I(t_k)$,$R(t_k)$ та $M(t_k)$, $k = \overline{1,92}$ були з 1%-похибкою;
- $\varphi_2 = 2.3566e + 06$ — випадок, коли всі спостереження $S(t_k)$, $I(t_k)$, $R(t_k)$ та $M(t_k)$, $k = \overline{1,92}$ були з 2%-похибкою.

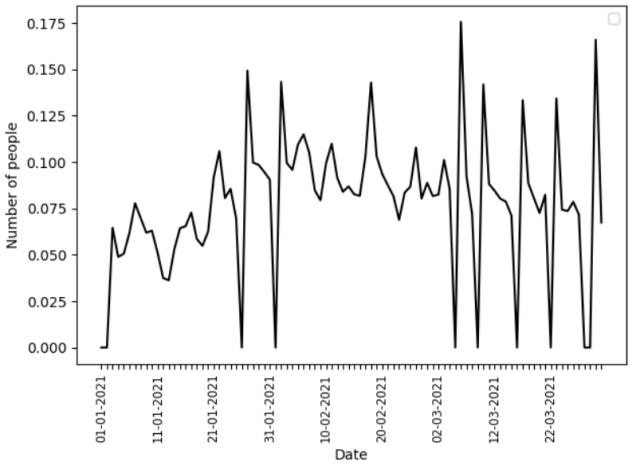

Рисунок 2 — Оптимальна оцінка параметру $\beta(t_k)$, $k = \overline{1,92}$ в Японії з 31 грудня 2020 р. по 1 квітня 2021 р., де прямою лінією представлено $\hat{\beta}(t_k)$, $k = \overline{1,92}$.

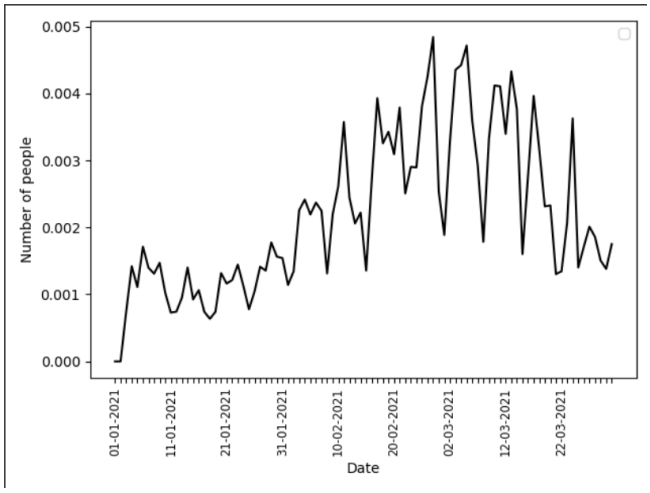

Рисунок 3 — Оптимальна оцінка параметру $\mu_2(t_k)$, $k = \overline{1,92}$ в Японії з 31 грудня 2020 р. по 1 квітня 2021 р., де прямою лінією представлено $\hat{\mu}_2(t_k)$, $k = \overline{1,92}$.

Як можна побачити на Рис. 4 похибка на початку роботи алгоритму є доволі значною, оскільки на початку роботи алгоритму покладається $\hat{\alpha}(t_1) = 0$, $\hat{\beta}(t_1) = 0$, $\hat{\mu}_2(t_1) = 0$, але з часом стає близькою до 0.

Тобто однією із особливостей знаходження оптимальних гарантованих оцінок за формулою (32) є те, що потрібна певна кількість вхідних даних для налаштування алгоритму на основі фільтра Каллмана-Бюсі. Тому адекватність оцінок евідомих параметрів потрібно оцінювати включно з інформацією про похибки цих оцінок.

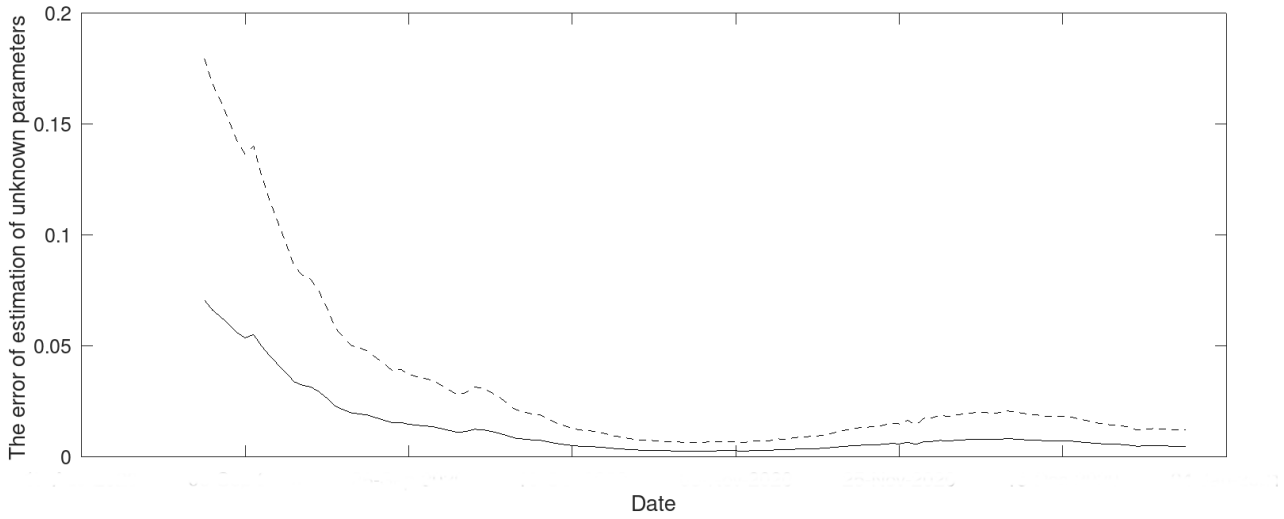

Рисунок 4 — Похибка оцінки невідомих параметрів моделі поширення COVID-19 Японії з 31 грудня 2020 р. по 1 квітня 2021 р., де суцільною лінією представлено похибку для $\varphi_1 = 1.1783e + 06$, пунктирною — для $\varphi_2 = 2.3566e + 06$.

Розділ 2. Огляд нейронних мереж

2.1 Поняття та переваги нейронних мереж

Перша хвиля інтересу до нейронних мереж (також відомих як «коннекціоністські моделі» або «паралельно розподілена обробка») з'явилася після представлення спрощених нейронів МакКалохом і Піттсом у 1943 році (McCulloch and Pitts, 1943) [60]. Ці нейрони були представлені як моделі біологічних нейронів і як концептуальні компоненти схем, які могли б виконувати обчислювальні завдання. Коли Мінський і Пеперт опублікували свою книгу «Персептрони» в 1969 році (Minsky & Papert, 1969), в якій вони показали недоліки моделей персептронів, більша частина фінансування на розвиток та дослідження нейронних мереж була перенаправлена, і дослідники залишили цю сферу діяльності. Лише декілька дослідників продовжували свої зусилля, зокрема Теуво Кохонен, Стівен Гроссберг, Джеймс Андерсон та Куніхіко Фукусіма[39]. Інтерес до нейронних мереж знову виник лише після деяких важливих теоретичних результатів , що були досягнуті на початку вісімдесятих років (насамперед відкриття зворотного поширення помилок – back-propagation error), а нові розробки апаратного забезпечення збільшили потужності обробки.

Штучні нейронні мережі можна охарактеризувати як «обчислювальні моделі» з особливими властивостями, такими як здатність адаптуватися або навчатися, узагальнювати, кластерувати або організовувати дані, і операція яких базується на паралельній обробці. Проте багато з вищезгаданих властивостей можна віднести до не нейронних моделей, що вже існують. Питання полягає в тому, наскільки нейронний підхід краще підходить для певних задач, ніж інші моделі. Наразі однозначної відповіді на це питання не знайдено.

Нейронні мережі побудовані за принципом функціонування біологічних нервових мереж. Їх схожість може бути описана на таких двох властивостях:

1. знання потрапляють в нейронну мережу із навколишнього середовища та використовуються в процесі навчання;

2. зв'язки між нейронами, які називаються синапсичними вагами, використовуються для накопичення знань.

Забезпечення потрібної структури взаємозв'язків у мережі досягається за рахунок знаходження коефіцієнтів синапсичних вагів. У процесі навчання нейронна мережа здатна виявляти складні залежності між вхідними і вихідними даними, а також виконувати узагальнення, що, у свою чергу, допомагає отримати вірний результат навіть за умови неповних та зашумлених початкових вхідних даних в навчальній виборці.

Використання нейронних мереж забезпечує наступні властивості систем [61]:

1. Відображення вхідної інформації у вихідну. Результат успішного навчання нейронною мережею має малу величину похибки.
2. Адаптивність. Нейронні мережі володіють властивістю адаптувати свої синапсичні ваги до зміни навколишнього середовища. Наприклад, для роботи в нестаціонарному середовищі (де статистика змінюється з часом) можуть бути створені нейронні мережі що змінюють синапсичні ваги в реальному часі. Але не завжди з адаптивності випливає стійкість, іноді отримується зовсім протилежний результат. Таким чином отримуємо дилему стабільності-пластичності. Щоб використовувати всі властивості адаптивності, основні параметри системи повинні бути достатньо стабільними для обминання зовнішніх перешкод, але й в той же час достатньо гнучкими для забезпечення реакції на суттєві зміни середовища.
3. Нелінійність. Штучні нейрони можуть бути лінійними та нелінійними. Нейронні мережі, сконструйовані із об'єднаних нелінійних нейронів, самі є нелінійними. Дана нелінійність розподілена по мережі.
4. Контекстна інформація. Знання представляються в самій структурі нейронної мережі за допомогою її стану активації.
5. Очевидність відповіді. В контексті задачі класифікації образів є можливість розробити нейрону мережу, яка має можливість збільшувати достовірність прийнятого рішення. Підвищення продуктивності нейронної мережі досягається при подальшому використанні даної інформації.

6. Відмовостійкість. Розподілений характер зберігання інформації дозволяє тримати рівень продуктивності майже на тому ж рівні при незначних пошкодженнях структури.
7. Однаковість аналізу та проектування. Одне й те ж проектне рішення нейронної мережі може використовуватися в багатьох предметних областях.
8. Масштабування. Паралельна структура нейронних мереж потенційно пришвидшує рішення деяких задач та забезпечує масштабування нейронних мереж в рамках VLSI (very large scale integration – надвеликі інтегральні схеми), однією з переваг якої є можливість представити достатньо складну поведінку за допомогою ієрархічної структури.

2.2 Модель нейрона

Нейрон представляє собою одиницю опрацювання інформації в нейронній мережі. Три основні елемента виділені на блок-схемі рис. 5.

1. Набір синапсів або зв'язків. Кожна одиниця характеризується своєю вагою або силою. Зокрема, сигнал x_j на вході синапса k, зв'язаного з нейроном j, множиться на вагу w_{kj} . Перший індекс синапсичної ваги w_{kj}, яка може мати як додатні, так і від'ємне значення, відноситься до нейрону, що розглядаються, а інший – до вхідного закінчення синапсису, з яким зв'язана дана вага. Ваги — дійсні числа.
2. Суматор (лінійна комбінація). Складає вхідні сигнали, зважені відносно відповідних синапсисів нейрона.
3. Функція активації (стиснення). Головною функцією є обмеження амплітуду вхідного сигналу нейрону. Зазвичай нормалізований діапазон амплітуд виходу нейрона лежить в інтервалі [0,1] або [-1,1].
4. Зміщення (поріг).

Іноді нейрон використовує зовнішньо забезпечене значення зміщення. В моделі нейрона, яка показана на рис. 5, включений пороговий елемент, який позначений символом b_k . Ця величина відображає збільшення або

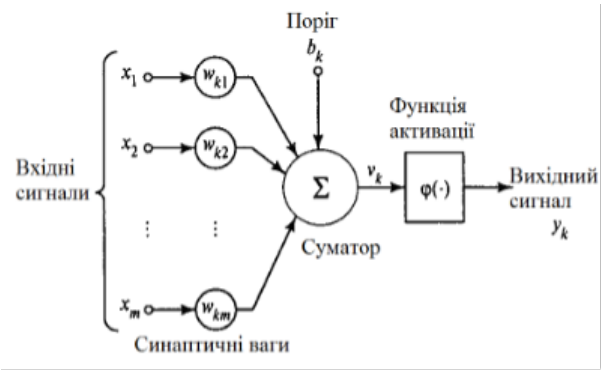

Рисунок 5 — Нелінійна модель нейрона.

зменшення вхідного сигналу, який подається на функцію активації. В математичному представленні функціонування нейрону k можна описати наступними рівняннями:

$$\sum_{j=1}^{m} w_{kj} \cdot x_j, \tag{38}$$

$$y_k = \varphi(u_k + b_k), \tag{39}$$

де x_1,x_2,...,x_m - вхідні сигнали; w_{k1},w_{k2},...,w_{km} - синапсичні ваги нейрона k ; u_k- лінійна комбінація вхідних впливів; b_k- поріг; φ - функція активації; y_k - вихідний сигнал нейрона. Використання порогу забезпечує ефект афінного перетворення виходу лінійного суматора. В моделі, що зображена на рис. 5, постсинапсичний потенціал вираховується наступним чином:

$$v_k = u_k + b_k. \tag{40}$$

В залежності від від'ємності або додатності значення зміщення індуковане локальне поле або потенціал активації нейрона k змінюється. Хоча й у загальному випадку вхідний сигнал, синапсичні ваги та поріг можуть приймати дійсні значення, та в багатьох практичних задачах вони здатні бути лише деякими фіксованими значеннями. Вид активації формує вихід (y_k), який може бути як дійсним, так і цілим.

Синапсичні зв'язки з додатніми вагами називають збуджуючими, а з від'ємними – гальмуючими [62].

2.3 Функції активації

Метою функції активації є додавання нелінійності до нейронної мережі. Функції активації, що представлені в формулах як $\varphi(v)$, вводять додатковий крок на кожному рівні під час прямого поширення, але його обчислення того варте.

Припустимо існування нейронної мережі без функцій активації. У цьому випадку кожен нейрон буде виконувати лише лінійне перетворення на входах, використовуючи вагові коефіцієнти та зміщення. Не має значення, скільки прихованих шарів буде приєднано до нейронної мережі, оскільки всі шари будуть вести себе однаково — композиція двох лінійних функцій сама є лінійною функцією. Хоча нейронна мережа стає простішою, вивчення будь-якої складної задачі неможливо, і модель стає просто моделлю лінійної регресії. Можна виділити три основні типи функцій активації:

1. Функція одиничного стрибка, або порогова функція, функція Хевісайда

$$\varphi(v) = \begin{cases} 0 & \text{v} < 0, \\ 1 & \text{v} >= 0. \end{cases} \tag{41}$$

Відповідно вихідний сигнал нейрона k такої функції можна представити як

$$y_k = \begin{cases} 0 & v_k < 0, \\ 1 & v_k >= 0, \end{cases} \tag{42}$$

де v_k - це індуковане локальне поле нейрона, тобто

$$v_k = \sum_{j=0}^{m} w_{kj} \cdot x_j + b_k, \tag{43}$$

що становить модель Мак-Калока-Пітца (McCal1och-Pitts mode1). В ній вихідний сигнал нейрона приймає значення 1, якщо індуковане локально поле цього нейрона не від'ємне, та 0 – в іншому випадку. Цей вираз описує властивість «все або нічого» даної моделі.

2. Кусково-лінійна функція.

$$\varphi(v) = \begin{cases} 0 & \text{v} <= \text{-0.5}, \\ v & \text{-0.5} < \text{v} < 0.5, \\ 1 & \text{v} >= 0.5, \end{cases} \tag{44}$$

де коефіцієнт посилення в лінійній області оператора рівний одиниці. Цю функцію активації можна розглядати як апроксимацію нелінійного посилювача.

3. Сигмоїдальна функція. Вважається найрозповсюдженішою функцією, яку використовують для створення штучних нейронних мереж. Вона є швидкозростаючою функцією, яка підтримує баланс між лінійною на нелінійної поведінкою. Прикладом сигмоїдальної функції може бути логістична функція, яка задається виразом:

$$\varphi(v) = \frac{1}{1 + e^{-av}}, \tag{45}$$

де α – параметр нахилу сигмоїдальної функції. Змінюючи цей параметр, можна побудувати функцію з різним нахилом. Сигмоїдальна функція вироджується в порогову в границі, коли параметр нахилу досягає нескінченності. Дана функція приймає нескінченну множину значень в діапазоні від 0 до 1. На відміну від порогової функції сигмоїдальна є диференційованою, причому на всій вісі абсцис. Крім того, вона має властивість підсилювати слабкі сигнали краще, ніж великі, та запобігає перенасичення від великих сигналів.

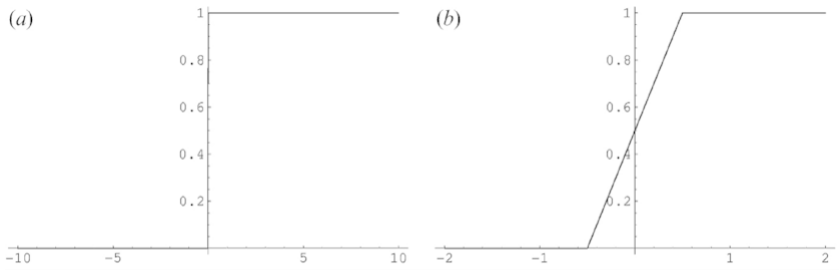

Рисунок 6 — Види активаційних функцій: функція одиничного скачка (а); кусково-лінійна функція (б)

Інколи потрібна функція активації , яка має область значень від -1 до +1. Порогову функцію, сигнум, в даному випадку можна визначити наступним чином:

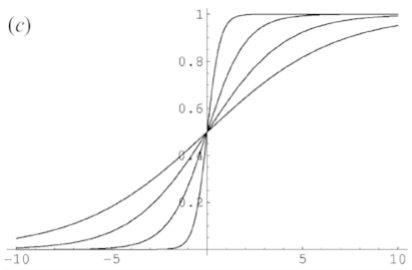

Рисунок 7 — Сигмоїдальна функція для різних величин значень параметра a (в)

$$\varphi(v) = \begin{cases} -1 & \text{-1 v <= 0,} \\ 0 & \text{v = 0 ,} \\ 1 & \text{v > 0.} \end{cases} \tag{46}$$

В даному випадку сигмоїдальна функція буде мати форму гіперболічного тангенса:

$$\varphi(v) = \tanh(v). \tag{47}$$

Під час навчання глибоких нейронних мереж на практиці зустрічаються дві проблеми, а саме градієнти, що зникають та “вибухають”.

Як і сигмоїдальна функція, певні функції активації стискають великий простір введення в невеликий вихідний простір між 0 і 1. Отже, велика зміна на вході сигмоїдної функції спричинить невелику зміну на виході, тобто похідна стає малою. Для неглибоких мереж лише з кількома шарами, які використовують ці активації, це не є великою проблемою, але при використанні більшої кількості шарів, градієнт може бути занадто малим, щоб навчання працювало ефективно. Проблема вибухових градієнтів полягає в накопиченні значних помилок градієнтів, що призводить до дуже великих оновлень ваг моделі нейронної мережі під час навчання, яке може спричинити неможливість завершення навчання. Значення ваг також можуть стати настільки великими, що переповнюються і призводять до того, що називається значеннями NaN. Хоча як сигмоїдна, так і $\tanh(v)$ стикаються з проблемою зникаючого градієнта, $\tanh(v)$ має нульовий центр, тобто градієнти не обмежені для руху в певному напрямку. Тому на практиці завжди віддають перевагу функції активації $\tanh(v)$, а не логістичній сигмоїдальній [62].

Назва	Формула	Область значень
Лінійна	$\varphi(v) = kv$	(-∞, ∞)
Напівлінійна	$\varphi(v) = \begin{cases} kv & \text{-1 v} > 0 \\ 0 & \text{v} <= 0 \end{cases}$	(0, ∞)
Логістична (сигмоїдальна)	$\varphi(v) = \frac{1}{1+e^{-av}}$	(0, 1)
Гіперболічний тангенс (сигмоїдальна)	$\varphi(v) = \frac{e^{av}-e^{-av}}{e^{av}+e^{-av}}$	(-1, 1)
Експоненціальна	$\varphi(v) = e^{-av}$	(0, ∞)
Синусоїдальна	$\varphi(v) = sin(v)$	(-1, 1)
Сигмоїдальна (раціональна)	$\varphi(v) = \frac{v}{a+\lvert v\rvert}$	(-1, 1)
Крокова (лінійна з насиченням)	$\varphi(v) = \begin{cases} -1 & \text{-1 v} <= 1 \\ v & \text{-1} < \text{v} < 1 \\ 1 & \text{v} >= 1 \end{cases}$	(-1, 1)
Кусково-лінійна	$\varphi(v) = \begin{cases} 0 & \text{v} <= \text{-0.5} \\ v & \text{-0.5} < \text{v} < 0.5 \\ 1 & \text{v} >= 0.5 \end{cases}$	(0, 1)
Порогова (одиничного скачка)	$\varphi(v) = \begin{cases} 0 & \text{v} < 0 \\ 1 & \text{v} >= 0 \end{cases}$	(0, 1)
Модульна	$\varphi(v) = \lvert v\rvert$	(0, ∞)
Знакова(сигнатурна)	$\varphi(v) = \begin{cases} 1 & \text{v} > 0 \\ -1 & \text{v} <= 0 \end{cases}$	(-1, 1)
Квадратична	$\varphi(v) = v^2$	(0, ∞)

Таблиця 1 — Традиційні функції активації

Розглянемо більш сучасні функції активації. ReLU (Rectified Linear Unit), або зрізаний лінійний вузол, створює враження лінійної функції, але дана функція має похідну і дозволяє здійснювати зворотне поширення та водночас робить його обчислювально ефективним. Функція ReLU не активує всі нейрони одночасно. Нейрони будуть дезактивовані, лише якщо вихідні дані лінійного перетворення менше 0.

Переваги використання ReLU як функції активації полягають у наступному:

- оскільки активується лише певна кількість нейронів, функція ReLU є набагато ефективнішою в обчислювальному відношенні порівняно з гіперболічним тангенсом та сигмоїдальною функцією;
- ReLU прискорює збіжність градієнтного спуску до глобального мінімуму функції витрат завдяки своїй лінійній властивості, що не має насичення.

Обмеження, з якими стикається ця функція, становить проблема під назвою Dying ReLU — ReLU, що "помирає". Від'ємні значення аргументу роблять значення похідної нульовим. З цієї причини під час процесу зворотного поширення ваги та зміщення для деяких нейронів не оновлюються. Це може створити мертві нейрони, які ніколи не активуються. Усі від'ємні вхідні значення відразу стають нульовими, що зменшує здатність моделі правильно підходити до даних або тренуватися на них.

Leaky ReLU (Нещільний ReLU) — це покращена версія функції ReLU для вирішення проблеми Dying ReLU, оскільки вона має невеликий позитивний нахил у області від'ємних значень. Переваги Leaky ReLU такі ж, як і в ReLU, але додатково вона дозволяє зворотне поширення навіть для від'ємних вхідних значень.

Зробивши цю незначну модифікацію для від'ємних вхідних значень, градієнт лівої частини графіка виявиться ненульовим значенням. Таким чином проблема мертвих нейронів в цій області буде уникнена. Обмеження, з якими стикається ця функція:

- прогнози можуть бути невідповідними для від'ємних вхідних значень;
- градієнт для від'ємних значень — це невелике значення, що робить вивчення параметрів моделі трудомістким.

Параметричний ReLU (Parametric ReLU) — це ще один варіант ReLU, який має на меті вирішити проблему зведення градієнта до нуля для лівої

половини осі. Ця функція надає нахил від'ємної частини функції як аргумент a. Виконуючи зворотне поширення, вивчається найбільш відповідне значення a. Параметрична ReLU використовується, коли leaky ReLU все ще не вдається вирішити проблему мертвих нейронів, і відповідна інформація не передається наступному шару. Обмеження цієї функції полягає в тому, що вона може працювати по-різному для різних проблем залежно від значення параметра нахилу a. Exponential Linear Units, або експоненціально-лінійний вузол ELU, також є варіантом ReLU, який змінює нахил від'ємної частини функції. ELU використовує логарифмічну криву для визначення від'ємних значень, на відміну від leaky ReLU та parametric ReLU з прямою лінією. Дана функція є сильною альтернативою ReLU через наступні переваги:

- згладжується повільно, поки його вихід не дорівнює -α, тоді як RELU різко згладжується;
- уникає "мертвої проблеми"ReLU шляхом введення кривої журналу для негативних значень вхідних даних.

Це допомагає мережі підштовхнути ваги та зміщення до зміни в правильному напрямку. Обмеження функції ELU:

- збільшений час обчислень завдяки доданої експоненційної операції;
- значення «a» може не вивчатися;
- проблема вибухового градієнта.

Функція SoftMax описується як комбінація кількох сигмоїдальних функцій, що обчислює відносні ймовірності. Подібно до функції сигмоподібної/логістичної активації, функція SoftMax повертає ймовірність кожного класу для проблеми класифікації, тому найчастіше вона використовується як функція активації для останнього шару нейронної мережі у випадку багатокласової класифікації. Swish — функція автоматичної активації, розроблена дослідниками Google. Swish перевершує або показує такі ж результати як і функція активації ReLU у глибоких мережах, що застосовуються до різних складних доменів, таких як класифікація зображень, машинний переклад тощо. Переваги функції активації Swish перед ReLU:

- є плавною функцією, що означає, що вона не змінює різко напрямок, як це робить ReLU в околі нуля;
- невеликі негативні значення були обнулені у функції активації ReLU, однак ці від'ємні значення все ще можуть бути релевантними для

фіксації закономірностей, що лежать в основі даних. Великі від'ємні значення обнулюються через розрідженість, що приносить позитивні результати;

– будучи немонотонною, покращує вираження вхідних даних і синапсичних зв'язків, які потрібно вивчити.

Функція активації лінійної одиниці помилки Гауса (GELU — Gaussian Error Linear Unit) сумісна з BERT, ROBERTa, ALBERT та іншими моделями нейролінгвістичного програмування та становить високопродуктивну функцію активації нейронної мережі. GELU — це $x \cdot F(x)$, де $F(x)$ — стандартна функція кумулятивного розподілу Гаусса. Нелінійність GELU зважує вхідні дані за їх значенням, а не за знаком, як у ReLU. Цей розподіл вибрано, оскільки вхідні нейрони мають тенденцію слідувати нормальному розподілу, особливо при пакетній нормалізації (batch normalization). Нелінійність GELU краща, ніж активація ReLU та ELU, і забезпечує покращення продуктивності в усіх задачах у сферах комп'ютерного зору, обробки природної мови та розпізнавання мовлення. Scaled Exponential Linear Unit (Маштабований експоненціально-лінійний вузол) — SELU було визначено в мережах, що самонормалізуються, і піклуються про внутрішню нормалізацію, що означає, що кожен рівень зберігає середнє значення та дисперсію від попередніх шарів. SELU дозволяє цю нормалізацію, коригуючи середнє значення та дисперсію. SELU має як позитивні, так і негативні значення для зсуву середнього, що було неможливим для функції активації ReLU, оскільки вона не може виводити негативні значення. Для регулювання дисперсії можна використовувати градієнти. Для збільшення функції активації потрібна область з градієнтом, більшим за одиницю. Функція має попередньо визначені значення альфа і лямбда.

Основна перевага SELU перед ReLU полягає у внутрішній норалізації, яка відбувається швидше, ніж зовнішня, що призводить до швидкої збіжності. SELU є відносно новою функцією активації і потребує більшої документації для таких архітектур, як CNN (згорткові нейронні мережі) і RNN (рекурентні нейронні мережі), де вона слабо досліджена.

Назва	Формула	Область значень
ReLU	$\varphi(v) = max(0, v)$	(-∞, ∞)
Leaky ReLU	$\varphi(v) = max(0{,}0.1 * v)$	(-∞, ∞)
Parametric ReLU	$\varphi(v) = max(v{,}a * v)$	(-∞, ∞)
SoftMax	$\varphi(v) = \frac{e^{vi}}{\sum_{i=1}^{m} e^{vi}}$	(-∞, ∞)
Swish	$\varphi(v) = sigmoid(v) \cdot v$	(-∞, ∞)
GELU	$\varphi(v) = 0.5v(1 + \tanh(\sqrt{\frac{2}{\pi}}(v + 0.045x^3)))$	(-∞, ∞)
SELU	$\varphi(\lambda{,}v) = \begin{cases} \lambda a(e^x - 1), \text{v} < 0 \\ \lambda v, \ \text{v}>=0 \end{cases}$	(-∞, ∞)

Таблиця 2 — Сучасні функції активації

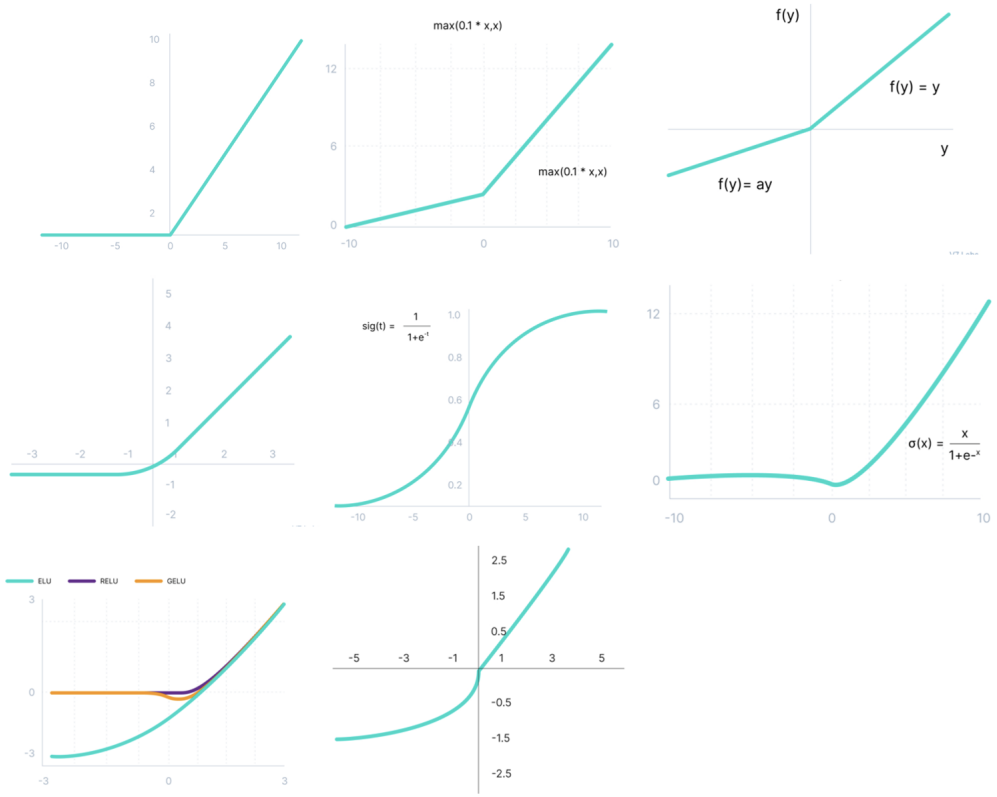

Рисунок 8 — Графіки сучасних функцій активації. ReLU, Leaky ReLU, Parametric ReLU, ELU, SoftMax, Swish, GELU, SELU зліва направо.

2.4 Класифікація нейронних мереж та їх властивості

Нейронна мережа представляє собою сукупність нейроннoподібних елементів, певним чином з'єднаних один з одним та з зовнішнім середовищем за допомогою зв'язків, визначених синапсними вагами. В залежності від функцій, які виконують нейрони в мережі, можна виділити три типи:

- Вхідні нейрони. В них зазвичай не виконуються обчислювальні процедури, а інформація передається з входу на вихід шляхом зміни їх активації;
- Вихідні нейрони, вихідні значення яких представляють виходи нейронної мережі; перетворення в них виконуються за виразами 38 та 39;
- Проміжні нейрони, які складають основу нейронних мереж, перетворення в яких виконуються також за виразами 38 та 39.

В переважній більшості нейронних моделей тип нейрона пов'язаний з його положенням в мережі. В процесі функціонування нейронної мережі виконується перетворення вхідного вектора в вихідний, опрацювання поданої інформації.

Конкретний вигляд перетворення даних, що виконується мережею, обумовлюється характеристиками нейроподібних елементів та особливостями її архітектури, а саме топологією міжнейронних зв'язків, вибором певної підмножин нейроподібних елементів для вводу та виведення інформації, способами навчання мереж, наявністю чи відсутністю конкуренції між нейронами, напрямком та способами керування та синхронізації передачі між нейронами.

З точки зору топології можна виділити три основних типи нейронних мереж:

- повнозв'язні;
- багатозв'язні або шаруваті;
- слабозв'язні (з локальними зв'язками).

В повнозв'язних нейронних мережах кожен нейрон передає вихідний сигнал іншим нейронам, а тому числі і самому собі. Всі вхідні сигнали подаються всім нейронам. Вихідними сигналами мережі можуть бути

всі або деякі вихідні сигнали нейронів після декількох тактів, або епох, функціонування мережі.

В багатозв'язних нейронних мережах нейрони об'єднані в шари. Шар містить сукупність нейронів з єдиними вхідними сигналами. Число нейронів в шарі може бути будь-яким та не залежить від кількості нейронів в інших шарах. В загальному випадку мережа складається з Q шарів, пронумерованих зліва на право. Зовнішні вхідні сигнали подаються на вхід нейронам вхідного шару (його часто нумерують як нульовий), а виходами мережі є вихідні сигнали останнього шару. Крім вхідного та вихідного шарів в багатошарових нейронних мережах є один або декілька прихованих шарів. Зв'язки від виходів нейронів деякого шару q до входів наступного шару (q+1) називають послідовними.

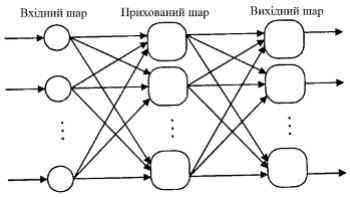

Рисунок 9 — Багатошарова (двошарова) мережа прямого розповсюдження.

В свою чергу, серед багатозв'язних нейронних мереж виділяють наступні типи.

1. Монотонні. Це частковий випадок багатозв'язних мереж з додатковими умовами на зв'язки та нейрони. Кожен шар, крім останнього (вихідного), розбитий на два блоки: збуджуючий та гальмуючий. Зв'язки між блоками також розділяються на гальмуючі та збуджуючі. Якщо від нейронів блоку A до нейронів блоку B ведуть лише збуджуючі зв'язки, то це означає, що будь-який вхідний сигнал блоку є монотонно неспадаючою функцією будь-якого вхідного сигналу блоку A. Якщо ці зв'язки тільки гальмуючі, то будь-який сигнал блоку B є незростаючою функцією будь-якого вхідного сигналу блоку A. Для нейронів монотонних мереж необхідна монотонна залежність вихідного сигналу нейрона від параметрів вхідних сигналів.
2. Мережі без зворотніх зв'язків. В таких мережах нейрони вхідного шару отримують вхідні сигнали, перетворюють їх та передають

нейронам першого прихованого шару, і так далі аж до вихідного, який видає сигнали для інтерпретатора та користувача. Якщо не оговорено інакше, то кожен вихідний шар q-го шару подається на вхід всіх нейронів (q+1) шару; також можливий варіант з'єднання q-го шару з довільним (q+p)- шаром. Серед багатозв'язних мереж без оберненого зв'язку розрізняють повнозв'язні (вихід кожного нейрона q-го шару зв'язаний з входом кожного нейрона (q+1)-го шару) та частково повнозв'язні. Класичним варіантом багатозв'язних мереж є повнозв'язні мережі прямого розповсюдження рис. 9.

3. Мережі зі зворотніми зв'язками. В цих мережах інформація з наступних шарів подається на попередні.

Серед останніх, в свою чергу, виділяють наступні:

- шарово-циклічні, які відрізняють тим, що шари замкнені в кільце: останній шар передає свої вихідні сигнали першому; всі шари рівноправні та можуть як отримувати вхідні сигнали, так і видавати вихідні;
- шарово-повнозв'язні складаються із шарів, кожен з яких представляє собою повнозв'язну мережу, а сигнали передаються як від шару до шару, так і всередині шару; в кожному шарі цикл роботи розкладається на три частини: отримання сигналів з попереднього шару, обмін сигналами всередині шару, вироблення вихідного сигналу та передача його до наступного шару.
- повнозв'язно-шарові, по своїй структурі аналогічні шарово-повнозв'язним мережам, але функціонують інакше — в них не розділяються фази обміну всередині шару та передачі наступному, на кожному такті нейрони всіх шарів приймають сигнали від нейронів як свого шару, так і наступних.

В слабкозв'язаних нейронних мережах нейрони розміщуються в вузлах прямокутних або гексагональних грат. Кожен нейрон зв'язаний з чотирма (окіл фон Неймана), шістьмома (окіл Голея) або вісьмома (окіл Мура) своїми найближчими сусідами. Відомі нейронні мережі можна розділити по типу структур нейронів на гомогенні (однорідні) та гетерогенні. Гомогенні мережі складаються з нейронів одного типу з єдиною функцією активації, а гетерогенні — з різними функціями активації. Існує поділ на бінарні та аналогові мережі. Перші з них оперують тільки двійковими сигналами, в той

час як вихід кожного нейрона може приймати значення або логічного нуля (гальмуючий стан), або логічної одиниці (збуджений стан).

Ще одна класифікація ділить нейронні мережі на синхронні та асинхронні. В першому випадку в кожен момент часу лише один нейрон змінює свій стан, в другому — стан змінюється відразу в цілої групи нейронів у всього шару.

Число шарів також може бути варіантом класифікації нейронної мережі. Теоретично число шарів та число нейронів в кожному шарі може бути довільним, але фактично воно обмежене ресурсами комп'ютера або спеціалізованих мікросхем, на яких зазвичай реалізується нейронна мережа. Чим складніша мережа, глибша у багатьох прикладах, тим складніші задачі вона може вирішувати.

Складність та особливість постановки задачі формують структуру нейронної мережі. Вирішення окремих типів задач потребує вже встановлені оптимальні конфігурації. Якщо ж задача не може бути зведена до якогось одного з наведених типів, то вирішується складна проблема синтезу нової конфігурації.

При цьому потрібно керуватися наступними правилами:

- можливості мережі зростають зі збільшенням числа нейронів мережі, густини зв'язків між ними та кількістю шарів;
- введення зворотніх зв'язків, як і збільшення можливостей мережі, піднімає питання про динамічну стійкість мережі;
- складність алгоритмів функціонування мережі, введення декількох типів синапсів сприяє посиленню потужності нейронної мережі.

Питання про необхідні та достатні властивості мережі для розв'язку задач того чи іншого роду представляє собою цілий напрямок нейрокомп'ютерної науки. Оскільки проблема синтезу нейронної мережі сильно залежить від поставленої задачі, чітко виділити основні тенденції рекомендацій вкрай складно. В більшості випадків оптимальний варіант отримується на основі інтуїтивного підбору, хоча в літературі наведені доведення того, що для будь-якого алгоритму існує нейронна мережа, яка може його реалізувати [62].

2.5 Топології нейронних мереж

- Персептрон. Найпростіша і найстаріша модель нейрона. Бере деякі вхідні дані, підсумовує їх, застосовує функцію активації та передає їх на вихідний рівень.
- Нейронні мережі прямої подачі (Feed Forward — FF), також досить старі — підхід походить із 50-х років. Вона є багатозв'язною повнозв'язною, активація перетікає від вхідного рівня до вихідного, без зворотних циклів, між входом і виводом є один шар (прихований шар). У більшості випадків цей тип мереж навчається за допомогою методу зворотного поширення.
- Радіальна базова мережа (Radial Basis Network — RBF) — це фактично FF мережа, які використовують радіальну базисну функцію як функцію активації замість логістичної функції. Логістична функція відображає деяке довільне значення в діапазоні від 0 до 1, що зручно для систем класифікації та прийняття рішень, але погано працює для неперервних значень. З іншого боку, радіальні базисні функції відповідають на питання «як далеко ми від цілі»? Це ідеально підходить для наближення функцій та керування машиною (наприклад, як заміна PID-контролерів).
- Глибинні нейронні мережі прямої подачі (Deep Feed Forward — DFF). Становлять собою FF, але з більш ніж одним прихованим шаром. Під час навчання традиційної FF передається лише невелика кількість помилок на попередній рівень. Через те, що побудова більшої кількості шарів призводила до експоненціального зростання часу навчання, DFF ставала досить непрактичною. Лише на початку 00-х було розроблено багато підходів, які дозволили ефективно тренувати DFF. Наразі вони становлять ядро сучасних систем машинного навчання, охоплюючи ті ж цілі, що й FF, але з набагато кращими результатами.
- Рекурентні нейронні мережі (Recurrent Neural Networks — RNN) представляють різні типи клітин — повторювані клітини. Першою мережею такого типу була так звана мережа Йордана, коли кожен з прихованих нейронів отримував свій вихід з фіксованою затримкою

— одну або кілька ітерацій. Якщо не брати до уваги цю властивість, то це була звичайна мережа прямої подачі. Існує багато варіацій — наприклад, передача стану вхідним вузлам, змінні затримки тощо, але основна ідея залишається незмінною. Цей тип нейронної мережі в основному використовується тоді, коли важливий контекст — коли рішення з минулих ітерацій або вибірок можуть вплинути на поточні. Найпоширенішими прикладами таких контекстів є тексти — слово можна проаналізувати лише в контексті попередніх слів чи речень.

– Long Short Term Memory (Довга короткочасна пам'ять) — LSTM. Цей тип мережі містить нейрон пам'яті (або клітину пам'яті), спеціальну комірку, яка може обробляти дані, коли дані мають часові проміжки (або затримки). RNN можуть обробляти тексти, «згадуючи» десять попередніх слів, а мережі LSTM можуть обробляти відеокадр, «згадуючи» те, що сталося багато кадрів тому. Мережі LSTM також широко використовуються для розпізнавання письма та мовлення. Клітини пам'яті насправді складаються з кількох елементів, які називаються воротами (gate), які повторюються і контролюють, як інформація запам'ятовується та забувається. Між блоками немає функції активації. Ворота мають свої власні ваги, а іноді й функції активації. На кожному зразку вони вирішують, чи передавати дані вперед, стерти пам'ять тощо. Вхідні ворота вирішують, скільки інформації з останнього зразка буде зберігатися в пам'яті. Вихідні ворота регулюють кількість даних, що передаються наступному шару, а ворота забуття контролюють міру розриву збереженої пам'яті.
– Вентильний рекурентний вузол (GRU, Gated Recurrent Unit) — це LSTM з різним періодом застосування воріт. Відсутність вихідних воріт полегшує повторення одного і того ж результату для конкретного входу кілька разів, і на даний момент вони найчастіше використовуються в синтезі звуку (музики) та мовлення.
– Автокодери (Autoencoder — AU) використовуються для класифікації, кластеризації та стиснення ознак. Навчання мереж FF для класифікації потребує «навчання з вчителем». AE, з іншого боку, можна навчати без вчителя. Їхня структура складається меншої кількісті прихованих комірок в порівнянні із кількістю вхідних комірок (а кількість вихідних комірок дорівнює кількості вхідних

комірок). Ці моделі навчаються шляхом надання вхідних даних для моделі як вхідних, так і цільових вихідних даних, вимагаючи, щоб модель відтворювала вхід, спочатку кодуючи його у стиснене представлення, а потім декодуючи його назад до оригіналу. Після навчання декодер відкидається, а кодер використовується в міру необхідності для створення компактних представлень вхідних даних [75].

- Варіаційні автокодери (Variational AE — VAE), порівняно з AE, стискають ймовірності замість ознак. Незважаючи на цю просту зміну, коли AE відповідають на питання «як дані можуть бути узагальнені?», VAE відповідають на питання «наскільки сильний зв'язок між двома подіями?» та «чи слід розподіляти помилку між двома подіями, чи вони абсолютно незалежні?».
- Розріджені Автокодери (Sparse AE — SAE). Вони є ще одним типом автокодера, який у деяких випадках може виявити деякі приховані шаблони групування в даних. Структура така ж, як і в AE, але кількість прихованих нейронів більше, ніж кількість нейронів вхідного/вихідного шару.
- Ланцюги Маркова (Markov Chains). Досить стара концепція графів, де кожне ребро має ймовірність. Раніше використовувалися для створення текстів із ймовірністю слідування деяких слів після якогось фіксованого слова. Не є нейронними мережами в класичному ключі, але їх можна використовувати для класифікації на основі ймовірностей (наприклад, байєсівські фільтри), для кластеризації і як скінченний автомат.
- Мережі Хопфілда (Hopfield Network — HN) навчаються на обмеженому наборі вибірок, тому вони відповідають на відомий зразок тим же самим зразком. Кожен нейрон служить як вхідний нейрон перед тренуванням, як прихований нейрон під час навчання та як вихідний нейрон при використанні. Оскільки HN намагаються відновити навчений зразок, їх можна використовувати для зменшення шуму та відновлення вхідних даних. Враховуючи половину вивченого зображення або послідовності, вони повернуть повний зразок.

- Машина Больцмана (Bolzmann Machine — BM). Є дуже схожими на мережі Хопфілда, де деякі нейрони позначаються як вхідні та залишаються прихованими. Вхідні нейрони стають вихідними, як тільки кожен прихований нейрон оновлює свій стан (під час навчання BM/HN оновлюють клітинки по черзі, а не паралельно). Це перша топологія мережі, яка була успішно отримана за допомогою алгоритму імітації відпалу. Кілька складених машин Больцмана можуть створювати так звану мережу глибоких переконань, яка використовується для виявлення та вилучення ознак.
- Обмежені Машини Больцмана (Restricted BM). RBM за своєю структурою нагадують BM, але через обмеження дозволяють навчатися за допомогою зворотного поширення так само, як і нейронні мережі прямої подачі (з тією лише різницею, що перед зворотним поширенням дані передаються назад на вхідний рівень один раз).
- Глибинні згорткові нейронні мережі (Deep Convolutional Neural Networks — DCN). Наразі є найбільш поширеними та прийнятними в колі дослідників штучні нейронні мережі. Вони містять нейрони згортки (або об'єднані шари) та ядра, кожне з яких виконує різні цілі. Ядра згортки фактично обробляють вхідні дані, а шари об'єднання спрощують це (в основному з використанням нелінійних функцій, таких як max), зменшуючи кількість непотрібних функцій. Зазвичай використовуються для розпізнавання зображень, вони працюють з невеликою підмножиною зображення (щось приблизно 20x20 пікселів). Вікно введення ковзає по зображенню, піксель за пікселем. Дані передаються до шарів згортки, які утворюють воронку (стиск виявлених ознак). З точки зору розпізнавання зображень, перший шар виявляє градієнти, другий — лінії, третій — форми і так далі в масштабі окремих об'єктів. DFF зазвичай приєднуються до останнього згорткового шару для подальшої обробки даних. Всі ворота LSTM об'єднані в так звані ворота оновлення (update gate), а ворота скидання (reset gate) тісно пов'язані до вводу. Вони споживають менше ресурсів, ніж LSTM, і майже так само ефективні.
- Антизгорткові нейронні мережі (Deconvolutional Network — DN). Представляють собою обернену задачу згорткових нейронних мереж.

- Глибока згортка інверсної графічної мережі (Deep Convolutional Inverse Graphics Network — DCIGN). Виглядає як поєднання DCN і DN, але насправді становить собою автокодер. DCN і DN не діють як окремі мережі, натомість вони є прокладками для введення і виведення мережі. В основному використовуються для обробки зображень, ці мережі можуть обробляти зображення, з якими вони раніше не були навчені. Ці сітки, завдяки своїм рівням абстракції, можуть видаляти певні об'єкти із зображення, перефарбовувати їх або замінювати.
- Генеративна змагальна мережа (Generative Adversarial Network — GAN). Дана нейронна мережа становить величезне сімейство подвійних мереж, які складаються з генератора і дискримінатора. Вони постійно намагаються обдурити один одного — генератор намагається згенерувати якісь дані, а дискримінатор, отримуючи вибіркові дані, намагається відрізнити згенеровані дані від вибірок. Цей тип нейронних мереж, який постійно розвивається, може генерувати реальні зображення, якщо розробник зможе підтримувати баланс навчання між цими двома мережами.
- Рідкий скінченний автомат (Liquid State Machine — LSM). Розріджена (не повнозв'язана) нейронна мережа, де функції активації замінені пороговими рівнями. Нейрон накопичує значення з послідовних вибірок і видає вихідні дані лише тоді, коли досягнуто порогове значення, знову встановлюючи внутрішній лічильник на нуль. Така ідея взята з людського мозку, і ці мережі широко використовуються в комп'ютерних системах зору та розпізнавання мовлення, але без серйозних проривів.
- Машина екстремального навчання (Extreme Learning Machine — ELM). Нейронна мережа є спробою зменшити складність мереж FF шляхом створення розріджених прихованих шарів із випадковими з'єднаннями. Вони вимагають меншої обчислювальної потужності, але фактична ефективність значною мірою залежить від завдання та даних.
- Мережа з відлуння стану (Echo State Network — ESN). Підтип рекурентних мереж зі спеціальним підходом до навчання. Дані передаються на вхід, а потім на вихід, якщо відстежуються протягом

кількох ітерацій (дозволяючи повторюваним функціям запускатися). Після цього оновлюються лише ваги між прихованими клітинками.

– Залишкова нейронна мережа (Deep Residual Network — DRN). Це глибока мережа, де деяка частина вхідних даних передається на наступні рівні. Ця функція дозволяє їм бути дійсно глибокими (до 300 шарів), але насправді вони є свого роду RNN без явної затримки.
– Самоорганізаційна карта Кохонена (Kohonen Network — KN). Вводить функцію «відстань до нейрона». В основному використовується для класифікації, цей тип мережі намагається налаштувати свої нейрони для максимальної реакції на певний вхід. Коли дейкий нейрон оновлюється, її найближчі сусіди також оновлюються. Як і SVM, ці мережі не завжди вважаються «справжніми» нейронними мережами.
– Машина опорних векторів (Support Vector Machine — SVM). Використовуються для завдань бінарної класифікації. Незалежно від того, скільки вимірів або вхідних даних мережа може обробити, відповідь завжди буде «так» або «ні». SVM не завжди вважаються нейронною мережею.
– Нейронна мережа Тюринга (Neural Turing Machine — NTM). Нейронні мережі — це свого роду чорні ящики, оскільки ми можемо навчати їх, отримувати результати, покращувати їх, але фактичний шлях прийняття рішень здебільшого прихований від розробника. NTM становить спробу виправити це і являє собою FF з витягнутими нейронами пам'яті. Деякі автори також кажуть, що це абстракція над LSTM. Пам'ять адресується її вмістом, і мережа може читати з пам'яті та записувати в неї залежно від поточного стану, що представляє повну за Тюрингом нейронну мережу.

2.6 Типи навчання нейронних мереж

1. Навчання із вчителем. Описує клас задач, який передбачає використання моделі для вивчення відображення між вхідними прикладами та цільовою змінною [62]. Моделі відповідають

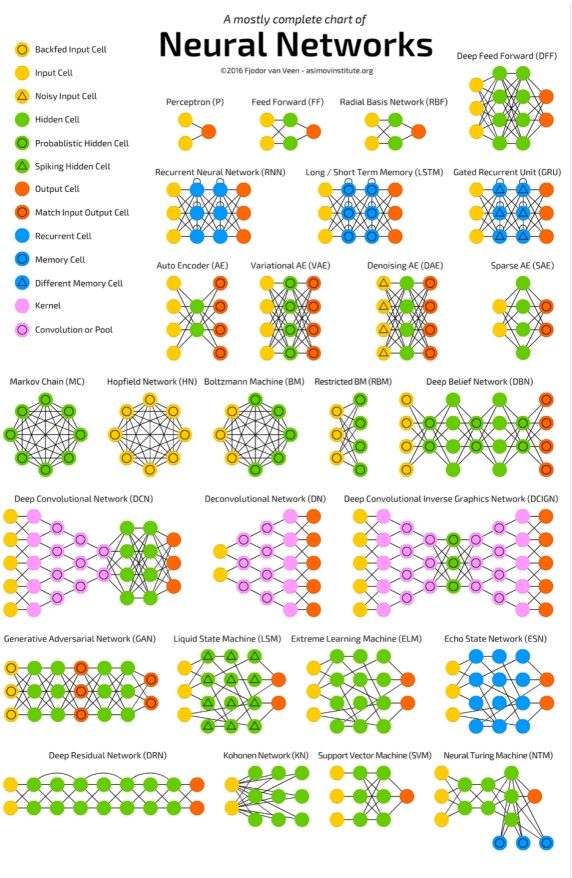

Рисунок 10 — Топології представлених нейронних мереж.

навчальним даним, які складаються з вхідних і вихідних даних, і використовуються для прогнозування на тестових наборах, де надаються лише вхідні дані, а результати моделі порівнюються з прихованими цільовими змінними та використовуються для оцінки навичок моделі. Існують два основних типи проблем навчання із вчителем: це класифікація, яка передбачає передбачення мітки класу, і регресія, яка передбачає прогнозування числового значення.

2. Описує клас проблем, який передбачає використання моделі для опису чи вилучення зв'язків у даних. Порівняно з навчанням з вчителем, даний вид навчання оперує лише вхідними даними без вихідних або цільових змінних. Існує багато типів такого навчання, але найбільш застосовним є в проблемах групування, що включає пошук групи в даних, і в оцінці щільності, що включає підсумовування розподілу даних. Можуть також використовуватися додаткові методи, такі як візуалізація, яка включає графіки або

графіки даних різними способами, і методи проекції, що передбачає зменшення розмірності даних.

3. Описує клас проблем, коли агент діє в середовищі і повинен навчитися діяти за допомогою зворотного зв'язку. Використання середовища означає, що немає фіксованого набору даних для навчання, а скоріше ціль або набір цілей, яких агент повинен досягти, дії, які він може виконувати, і зворотний зв'язок щодо ефективності досягнення мети.
4. Напівкероване навчання. Це навчання з вчителем, де навчальні дані містять велику кількість немаркованих прикладів. Метою напівкерованої моделі навчання є ефективне використання всіх доступних даних, а не лише маркованих даних, як у навчанні із вчителем. Для ефективного використання немаркованих даних може знадобитися використання таких. методів, таких як кластеризація (або похідних від них) та оцінка щільності. Після того, як групи або шаблони будуть виявлені, методи для навчання із вчителем можуть бути використані для позначення непозначених прикладів, які пізніше будуть використані для передбачення.
5. Навчання з самоконтролем. Відноситься до проблеми навчання без вчителя, яка оформляється як проблема навчання із вчителем з метою застосування його алгоритмів для її вирішення. Алгоритми навчання із вчителем використовуються для розв'язування альтернативної або передтекстної задачі, результатом якої є модель або уявлення, які можуть бути використані при розв'язанні вихідної (фактичної) задачі моделювання. Поширеним прикладом самоконтрольованого навчання є комп'ютерний зір, де доступний набір зображень без міток, який можна використовувати для навчання контрольованої моделі, наприклад, створення зображень у відтінках сірого та наявність моделі передбачити відображення кольору (розфарбовування) або видалення блоків із зображення. Загальним прикладом алгоритмів навчання з самоконтролем є автокодери. Хоча автокодери навчаються за допомогою методу навчання з вчителем, вони вирішують проблему навчання без вчителя, а саме, вони є типом методу проекції для зменшення розмірності вхідних даних. Іншим прикладом самоконтрольованого

навчання є генеративні опозиційні мережі, або Generative Adversarial Networks — GAN. Це генеративні моделі, які найчастіше використовуються для створення синтетичних фотографій з використанням лише колекції немаркованих прикладів із цільової області.

6. Індуктивне навчання передбачає використання доказів для визначення результату. Індуктивне міркування стосується використання конкретних випадків для визначення загальних результатів, напр. специфічне до загального. Більшість моделей машинного навчання навчаються за допомогою типу індуктивного висновку або індуктивного міркування, коли загальні правила (модель) вивчаються з конкретних історичних прикладів (даних). Підгонка моделі машинного навчання — це процес індукції. Модель є узагальненням конкретних прикладів у наборі навчальних даних. За допомогою навчальних даних створюється модель або гіпотеза щодо проблеми, і вважається, що вона зберігає нові невидимі дані пізніше, коли модель буде використана.
7. Дедукція або дедуктивний висновок відноситься до використання загальних правил для визначення конкретних результатів. Дедукція є зворотною до індукції. Якщо індукція йде від конкретного до загального, то дедукція йде від загального до конкретного. В контексті машинного навчання, як тільки ми використовуємо індукцію, щоб пристосувати модель до навчального набору даних, модель можна використовувати для прогнозування. Використання моделі є різновидом дедукції або дедуктивного висновку.
8. Трансдукційне або трансдуктивне навчання використовується в області статистичної теорії навчання для позначення прогнозування конкретних прикладів із конкретними прикладами з певної області. Це відрізняється від індукції, яка передбачає вивчення загальних правил на конкретних прикладах, наприклад, специфічнго до конкретного. На відміну від індукції, узагальнення не потрібно, замість цього використовуються безпосередньо конкретні приклади. Класичним прикладом трансдуктивного алгоритму є алгоритм k-найближчих сусідів, який не моделює навчальні дані, а натомість використовує їх безпосередньо щоразу, коли потрібне передбачення.

Отже, індукція — це процесс вивчення моделі на основі конкретних прикладів. Дедукція — використання отриманої моделі для прогназування. Трансдукція — використання конкретних прикладів для формування прогнозу.

9. Багатозадачне навчання. Цей це тип навчання з вчителем, який передбачає встановлення моделі на одному наборі даних, що вирішує декілька пов'язаних проблем.Це включає в себе розробку моделі, яку можна навчати на кількох пов'язаних завданнях таким чином, що ефективність моделі покращується шляхом навчання між завданнями в порівнянні з навчанням для будь-якої окремої задачі. Багатозадачне навчання може бути корисним підходом до вирішення проблем, коли є велика кількість вхідних даних, позначених для одного завдання, які можна спільно використовувати з іншим завданням із набагато менш позначеними даними [74].
10. Активне навчання. Є технікою, де модель може запитувати оператора-людину під час процесу навчання, щоб усунути неоднозначність під час процесу навчання. Даний тип навчання прагне досягти такої ж або кращої ефективності так званого «пасивного» навчання із вчителем, хоча більш ефективного щодо того, які дані збирає чи використовує модель. Активне навчання є корисним підходом, коли доступних даних небагато, а нові дані дорого збирати чи маркувати. Активний процес навчання дозволяє спрямувати вибірку предметної області таким чином, щоб мінімізувати кількість вибірок і максимізувати ефективність моделі.
11. Онлайн навчання. Передбачає використання наявних даних та оновлення моделі безпосередньо перед потрібним прогнозом або після останнього спостереження. Онлайн-навчання підходить для тих проблем, де спостереження надаються з часом і де очікується, що розподіл ймовірностей спостережень також зміниться з часом. Тому очікується, що модель буде змінюватися так само часто, щоб вловити та використати ці зміни [75]. Цей підхід також використовується в алгоритмах, де спостережень може бути більше, ніж можна розумно вмістити в пам'ять, тому навчання виконується поступово над спостереженнями, такими як потік даних. Одним із прикладів онлайн-навчання є так званий стохастичний або онлайн-градієнтний

спуск, який використовується для створення штучної нейронної мережі [74].

12. Трансферне навчання. Це тип навчання, коли модель спочатку навчають одному завданню, а потім частину або всю модель використовують як відправну точку для пов'язаного завдання. Це корисний підхід до проблем, де є завдання, пов'язане з основним завданням, що цікавить, і пов'язане завдання має велику кількість даних. Це відрізняється від багатозадачного навчання, оскільки завдання вивчаються послідовно під час трансферного навчання, тоді як багатозадачне навчання прагне до гарного виконання всіх розглянутих завдань однією моделлю одночасно. Прикладом є класифікація зображень, де передбачено модель, яку можна навчати на великому сеті загальних зображень, а ваги моделі можна використовувати як відправну точку під час навчання на меншому, більш конкретному наборі даних, наприклад, собаках і кішках. Функції, які вже засвоїла модель у більш широкому завданні, такі як вилучення ліній і візерунків, будуть корисними для нового пов'язаного завдання [74].
13. Ансамблеве навчання. В даному підході два або більше режимів підходять до одних і тих же даних, а прогнози кожної моделі поєднуються. Метою ансамблевого навчання є досягнення кращої продуктивності за допомогою ансамблю моделей порівняно з будь-якою окремою моделлю. Це включає в себе як рішення, як створити моделі, що використовуються в ансамблі, так і як найкраще поєднати прогнози учасників ансамблю. Таке навчання є корисним підходом для покращення навичок прогнозування в проблемній області та зменшення дисперсії алгоритмів стохастичного навчання, таких як штучні нейронні мережі [74].

Розділ 3. Використання нейронних мереж для побудови прогнозних оцінок динаміки процесу поширення інфекційних захворювань

3.1 Використання нейронних мереж для прогнозування COVID-19

Існує декілька епідеміологічних моделей, що використовуються у всьому світі для прогнозування кількості інфікованих осіб та рівня смертності від спалаху COVID-19. Вдосконалення моделей прогнозування має надзвичайно важливе значення для вживання належних дій. Через відсутність суттєвих даних та невизначеність епідеміологічні моделі вважаються недостатніми в сенсі надання більш високої точності для довгострокового прогнозування [37]. Оскільки різні країни вживають різних заходів для уповільнення спалаху, моделі на основі SIR мають бути адаптованими відповідно до місцевих умов [38].

Модель SEIR є одними з найпопулярніших інструментів для прогнозування помітних спалахів. В ній часто ретельно оцінюють інкубаційний період інфікованої людини, щоб отримати більш точні прогнози. У разі спалахів вітряної віспи та вірусу Зіка моделі SEIR показали підвищену точність [39] [40]. Особливістю моделі SEIR є інкубаційний період, що становить випадкову величину. Крім того, подібно до стандартних моделей на основі SIR, моделі SEIR працюють на ідеології рівноваги без хвороб [41] [42]. Однак слід зазначити, що моделі SEIR не моделюють якісно ситуації, де мережа контактів є нестаціонарною в часі [43]. Соціальне змішування як критичний фактор нестаціонарності визначає репродуктивне число R0, тобто кількість сприйнятливих до інфекції особ. Значення R0 для COVID-19 оцінювалося в 4, що значною мірою спровокувало пандемію. Карантинні заходи спрямовані на зниження значення R0 до 1. Тим не менш, повідомляється, що моделі SEIR важко підлаштувати у випадку COVID-19 через нестаціонарність соціальних контактів та їх стихійність. Тому для розробки більш точних моделей на основі SIR необхідна поглиблена інформація про соціальний рух та якість карантинних заходів. Ще одним

недоліком моделей на основі SIR є короткостроковість. Для довгострокового прогнозування точність більшості моделей, заснованих на SIR, знижується.

Через наявність невизначеності та високий ступінь складності в розвитку епідеміологічних моделей, машинне навчання все частіше розглядається як потенційна технологія. Воно вже показало значні результати у розробці кращих моделей на основі SIR з вищою продуктивністю, здатністю до узагальнення та надійністю [44] [45]. Машинне навчання вже визнали як обчислювальну техніку з великим потенціалом у прогнозуванні спалахів вірусів. Примітні алгоритми машинного навчання включають, наприклад, випадковий ліс для свинячої чуми [46], нейронні мережі для грипу H1N1, лихоманки денге та норовірусу Oyster [47], генетичне програмування для норовірусу Oyster [48], класифікацію і дерево регресії (CART) для денге [49], байєсівська мережа для лихоманки денге [50], LogitBoost для денге [51], мультирегресія та наївне прогнозування спалаху денге [52].

Машинне навчання часто використовувалося як додатковий обчислювальний інструмент для покращення моделей на основі SIR [47]— [51]. Тим не менш, існує прогалина у використанні машинного навчання у випадку COVID-19, хоча достатньо багато нових дослідницьких робіт вказують на величезний потенціал машинного навчання для боротьби з COVID-19. Машинне навчання показало перспективні результати в кількох аспектах для пом'якшення та запобіганню поширення вірусів, і було схвалено науковим співтовариством, наприклад, для ідентифікації випадків [53], класифікації нових патогенів [54], модифікації моделей на основі SIR [55], діагностики [56] [57], прогнозу виживання [58] та прогнозу попиту у відділенні інтенсивної терапії [59]. Серед застосувань машинного навчання наявне вдосконалення існуючих моделей прогнозування, виявлення вразливих груп, рання діагностика, просування доставки ліків, оцінка ймовірності наступної пандемії, розвиток інтегрованих систем просторово-часового прогнозування та інтелектуальний аналіз даних у соціальних мережах.

Машинне навчання можна використовувати для попередньої обробки даних. Підвищення якості даних може особливо покращити якість моделі на основі SIR. Наприклад, кількість випадків, про які повідомляє Worldometer, не є точно кількістю інфікованих (але ще не заразні) (E в моделі SEIR), або обчислення кількості інфікованих людей, що можуть інфікувати інших (I в

SEIR), не може бути легко визначено, оскільки багато людей, які можуть бути інфікованими, не з'являються на тестування, а кількість людей, які потрапили до лікарні та померли, не будуть справляти ефект на R, оскільки більшість випадків коронавірусної хвороби одужують, не потрапляючи до лікарні. Враховуючи цю проблему даних, задовільно підігнати моделі SEIR досить складно. Машинне навчання здатне оцінювати відсутню інформацію про кількість заражених E або інфікованих осіб I.

Наразі, існує декілька найбільш поширених підходів для даної задачі, проте дедалі більше розвиваються гібридні та специфічні методи, що спрямовані на покращення точності, а також боротьбу із невизначеністю за допомогою включення до аналізу нетривіальних даних.

3.2 Найуживаніші стратегії машинного навчання для прогнозування COVID-19

3.2.1 Узагальнений огляд

Виходячи з нелінійної та комплексної природи даного захворювання та складності оцінки особливостей передачі вірусу за допомогою традиційних моделей епідемії, для прогнозування його поширення застосовуються методи штучного інтелекту.

Виходячи з важливості підходів машинного та глибокого навчання для оцінки тенденції поширення COVID-19, було проведено огляд досліджень, які використовували ці стратегії для прогнозування кількості нових випадків COVID-19. Найбільш використовуваними моделями були: адаптивна нейро-нечітка система висновку (adaptive neuro-fuzzy inference system), довготривала короткочасна пам'ять (LSTM), рекурентна нейронна мережа (RNN) та багатошаровий персептрон (MLP) [63]. Як показники ефективності для порівняння точності моделей були обрані параметри кореневої середньоквадратичної помилки (RMSE), середньої абсолютної помилки (MAE), коефіцієнта детермінації R2 (R2) та середньої абсолютної процентної помилки (MAPE). Значення R2 коливаються від 0,64 до 1 для

штучної нейронної мережі (ANN) і двонаправленої довгострокової пам'яті (LSTM) відповідно. Адаптивна нейро-нечітка система висновку (ANFIS), авторегресійне інтегроване ковзне середнє (ARIMA) і багатошаровий персептрон (MLP) також мають значення R2, близькі до 1. ARIMA та LSTM мали найвищі значення MAPE [63].

У сукупності ці моделі здатні ідентифікувати параметри навчання, які впливають на відмінності в поширенні COVID-19 у різних регіонах або групах населення, поєднуючи численні методи втручання та реалізуючи сценарії «what-if», інтегруючи дані про хвороби, що мають аналогічні тенденції з COVID-19. Таким чином, застосування цих методів допомогло б у точному формуванні політики, щоб розробити найбільш відповідні заходи та уникнути неефективних обмежень. Ці моделі прогнозування можуть також оцінити вплив кліматичних факторів на рівень зараження або поширення COVID-19, що сприяє реалізації конкретних стратегій для кожного стану. Крім того, дані, отримані з цих моделей, можуть бути використані для визначення гарячих точок для COVID-19 для організації профілактичних заходів для окремих регіонів.

Включення даних про стан здоров'я постраждалих осіб, включаючи загальний стан здоров'я та відповідні фактори ризику, підвищить точність цих моделей. Більшість запропонованих моделей були ефективними для короткострокового прогнозування параметрів, пов'язаних з COVID-19. Їх ефективність у довгостроковій перспективі повинна бути підтверджена в подальших дослідженнях. Розглянемо більш детально запропоновані моделі.

Адаптивна нейро-нечітка система висновку (ANFIS)

ANFIS — це тип штучної нейронної мережі, заснованої на системі нечіткого висновку Такагі-Сугено. Архітектура ANFIS має п'ять шарів, а саме шар фазифікації, шар, який генерує сили спрацьовування для правил (рівень правил), шар, який нормалізує обчислені сили спрацьовування, шар, який отримує як вхідні дані нормовані значення та параметри наслідків, і шар, який повертає кінцевий вихід [64]. Аль-Канес та інші [65] розробили оновлений тип моделі ANFIS для оцінки кількості інфікованих у чотирьох країнах, а саме в Італії, Ірані, Кореї та США. Їхня модель була заснована на новому оптимізаторі, натхненному природою, а саме на алгоритмі морських хижаків (Marine Predators Algorithm - MPA). Цей алгоритм оптимізував змінні ANFIS, підвищивши ефективність його прогнозування. Вони показали

перевагу методу MPA-ANFIS над раніше запропонованими моделями прогнозування з точки зору кращих значень RMSE, MAE, MAPE та R2 [65].

В іншому дослідженні ANFIS було посилено за допомогою вдосконаленого алгоритму запилення квітів (Flower Pollination Algorithm - FPA) та алгоритму рою салпів (Salp Swarm Algorithm - SSA). Потім запропонована модель FPASSA-ANFIS була оцінена за допомогою офіційних даних, отриманих із сайту ВООЗ. Більше того, точність запропонованої моделі була оцінена за допомогою двох різних наборів даних щотижневих випадків грипу [65]. Alsayed та ін. [66] спрогнозували пік епідемії в Малайзії, використовуючи модель SEIR. Вони також використовували модель ANFIS для короткочасного прогнозування кількості інфікованих осіб. Було продемонстровано вплив заходів на відтермінування піку епідемії. У цьому дослідженні зазначено значення RMSE, R2 і MAPE як 46,87, 0,9973 і 2,79 відповідно [66]. Таким чином, це дослідження повідомило про найкращі вимірювання продуктивності за допомогою цього методу.

Бехнуд та ін. [67] використовували інтеграцію алгоритму оптимізації вірусів (VOA) та ANFIS для оцінки впливу численних кліматичних параметрів і щільності населення на поширення COVID-19. Вони продемонстрували чудовий вплив щільності населення на продуктивність розроблених ними моделей, наголошуючи на важливості соціального дистанціювання для зниження рівня зараження та поширення COVID-19. Повідомлялося, що значення RMSE, MAE та R2 становлять 22,47, 7,33 та 0,83 відповідно [67].

Авторегресійне інтегроване ковзне середнє (ARIMA)

Як тип методу одновимірного регресійного аналізу, ARIMA прогнозує майбутні значення відповідно до відмінностей між значеннями замість фактичних цифр. Як узагальнення моделі авторегресивної ковзної середньої (ARMA), ARIMA пристосована до даних часових рядів для кращого розуміння даних або прогнозування майбутніх точок у цих рядах. Альзахрані та ін. [68] використовували модель ARIMA, щоб передбачити приблизний щоденний приріст інфікованих у Саудівській Аравії. Була досліджена перевага ARIMA над моделлю авторегресії, ковзним середнім та інтеграцією ARMA та ARIMA. Використовуючи ARIMA, вони повідомили про значення RMSE, MAE, R2 і MAPE як 21,17, 14,93, 0,99 і 2,16 відповідно [68].

Мультишаровий персептрон (Multilayer Perceptron — MLP)

Кар та ін. [69] використовували вільно доступний набір даних часових рядів для розробки своєї моделі. Вони використовували цей набір даних для навчання моделі MLP. Найкращі розроблені моделі мали 4 прихованих шари з 4 нейронами в кожному. Ця модель мала відповідні заходи для прогнозування померлих та підтверджених випадків, але мала низьку надійність для пацієнтів, які одужали [69]. Пінтер та ін. [70] використовували гібридні стратегії машинного навчання ANFIS та MLP-імперіалістичного конкурентного алгоритму (MLP-ICA) для прогнозування часових рядів випадків COVID-19 і смертності. Короткочасне спостереження підтвердило точність запропонованої моделі. Автори припускають, що модель зберігає свою точність за умови відсутності істотних перерв [70].

Довга короткочасна пам'ять (Long Short-Term Memory — LSTM) LSTM з модулем обробки природної мови (NLP) використовувався для оцінки частоти зараження та підвищення точності прогнозування моделі [71]. LSTM може ефективно покращити градієнтний вибух і зникнення градієнта в процесі навчання, представивши блок каруселі постійної помилки [71]. LSTM перевершує традиційну рекуррентну нейронну мережу (RNN) з точки зору її хорошого сприйняття довгострокової залежності послідовностей, тому є відповідним для категоризації, обробки та прогнозування даних довгої послідовності [43].

Аора та ін. [72] використовували пов'язані з RNN варіанти LSTM на індійському наборі даних пацієнтів з COVID-19, щоб спрогнозувати кількість позитивних випадків. На основі найнижчого рівня помилок була обрана модель LSTM для прогнозування щоденних і щотижневих нових випадків COVID-19 з приблизними показниками помилок 3 та 8 відсотків відповідно. Згодом вони класифікували штати Індії на різні зони на основі масштабів позитивних випадків та щоденної ескалації для розпізнавання гарячих точок COVID-19 [72]. Також була застосована двонаправлена мережа LSTM, щоб отримати надійне узагальнення RNN. Цей метод використовувався для прогнозування нових випадків COVID-19 в Італії, Іспанії, Франції, Німеччині, США та Швеції [73].

3.3 Розробка прототипу інтелектуального аналізу процесів поширення інфекційних захворювань з урахуванням життєвого циклу населення на мові Python

3.3.1 Технології для створення нейронних мереж на базі Python

Python довгий час залишається найкращим вибором для розробників машинного навчання та штучного інтелекту. Дана мова програмування пропонує розробникам одні з найкращих можливостей і функцій, які не тільки підвищують їх продуктивність. Розроблені великі бібліотеки допомагають полегшити робоче навантаження. Ознаки, які забезпечують Python місце серед провідних мов програмування для машинного навчання, глибинного навчання та штучного інтелекту:

1. Безкоштовний та відкритий вихідний код робить його дружнім до спільноти та гарантує покращення в довгостроковій перспективі;
2. Вичерпні бібліотеки забезпечують рішення для кожної існуючої проблеми;
3. Плавне впровадження та інтеграція роблять його доступним для людей з різним рівнем навичок;
4. Підвищення продуктивності за рахунок скорочення часу на кодування та адаптування під задачу;
5. Може використовуватися для м'яких обчислень (Soft Computing), а також для обробки природної мови (NLP);
6. Безперебійно працює з модулями коду C і C++. Багато бібліотек написані на даних мовах, що покращує швидкість виконання коду.

Розглянемо найпопулярніші бібліотеки на Python.

TensorFlow — це швидка, гнучка та масштабована бібліотека машинного навчання з відкритим кодом для досліджень і виробництва. Одна з найкращих бібліотек, доступних для роботи з машинним навчанням на Python. Запропонований Google, TensorFlow полегшує створення моделі ML як для новачків, так і для професіоналів. Дана бібліотека дозволяє створювати та навчати моделі не тільки на комп'ютерах, а й на серверах та мобільних телефонах. Вона підтримує виконання обчислень на різних

типах пристроїв, включаючи CPU та GPU. Області використання :робота з глибокими нейронними мережами, обробка природної мови, рівняння в частинних похідних, можливості абстракції, розпізнавання зображень, тексту та мовлення.Основним завданням є створення моделей глибинного навчання.

Keras — одна з найпопулярніших бібліотек нейронних мереж з відкритим кодом для Python. Спочатку розроблений інженером Google для ONEIROS, Open-Ended Neuro Electronic Intelligent Robot Operating System, Keras незабаром був підтриманий в основній бібліотеці TensorFlow, що робить його доступним поверх TensorFlow. Keras містить кілька будівельних блоків та інструментів, необхідних для створення нейронної мережі, наприклад: нейронні шари, функції активації та втрат, пакетна нормалізація (batch normalization), випадання (dropout), агрегування (pooling).

Keras покращує зручність використання TensorFlow за допомогою цих додаткових функцій для програмування задач машинного та глибинного навчання. Поряд зі стандартними нейронними мережами також існує підтримка згорткової та рекурентної нейронної мережі.

PyTorch, розроблена Facebook, також існує для C++ із своїм інтерфейсом. Вважається одною з головних претенденток на те, щоб стати найкращим фреймворком машинного навчання та глибокого навчання, конкуруючи з TensorFlow. Деякі важливі функції, які відрізняють PyTorch від TensorFlow: тензорні обчислення з можливістю прискореної обробки за допомогою графічних процесорів (оновлення TensorFlow 2.0 тепер також це дозволяє); легко вивчати, використовувати та інтегрувати з рештою екосистеми Python; підтримка нейронних мереж, побудованих на основі системи автоматичного диференціювання на основі стрічки. Різні модулі, які є в PyTorch, допомагають створювати й навчати нейронні мережі: тензори (torch.Tensor); оптимізатори (torch.optim); нейронні мережі (nn); автоград. Основним завданням є розробка та навчання моделей глибинного навчання.

Scikit-learn — ще одна активно використовувана бібліотека машинного навчання для Python. Вона включає просту інтеграцію з різними бібліотеками програмування для задач машинного навчання, такими як NumPy і Pandas. Scikit-learn містить підтримку різних алгоритмів, таких як класифікація, регресія, кластеризація, зменшення розмірності, вибір моделі та попередня обробка.Побудований навколо ідеї бути простим у використанні, але залишатися гнучким, Scikit-learn зосереджений на моделюванні даних, а

не на інших завданнях, таких як завантаження, обробка, маніпулювання та візуалізація даних. Тому основним завданням є моделювання.

Pandas — це бібліотека аналізу даних Python, яка використовується в основному для маніпуляції та аналізу даних. Вона часто вступає в дію до того, як набір даних буде підготовлений до навчання. Pandas полегшує роботу з часовими рядами та структурованими багатовимірними даними. Можливості обробки даних у Pandas: зміна розмірності та поворот набору даних, об'єднання наборів даних різними шляхами, обробка відсутніх даних і вирівнювання даних, різні параметри індексації, параметри фільтрації даних. Pandas використовують DataFrames, який є лише технічним терміном для двовимірного представлення даних, пропонуючи програмістам об'єкти DataFrame. Основним завданням є маніпулювання та аналіз даних. Схожою в розумінні обробки даних також є бібліотека NumPy.

NLTK означає Natural Language Toolkit і є бібліотекою Python для роботи з обробкою природної мови. Вважається однією з найпопулярніших бібліотек для роботи з даними людської мови. NLTK пропонує прості інтерфейси разом із широким набором лексичних ресурсів, таких як FrameNet, WordNet, Word2Vec та деякі інші. Основні можливості NLTK: пошук ключових слів у документах, токенізація та класифікація текстів, розпізнавання голосу та почерку, лематизація та стемінг слів.

Також існують бібліотеки Spark MLlib, Theano та MXNet.

3.3.2 Архітектура моделі LSTM для прогнозу розповсюдження COVID-19

Для реалізації гібридного методу моделювання розповсюдження вірусу на основі SIRM моделі із гарантованими оцінками параметрів та нейронної мережі у даній роботі була обрана модель мультизмінної LSTM, тобто із декількома ознаками. Ознаками виступають декілька паралельних часових рядів.

На рис. ?? кожен рядок несе цілий вектор від виходу одного вузла до входу інших. Рожеві кола представляють точкові операції, наприклад, додавання векторів, а жовті прямокутники — це вивчені шари нейронної

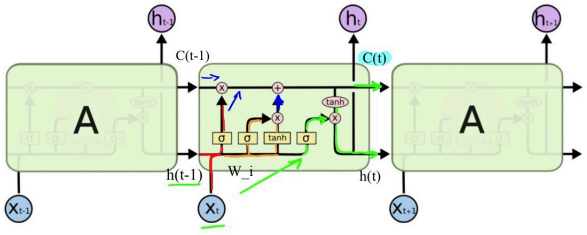

Рисунок 11 — Архітектура використаної нейронної мережі

мережі. Злиття рядків позначає конкатенацію, а розгалуження рядків означає, що його вміст копіюється, а копії спрямовуються в різні місця.

Ворота забуття (або вентилі) вирішують, яку інформацію слід викинути або зберегти. Інформація з попереднього прихованого стану (h(t-1)) та інформація з поточного входу (x(t)) передається через сигмовидну функцію (виділено червоним) та на виході отримується значення від 0 до 1. Значення ближче до нуля означає забуття, а ближче до 1 — зберігання. Щоб оновити стан комірки, маємо вхідні ворота. Спочатку передається попередній прихований стан (h(t-1)) і поточний вхід (x(t)) у сигмовидну функцію (виділено оранжевим). Це визначає, які значення будуть оновлені шляхом перетворення на значення від 0 до 1, де близькість до 1 позначає важливість інформації. Прихований стан (h(t-1)) і поточний вхід (x(t)) передаються у функцію tanh, щоб стискувати значення в проміжок від 1 і 1, що додатково допомогає регулювати мережу. Вихід tanh множиться вихід сигмовидної форми, причому сигмовидний вихід вирішує, яку інформацію важливо зберігати з виходу tanh.

Після виконаних дій маємо достатньо інформації для оновлення стану клітини. По-перше, стан клітини точково множиться на вектор забуття (виділено синім). Це може скинути значення в стані комірки, якщо її помножити на значення, близькі до 0. Вихідні дані з вхідних воріт поточково додаємо до останнього отриманого результату, що оновить стан комірки (C(t)). до нових значень, які нейронна мережа вважає релевантними. Це дає нам новий стан комірки.

Далі вихідні ворота вирішують, яким має бути наступний прихований стан h(t) (виділено зеленим). Відомо, що прихований стан містить інформацію про попередні введення. Він також використовується для прогнозу. Попередній прихований стан і поточний вхід передаються у

сигмовидну функцію, а змінений стан комірки — у функцію tanh. Вихідний сигнал сигмовидної функції при виході tanh визначає, яку інформацію має нести прихований стан. Таким чином отримаємо прихований стан. Новий стан комірки та нове прихований стан потім переносяться на наступний часовий крок.

Отже, ворота забуття вирішують, що важливо утримати від попередніх кроків. Вхідні ворота відповідальні за інформацію, що необхідно додати з поточного кроку. Вихідні ворота визначають, яким має бути наступний прихований стан.

3.3.3 Застосування та порівняння LSTM та SIRM-LSTM моделей для прогнозу захворювання

При реалізації даної структури найважливішим кроком є саме обробка даних. Оскільки використовується бібліотека Keras, то розробнику пропонується лише правильно підготувати вхідні дані (особливо це стосується розмірностей). У даній роботі головним чином потрібно було притримуватись розмірності [кількість прикладів для навчання, крок , кількість ознак (часових рядів)]. Кроком було обрано два дні, тобто на основі двох днів будується прогноз на третій день. Побудова моделі займає лише декілька рядків коду, похідні автоматично рахуються при прогонці алгоритму для кожної епохи. Розробник лише вказує кількість нейронів на кожному шарі, а також яким цей шар має бути, який алгоритм оптимізації використовується та яка похибка буде обрахована.

З огляду на практичність та гарну результативність було застосовано оптимізаційний алгоритм ADAM (Adaptive Moment Estimation) — оцінка адаптивного моменту. Він є ефективним з огляду обчислень, має невеликі вимоги до пам'яті, інваріантний до діагонального масштабування градієнтів, добре підходить для задач, які є великими з точки зору даних та/або параметрів, а також підходить для проблем із дуже шумними/або рідкими градієнтами.

Середня квадратична помилка, що була задіяна у даній роботі, за замовчуванням використовується для задач регресії. Ця функція втрат

є першою, яку потрібно оцінити, і змінити лише за наявності поважної причини. Математично, вона є бажаною функцією втрат на основі висновку, що базується на методі максимальної вирогідності, якщо розподіл цільової змінної є гаусівським. Середня квадратична похибка розраховується як середнє з квадратів різниць між прогнозованим і фактичним значеннями. Результат завжди позитивний, незалежно від знака прогнозованого та фактичного значень, а ідеальне значення дорівнює 0. Зведення в квадрат означає, що більші відстані між реальним та прогнозованим призводять до більшої кількості помилок, ніж менші, тобто модель карається за більші помилки. Виконується нормалізація даних стискуванням до проміжку від -1 до 1. Для графічного представлення використовуються оборотні перетворення. Маємо три шари побудованої моделі : LSTM (100 нейронів), LSTM-1 (50 нейронів), Dense(3)(або 6 нейронів в залежності від кількості ознак). Останній становить собою звичайний повнозв'язний шар нейронної мережі.

Було проведено декілька експериментів із різними кількостями епох : 50, 100, 300, 500, 1000.

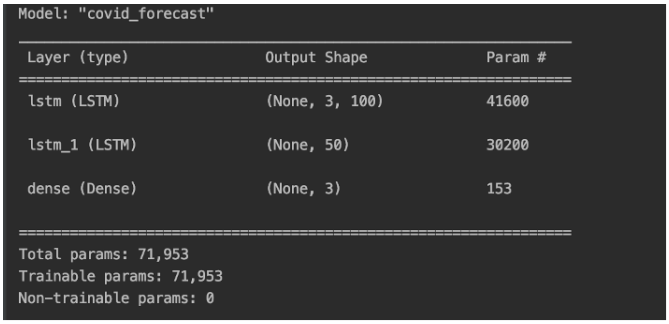

Рисунок 12 — Вигляд нейронної мережі в Keras

На практиці будемо порівнювати дві нейронні мережі: одна звичайна LSTM із трьома часовими рядами (приріст випадків одужання, інфікування та смерті) та друга гібридна. Гібридністю є використання не тільки вищевказаних приростів, а й результатів обчислення гарантованих оцінок параметрів запропонованої моделі SIRM із урахуванням життєвого циклу населення на прикладі країни Японії. Тобто буде додано ще три часових ряди із значеннями параметрів. Часовим проміжком було обрано спочатку 485 днів — з 1 січня 2021 року до 1 травня 2022 року. Але використання алгоритму знаходження оцінок несе у собі рекурентність, а отже накопичення помилки.

Тому в ході дослідження було виявлено, що не є раціональним використання гібридної моделі на строк більший, ніж 90 днів.

Кінцевий обраний період становив проміжок від 1 січня 2021 до 1 квітня 2022 року. Досліджена література також показує недоцільність використання моделей розповсюдження вірусних хвороб типу SIR для довготривалих прогнозів [37]. Множину даних для тестування та для тренування було поділено у відношення 1:9.

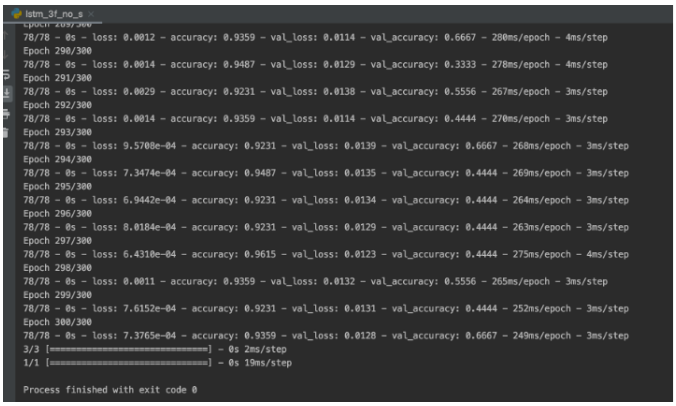

Рисунок 13 — Результат навчання негібридної мережі із трьома часовими рядами для 90 днів

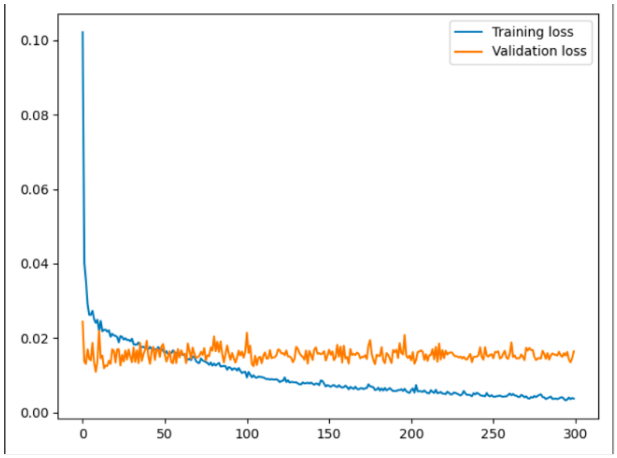

Рисунок 14 — Функція втрат LSTM 90 днів

Беручи до уваги, що обраний часовий проміжок дає доволі невелику вибірку, отримаємо непогані результати із досяжною точністю в 93 відсотки. На тестових даних видно, що настільки мала вибірка не дає відловити гарно піки хвороби, оскільки навіть з достатньою вибіркою задача не є простою.

Наведемо приклад відпрацювання нейронної мережі на 485 днях.

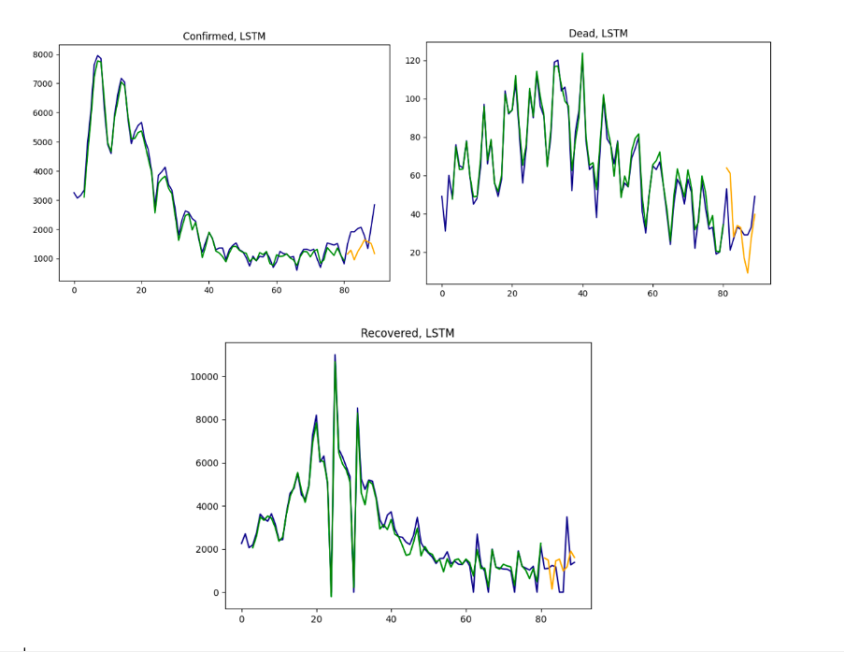

Рисунок 15 — Результати навчання та прогнозу для LSTM мережі 90 днів

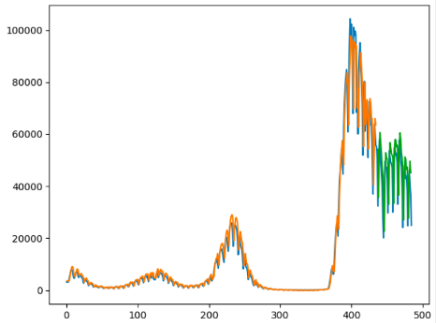

Рисунок 16 — Результати навчання та прогнозу для LSTM мережі 485 днів

На рис. 16.бачимо, що точність зростає (становить 98 відсотків), проте піки все ще є недосяжними.

Порівняємо тепер із результатами роботи гібридної SIRM-LSTM моделі рис. 18., розширивши кількість ознак до шести, проте залишаючи невеликий проміжок часу в 90 днів.

Оцінки параметрів були обраховані в роботі попереднього року за допомогою алгоритму знаходження гарантованих оцінок із [34]. Тобто після отримання даних в результаті опрацювання рекурентними формулами SIRM моделі та вищезгаданого алгоритму, отримуємо вектор оцінок α, β та μ_2. Саме вони стають додатковими часовими рядами розробленої гібридної моделі.

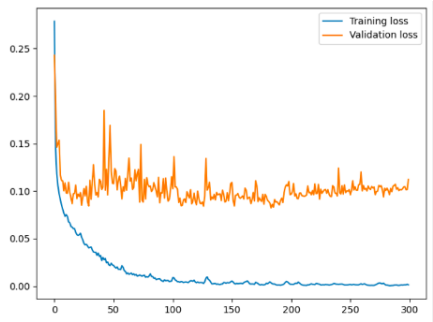

Рисунок 17 — Функція втрат SIRM-LSTM 90 днів

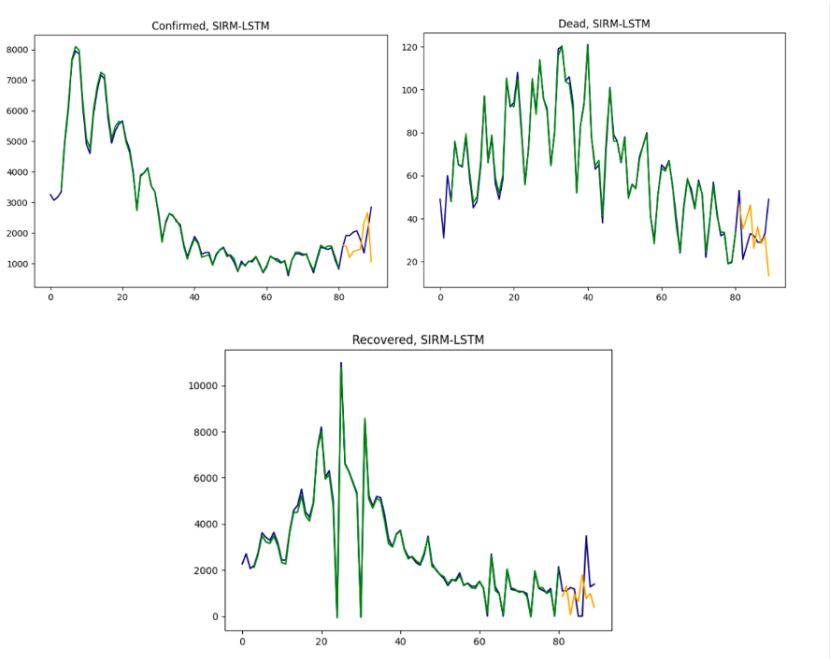

Рисунок 18 — Результати навчання та прогнозу для SIRM-LSTM мережі 90 днів

Бачимо, що навіть вдвічі збільшивши кількість часових рядів, але не розширивши часовий проміжок, було отримано доволі точні результати, що зберігають основну тенденцію. Зрозуміло, що знову передбачення піків є нетривіальною задачею, але на даних для тренування графічно алгоритм показує кращі результати. Було досягнуто точність в 83 на вибірці для тренування. Покращення результатів точності для тестових даних потребує набагато більшої кількості даних. Експерименти були порівняні на прикладі 300 епох. Збільшення кількості епох покращувало результати, але, наприклад, при 1000 епохах шанс перенавчання дуже різко зростає, тому це не є доречним варіантом.

ВИСНОВКИ

Дана робота є комплексним дослідженням, яке пов'язане з практичним розв'язанням задач оцінювання в моделях поширення інфекційних захворювань. Наведені алгоритми оцінювання параметрів дають можливість побудови математичних моделей поширення інфекційних захворювань на основі спостережень, а також знаходження прогнозних оцінок прогнозів динаміки цих процесів.

У роботі наведені результати аналізу основних підходів до математичного моделювання процесів поширення інфекційних захворювань, зокрема імітаційні моделі, методів оцінювання в моделях популяційної динаміки.

Було проаналізовано основі принципи моделювання та прогнозування розповсюдження вірусної хвороби за допомогою SIR моделей та засобів машинного навчання, а також труднощі, які виникають при постановці такої задачі. Недоліком SIR моделей є короткочасність прогнозів, а також зростання складності системи при додаванні додаткової інформації для покращення точності. Наведений у роботі алгоритм із знаходження аналітичних розв'язків системи нелінійних диференціальних рівнянь є допоміжним засобом в даному моделюванні.

Аналіз методів глибинного навчання дозволив виявити зручну та ефективну модель для прогнозування коронавірусу навіть на невеликій виборці, оскільки дані засоби мають кращу здатність працювати із зашумленими та неповними даними. Запропонована у роботі гібридна модель об'єднує в собі школу системного аналізу у ролі гарантованих оцінок параметрів SIRM моделей та останні технологічні досягнення у штучному інтелекті у ролі оптимізованої для прогнозування часових рядів нейронної мережі LSTM. Було порівняно точності навчання для нейронної мережі із двома типами вхідних даних, а саме із трьома паралельними часовими рядами із приростом кількості людей, що захворіли, померли та одужали, а також із шістьма часовими рядами, що включають ще додаткові вектори отриманих гарантованих оцінок. Розширення кількості ознак як вхідних даних вдвічі зменшило точність лише на 10 відсотків. Збільшення кількості

епох для нейронної мережі покращує результат на декілька відсотків, але таким чином постає проблема перенавчання.

Запропонована гібридна модель є новою та показує задовільні результати на малій виборці. Графічні представлення прогнозів виглядають впевнено та слідують головній тенденції розповсюдження захворювання. Головною перепоною для покращення результатів є забезпечення більшої кількості вхідних даних шляхом використання додаткових числених методів для рекурентного підрахунку гарантованих оцінок.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Kermack W.O., McKendrick A.G. A Contribution to the Mathematical Theory Of Epidemics. *Proceedings of the Royal Society of London.* 1927. Series A, vol. 115, no. 772.

2. Хусаїнов Д.Я. Аналіз моделі розповсюдження захворювання. *Problems of Dicision Making under Uncertainties (PDMU-2013)*: тези XXII міжнародної конференції, Ялта-Форос, Україна, 23-27 вересня 2013. Київ, 2013. С. 138.

3. Хусаїнов Д.Я., Шатирко А.В. Аналіз однієї епідеміологічної моделі з післядією. *Вісник Київського національного університету імені Тараса Шевченка. Кібернетика.* 2017. В. 1. С. 36—44.

4. Miller J.C. Mathematical models of SIR disease spread with combined non-sexual and sexual transmission routes. *Infectious Disease Modelling.* 2017. №2. P. 35—55.

5. Adda P., Bichara D. Global stability for SIR and SIRS models with differential mortality. *arXiv preprint arXiv*: 1112.2662. 2011. Режим доступу *https* : *//arxiv.org/pdf/*1112.2662.*pdf*

6. Allen L. J. Some discrete-time SI, SIR, and SIS epidemic models. *Mathematical biosciences.* 1994. Vol. 124(1). P. 83—105.

7. Hu H., Yuan X., Huang L., Huang C. Global dynamics of an SIRS model with demographics and transfer from infectious to susceptible on heterogeneous networks. *Math. Biosci. Eng.* 2019. Vol. 16(5). P. 5729—5749.

8. Kröger M., Schlickeiser R. Analytical solution of the SIR-model for the temporal evolution of epidemics. Part A: time-independent reproduction factor. *Journal of Physics A: Mathematical and Theoretical.* 2020. Vol. 53(50). 505601.

9. Harko T., Lobo F.S.N., Mak M.K. Exact analytical solutions of the Susceptible-Infected-Recovered (SIR) epidemic model and of the SIR model

with equal death and birth rates. *Applied Mathematics and Computation.* 2014. №236. P. 184—194.

10. Ndiaye B. M., Tendeng L., Seck D. Comparative prediction of confirmed cases with COVID-19 pandemic by machine learning, deterministic and stochastic SIR models. *arXiv preprint arXiv:2004.13489.* 2020. Режим доступу: *https* : *//arxiv.org/pdf/*2004.13489*.pdf*

11. Martsenyuk V. P., Andreychyn M. A., Sverstiuk A.S., Kopcha V.S., Chaychuk O.T., Panychev V. O. Ідентифікація параметрів у SIR-моделях за результатами пандемії COVID-19 в Тернопільській області. *Інфекційні хвороби.* 2020. № 2. С. 15—21.

12. Biswas K., Khaleque A., Sen P. Covid-19 spread: Reproduction of data and prediction using a SIR model on Euclidean network. *arXiv preprint arXiv:2003.07063.* 2020. Режим доступу: *https* : *//arxiv.org/pdf/*2003.07063*.pdf*

13. Piazzola C., Tamellini L., Tempone R. A note on tools for prediction under uncertainty and identifiability of SIR-like dynamical systems for epidemiology. *Mathematical Biosciences.* 2020. 108514. Режим доступу: *https* : *//arxiv.org/pdf/*2008.01400*.pdf*

14. Rajesh A., Pai H., Roy V., Samanta S., Ghosh S. CoVID-19 prediction for India from the existing data and SIR (D) model study. *medRxiv.* 2020. Режим доступу: *https* : *//www.medrxiv.org/content/medrxiv/early/*2020/05/08/2020.05.05.20085902*.full.pdf*

15. Fanelli D., Piazza F. Analysis and forecast of COVID-19 spreading in China, Italy and France. *Chaos, Solitons and Fractals.* 2020. Vol. 134. 109761. Режим доступу: *https* : *//www.ncbi.nlm.nih.gov/pmc/articles/PMC*7156225/

16. Maleki M., Mahmoudi M. R., Wraith D., Pho, K. H. Time series modelling to forecast the confirmed and recovered cases of COVID-19. *Travel medicine and infectious disease.* 2020. Vol. 37. 101742. Режим доступу: *https* : *//www.ncbi.nlm.nih.gov/pmc/articles/PMC*7219401/

17. Cantó B., Coll C., Sánchez E. Estimation of parameters in a structured SIR model. *Advances in Difference Equations.* 2017. Vol. 1. P. 1—13.

18. Marinov T. T., Marinova R. S. COVID–19 analysis using inverse problem for coefficient identification in SIR epidemic models. *Chaos, Solitons and Fractals: X.* 2020. Vol. 100041.

19. Lounis M., Bagal D. K. Estimation of SIR model's parameters of COVID-19 in Algeria. *Bulletin of the National Research Centre.* 2020. Vol. 44(1). P. 1—6.

20. Roques L., Klein E. K., Papaix J., Sar A., Soubeyrand S. Using early data to estimate the actual infection fatality ratio from COVID-19 in France. *Biology.* 2020. Vol. 9(5). 97.

21. Milan B. Estimation of the final size of the coronavirus epidemic by the SIR model. *Online paper, ResearchGate.* 2020. P. 1-12.

22. Postnikov E. B. Estimation of COVID-19 dynamics "on a back-of-envelope": Does the simplest SIR model provide quantitative parameters and predictions? *Chaos, Solitons and Fractals.* 2020. Vol. 135. 109841.

23. Shapiro M. B., Karim F., Muscioni G., Augustine A. S. Are we there yet? An adaptive SIR model for continuous estimation of COVID-19 infection rate and reproduction number in the United States. *medRxiv preprint.* 2020. Режим доступу: *https* : *//www.medrxiv.org/content/medrxiv/early/* 2020/09/15/2020.09.13.20193896.*full.pdf*

24. Бакан Г.М. Эллипсоидальные алгоритмы гарантированого оценивания и рекуррентный метод наименьших квадратов в задачах фыльтрации состояний динамических систем. *Проблемы управления и информатики.* 1997. №3. С. 34—48.

25. Кунцевич В.М. Управление в условиях неопределенности: гарантированные результаты в задачах управления и идентификации. Киев: Наукова думка, 2006. 264 с.

26. Gubarev V.F. Method of iterative identification of multidimensional systems by uncertain data. Part I. Theoretical aspects. *Journal of Automation and Information Sciences.* 2006. Vol. 38, №9. P. 12—28.

27. Gubarev V.F., Shevchenko V.N., Gummel A.V. State estimation for systems subjected to bounded uncertainty using moving horizon approach. *IFAC Proceedings Volumes*. 2009. Vol. 42, №10. P. 910–915.

28. Губарев В.Ф., Дарьин А.Н., Лысюченко И.А. Нелинейный оцениватель состояния по заданным на скользящем интервале и возможность его применения в задаче ориентации космического аппарата. *Проблемы управления и информатики.* 2011. №1. С. 118—132.

29. Nakonechnyi O.G. Beast-mean estimates in model of information confrontation. *PROBLEMS OF DECISION MAKING UNDER UNCERTAINTIES*: abstracts of XXIV International Conference, Cesky Rudolec, Czech Republic, September 1-5 2014. Kyiv, 2014. P.114—115.

30. Martynyuk A.A., Khusainov D. Ya., Chernienko V.A. Integral estimates of solutions to nonlinear systems and their applications. *Nonlinear Dynamics and Systems Theory.* 2016. №1. P. 1—11.

31. Наконечний О.Г. Оцінювання параметрів в умовах невизначеності. *Наукові записки Київського національного університету, факультет кібернетики.* 2004. Т.VII. С. 102—111.

32. Ramsay J. O. Hooker G., Campbell D., Cao J. Parameter estimation for differential equations: a generalized smoothing approach. *Royal Statistic Society.* 2007. Vol. 69, №5. P. 741—796.

33. Aster R.C., Borchers B., Thurber C.H. Parameter estimation and inverse problems: eBook. Academic Press, 2013. P. 376.

34. Nakonechnyi O.G., Zinko P.M., Shevchuk I.M. Estimate of parameters of difference equations under uncertainty. *Ukrainian conference on applied mathematics*: abstracts of conference, Lviv, Ukraine , September 28-30 2017. Lviv, 2017. P.86.

35. Наконечный А.Г. Зинько П.Н., Шевчук Ю.М. Гарантированные оценки нестационарных параметров разностных уравнений в условиях неопределенности. *Проблемы управления и информатики.* 2018. № 6. С. 41—54.

36. Novel Coronavirus (COVID-19) Cases Data. Режим доступу: $https://data.humdata.org/dataset/novel-coronavirus-2019-ncov-cases$

37. Pinter, G., Felde, I., Mosavi, A., Ghamisi, P., Gloaguen, R. COVID-19 Pandemic Prediction for Hungary; A Hybrid Machine Learning Approach. Mathematics 2020

38. Maier, B.F., Brockmann, D. Effective containment explains sub-exponential growth in confirmed cases of recent COVID-19 outbreak in Mainland China. medRxiv 2020.

39. Pan, J.R., Huang, Z.Q., Chen, K. Evaluation of the effect of varicella outbreak control measures through a discrete time delay SEIR model. Chin. J. Prev. Med. 2012, 46, P. 343–347.

40. 4 Zha, W.T., Pang, F.R., Zhou, N., Wu, B., Liu, Y., Du, Y.B., Hong, X.Q., Lv, Y. Research about the optimal strategies for prevention and control of varicella outbreak in a school in a central city of China: Based on an SEIR dynamic model. Epidemiol. Infect. 2020.

41. Dantas, E., Tosin, M., Cunha, A., Jr. Calibration of a SEIR–SEI epidemic model to describe the Zika virus outbreak in Brazil. Appl. Math. Comput. 2018, 338, P. 249–259.

42. Leonenko, V.N., Ivanov, S.V. Fitting the SEIR model of seasonal influenza outbreak to the incidence data for Russian cities. Russ. J. Numer. Anal. Math. Model. 2016, 31, P. 267–279.

43. Imran, M., Usman, M., Dur-e-Ahmad, M., Khan, A. Transmission Dynamics of Zika Fever: A SEIR Based Model. Differ. Equ. Dyn. Syst. 2020.

44. Burke, R.M., Shah, M.P., Wikswo, M.E., Barclay, L., Kambhampati, A., Marsh, Z., Cannon, J.L., Parashar, U.D., Vinjé, J., Hall, A.J. The Norovirus Epidemiologic Triad: Predictors of Severe Outcomes in US Norovirus Outbreaks, 2009–2016. J. Infect. Dis. 2019, 219, P. 1364–1372.

45. Kleiven, E.F., Henden, J.A., Ims, R.A., Yoccoz, N.G. Seasonal difference in temporal transferability of an ecological model: Near-term predictions of lemming outbreak abundances. Sci. Rep. 2018, P. 8.

46. Liang, R., Lu, Y., Qu, X., Su, Q., Li, C., Xia, S., Liu, Y., Zhang, Q., Cao, X., Chen, Q., et al. Prediction for global African swine fever outbreaks based on a combination of random forest algorithms and meteorological data. Transbound. Emer. Dis. 2020, 67, P.935–946.

47. Anno, S., Hara, T., Kai, H., Lee, M.A., Chang, Y., Oyoshi, K., Mizukami, Y., Tadono, T. Spatiotemporal dengue fever hotspots associated with climatic factors in taiwan including outbreak predictions based on machine-learning. Geospat. Health 2019, 14, P. 183–194.

48. Chenar, S.S., Deng, Z. Development of genetic programming-based model for predicting oyster norovirus outbreak risks. Water Res. 2018, 128, P. 20–37.

49. Titus Muurlink, O., Stephenson, P., Islam, M.Z., Taylor-Robinson, A.W. Long-term predictors of dengue outbreaks in Bangladesh: A data mining approach. Infect. Dis. Model. 2018, 3, P. 322–330.

50. Raja, D.B., Mallol, R., Ting, C.Y., Kamaludin, F., Ahmad, R., Ismail, S., Jayaraj, V.J., Sundram, B.M. Artificial intelligence model as predictor for dengue outbreaks. Malays. J. Public Health Med. 2019, 19, P. 103–108.

51. Iqbal, N., Islam, M. Machine learning for dengue outbreak prediction: A performance evaluation of different prominent classifiers. Informatica 2019, 43, P. 363–371.

52. Agarwal, N., Koti, S.R., Saran, S., Senthil Kumar, A. Data mining techniques for predicting dengue outbreak in geospatial domain using weather parameters for New Delhi, India. Curr. Sci. 2018, 114, P. 2281–2291.

53. Rao, A.S.S., Vazquez, J.A. Identification of COVID-19 can be quicker through artificial intelligence framework using a mobile phone-based survey in the populations when cities/towns are under quarantine. Infect. Control Hosp. Epidemiol. 2020, P. 1–18.

54. Randhawa, G.S., Soltysiak, M.P., El Roz, H., de Souza, C.P., Hill, K.A., Kari, L. Machine learning using intrinsic genomic signatures for rapid classification of novel pathogens: COVID-19 case study. bioRxiv 2020.

55. Yang, Z., Zeng, Z., Wang, K., Wong, S.S., Liang, W., Zanin, M., Liu, P., Cao, X., Gao, Z., Mai, Z. Modified SEIR and AI prediction of the epidemics trend of COVID-19 in China under public health interventions. J. Thorac. Dis.2020, 12, P. 165.

56. Barstugan, M., Ozkaya, U., Ozturk, S. Coronavirus (covid-19) classification using ct images by machine learning methods. arXiv 2020, arXiv:2003.09424.

57. Apostolopoulos, I.D., Mpesiana, T.A. Covid-19: Automatic detection from x-ray images utilizing transfer learning with convolutional neural networks. Phys. Eng. Sci. Med. 2020, 1.

58. Yan, L., Zhang, H.T., Xiao, Y., Wang, M., Sun, C., Liang, J., Li, S., Zhang, M., Guo, Y., Xiao, Y. Prediction of survival for severe Covid-19 patients with three clinical features: Development of a machine learning-based prognostic model with clinical data in Wuhan. medRxiv 2020.

59. Grasselli, G., Pesenti, A., Cecconi, M. Critical care utilization for the COVID-19 outbreak in Lombardy, Italy: Early experience and forecast during an emergency response. JAMA 2020.

60. Patrick van der Smagt, Ben Kröse An Introduction to Neural Networks. P. 22.

61. Simon Haykin Neural Networks: A Comprehensive Foundation. P. 241.

62. Christopher M. Bishop Pattern Recognition and Machine Learning. P. 20-40.

63. Ghafouri-Fard S., Mohammad-Rahimi H., Motie P., Minabi M.A..S, Taheri M., Nateghinia S. Application of machine learning in the prediction of COVID-19 daily new cases: A scoping review. Heliyon. 2021;7(10).

64. Kraboga D.,Kaya E. Adaptive network based fuzzy inference system (ANFIS) training approaches: a comprehensive survey Artif. Intell. Rev., 52 (4) (2019), P. 2263-2293.

65. Al-Qaness M.A.A., Ewees A.A., Fan H., Abualigah L., Elaziz M.A. Marine predators algorithm for forecasting confirmed cases of COVID-19 in Italy, USA, Iran and Korea Int. J. Environ. Res. Publ. Health, 17 (10) (2020).

66. Alsayed A.,Sadir H.,Kamil R., Sari H. Prediction of epidemic peak and infected cases for COVID-19 disease in Malaysia, 2020 Int. J. Environ. Res. Publ. Health, 17 (11) (2020)

67. Behnood A., Mohammadi Golafshani E., Hosseini S.M. Determinants of the infection rate of the COVID-19 in the U.S. using ANFIS and virus optimization algorithm (VOA) Chaos, Solit. Fractals, P. 139.

68. Alzahrani S.I., Aljamaan I.A., Al-Fakih E.A. Forecasting the spread of the COVID-19 pandemic in Saudi Arabia using ARIMA prediction model under current public health interventions J Infect Public Health, 13 (7) (2020), P. 914-919.

69. Car Z., Baressi Šegota S., Anđelić N., Lorencin I.,Mrzljak V. Modeling the spread of COVID-19 infection using a multilayer perceptron Comput Math Methods Med, 2020 (2020), P. 571.

70. Pinter G., Felde I., Mosavi A., Ghamisi P., Gloaguen R. COVID-19 pandemic prediction for Hungary; A hybrid machine learning approach Mathematics, 8 (6) (2020).

71. Zheng N., Du S., Wang J., Zhang H., Cui W., Kang Z., et al. Predicting COVID-19 in China using hybrid AI model IEEE Trans Cybern, 50 (7) (2020), P. 2891-2904.

72. Arora P., Kumar H., Panigrahi B.K. Prediction and analysis of COVID-19 positive cases using deep learning models: a descriptive case study of India Chaos, Solit. Fractals, 139 (2020), P. 110017.

73. Fokas A.S., Dikaios N., Kastis G.A. Mathematical models and deep learning for predicting the number of individuals reported to be infected with SARS-CoV-2 J. R. Soc. Interface, 17 (169)

74. Goodfellow I., Bengio Y., Courville A. Deep Learning (Adaptive Computation and Machine Learning series) P.233-250.

75. Machine Learning: A Probabilistic Perspective (Adaptive Computation and Machine Learning series) by Kevin P. Murphy P.150-187.

ДОДАТОК 1. Програмний код нейронної мережі для прогнозування динаміки

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import datetime
from numpy import hstack
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from sklearn.preprocessing import MinMaxScaler


m = 90 # досліджуваний період
n = 367  # перший досліджуваний день

N = 126264931  # населення Японії 2019
N_0 = 126529100  # населення 2018
M_0 = 1351035  # deaths in 2019
G = 921734  # birth rate 2019

sliced_growth = [[], [], []]  # confirmed death recovered but by day growth

confirmed = pd.read_csv('time_series_covid19_confirmed_global.csv')
dead = pd.read_csv('time_series_covid19_deaths_global.csv')
recovered = pd.read_csv('time_series_covid19_recovered_global.csv')


def data_process(dataset):
    dataset.rename(columns={'Country/Region': 'Country'}, inplace=True)
    index_Japan = dataset.loc[dataset['Country'] == 'Japan'].index[0]
    return index_Japan


end_date = '4/1/21'
```

```
35
def make_into_growth(list_of_country_index_ds):
    sliced = list_of_country_index_ds[0].loc[[
   list_of_country_index_ds[1]], '12/31/20':end_date]
    growth_by_day = np.diff(sliced.values, axis=1)
    begin_conditions = list_of_country_index_ds[0].loc[
   list_of_country_index_ds[1], '12/31/20']
40    return growth_by_day, begin_conditions


list_of_country_index_confirmed_with_ds = [confirmed,
   data_process(confirmed)]
list_of_country_index_dead_with_ds = [dead, data_process(dead)
   ]
45 list_of_country_index_recovered_with_ds = [recovered,
   data_process(recovered)]

growth_confirmed, begin_con_confirmed = make_into_growth(
   list_of_country_index_confirmed_with_ds)
growth_dead, begin_con_dead = make_into_growth(
   list_of_country_index_dead_with_ds)
growth_recovered, begin_con_recovered = make_into_growth(
   list_of_country_index_recovered_with_ds)
50 # print(growth_confirmed[0])

dates = []
start = datetime.datetime.strptime("01-01-2021", "%d-%m-%Y")
end = datetime.datetime.strptime("01-04-2021", "%d-%m-%Y")
55 date_generated = [start + datetime.timedelta(days=x) for x in
   range(0, (end - start).days)]

for date in date_generated:
    dates.append(date.strftime("%d-%m-%Y"))

60 croped_dates = [dates[i] if i % 50 == 0 else ' ' for i in
   range(len(dates))]
labels = ['Сприйнятливі до захворювання', 'Кількість інфікован
   их', 'Кількість одужалих', 'Померлі']


# convert an array of values into a dataset matrix
65 def split_sequences(sequences, n_steps):
```

```
    X, y = list(), list()
    for i in range(len(sequences)):
        # find the end of this pattern
        end_ix = i + n_steps
        # check if we are beyond the dataset
        if end_ix > len(sequences) - 1:
            break
        # gather input and output parts of the pattern
        seq_x, seq_y = sequences[i:end_ix, :], sequences[
    end_ix, :]
        X.append(seq_x)
        y.append(seq_y)
    return np.array(X), np.array(y)


dataset_1 = growth_confirmed[0][0:-1]
dataset_1 = np.reshape(dataset_1, (dataset_1.shape[0], 1))
dataset_2 = growth_dead[0][0:-1]
dataset_2 = np.reshape(dataset_2, (dataset_2.shape[0], 1))
dataset_3 = growth_recovered[0][0:-1]
dataset_3 = np.reshape(dataset_3, (dataset_3.shape[0], 1))
dataset = hstack((dataset_1, dataset_2, dataset_3))



scaler = MinMaxScaler(feature_range=(0, 1))
dataset = scaler.fit_transform(dataset)

n_steps = 3
n_features = 3
X, y = split_sequences(dataset, n_steps)
# split into train and test sets
train_size = int(len(X) * 0.9)
test_size = len(X) - train_size
X_train, X_test = X[0:train_size, :], X[train_size:len(X), :]
Y_train, Y_test = y[0:train_size, :], y[train_size:len(y), :]

model = Sequential(name='covid_forecast')
model.add(LSTM(100, activation='relu', return_sequences=True,
    input_shape=(n_steps, n_features)))
model.add(LSTM(50, activation='relu'))
model.add(Dense(n_features))
```

```
model.compile(loss='mean_squared_error', optimizer='adam',
    metrics=["accuracy"])
model.summary()
# fit model
history = model.fit(X_train, Y_train, epochs=50, batch_size=1,
     verbose=2, validation_data=(X_test, Y_test))
# make predictions
trainPredict = model.predict(X_train)
testPredict = model.predict(X_test)
unscaled_train = scaler.inverse_transform(trainPredict)
unscaled_test = scaler.inverse_transform(testPredict)

plt.plot(history.history['loss'], label='Training loss')
plt.plot(history.history['val_loss'], label='Validation loss')
plt.legend()
plt.show()
plt.plot(history.history['accuracy'], label='Training acc')
plt.plot(history.history['val_accuracy'], label='Validation
    acc')
plt.legend()
plt.show()
plot_time = np.arange(80, 80+len(unscaled_test), 1)
green_plot_time = np.arange(3, 3+len(unscaled_train), 1)

plt.plot(dataset_1, color='navy')
plt.plot(green_plot_time, unscaled_train[:, 0:1], color='green
    ')
plt.plot(plot_time, unscaled_test[:, 0:1], color='orange')
plt.title('Confirmed, LSTM')
plt.show()
plt.plot(dataset_2, color='navy',  label='Dead')
plt.plot(green_plot_time, unscaled_train[:, 1:2], color='green
    ')
plt.plot(plot_time, unscaled_test[:, 1:2], color='orange')
plt.title('Dead, LSTM')
plt.show()
plt.plot(dataset_3, color='navy', label='Recovered')
plt.plot(green_plot_time, unscaled_train[:, 2:3], color='green
    ')
plt.plot(plot_time, unscaled_test[:, 2:3], color='orange')
plt.title('Recovered, LSTM')
plt.show()
```

ДОДАТОК 2. Код програмної реалізації знаходження оцінок параметрів SIRM-моделі

```
from numpy import hstack
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from sklearn.preprocessing import MinMaxScaler
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import datetime


m = 90  # досліджуваний період
n = 367  # перший досліджуваний день

N =  126264931 # населення Японії 2019
N_0 = 126529100 # населення 2018
M_0 =  1351035  # deaths in 2019
G =  921734  # birth rate 2019

sliced_growth = [[], [], []]  # confirmed death recovered but by day growth

begin = [[], [], []]

confirmed = pd.read_csv('time_series_covid19_confirmed_global.csv')
dead = pd.read_csv('time_series_covid19_deaths_global.csv')
recovered = pd.read_csv('time_series_covid19_recovered_global.csv')


def data_process(dataset):
    dataset.rename(columns={'Country/Region': 'Country'}, inplace=True)
    index_Japan = dataset.loc[dataset['Country'] == 'Japan'].index[0]
    return index_Japan
```

```
35
list_of_country_index_confirmed_with_ds = [confirmed,
   data_process(confirmed)]  # датасет и нужные индексы для ст
   ран
list_of_country_index_dead_with_ds = [dead, data_process(dead)
   ]
list_of_country_index_recovered_with_ds = [recovered,
   data_process(recovered)]


40
end_date = '4/1/21'
start_date = '12/31/20'

def make_into_growth(list_of_country_index_ds):
45     sliced = list_of_country_index_ds[0].loc[[
   list_of_country_index_ds[1]], start_date:end_date]
     growth_by_day = np.diff(sliced.values, axis=1)
     begin_conditions = list_of_country_index_ds[0].loc[
   list_of_country_index_ds[1], start_date]
     return growth_by_day, begin_conditions


50
def sir_one_country():
     growth_confirmed, begin_con_confirmed = make_into_growth(
   list_of_country_index_confirmed_with_ds)
     growth_dead, begin_con_dead = make_into_growth(
   list_of_country_index_dead_with_ds)
     growth_recovered, begin_con_recovered = make_into_growth(
   list_of_country_index_recovered_with_ds)
55     I = [begin_con_confirmed - begin_con_dead -
   begin_con_recovered]
     R = [begin_con_recovered]
     M = [begin_con_dead]
     S = [(N - (n - 1) * M_0 * N / (N_0 * 365) + (n - 1) * G *
   N /
            (N_0 * 365) - I[0] - R[0] - M[0])]
60

     m_1 = m_3 = M_0 / (365 * N_0)
     gamma = (G * N/ (365 * N_0))
     for i in range(m - 1):
         S.append(int((1 - m_1) * S[i] - growth_confirmed[0][i]
    + gamma))
```

```
        M.append(int(M[i] + growth_dead[0][i]))
        R.append(int((1 - m_3) * R[i] + growth_recovered[0][i
    ]))
        I.append(int(I[i] + growth_confirmed[0][i] -
    growth_recovered[0][i] - growth_dead[0][i]))

    return S, I, R, M, m_1, gamma


SIRM = sir_one_country()

dates = []
start = datetime.datetime.strptime("01-01-2021", "%d-%m-%Y")
end = datetime.datetime.strptime("01-04-2021", "%d-%m-%Y")
date_generated = [start + datetime.timedelta(days=x) for x in
    range(0, (end - start).days)]

for date in date_generated:
    dates.append(date.strftime("%d-%m-%Y"))

croped_dates = [dates[i] if i % 10 == 0 else ' ' for i in
    range(len(dates))]
labels = ['Сприйнятливі до захворювання', 'Кількість інфікован
    их', 'Кількість одужалих', 'Померлі']


def draw(comp, i, var_label):
    plt.xlabel('Date')
    plt.ylabel('Number of people')
    plt.xticks(range(len(croped_dates)), croped_dates,
    rotation='vertical', size='small')
    plt.plot(dates, comp, label=var_label[i], color='black')
    plt.legend()
    plt.show()


draw(SIRM[1], 1, labels)

def calculate_F(a,b,matrix_f, P, F):
    v4_list = []
    flagg = True
    for i in range(a, b):
        v1 = P[i].dot(matrix_f[i].transpose())
```

```
        v2 = matrix_f[i].dot(P[i])
        v3 = v2.dot(matrix_f[i].transpose())
        try:
            v4 = np.linalg.inv(v3 + np.eye(4))
        except:
            flagg = False
        if flagg is False:
            F.append(F[i-1])
        else:
            v4_list.append(v4)
            v5 = v1.dot(v4)
            F.append(v5)
        tmp = np.eye(3) - F[i].dot(matrix_f[i])
        tmp1 = tmp.dot(P[i]).dot(tmp.transpose())
        P.append(tmp1 + np.eye(3) + F[i].dot(F[i].transpose())
    )


def calculate_mark_a(SIRM):
    mark_a = [np.array([[0],
                        [0],
                        [0]])]
    matrix_f = []
    matrix_g = []
    P = [np.zeros((3, 3))]
    F = []
    y = []
    for i in range(m):
        matrix_f.append(np.array([[-SIRM[0][i] * SIRM[1][i],
    0, 0],
                                  [SIRM[0][i] * SIRM[1][i], -
    SIRM[1][i], -SIRM[1][i]],
                                  [0, SIRM[1][i], 0],
                                  [0, 0, SIRM[1][i]]]))

        matrix_g.append(np.array([[(1 - SIRM[4]) * SIRM[0][i]
    + SIRM[5]],
                                  [SIRM[1][i]],
                                  [(1 - SIRM[4]) * SIRM[2][i
    ]],
                                  [SIRM[3][i]]]))

    calculate_F(0,m-1,matrix_f, P, F)
```

```
140
    for i in range(m-1):
        y.append(np.array([[SIRM[0][i + 1]],
                           [SIRM[1][i + 1]],
                           [SIRM[2][i + 1]],
145                        [SIRM[3][i + 1]]]) - matrix_g[i])
    for i in range(m-1):
        etw = mark_a[i] + F[i].dot(y[i] - matrix_f[i].dot(
   mark_a[i]))
        if etw[0] < 0:
            etw[0] = 0
150     if etw[1] < 0:
            etw[1] = 0
        if etw[2] < 0:
            etw[2] = 0
        mark_a.append(etw)
155    return mark_a

lab = ['Оптимальна оцінка альфа', 'Оптимальна оцінка бета', 'О
   птимальна оцінка мю_2']

mark_a_for_certain_country = calculate_mark_a(SIRM)
160 list_mark_alpha_for_certain_country = []
list_mark_beta_for_certain_country = []
list_mark_mu2_for_certain_country = []
for i in range(m):
    list_mark_alpha_for_certain_country.append(
   mark_a_for_certain_country[i][0].tolist()[0])
165    list_mark_beta_for_certain_country.append(
   mark_a_for_certain_country[i][1].tolist()[0])
    list_mark_mu2_for_certain_country.append(
   mark_a_for_certain_country[i][2].tolist()[0])
```