Харківський національний університет імені В. Н. Каразіна
Факультет математики і інформатики
Кафедра прикладної математики

Кваліфікаційна робота

освітньо-кваліфікаційний рівень: **бакалавр**

на тему

«Математичне моделювання динаміки пандемії ковід-19 з використанням стохастичних методів»

Виконала: студентка групи МП41 IV курсу
(перший бакалаврський рівень)
спеціальності 113
«Прикладна математика»
освітньої програми
«Прикладна математика»
Матвєєва Анна Юріївна

Керівник: доктор фіз.-мат. наук ,
професор кафедри
прикладної математики
Кізілова Наталія Миколаївна

Рецензент: в.о. завідувача кафедри
теоретичної та прикладної інформатики,
к.т. н., доцент
Меняйлов Євген Сергійович

Харків – 2024 рік

Анотації

Матвєєва А.Ю. Математичне моделювання динаміки пандемії COVID-19 з використанням стохастичних методів.

У роботі розглянуті стохастичні математичні методи, а саме стохастичний градієнтний спуск та генетичний алгоритм. За допомогою цих методів ми намагались змоделювати розповсюдження COVID-19 (підрахувати вірогідність смертності від цього захворювання при певних характеристиках пацієнта) а також порівняти якість роботи зазначених алгоритмів.

В результаті було отримано програми розрахунку вірогідності смертності від COVID-19 та порівняна швидкість та якість двох зазначених методів.

Matvieieva A.Y. Mathematical modeling of the dynamics of the COVID-19 pandemic using stochastic methods.

The paper considers stochastic mathematical methods, namely stochastic gradient descent and genetic algorithm. With the help of these methods, we tried to model the spread of COVID-19 (to calculate the probability of death from this disease with certain characteristics of the patient) and compare the quality of these algorithms.

As a result, we obtained programs for calculating the probability of death from COVID-19 and compared the speed and quality of the two methods.

ЗМІСТ

ВСТУП

Актуальність теми дослідження. Дана тема є дуже актуальною в контексті світової пандемії, що спричинена COVID-19. Відомо, що дана хвороба може призвести до смертельних наслідків, особливо у людей з певними ризиковими факторами, такими як похилий вік, наявність певних захворювань та імунодефіцитні стани. Моделювання розповсюдження COVID-19 дозволяє оцінити рівень загрози, яку представляє цей вірус для різних категорій населення. Такі дослідження можуть допомогти у прийнятті рішень щодо необхідних заходів для запобігання та контролю поширення вірусу.

Також моделювання розповсюдження COVID-19 є важливим для розробки ефективних стратегій лікування та профілактики хвороби. Результати таких досліджень можуть допомогти у встановленні найбільш ефективних методів лікування та профілактики COVID-19, а також визначити, які групи населення мають найбільший ризик від хвороби.

У цілому, моделювання розповсюдження COVID-19 має велику актуальність в контексті світової пандемії, оскільки від нього залежать не тільки життя та здоров'я населення, але й дії урядів та міжнародних організацій у боротьбі з пандемією.

Мета дослідження – змоделювати розповсюдження COVID-19 та дослідити математичні методи, що дозволяють оцінити ризик смертності рецепієнтів з різних вікових груп, що мають певні захворювання та імунодефіцитні стани.

Завдання дослідження:

1. Зібрати та проаналізувати статистичні дані про кількість хворих на COVID-19 та кількість смертей від цієї хвороби в різних регіонах Мексики.
2. Дослідити математичні методи, які дозволяють передбачити ризик смертності від COVID-19 в різних категоріях населення.

3. Порівняти методи передбачення смертності і виявити який з них є більш ефективним та швидким.

Об'єкт дослідження – моделі розповсюдження COVID-19.

Предмет дослідження - математичні методи розрахунку розповсюдження COVID-19 в різних категоріях населення.

Такі дослідження можуть виявити групи населення, які мають підвищений ризик смертності від COVID-19, або інші аспекти впливу, та допомогти розробити належні заходи для захисту громадян. Таким чином, предметом дослідження є аналіз різних факторів та груп населення, які можуть впливати на ризик смертності від COVID-19.

Для виконання поставлених завдань були використані наступні ***методи*** дослідження:

1) теоретичні методи: аналіз, синтез та узагальнення інформації щодо різних аспектів даного вірусу та нормативних документів з питань виникнення, симптомів та розповсюдження COVID-19; аналіз груп зони ризику при наявності COVID-19 та інших факторів, що можуть впливати на розповсюдження вірусу;
2) математичні методи: стохастичні методи, їх застосування.

Наукова новизна і теоретичне значення: вивчення поширення COVID-19 дає результати порівняння математичних методів з метою встановлення моделей розповсюдження COVID-19 рецепієнтів з різних вікових груп, що мають певні захворювання, імунодефіцитні стани або інші фактори, що впливають на розповсюдження вірусу.

Наприклад, дослідження може виявити, які захворювання або стани здоров'я можуть збільшити ризик смертності від COVID-19, і визначити, які групи населення мають підвищений ризик. Це може бути корисно для розробки ефективних стратегій профілактики та лікування.

Отже, наукова новизна та теоретичне значення дослідження розповсюдження COVID-19 полягає в тому, що це дослідження може

допомогти встановити зв'язки між різними факторами та ризиками, пов'язаними з цією хворобою, а також показати ефективність різних методів лікування та профілактики.

Практичне значення результатів дослідження. Результати дослідження моделювання розповсюдження COVID-19 можуть мати значення для практики лікування та профілактики цієї хвороби, бути корисними для лікарів, задля визначення методів лікування. Дослідження також може мати значення для широкої громадськості, яка шукає інформацію про ризики COVID-19. Результати дослідження можуть бути корисними для поширення інформованості про ризики, пов'язані з цією хворобою. Також, дане дослідження в майбутньому можна буде використовувати для вивчення розповсюдження інших вірусів.

РОЗДІЛ 1
ТЕОРЕТИЧНІ АСПЕКТИ КОВІДУ ТА ЙОГО РОЗПОВСЮДЖЕННЯ СЕРЕД НАСЕЛЕННЯ

1.1. Виникнення COVID-19. Симптоматика та розповсюдження COVID-19

COVID-19 - це хвороба, яку спричинює новий корона-вірус SARS-CoV-2. Виникнення цього вірусу пов'язують з ринком мокрих продуктів у місті Ухань, провінція Хубей, Китай, в грудні 2019 року. Вчені вважають, що цей вірус міг бути переданий людям через тварин (зокрема, через шиншил, які продавалися на ринку). Інша версія вказує на те, що вірус виник в результаті мутації вже існуючого корона-вірусу.

Згідно з офіційними даними Всесвітньої організації охорони здоров'я (ВООЗ), до 10 травня 2023 року в світі зареєстровано більше 543 мільйонів випадків COVID-19, зокрема понад 7 мільйонів смертей. Тому дослідження характеристик та механізмів поширення цієї хвороби має важливе значення для здоров'я людей та глобальної боротьби з цією пандемією [1].

COVID-19 може проявлятися різноманітними симптомами, від легких до важких форм. Зазвичай симптоми починають проявлятися через 2-14 днів після контакту з інфікованою людиною.

Серед основних симптомів COVID-19 можна виділити:

- Гарячку;
- Кашель;
- Затруджене дихання;
- Слабкість та втому;
- Біль у м'язах;
- Біль у горлі;
- Головний біль;

•Втрата смаку та нюху.

Також, можуть спостерігатися інші симптоми але вони менш типові.

Важливо зауважити, що деякі люди можуть бути інфіковані корона-вірусом, але не відчувати жодних симптомів. Такі люди можуть все ж стати переносниками вірусу та передавати його іншим людям [2].

COVID-19 поширюється серед населення головним чином повітряно-краплинним шляхом під час прямих контактів з інфікованими людьми.

Основні шляхи передачі включають:

•Повітряно-краплинний шлях: Вірус може передаватися через краплі, які виділяються під час кашлю, чхання, розмови або спільного використання предметів з інфікованою особою. Ці краплі мають розмір більше 5 мікрометрів і можуть попадати на слизові поверхні людей навколо, або крізь їх верхні дихальні шляхи до легені.

•Контактний шлях: Вірус може передаватися через прямий контакт з інфікованими поверхнями або предметами, на яких знаходяться живі віруси. Якщо людина торкається такої поверхні або предмета, а потім торкається своїх очей, носа або рота, він може заразитися.

Вірус також може передаватися повітряними краплями, особливо в закритих приміщеннях, де може бути обмежений доступ до свіжого повітря. Ці рекомендації базуються на наукових дослідженнях та офіційних рекомендаціях організацій, таких як Всесвітня організація охорони здоров'я (ВООЗ) і Центри контролю та профілактики хвороб (CDC)[3].

РОЗДІЛ 2
ОПИС МАТЕМАТИЧНИХ МЕТОДІВ ДОСЛІДЖЕННЯ

2.1. Опис методу «Стохастичного градієнту»

Стохастичний градієнтний спуск — це дуже популярний і поширений алгоритм, який використовується в різних алгоритмах машинного навчання, і, що найважливіше, є основою нейронних мереж.

Градієнт простою мовою означає нахил або нахил поверхні. Отже, градієнтний спуск буквально означає спуск по схилу, щоб досягти найнижчої точки цієї поверхні. Уявімо собі двовимірний графік, такий як парабола на малюнку нижче.

На Рис.4 найнижча точка на параболі знаходиться в точці x = 1,5. Мета алгоритму градієнтного спуску полягає в тому, щоб знайти таке значення «x», щоб «y» було мінімальним. «y» тут називається цільовою функцією, на якій працює алгоритм градієнтного спуску, щоб опуститися до найнижчої точки.

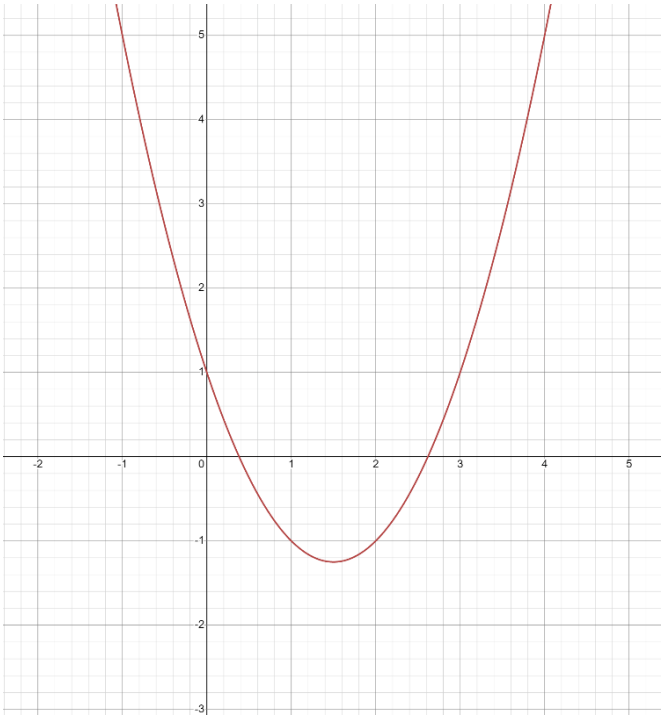

Рис.4 (Параболічна функція з двома вимірами (x, y))

Градієнтний спуск — це ітераційний алгоритм, який починається з випадкової точки на функції та рухається вниз по схилу кроками, доки не досягне найнижчої точки цієї функції.

Алгоритм градієнтного спуску полягає в тому, щоб почати з випадкової точки (у нашому прикладі параболи почати з випадкової «x») і знайти спосіб оновити цю точку з кожною ітерацією, щоб ми спускалися по схилу.

Кроки алгоритму такі[4]:

1. Знаходимо нахил цільової функції відносно кожного параметра/ознаки. Іншими словами, обчислюємо градієнт функції;
2. Вибираємо довільне початкове значення для параметрів;
3. Оновлюємо функцію градієнта, підключивши значення параметрів;
4. Обчислюємо розмір кроку для кожної функції як:
   розмір кроку = градієнт * швидкість навчання;
5. Обчислюємо нові параметри як:
   нові параметри = старі параметри - розмір кроку
6. Повторюємо кроки 3–5, поки градієнт не стане майже нульовим.

«Швидкість навчання», згадана вище, є гнучким параметром, який сильно впливає на збіжність алгоритму. Більші швидкості навчання змушують алгоритм робити великі кроки вниз по схилу, і він може перескочити через мінімальну точку, таким чином пропустивши її. Отже, завжди добре дотримуватися низького рівня навчання, наприклад 0,01. Можна також математично показати, що алгоритм градієнтного спуску робить більші кроки вниз по схилу, якщо початкова точка знаходиться високо, і робить невеликі кроки, коли він наближається до шуканого значення.

Алгоритм градієнтного спуску має кілька недоліків.

Наприклад, у нас є 10 000 точок даних і 10 функцій. Сума квадратів залишків складається з стільки доданків, скільки точок даних, тобто 10000

доданків у нашому випадку. Нам потрібно обчислити похідну цієї функції по відношенню до кожної функції, тому фактично ми будемо робити 10000 * 10 = 100 000 обчислень за ітерацію. Зазвичай потрібно 1000 ітерацій, фактично ми маємо 100 000 * 1000 = 100000000 обчислень для завершення алгоритму. Це значною мірою накладні витрати, тому градієнтний спуск відбувається повільно на великих даних. Саме тому краще використовувати саме «стохастичний» градієнтний спуск.

Стохастичний градієнтний спуск:

Під час вибору точок даних на кожному кроці обчислюються похідні. Методом стохастичного градієнтного спуску випадково вибирається одна точка даних із усього набору даних на кожній ітерації, щоб значно скоротити обчислення.

Тобто основною проблемою методу стохастичного градієнту може бути його повільність при великій вибірці даних.

Алгоритм градієнтного спуску є наближеним та ітераційним методом математичної оптимізації і може використовуватись для наближення до мінімуму будь-якої диференційованої функції.

1. Функція витрат: мета оптимізації:

Функція витрат є функцією, яку потрібно мінімізувати (або максимізувати), змінюючи змінні рішення. Багато методів машинного навчання вирішують проблеми оптимізації під поверхнею. Вони прагнуть мінімізувати різницю між фактичними та прогнозованими результатами шляхом коригування параметрів моделі.

У задачі регресії ми зазвичай маємо вектори вхідних змінних $x = (x_1, \dots, x_r)$і фактичні результати y. Ви хочете знайти модель, яка відображає x на прогнозовану відповідь $f(x)$, щоб $f(x)$ був якомога ближчим до y.

Наша мета — мінімізувати різницю між прогнозом $f(x)$ і фактичними даними y. Ця різниця називається залишком.

У задачі цього типу потрібно мінімізувати суму квадратів залишків (SSR), де SSR=$\sum_i \big(y_i - f(x_i)\big)^2$ для всіх спостережень i = 1, …, n, де n – загальна кількість спостереження. Крім того, можна використовувати середню квадратичну помилку (MSE = SSR/n) замість SSR.

І SSR, і MSE використовують квадрат різниці між фактичним і прогнозованим результатами. Чим менша різниця, тим точніший прогноз. Різниця в нуль означає, що прогноз дорівнює фактичним даним.

SSR або MSE мінімізується шляхом налаштування параметрів моделі. Наприклад, у лінійній регресії якщо ми хочемо знайти функцію $f(x) = b_0 + b_1x_1 + \cdots + b_rx_r$, то потрібно визначити ваги b_0, b_1, …, b_r, які мінімізують SSR або MSE.

У задачі класифікації результати y є категоріальними, часто 0 або 1. Наприклад, можна спробувати передбачити, чи є електронний лист спамом чи ні. У випадку двійкових виходів зручно мінімізувати функцію крос-ентропії, яка також залежить від фактичних виходів y_i і відповідних прогнозів $p(x_i)$:

$$H = -\sum_i \big(y_i\, log\big(p(x_i)\big) + (1 - y_i) \cdot log\big(1 - p(x_i)\big)\big)$$

У логістичній регресії, яка часто використовується для вирішення проблем класифікації, функції $p(x)$ і $f(x)$ визначаються наступним чином:

$$P(x) = \frac{1}{1 + exp\big(-f(x)\big)}$$

$$f(x) = b_0 + b_1x_1 + \cdots + b_rx_r$$

2. Градієнт функції. Підготовка до обчислення:

В аналізі похідна функції показує, наскільки змінюється значення, коли ми змінюємо його аргумент (або аргументи). Похідні важливі для оптимізації, оскільки нульові похідні можуть вказувати на мінімум, максимум або сідлову точку.

Градієнт функції C кількох незалежних змінних $v_1, \ldots, v_r$ позначається як $\nabla C(v_1, \ldots, v_r)$ і визначається як вектор-функція часткових похідних C за кожною незалежною змінною: $\nabla C = (\partial C/\partial v_1, \ldots, \partial C/v_r)$.

Ненульове значення градієнта функції C в даній точці визначає напрямок і швидкість найшвидшого зростання C. Під час роботи з градієнтним спуском нас цікавить напрямок найшвидшого *зменшення* функції витрат. Цей напрямок визначається від'ємним градієнтом, $-\nabla C$.

3. Швидкість навчання алгоритму:

Щоб зрозуміти алгоритм градієнтного спуску, уявімо краплю води, що ковзає по стінці миски, або м'яч, що котиться з пагорба. Крапля і кулька прагнуть рухатися в напрямку найшвидшого зниження, поки не досягнуть дна. З часом вони набирають обертів і прискорюються.

Ідея градієнтного спуску схожа: ми починаємо з довільно вибраного положення точки або вектора $v = (v_1, \ldots, v_r)$ і переміщуємо його ітераційно в напрямку найшвидшого зменшення функції витрат. Як згадувалося, це напрямок вектора від'ємного градієнта, $-\nabla C$.

Якщо у нас є випадкова початкова точка $v = (v_1, \ldots, v_r)$, ми оновлюємо її або переміщуємо в нову позицію в напрямку від'ємного градієнта: $v \rightarrow v - \eta \nabla C$, де η є невеликим додатним значенням, яке називається швидкістю навчання.

Швидкість навчання визначає, наскільки великий крок оновлення або переміщення. Це дуже важливий параметр. Якщо η занадто мале, алгоритм може сходитися дуже повільно. Великі значення η також можуть спричинити проблеми зі збіжністю або зробити алгоритм розбіжним.

Необхідність винаходження стохастичного градієнтного виникла через такі причини:

1. У машинному навчанні зазвичай є дуже великі набори даних із мільйонами або мільярдами точок даних. Використання традиційних алгоритмів градієнтного спуску на таких великих наборах даних є

обчислювально дорогим і непрактичним. У таких випадках SGD працює добре, оскільки використовує випадково вибрані точки даних для оновлення параметрів моделі, роблячи процес швидшим і ефективнішим.

2. У багатьох задачах машинного навчання цільова функція не є опуклою, що означає наявність кількох локальних мінімумів. Традиційні алгоритми градієнтного спуску можуть застрягти в одному з локальних мінімумів, що призведе до неоптимальних моделей. SGD має меншу ймовірність застрягти, оскільки він оновлює параметри, використовуючи лише кілька точок даних одночасно, що підвищує ймовірність знайти глобальний мінімум.
3. У деяких програмах машинного навчання нові дані постійно генеруються та додаються до набору даних. У таких випадках необхідно оновлювати параметри моделі в режимі реального часу, у міру появи нових даних. SGD добре підходить для таких завдань, оскільки він оновлює параметри моделі за допомогою невеликих пакетів даних і може застосовуватися до даних, щойно вони надходять.

Порівняємо стохастичний градієнтний спуск і звичайний градієнтний спуск. Стохастичний градієнтний спуск має такі переваги (Рис. 5):

1. Стохастичний градієнтний спуск набагато швидше, ніж звичайний градієнтний спуск, особливо коли набір даних великий. Це пояснюється тим, що SGD оновлює параметри моделі лише після кожної окремої точки даних, тоді як GD оновлює параметри після кожної групи точок даних.
2. SGD більш стійкий до шуму в даних, ніж GD. Це пояснюється тим, що стохастичний градієнтний спуск використовує лише одну точку даних для оновлення параметрів, тому на нього менш імовірно впливають викиди або шум у даних.

3. Стохастичний градієнтний спуск є більш ефективним, ніж GD, з точки зору використання пам’яті. Це пояснюється тим, що SGD потрібно зберігати лише поточні параметри та градієнт функції втрат, тоді як градієнтний спуск потрібно зберігати весь набір даних.

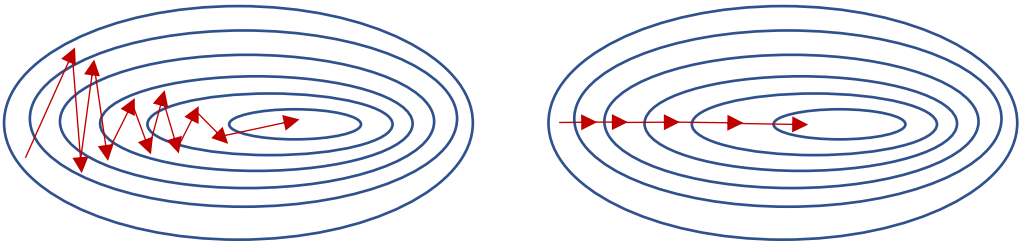

Рис.5 (стохастичний градієнтний спуск та градієнтний спуск)

2.2. Опис методу «Генетичного алгоритму»

Генетичний алгоритм (GA) - це стохастичний метод, який моделює природну еволюцію в просторі розв’язків задач оптимізації. Він діє на сукупності потенційних рішень (тобто індивідуумів) у кожній ітерації (тобто покоління).

Генетичний алгоритм є варіантом стохастичного променевого пошуку, у якому стани-наступники формуються шляхом комбінування двох батьківських станів, а не за допомогою модифікації єдиного стану. Робота алгоритмів GA починається з множини k сформованих випадковим чином станів, які називаються популяцією. Кожен стан, або індивідуум, представлений у вигляді рядка символів зі скінченного алфавіту, найчастіше у вигляді рядка з нулів (0) і одиниць (1).

Ключові компоненти генетичних алгоритмів:

1. Ініціалізація заповнення

Починаємо з випадково згенерованої сукупності H особин (розв'язків), де кожна особина представлена хромосомою (вектором параметрів).

2. Фітнес-функція

   Фітнес-функція $f: P^n \to P$ використовується для оцінки придатності кожної особини в популяції. Вищі значення придатності вказують на кращі рішення.

3. Вибір

   Вибираємо особин із поточної популяції, щоб сформувати пул для схрещування на основі їх «фізичної підготовки». Загальні методи відбору включають вибір колеса рулетки, турніру та рангу.

4. Рекомбінація

   Створення потомства шляхом поєднання генетичної інформації двох батьків. Найпоширеніші методи кросинговеру включають одноточковий, двоточковий та рівномірний перехрест.

   Математично, якщо $p_1 = (x_1, x_2, \dots, x_n)$ та $p_2 = (i_1, i_2, \dots, i_n)$ є двома батьківськими векторами параметрів, одноточкова рекомбінація може дати нащадків O_1 та O_2 такі як:

   $O_1 = (x_1, x_2, \dots, x_k, i_{k+1}, \dots, i_n)$ $\quad$ $O_2 = (i_1, i_2, \dots, i_k, x_{k+1}, \dots, x_n)$

   Тут k це точка перетину, обрана випадковим чином.

5. Мутація

   Застосовуючи випадкові зміни до окремих нащадків, щоб зберегти генетичну різноманітність. Для векторів $c = (x_1, x_2, \dots, x_n)$ мутація може змінитися від x_i до x_i' з невеликою ймовірністю P_m.

   Якщо x_i змінено на x_i' операцію мутації можна записати як:

   $x_i' = x_i + d$ де d є маленькою випадковою величиною.

6. Заміна

Замінюємо частину чи всю популяцію новим потомством. Стратегії включають заміну поколінь (замінюється вся популяція) і заміну в стаціонарному стані (замінюється тільки декілька особин).

Етапи алгоритму виглядають так:

1. Ініціалізація населення $P(0)$ з N випадковими особами.
2. Оцінка «фізичної форми» $f(c)$ для кожної особи c з популяції.
3. Повторюємо, доки не буде виконано критерій зупинки (наприклад, максимальна кількість поколінь або задовільний рівень «фізичної підготовки»):
    - Вибираємо особин з популяції на основі придатності для формування пулу для схрещування.
    - Схрещування пари особин із пулу для отримання потомства.
    - Мутування потомства з невеликою ймовірністю.
    - Оцінка придатності нового потомства.
    - Заміна частини або всієї популяції новим потомством.

Математичне формулювання:

- Популяція при генерації t: $P(t) = \{c_1(t), c_2(t), \dots, c_N(t)\}$
- Оцінка «фізичної форми» $f\big(c_i(t)\big)$ для кожного $c_i(t) \in P(t)$
- Вибір: на основі ймовірностей, пропорційних придатності,

$$P(select\ c_i) = \frac{f(c_i)}{\sum_{j=1}^{N} f(c_j)}$$

- Схрещування: для обраних особин p_1 та p_2 породжуємо потомство O_1 та O_2
- Мутація: для кожного гена в потомстві з імовірністю P_m, змінюємо ген
- Заміна: створюємо нову популяцію $P(t+1)$ від поточної популяції та потомства

Алгоритм збігається до оптимального або близького до оптимального розв'язку протягом послідовних поколінь. Ефективність та результативність

GA залежить від таких параметрів, як розмір популяції N, швидкості схрещення p_c, швидкості мутації p_m та тиску відбору.

Якщо показувати роботу даного алгоритму на блок схемі то вона виглядає таким чином (Рис. 6):

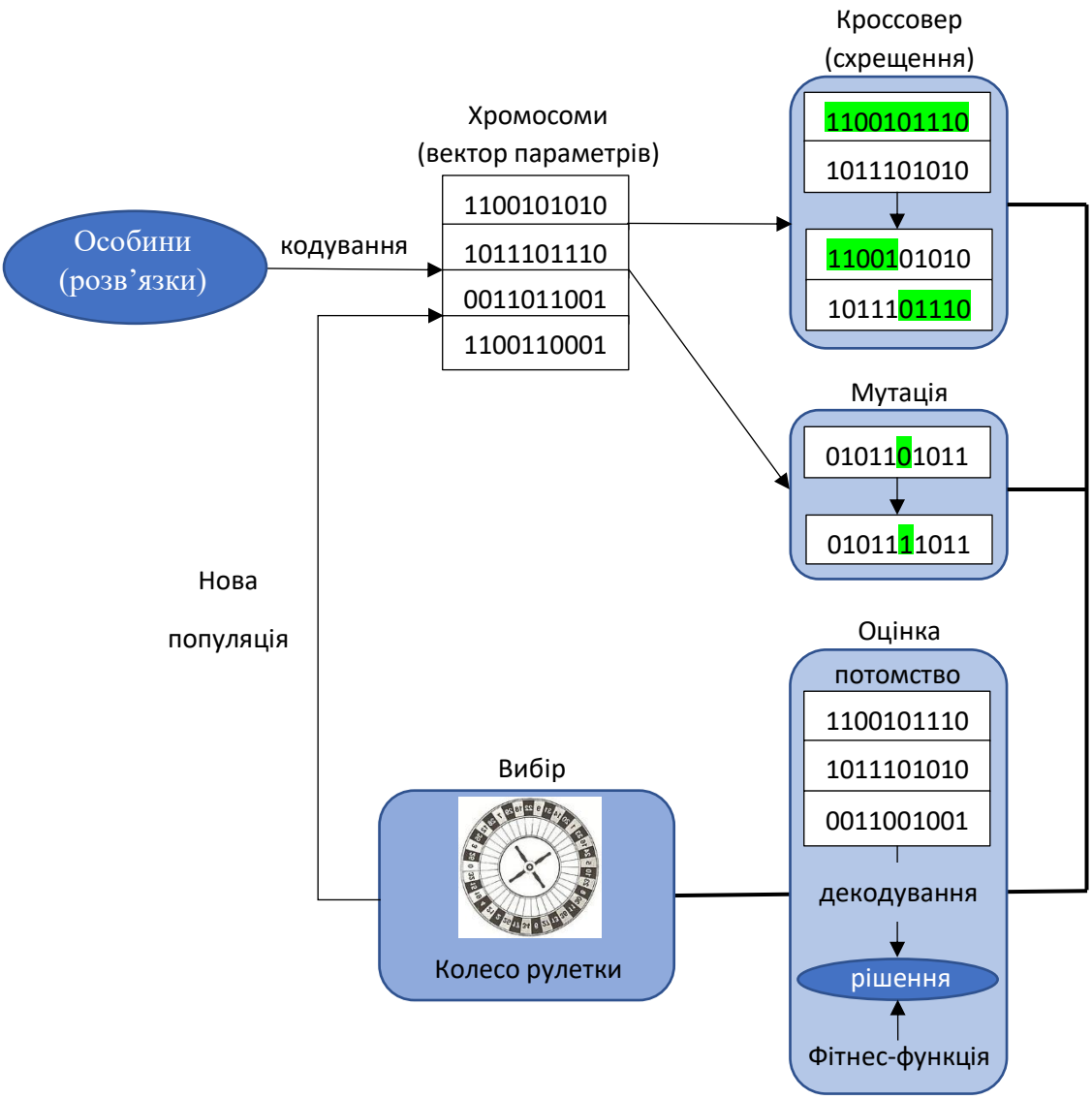

Рис. 6 (блок-схема роботи алгоритму)

Розглянемо приклад, що представлено нижче[5]:

Нехай ми маємо функцію $f(x) = 2x^2 + 1$ де $x \in [0; 15]$.

Ми маємо виявити таку точку з 16 можливих (0,1,…,15) в якій функція має екстремум, це і буде задачею оптимізації. Множина {0,1,..., 15} це

простір пошуку та множина потенційних розв’язків задачі. Ці 16 чисел називаються фенотипом або значенням параметра. Найкращим або оптимальним розв'язком називають такий розв'язок, який оптимізує функцію.

Значення параметра $x \in [0; 15]$ можна закодувати таким чином:

0000 0001 0010 0011 0100 0101 0110 0111

1000 1001 1010 1011 1100 1101 1110 1111

Це двійкове кодування – поширений спосіб який являє собою запис десяткових цифр у двійковій системі. Представлені кодові послідовності також називаються хромосомами або набором параметрів, у даному прикладі вони також являються генотипами. Кожен з цих наборів параметрів складається з 4 генів (тобто кожна з цих двійкових послідовностей складається з 4 бітів). Значення гена в конкретній позиції називається алеллю, яка приймає в даному випадку значення 0 або 1.

З цих 16 наборів параметрів вибираються особини (розв’язки), які в кінцевому результаті і складають популяцію. Наприклад, якщо нам треба скласти популяцію з чисельністю 5 то підійде множина наборів параметрів: {0001, 0101, 1011, 0111, 0100} при розкодуванні маємо набір наступних фенотипів: {1, 5, 11, 7, 4}, а функцію пристосованості ми задавали на початку, а саме $f(x) = 2x^2 + 1$.

Значення цієї функції для конкретних x визначає пристосованість відповідних наборів параметрів в популяції, тобто для фенотипів, які відповідають певним генотипам.

Висновок

Після опрацювання за допомогою даних алгоритмів набору великих даних (а саме 1 048 576 унікальних пацієнтів) були отримані такі результати [6]:

Точність даних методів складає (Рис. 7, 8):

```
# Step 1: Multiply the solution with the test data to get the predicted output
output = np.sum(solution*X_test, axis=1)

# Step 2: Apply a threshold to the predicted output to classify it into the correct classes
predicted_output = np.where(output > 0.5, 1, 0)

# Step 3: Use the accuracy_score function to compare the predicted output with the actual output
accuracy = accuracy_score(y_test, predicted_output)

print("Accuracy of the genetic algorithm solution on the test data: ", accuracy)
```

[14] ✓ 0.0s

```
Accuracy of the genetic algorithm solution on the test data:  0.9441241685144124
```

Рис. 7 (скріншот підрахунку точності генетичного алгоритму)

```
clf = make_pipeline(StandardScaler(), SGDClassifier(max_iter=1000, tol=1e-3,
                                                    alpha=0.0001,
                                                    loss='log_loss',
                                                    penalty='elasticnet',
                                                    l1_ratio=0.15))
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
accuracy_score(y_test, y_pred)
```

[17] ✓ 2.8s

```
0.9355601649858141
```

Рис. 8 (скріншот підрахунку точності алгоритму стохастичного градієнтного спуску)

Як можна побачити, генетичний алгоритм в цілому є більш точним.

Зазначимо, що швидкість роботи алгоритмів порівнювати не є доцільним через наявність в алгоритмах випадкових величин. Тобто, при

першій прогонці алгоритми мали такий час виконання: генетичний алгоритм – 37 хвилин 4.4 секунди, а стохастичний градієнт – 27 хвилин 12.6 секунд. При другій прогонці: 36 хвилин 55.4 секунди та 22 хвилини 55.8 секунд відповідно (хоча можна зазначити більшу швидкість стохастичного градієнтного спуску).

При підрахунках точності смертності для «живих» та «мертвих» пацієнтів окремо маємо такі результати (Рис. 9, 10):

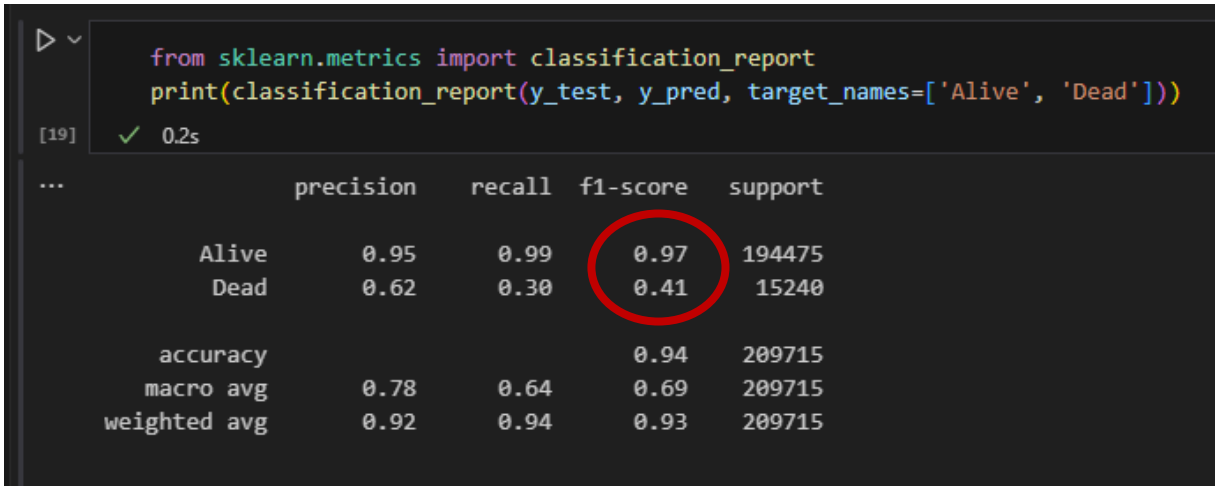

Рис. 9 (точність розрахунків програми стохастичного градієнтного спуску для «живих» та «мертвих» пацієнтів окремо)

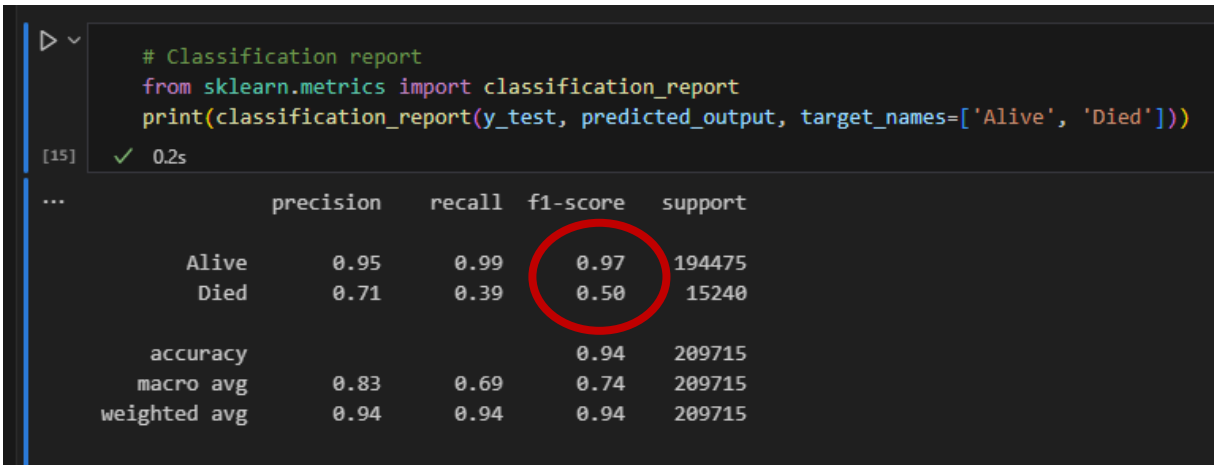

Рис. 10 (точність розрахунків програми генетичного алгоритму для «живих» та «мертвих» пацієнтів окремо)

Під «живими» та «мертвими» мається на увазі, що алгоритм окремо рахує точність прогнозування що при певних характеристиках пацієнта він виживе чи помре. Тобто результат програми порівнюється з реальним станом даного пацієнта з вибірки і вираховується похибка.

З даних результатів можна побачити, що прогнозування саме смертності а не виживання певного пацієнта є більш точним для обох методів (і однакове, в обох випадках 97% точності).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Всесвітня організація охорони здоров'я (ВООЗ): https://www.who.int/emergencies/disease-outbreak-news/item/2020-DON-12-january-2020-novel-coronavirus-outbreak
2. Всесвітня організація охорони здоров'я (ВООЗ): https://www.who.int/health-topics/coronavirus#tab=tab_3
3. Центри контролю та профілактики захворювань. (2021, 5 квітня). Як захистити себе та інших https://www.cdc.gov/coronavirus/2019-ncov/prevent-getting-sick/prevention.html
4. https://realpython.com/gradient-descent-algorithm-python/
5. http://www.znannya.org/?view=ga_general
6. https://drive.google.com/drive/folders/1tsplnDxglKBm6tvun6Zy-OniVgADJBTj?usp=sharing