огляд літератури

Іван Кущ

February 2025

1 Машине Навчання

Машинне навчання та штучний інтелект – це не просто інструменти, а нова філософія роботи з інформацією. Вони змінюють підхід до наукових досліджень, роблячи їх швидшими, точнішими та масштабнішими. У майбутньому ці технології можуть привести до відкриттів, які ми сьогодні навіть не можемо уявити. Машинне навчання (ML) ділиться на три основні категорії: навчання з учителем (Supervised Learning, SL), навчання без учителя (Unsupervised Learning, UL) і навчання з підкріпленням (Reinforcement Learning, RL).

У методі навчання з учителем(Supervised Learning, SL) , модель навчається на розмічених даних, тобто кожен вхідний об'єкт супроводжується відповідним вихідним значенням. Мета - знайти функцію f, яка апроксимує залежність між входами X і виходами Y:

$$Y = f(X) + \epsilon$$

де ϵ – випадкова помилка.

Навчання без вчителя (Unsupervised Learning) Цей метод використовується, коли дані не мають розмічених міток. Алгоритм шукає приховані структури в даних.Прикладом такого метроду , є метод кластеризації-об'єднує схожі об'єкти.

В навчання з підкріпленням (Reinforcement Learning) агент взаємодіє зі середовищем та отримує винагороду за дії. Мета - максимізувати функцію винагороди:

$$Q(s, a) = r + \gamma \max_{a'} Q(s', a')$$

де $Q(s, a)$ - цінність дії a в стані s, r - винагорода, γ - коефіцієнт дисконтування.

2 Методи теоретичного розрахунку рухливості носіїв заряду

Перш ніж застосовувати машинне навчання для побудови моделей рухливості, необхідно мати фізично обґрунтовані теоретичні оцінки, які слугують базовими орієнтирами або джерелами навчальних даних. У цій роботі було розглянуто три основні аналітичні методи — Классена, Арори та Флетчера, кожен із яких має свої переваги, обмеження та сферу застосування.

2.1 Модель Классена

Метод Классена є уніфікованим підходом для моделювання рухливості електронів і дірок у напівпровідниках. Його основна перевага полягає в можливості одночасного врахування кількох механізмів розсіювання: на фононах (ґратці) та на іонізованих домішках. Модель має вигляд:

$$\mu^{-1} = \mu_L^{-1} + \mu_{DA}^{-1},$$

де μ_L — рухливість, обмежена фононним розсіюванням, а μ_{DA} — домішковим.

Фононна складова моделюється степеневим законом:

$$\mu_L = \mu_{\max} \cdot \left(\frac{300}{T}\right)^{2.25},$$

а домішкова — складнішою функцією температури, ефективних концентрацій та коефіцієнтів іонізації. Наприклад:

$$\mu_{DA} \propto \frac{(T/300)^{3\alpha-1.5}}{N_{\text{eff}}^{\alpha}},$$

де N_{eff} — ефективна концентрація домішок з урахуванням температурних залежностей та іонізації, а α — параметр підгонки.

Ця модель добре описує поведінку в широкому температурному діапазоні та для різних рівнів легування, однак потребує численних емпіричних коефіцієнтів.

2.2 Модель Арори

Підхід Арори пропонує спрощену, але адаптивну модель, що концентрується на залежностях рухливості від температури та загальної концентрації носіїв:

$$\mu = \mu_{\min} + \frac{\mu_{\max}}{1 + \left(\frac{N}{N_0}\right)^{\alpha}}.$$

Це вираження дозволяє плавно інтерполювати між максимальною та мінімальною рухливістю, враховуючи насичення при високій концентрації носіїв. Додатково, усі параметри цієї моделі — $\mu_{\min}$, $\mu_{\max}$, N_0, α — масштабуються із температурою за степеневими законами:

$$\mu_{\min}(T) = \mu_{\min}^{\text{eff}} \left(\frac{T}{T_{\text{eff}}}\right)^{\beta_1}, \quad \alpha(T) = \alpha_0^{\text{eff}} \left(\frac{T}{T_{\text{eff}}}\right)^{\beta_4},$$

та інші подібні вирази. Цей метод менш фізично деталізований, ніж Классена, однак легший у реалізації і дає стабільні результати для типових умов.

2.3 Модель Флетчера

Метод Флетчера базується на простому підході складання зворотних рухливостей, але із введенням температурно залежної колізійної складової:

$$\mu_{n,p}^{-1} = \mu_{\text{in}}^{-1} + \mu_{\text{cc}}^{-1},$$

де:

$$\mu_{\text{cc}} = \left(\frac{T}{T_{\text{eff}}}\right)^{3/2} \cdot F_1 \cdot \sqrt{np} \cdot \ln\left(1 + \left(\frac{T}{T_{\text{eff}}}\right)^{2} \cdot \frac{1}{(np)^{1/3}} F_2\right).$$

Цей метод дозволяє враховувати як температуру, так і взаємодію електронів та дірок, що особливо важливо в умовах високих концентрацій або нерівноважних станів. Він також відзначається простотою реалізації при збереженні розумного рівня точності.

Порівняння методів

- **Классен:** найбільш повна модель, підходить для симуляцій на всьому діапазоні температур та легування; складна у реалізації, потребує великої кількості емпіричних параметрів.
- **Арора:** компроміс між точністю та простотою, добре працює при середніх температурах та концентраціях.
- **Флетчер:** підходить для попередніх оцінок або швидких розрахунків у системах з відомими температурними профілями.

Таким чином, використання одразу кількох теоретичних моделей дозволяє гнучко адаптувати обчислення під конкретні умови, а також створює якісну основу для побудови та навчання моделей машинного навчання.

2.4 Основні Властивості Випадкового лісу (Random Forest, RF)

У сучасному світі величезний потік інформації вимагає ефективних методів її аналізу. Одним із найбільш популярних і потужних алгоритмів машинного навчання є випадковий ліс (Random Forest, RF). Він поєднує простоту дерева рішень з ансамблевим підходом, що дозволяє підвищити точність і зменшити ризик переобучення. Випадковий ліс є ансамблевим методом, який базується на методі бутстреп-агрегування (bagging). Основна ідея полягає в тому, щоб створити кілька незалежних дерев рішень і об'єднати їхні результати.У кожному вузлі дерева випадковим чином вибирається тільки частина ознак, що зменшує кореляцію між деревами.Дерево росте до максимальної глибини або досягнення порогу розбиття.Для класифікації використовується голосування більшості: клас, що зустрічається найчастіше серед дерев, обирається як остаточний. Нехай маємо датасет:

$$D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$$

де x_i — вектор ознак, а y_i — мітка класу (у класифікації) або числове значення (у регресії).

Остаточний клас $\hat{y}$ визначається більшістю голосів:

$$\hat{y} = \arg\max_c \sum_{i=1}^{m} \mathbb{I}(T_i(x) = c)$$

де:

- m — кількість дерев у лісі, - $T_i(x)$ — прогноз i-го дерева, - c — один із класів, - $\mathbb{I}(\cdot)$ — індикаторна функція, що дорівнює 1, якщо прогноз дерева дорівнює класу c, і 0 в іншому випадку.

Ансамблевий підхід дозволяє випадковому лісу перевершувати класичні моделі, такі як лінійна регресія або поодинокі дерева рішень, особливо у випадку складних нелінійних залежностей.

Для кожного дерева використовується випадкова підмножина ознак, що зменшує кореляцію між деревами та підвищує стійкість алгоритму.

Формально, помилка узгодженого ансамблю (ensemble error) визначається як:

$$E = \rho E_T + \frac{1-\rho}{m} E_I$$

де:

- E — загальна помилка ансамблю,
- ρ — середня кореляція між деревами,
- E_T — помилка окремого дерева,
- E_I — незалежна помилка.

Якщо $m \to \infty$ і ρ мале, тоді $E \to E_I$, тобто помилка зменшується!

Оскільки алгоритм використовує бутстреп-вибірки, окремі аномальні точки не впливають на всі дерева, що робить модель стійкою до викидів (outliers).

Формально, випадковий ліс може зменшувати вплив викидів на оцінку середнього значення в задачах регресії:

$$\hat{y}_{RF} = \frac{1}{m} \sum_{i=1}^{m} T_i(x) \approx \text{median}(T_i(x))$$

Медіана є більш стійкою до викидів, ніж середнє значення!

Випадковий ліс дозволяє визначити, які ознаки мають найбільший вплив на прогноз.

Метрика важливості ознаки (Gini Importance) обчислюється як:

$$I(X_j) = \sum_{t \in T} p(t)(1-p(t)) - \sum_{k \in \{left, right\}} p(t_k)(1-p(t_k))$$

де:

- $I(X_j)$ — важливість ознаки X_j,
- $p(t)$ — частка вибірки, яка потрапляє у вузол t,
- $p(t_k)$ — частки після розбиття вузла на дочірні вузли.

Це допомагає проводити відбір ознак, спрощуючи модель!

Головний недолік випадкового лісу — це час навчання та використання, оскільки він створює багато дерев і робить численні обчислення.

Складність алгоритму:

$$O(m \cdot n \log n)$$

де:

- m — кількість дерев,
- n — кількість зразків у навчальній вибірці.

Чим більше дерев, тим вища точність, але тим довше час навчання!

На відміну від окремого дерева рішень, де логіка ухвалення рішення є прозорою, випадковий ліс працює як чорний ящик.

Він не надає чітких правил ухвалення рішень, а лише усереднює прогнози дерев.

Приклад: В індивідуальному дереві ми можемо чітко сказати, що "Якщо дохід $>$ 50 тис., то людина бере кредит але у випадковому лісі кожне дерево може мати інший критерій.

Це ускладнює використання у чутливих сферах, таких як медицина чи право.

Якість випадкового лісу залежить від:

- Кількості дерев m (замало — низька точність, забагато — довге навчання),
- Максимальної глибини дерев (надмірно глибокі дерева можуть переобучатися),
- Кількості ознак для поділу (якщо вибирати забагато, дерева стають подібними).

Оптимальне налаштування:

$$m = 100 - 500 \quad \text{дерев}, \quad k = \frac{p}{\sqrt{m}}, \quad \text{де } p^{-}\text{загальна кількість ознак.}$$

2.5 Основні Властивості Символічної регресії (Symbolic Regression, SR)

Випадковий ліс, демонструючи вражаючу ефективність у задачах прогнозування, страждає від проблеми "чорної скриньки". Хоча модель здатна генерувати якісні результати, її внутрішня структура залишається непрозорою, що ускладнює інтерпретацію отриманих висновків та розуміння факторів, які впливають на результат.Для більш якіснох інтерпретації наших даних та їх передбаченню був сворений метод - Символічна регресія(Symbolic Regression, SR). Символічна регресія (Symbolic Regression, SR) — це потужний метод машинного навчання, який дозволяє автоматично знаходити математичні функції, що найкраще описують дані, без необхідності заздалегідь визначати функціональну форму. Цей метод є особливо корисним у випадках, коли немає можливості вивести функцію з перших принципів, або коли форма розподілу даних є довільною та складною для опису традиційними методами. Однією з ключових властивостей символічної регресії є її здатність автоматично шукати функції, які найкраще описують дані. На відміну від традиційних методів регресії, таких як поліноміальна регресія, які вимагають заздалегідь визначеної функціональної форми, SR дозволяє моделі самостійно знаходити оптимальну функцію. Це досягається за рахунок використання генетичного програмування, де функції представлені у вигляді дерев виразів, які еволюціонують протягом процесу навчання.

У генетичному програмуванні функції генеруються шляхом мутацій (зміни вузлів дерева) та кросоверу (обмін піддеревами між різними кандидатами). Цей процес дозволяє SR шукати функції в широкому просторі можливих математичних виразів, не обмежуючись попередньо заданими формами. Таким чином, SR усуває необхідність емпіричного підбору функцій, що є трудомістким і часто неточним процесом.

SR дозволяє знаходити функції, які точно описують такі розподіли, навіть якщо вони не мають явного аналітичного виразу. Це досягається за рахунок того, що SR не обмежується попередньо заданими функціональними формами, а шукає їх у широкому просторі можливих математичних виразів. Сума квадратів похибок (SSE) є широко використовуваною метрикою оцінки якості моделі в алгоритмах символічної регресії.

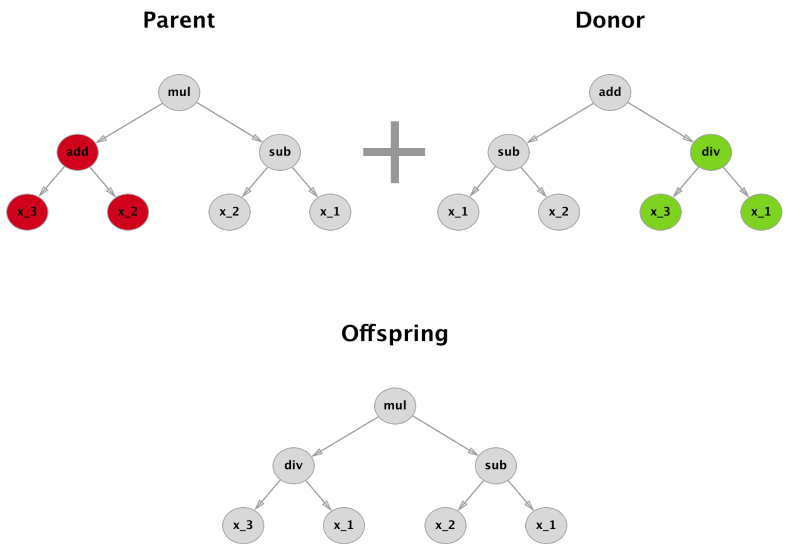

Рис. 1: Підхід генетичного програмування до символічної регресії. Функції представлені деревами виразів. Нові функції генеруються за допомогою мутації деревних вузлів.

Вона базується на обчисленні різниці між прогнозованими та фактичними значеннями цільової змінної та визначається наступним чином

$$\text{SSE} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{1}$$

де y_i - фактичне значення, $\hat{y}_i$ - прогнозоване значення, n - кількість спостережень.

Середньоквадратична похибка (MSE) є однією з найбільш поширених метрик для оцінки якості регресійних моделей. Вона відображає середнє значення квадрату різниці між прогнозованими та фактичними значеннями цільової змінної. MSE чутлива до викидів, оскільки вони зводяться в квадрат, що може значно збільшити значення метрики.

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{2}$$

де:

- n - кількість спостережень;
- y_i - фактичне значення i-го спостереження;
- $\hat{y}_i$ - прогнозоване значення i-го спостереження.

В процесі символічної регресії (SR) початкова популяція розв'язків формується випадковим чином з нескінченного пулу операторів та терміналів. Розмір популяції задається користувачем. Кожен окремий розв'язок може характеризуватися різною мірою придатності, впливаючи на загальну ефективність. Алгоритм SR постійно оцінює якість кожного розв'язку, відбираючи для подальшої еволюції найбільш перспективні варіанти, що демонструють мінімальну похибку, та відкидаючи неефективні.В алгоритмі

символічної регресії (SR) еволюційний процес характеризується ітеративним зменшенням середньої помилки за рахунок елімінації розв'язків з низькою ефективністю. Завершення алгоритму SR відбувається при виконанні однієї з наступних умов: досягнення максимальної кількості ітерацій, заданої користувачем; досягнення оптимального розв'язку з нульовою помилкою; досягнення прийнятного рівня помилки, встановленого користувачем (наприклад, 0,01). Зазвичай використовується комбінація цих критеріїв. Важливо зазначити, що SR може генерувати не тільки окреме рівняння, але й набір рішень, що відрізняються за складністю. Символічна регресія є ефективним методом, який дозволяє значно скоротити час і зусилля, необхідні для моделювання даних. Завдяки автоматизації процесу пошуку функцій, SR усуває необхідність ручного підбору функціональних форм, що є трудомістким і часто неточним процесом. Це особливо важливо в таких галузях, як фізика високих енергій, де дані можуть мати складні форми, і традиційні методи моделювання можуть бути недостатньо ефективними.

```
    model = PySRRegressor(
        model_selection="best",
        # model_selection="accuracy",
        # elementwise_loss = "L1DistLoss()",
        # loss="L1DistLoss()",
        populations = 3*cpu_count(),
        population_size = 1000,
        ncycles_per_iteration=500,
        # population_size = 300,
        # ncycles_per_iteration=3000,

        # niterations = 100,
        niterations = 500000,

        timeout_in_seconds=2 * 60 * 60,
        warm_start=True,
        early_stop_condition=("stop_if(loss, complexity) = loss < 1e-8 && complexity < 10"), # Stop early if we find

        maxsize = 30,
        # maxdepth= 7,

        # binary_operators=["*", "+", "/", "pow",#"-",
        #                   "Zved(x,y)=x*y/(x+y)"
        #                  ],
        # unary_operators=["exp", "log", "tanh", "sin","neg",
        #                  "PowInv(x)=1.0f0/(1.0f0+x)"],
        complexity_of_constants = 1,
        complexity_of_variables = 1,

        complexity_of_operators={"tanh": 3, "sin":3},
        constraints={
            "pow": (-1, 5),
            "exp": 4,
            },
        # extra_sympy_mappings={
        #      "PowInv": lambda x: 1/(1+x),
        #      "Zved": lambda x,y: x*y/(x+y)
        # },
        # nested_constraints={
        #     "sin": {"sin": 0},
        #     "tanh": {"tanh": 0},
        #     "PowInv": {"PowInv": 0, "tanh": 0,},
        #     "Zved": {"Zved": 0, "tanh": 0,},
        #                 },

        parsimony=1e-5 ,
        # weight_optimize = 0.001,
        adaptive_parsimony_scaling = 100 ,
        # warmup_maxsize_by = 0.1,

        verbosity = 1,
        print_precision = 3,
# ----------------------------------------------
        # batching = True,
        # batch_size = 8,
# ----------------------------------------------------
        # temp_equation_file = True,
        # temp_equation_file = False,
        # tempdir = "temp",
        # delete_tempfiles = False,
        # delete_tempfiles = True,



        procs= cpu_count()
```

Рис. 2: Основний код моделі

В ході багатьох спроб підбору найкращих параметрів, було прийнято метод оцінки моделі `"model selection = "best"` — цей метод оцінки моделі оптимізує фінальний результат навчання, між складністю моделі та середньою квадратичною помилкою. Тобто не дає "дереву"розроститися до неінтерпритованих варіантів і при цьому шукає максимально якісну апроксимацію даних. `"populations = 3 * cpu count()"` — Кількість окремих популяцій у стилі "island model"— кожна популяція еволюціонує незалежно, іноді обмінюючись найкращими формулами. Параметр покращує різноманіття рішень, зменшує ймовірність застрягнути в локальному мінімумі. Проте дуже залежить від числа процесорів, бо можна паралелити. `"population size = 1000"` — Кількість формул

в одній популяції. Працює за доволі простим принципом: більше = більше варіантів, але й довше обчислення. В описі цієї бібліотеки автор рекомендує значення від 100 до 1000, залежно від задачі й ресурсів. Параметр `"ncycles per iteration = 500"` це скільки еволюційних циклів робиться всередині кожної ітерації (до оновлення результату). Також модель можна контролювати, а саме скільки максимальних ітерацій всього процесу буде проведено, можна обмежувати модель за кількістю ітерацій, за цей метод відповідає параметр `"niterations"`, проте в нього є великий недолік, що модель може застрягти в локальному мінімумі і процес просто заступориться, проте не закінчиться до кінця ітерацій. Звісно, можна поставити обмеження по похибці, проте дуже важко в реальному часі спрогнозувати, як поведе модель до моменту поки не побачиш результати. Тому було вирішено обмежувати модель по часу, тоді ми не ризикуємо витрачати час та ресурси комп'ютера в "холосту". Так як під час навчання автоматично створюється файл, за допомогою якого можна запустити модель на місці, на якому вона закінчила навчання по закінченню часу, за цей метод відповідає параметр `"timeout in seconds"`. В ході роботи було виявлено, що на один пакет даних, модель знаходить найкращі формули за 5-6 годин. `"warm start"` — Якщо вже був попередній запуск і є збережені тимчасові дані, алгоритм продовжить з того місця. Також ми можемо задати максимальний розмір формули, що генерується, задавши параметр `"maxsize"`. Розмір формули рахується за кількістю всіх елементів у рівнянні (оператори + змінні + константи). Параметри `"complexity of constants"`, `"complexity of variables"` та `"complexity of operators"` відповідають як раз за ціну одного елементу в рівнянні: константи, змінні та оператори (повноцінні функції як "sin"та "cos") відповідно. Також, аби фінальна формула не використовувала лише один з якихось операторів, використовується метод `"constraints"` — він обмежує можливу кількість користування операторами. `"parsimony"` — ще один з операторів, що допомагає уникати занадто складних формул, навіть якщо вони точні, балансує між точністю й простотою. Проте є можливість більш продвинуто впливати на параметр `"parsimony"` прямо під час роботи програми, це метод `"adaptive parsimony scaling"` — він автоматично масштабує вплив `parsimony` залежно від значень втрат (loss). Це особливо корисно, коли у нас loss змінюється в широкому діапазоні — підтримує баланс між loss і complexity. `"verbosity"` — це параметр, що відповідає за рівень виводу в консоль, 1 = базова інформація (поточний лідер, ітерації тощо) чи 0 = тихий режим. `"procs"` — Скільки процесів запускати паралельно. В ході налаштування моделі для оптимізації важливими параметрами стали `Binary_operators` та `Unary_operators`. Перші визначають, які бінарні операції можуть бути використані в процесі формування моделей. Це дозволяє регулювати складність рівнянь, які можна отримати, а також зберігати контроль над точністю та інтерпретованістю кінцевої формули. Операторів може бути кілька, і їх використання в кожному окремому випадку має критичне значення для балансу між ефективністю та складністю.

`Unary_operators`, в свою чергу, обмежують можливості роботи з унарними операціями, що дає можливість точніше контролювати математичні трансформації і спрощувати процес знаходження оптимальних рішень. Вони допомагають моделі уникати надмірної складності в процесі побудови формул.

Параметр `Extra_sympy_mappings` додає додаткові відображення в процес розв'язування задачі, що може бути корисним при роботі з складними виразами, де є потреба в інтеграції нестандартних математичних функцій, що не входять до стандартного набору.

`Nested_constraints` служать для введення додаткових обмежень на внутрішню структуру моделей. Цей параметр дозволяє розширювати можливості моделі, не порушуючи загальних правил побудови, таким чином, забезпечуючи більшу гнучкість і можливість оптимізації для складних систем, де потрібен контроль над кожним окре-

мим рівнянням і його складовими. Метод `Extra_sympy_mappings` дозволяє створювати унікальні оператори, які можна використовувати під час побудови математичних моделей. Цей параметр дає можливість додавати кастомізовані функції, що можуть бути зручними для вирішення конкретних задач, де стандартні оператори не дають достатньо гнучкості або точності.

Наприклад, у нашій реалізації додано два унікальні оператори:

- `PowInv`: функція, що обчислює обернену залежність від виразу $1 + x$, тобто:

  $$\texttt{PowInv}(x) = \frac{1}{1+x}$$

  Цей оператор може бути корисним при моделюванні функцій, де існує потреба в оберненій залежності від змінної, що часто зустрічається в економіці, фізиці та біології.

- `Zved`: функція, що обчислює середнє гармонійне двох чисел x і y, тобто:

  $$\texttt{Zved}(x, y) = \frac{x \cdot y}{x+y}$$

  Цей оператор використовується, коли необхідно працювати з величинами, що мають взаємозв'язок, який можна описати через середнє гармонійне, наприклад, у випадках з парними чи зваженими обчисленнями в задачах з теорії ймовірностей або статистики.

Ці оператори додаються до моделі і дозволяють збільшити її точність та гнучкість, зокрема при роботі з нестандартними функціями чи специфічними залежностями в даних. Завдяки такому підходу можна значно розширити можливості математичних моделей і знаходити більш точні та ефективні рішення для різних задач. Метод `Nested_constraints` дозволяє вводити додаткові обмеження на використання операцій у формулюванні математичних моделей. Це дає змогу контролювати, які саме оператори можуть бути використані разом у одній формулі, тим самим запобігаючи надмірній складності чи непотрібним комбінаціям операторів, що можуть погіршити точність чи інтерпретованість моделі.

У нашому випадку були додані наступні обмеження:

- Для оператора `tanh`: обмеження на використання іншого `tanh` у тій самій формулі. Тобто, оператор `tanh` не може бути комбінований з собою у жодній з формул, що допомагає уникнути надмірної складності та багатократних використань цього оператора.

- Для оператора `PowInv`: обмеження на використання як оператора `PowInv`, так і оператора `tanh` у тій самій формулі. Це забезпечує, що жодна з формул не буде містити одночасно ці два оператора, що дозволяє підтримувати певну рівновагу між різними видами трансформацій в моделі та запобігає занадто складним комбінаціям.

- Для оператора `Zved`: обмеження на використання як оператора `Zved`, так і оператора `tanh` у тій самій формулі. Це обмеження має схожий ефект, як і для `PowInv`, обмежуючи комбінації цих двох операторів для забезпечення більш простих та інтерпретованих моделей.

Ці обмеження дозволяють моделі працювати більш ефективно, підтримуючи баланс між складністю та точністю, що є особливо важливим при роботі з великими обсягами даних або складними математичними моделями.

3 Застосування символічної регресії у різних фізичних задачах

У цьому розділі більш детально розглянеться приклади використання SR та порівняння результатів з іншими методами ML та AI .

3.1 Розрахунок швидкості потоку в розріджених середовищах

Вивчення потоків розрідженого газу у циліндричних трубах є ключовим для низки інженерних застосувань — від мікрофлюїдних пристроїв до вакуумних систем і термоядерних реакторів. У таких задачах класичні методи, що базуються на кінетичній теорії, забезпечують високу точність, проте вимагають значних обчислювальних ресурсів. Тому зростає інтерес до методів машинного навчання, здатних швидко та точно відтворити поведінку системи на основі вже обчислених даних.

У статті було продемонстровано ефективність використання методів випадкового лісу та символічної регресії для апроксимації зменшеної витрати маси газу W як функції трьох безрозмірних параметрів: відношення тисків $p = P_2/P_1$, відносної довжини труби $l = L/R$ та параметра розрідженості δ, що пов'язаний із числом Кнудсена.

У рамках дослідження, на основі згенерованого кінетичного набору даних було побудовано кілька аналітичних формул для $W(p, l, \delta)$ за допомогою символічної регресії. Зокрема, було отримано загальну апроксимацію, що охоплює весь діапазон параметрів, з максимальною абсолютною похибкою менше 17%. Для підвищення точності дані були розділені на три підгрупи відповідно до значень δ, для яких побудовано окремі формули, що досягли похибки не більше 9%.

Особливістю методу є можливість отримання саме аналітичного вигляду функцій, що, на відміну від «чорних скринь» типу нейронних мереж, дозволяє не тільки прогнозувати, а й аналізувати фізичний зміст процесів. При цьому, формули не потребують великої кількості параметрів або ітеративних процедур, як це часто буває у напівемпіричних підходах.

У випадку $\delta \to 0$ та $l \to 0$, тобто у вільномолекулярному режимі, всі моделі, побудовані методом символічної регресії, коректно наближаються до теоретично відомої залежності $W = 1 - p$. Це підтверджує фізичну обґрунтованість отриманих виразів.

Таким чином, використання символічної регресії для задач цього типу дозволяє суттєво знизити обчислювальні витрати, зберігаючи при цьому точність і фізичну інтерпретованість моделі. Подібний підхід може бути легко адаптований для інших задач кінетичної теорії, таких як моделювання взаємодії частинок у газових сумішах або вивчення ефектів на границях розділу.

3.2 Аналітичне моделювання сонічних кристалів зі змінним радіусом

Сонічні кристали — це періодичні акустичні структури, здатні змінювати характеристики поширення звукових хвиль завдяки своєму геометричному або матеріальному профілю. Незважаючи на більш ніж тридцятирічну історію вивчення, кількість точних аналітичних виразів для таких структур залишається обмеженою лише найпростішими випадками. Це значно ускладнює практичне проєктування систем з заданими резонансними або забороненими зонами.

В основі задачі лежить так зване рівняння Вебстера — модифікація хвильового рівняння, що описує поширення квазіплощинних хвиль у хвилеводах із змінним поперечним перерізом. Якщо форма хвилеводу змінюється періодично, то на цю структуру мо-

жна застосувати теорію Флоке-Блоха, що дозволяє обчислити дисперсійну залежність — тобто залежність хвильового числа від частоти. Вона є ключовою для виявлення зон заборонених частот (bandgaps), в яких хвилі не можуть поширюватися.

У дослідженні [[?]] запропоновано новий підхід до побудови компактних аналітичних формул, що пов'язують геометрію періодичної комірки з характеристиками дисперсії. Замість прямого чисельного розв'язку рівнянь було згенеровано великий набір даних, які потім проаналізовано за допомогою методів фізично-обґрунтованого машинного навчання, зокрема — символічної регресії.

Геометрія одиничної комірки хвилеводу параметризована за допомогою кубічного сплайну з чотирма керуючими точками, що забезпечує достатню гнучкість при одночасному збереженні гладкості та симетрії. Основними змінними стали чотири безрозмірні параметри: зміщення і рознесення по координаті x, а також зміщення і різниця по радіусу r. Ці параметри дозволяють компактно описувати форму комірки без втрати загальності.

Було отримано формули, які з високою точністю апроксимують як центри, так і ширини перших трьох зон заборони, а також — повну дисперсійну залежність. Для прикладу, центр першої зони заборони описується виразом:

$$m_1 = \pi + 0.38 \cdot \max(0.17, x_d r_d),$$

а ширина цієї зони:

$$w_1 = 2.61 \cdot [\max(r_m, r_d) - \min(0, r_m)] .$$

Ці формули є не лише зручними для швидкої оцінки параметрів системи, але й демонструють хорошу узгодженість із фізичними принципами: наприклад, залежність ширини першої зони переважно від об'єму одиничної комірки, а не її довжини. Такий підхід відкриває можливості для інженерного проектування акустичних метаструктур без необхідності постійного використання чисельних симуляцій.

Крім того, було отримано й узагальнене рівняння для передбачення повної дисперсійної залежності у вигляді:

$$\cos\xi = b_0(\kappa) + b_1(\kappa)c_1 + b_2(\kappa)c_2 + b_3(\kappa)c_3,$$

де $b_i(\kappa)$ — функції хвильового числа, а c_i — функції геометричних параметрів. Така декомпозиція дозволила відокремити залежність від частоти та геометрії, що підвищує гнучкість моделі.

Усі отримані формули продемонстрували високу точність як на навчальному, так і на тестовому наборі даних. Важливо, що побудовані залежності не тільки описують поведінку системи, але й мають зрозумілу фізичну інтерпретацію. Наприклад, у випадку хвилеводу з постійним радіусом модель повертається до стандартної залежності $\cos\xi = \cos\kappa$, що відповідає неперервному середовищу.

Таким чином, використання символічної регресії в задачах акустики дозволяє будувати прозорі й ефективні моделі для складних періодичних структур. Це відкриває перспективи для розв'язання аналогічних задач у фотоніці, механіці, електродинаміці та інших галузях, де важливими є дисперсійні властивості середовища.

3.3 Інтерпретоване машинне навчання для задач фонового віднімання в адронних струменях

У фізиці важких іонних зіткнень струмені (джети), що виникають унаслідок жорсткого розсіювання кварків і глюонів, є чутливими зондами до властивостей кварк-глюонної

плазми. Проте точність таких вимірювань значно обмежується флуктуаційним фоном, утвореним м'якими частинками з гідродинамічним походженням. Традиційні методи віднімання фону, зокрема area-based підхід, демонструють високу варіативність результатів, особливо в області малих імпульсів p_T.

У дослідженні [?] вперше було застосовано методи інтерпретованого машинного навчання для задач реконструкції струменевого імпульсу, зокрема поєднання глибокої нейронної мережі та символічної регресії. Після навчання моделі нейромережі на даних згенерованих за допомогою симуляцій PYTHIA + TENNGEN, автори використали бібліотеку PySR для отримання аналітичного виразу, що апроксимує поведінку нейромережі:

$$p_{T,\text{corr}} = p_{T,\text{tot}} - C_1 \cdot (N_{\text{tot}} - C_2),$$

де $p_{T,\text{tot}}$ — повний імпульс джета, N_{tot} — загальна кількість частинок у струмені, а C_1, C_2 — коефіцієнти, які підбирає PySR. Отримана формула виявилась тісно пов'язаною з запропонованим авторами **мультиплікативним методом**, в якому корекція імпульсу базується на числі частинок, а не на площі струменя.

На відміну від area-методу:

$$p_{T,\text{corr}} = p_{T,\text{tot}} - \rho A,$$

де ρ — густина імпульсу на одиницю площі, а A — площа струменя, новий підхід значно знижує вплив флуктуацій, спричинених гідродинамічними ефектами та статистикою числа частинок. Згідно з формулою стандартного відхилення залишку імпульсу δp_T, останні два члени повністю усуваються при використанні множинного підходу:

$$\sigma_{\delta p_T} = \sqrt{N \cdot \sigma^2_{p_T} + \text{флуктуації від } v_n},$$

де N — кількість частинок, v_n — коефіцієнти азимутальної анізотропії, що виникають внаслідок колективного потоку.

Окрім того, застосування символічної регресії дозволило авторам не лише зменшити модельну залежність від нейромережі, а й надати отриманим виразам фізичну інтерпретацію. Коєфіцієнти C_1 та C_2 були порівняні з середніми значеннями параметрів ρ_{Mult} та N_{signal} з мультиплікативного методу, і виявилися узгодженими, особливо для колізій типу Au+Au на RHIC.

Цей підхід, що поєднує гнучкість глибокого навчання із прозорістю символічного моделювання, є яскравим прикладом того, як машинне навчання може не лише замінити, але й розширити класичні методи фізичного аналізу, зберігаючи при цьому інтерпретованість та обґрунтованість отриманих результатів.

3.4 Масштабне співвідношення для чорних дір у спіральних галактиках

Вивчення зв'язку між масою надмасивної чорної діри (SMBH) та структурно-кінематичними властивостями її галактики-хазяїна залишається одним із ключових напрямків сучасної астрофізики. Вже понад два десятиліття науковці виявляють та аналізують так звані *масштабні співвідношення* — емпіричні залежності, які дозволяють оцінити масу чорної діри на основі більш доступних до спостережень параметрів, таких як світність або швидкість обертання галактики. Проте більшість таких залежностей базується на масивних еліптичних галактиках, де спостереження є простішими. Це обмежує точність екстраполяції до легших, пізнього типу спіральних галактик, де очікується наявність проміжних за масою чорних дір (IMBHs).

У роботі Davis & Jin (2023) запропоновано нову, трипараметричну залежність маси чорної діри від кутового параметра спіралі ϕ (так званий pitch angle) та максимальної швидкості обертання диска $v_{\max}$. Автори вперше використали комбінацію цих двох параметрів, характерних саме для спіральних галактик, щоб отримати *фундаментальну площину* — узагальнене співвідношення, яке значно зменшує статистичне розсіювання в оцінках.

Залежність була побудована за допомогою методів машинного навчання. На першому етапі використовувалася символічна регресія (пакет PySR), яка автоматично шукала математичний вираз, що найкраще описує дані. Далі для уточнення коефіцієнтів та врахування похибок вимірювань було застосовано метод Hyper-Fit. Остаточна формула для логарифма маси чорної діри $M_\bullet$ (у сонячних масах) має вигляд:

$$M_\bullet = \alpha \cdot (\tan|\phi| - 0.24) + \beta \cdot \log\left(\frac{v_{\max}}{211\ \text{км/с}}\right) + \gamma,$$

де знайдені коефіцієнти: $\alpha = -5.58 \pm 0.06$, $\beta = 3.96 \pm 0.06$, $\gamma = 7.33 \pm 0.05$, а внутрішнє розсіювання $\sigma = 0.22 \pm 0.06$ dex — одне з найменших серед усіх відомих масштабних співвідношень.

Фізичне тло цієї залежності цілком узгоджується з попередніми спостереженнями: масивні чорні діри трапляються в галактиках з щільно закрученими спіральними рукавами (малий ϕ) та високою швидкістю обертання, тоді як легкі — в об'єктах із розгорнутими рукавами та повільним обертанням. Цей зв'язок підкреслює глибоку взаємодію між SMBH та глобальною структурою галактики, на противагу традиційним підходам, які акцентувалися на центральних (балджевих) властивостях.

Особливу увагу автори звертають на можливість використання цієї залежності для пошуку IMBH — об'єктів із масою 10^3–10^5 $M_\odot$, що залишаються поза зоною надійного динамічного вимірювання. За допомогою отриманої моделі було визначено умови, за яких маса чорної діри ймовірно менша за 10^5 $M_\odot$: $|\phi| > 26.8°$ та $v_{\max} < 130$ км/с. Таким чином, модель дозволяє визначити кандидатів у IMBH-хости вже на основі архівних даних.

Окрім цього, нове співвідношення є надзвичайно практичним: pitch angle легко оцінити навіть з немаркованих зображень, а дані про $v_{\max}$ доступні в численних базах радіоспостережень. Це робить модель універсальним інструментом для великих оглядів, зокрема — на високих червоних зміщеннях, де інші методи обмежені.

Отримане тривимірне співвідношення між $M_\bullet$, ϕ та $v_{\max}$ значно випереджає попередні моделі як за точністю, так і за інтерпретованістю. З урахуванням доступності необхідних параметрів, воно відкриває нові можливості в дослідженні еволюції галактик, статистики чорних дір і потенційного відкриття довгоочікуваних IMBH.

Література

[1] Davis, B. L., & Jin, K. (2023). A Fundamental Plane for Spiral Galaxies Hosting Supermassive Black Holes. *The Astrophysical Journal Letters*, **949**(2), L41. https://doi.org/10.3847/2041-8213/acdc64

[2] Pichugin, K., Fleming, A. J., & Maksymov, I. S. (2021). Symbolic regression of band gaps and dispersion relations of axisymmetric sonic crystals. *Journal of Sound and Vibration*, **597**, 118821. https://doi.org/10.1016/j.jsv.2021.118821

[3] Cranmer, M., Sanchez-Gonzalez, A., Battaglia, P., Xu, R., Cranmer, K., & Greydanus, S. (2020). PySR: Fast & Interpretable Symbolic Regression in Python and Julia. Available at: https://github.com/MilesCranmer/PySR (Accessed April 2025)