політики в галузі гігієни праці та соціального захисту працюючого населення [1-10].

Список літератури

1. Державна служба статистики України.// http://www.ukrstat.gov.ua
2. «Надопомогуспеціалістузохоронипраці»: Наук. - виробн. журнал. К.: ДП «Редакція журналу «Охорона праці» . - 2007-2015. - №№1-12.
3. **Риженко С.А., Лисий А.Ю., Капшук В.Г., Грузін І.І., Ткач Л.А.** Особливості професійної захворюваності опорно-рухового апарату робочих промислових підприємств Кривбасу. Матеріали науково-практичної конференції з нагоди 85-річчя кафедри гігієни праці і професійних хвороб НМУ ім.. О.О. Богомольця та 120-річчя від дня народження професора В.Я. Підгаєцького «Пріоритетні проблеми гігієни праці, професійної та виробничо-зумовленої захворюваності в Україні» . Київ, 2008.
4. **Риженко С.А., Лисий А.Ю., Грузін І.І., Погорєлова Л.О., Слюта Т.В., Ткач Л.А., Громик Т.М.** До питання оптимізації моніторингу шкідливих речовин в виробничих приміщеннях промислових підприємств Кривбасу. Сборникматериалов 12-й итоговойрегиональнойконференции. Эпидемиология, экология и гигиена. Харьков, 2009.
5. **Глембоцька А.** Своєчасне запобігання профзахворюванням у сучасних реаліях. СЕС.Профілактична медицина, Київ, № 2, 2011.
6. **Ткач Л.А.** Проблемні питання професійної захворюваності працівників промислових підприємств Кривбасу: Медицинапрацітапрофпатології. - Кривий Ріг.
7.http://cyberleninka.ru/article/n/analiz-sostoyaniya-professionalnoy-zabolevaemosti-i-proizvodstvennogo-travmatizma-gornometallurgicheskogo-kompleksa#ixzz3z8Y3tXOg
8. Environment, Health and Safety Committee OCCUPATIONAL HEALTH AND SAFETY MANAGEMENT SYSTEMS http://www.rsc.org/images/Occupational-Health-and-Safety-Management-Systems_tcm18-240421.pdf
9.http://www.hse.gov.uk/statistics/overall/hssh1415.pdf
10. http://dnop.kiev.ua/web/index.php?option=com_content&task=view&id=6387&Itemid=137

Рукопис подано до редакції 13.04.2018

УДК 519.6:004.8

Н.Н. ШАПОВАЛОВА, О.Г. РИБАЛЬЧЕНКО, ст. викладачі,
Д.І. КУРОПЯТНИК, студент, Криворізький національний університет

ПОРІВНЯЛЬНИЙ АНАЛІЗ МЕТОДІВ ОПТИМІЗАЦІЇ ФУНКЦІОНАЛУ ЯКОСТІ МОДЕЛЕЙ МАШИННОГО НАВЧАННЯ

Мета. Визначити ефективність методів оптимізації функціоналу якості моделей машинного навчання в залежності від виду критерію оцінки якості алгоритму, розміру навчальної вибірки, порівняти методи за критеріями стабільності отримання рішення та обчислювальної складності, розробити рекомендації по застосуванню розглянутих методів за певних початкових умов задачі багатовимірної оптимізації. Вибір методу оптимізації оціночного функціоналу на етапі формування математичної моделі є важливим фактором ефективності побудованої алгоритму машинного навчання, зокрема за умов багатофакторних цільових функцій та великих обсягів навчальної вибірки. Загальноприйнятні в практиці машинного навчання методи оптимізації функціоналу якості не завжди враховують вид цільової функції, що призводить до значного зростання часу навчання моделі та зниження її якості в цілому.

Методи. Використано числовий експеримент навчання регресійних моделей і системний аналіз методів пошуку оптимальних значень параметрів критерію якості задач класу навчання по прецедентах: градієнтного спуску, симплекс-методу Нелдера-Міда, імітації відпалу, генетичного алгоритму.

Наукова новизна. Проведено порівняння ефективності методів багатовимірної оптимізації та аналіз доцільності їх застосування до різних типів оптимізаційних функцій в машинному навчанні на різних обсягах навчальної вибірки.

Практична значимість виконаної роботи полягає в обґрунтуванні застосування того чи іншого оптимізаційного методу в залежності від виду оціночного функціоналу якості та розміру простору ознак задачі машинного навчання, визначенні обчислювальної складності застосованих алгоритмів. Вибір методу оптимізації на етапі постановки задачі значно підвищує ефективність моделі машинного навчання.

Результати. Розроблено бібліотеку opti_methods методів багатовимірної оптимізації оціночного функціоналу якості моделей задач машинного навчання для мови програмування Python 3, розроблено рекомендації щодо використання певного оптимізаційного метода в залежності від виду критерію якості навчання моделей та розміру навчальної вибірки.

Ключові слова: оптимізація, машинне навчання, функціонал якості, обчислювальна складність, метод градієнтного спуску, генетичний алгоритм, алгоритм імітації відпалу, метод Нелдера-Міда.

doi: 10.31721/2306-5451-2018-1-46-104-112

Проблема та її зв'язок з науковими і практичними задачами. Машинне навчання (МН) займає все більше місце в нашому житті з огляду на величезний спектр його застосування. Згідно недавнього звіту дослідницької компанії Gartner, яка регулярно оновлює свій «цикл зрілості технологій», на сьогоднішній день з усіх інформаційних технологій на піку очікувань знаходиться саме МН [1].

З кожним днем сфера застосування МН розширюється. Повсюдна інформатизація призводить до накопичення величезних обсягів даних у науці, виробництві, бізнесі, транспорті, охороні здоров'я. Дані стають одним з найцінніших ресурсів сьогодення. Оперуючи достатньо великими обсягами систематизованої інформації і значними обчислювальними потужностями, люди навчились виявляти закономірності, приховану структуру даних, що дозволило робити достовірні прогнози в багатьох сферах людської діяльності. Виникаючі при цьому задачі прогнозування, управління та прийняття рішень часто зводяться до навчання за прецедентами. Раніше, коли таких даних не було, або вони не були систематизовані належним чином, такі задачі або взагалі не ставилися, або вирішувалися зовсім іншими методами.

Проблема визначення найбільш придатних, гнучких алгоритмів МН, що за умов впливу зовнішніх параметрів не вимагатимуть перегляду або заміни всієї моделі, стає дедалі актуальнішою, а вибір методів оптимізації функціоналу якості вимагає ґрунтованого підходу.

В статті розглянуто чотири види оптимізаційних методів: метод Нелдера-Міда, градієнтного спуску, генетичний алгоритм і алгоритм імітації відпалу. Порівняльний аналіз методів у залежності від виду оптимізаційної функції, розміру і ступеню збалансованості вибірки, умов збіжності тощо, дозволить сформулювати рекомендації щодо доцільності використання кожного з розглянутих варіантів із урахуванням вищезазначених умов.

Аналіз досліджень і публікацій. Найбільш поширений тип задач МН – це задачі навчання з учителем [2, 3]. Кожен прецедент являє собою пару «об'єкт – відповідь». Потрібно знайти функціональну залежність відповідей від описів об'єктів і побудувати алгоритм, який бере на вході опис об'єкта і видає на виході відповідь. До цього типу належать, зокрема, задачі класифікації і регресії.

В задачі класифікації безліч допустимих відповідей визначено. Їх називають мітками класів. Клас – це безліч всіх об'єктів із даним значенням мітки. Задача регресії відрізняється тим, що допустимою відповіддю є дійсне число або числовий вектор.

Для вирішення задачі навчання за прецедентами в першу чергу фіксується модель відновлюваної залежності. Потім вводиться функціонал якості, значення якого показує, наскільки адекватно модель описує спостережувані дані. Функціонал якості зазвичай визначається як середня помилка відповідей, виданих алгоритмом, за всіма об'єктами вибірки. Алгоритм навчання шукає такий набір параметрів моделі, при якому функціонал якості на заданій навчальній вибірці приймає оптимальне значення.

Процес настройки моделі за вибіркою даних у більшості випадків зводиться до застосування методів оптимізації. Методи математичного програмування дають велику різноманітність алгоритмів розв’язання даного завдання. В машинному навчанні традиційно використовують градієнтні методи, але за умови, що функція помилки моделі гладка і диференційована [4, 5]. Окрім того, існують недоліки цієї групи методів:

алгоритм може не сходитися чи сходитися занадто повільно;

як правило, функціонал якості багатоекстремальний і процес градієнтного спуску може «застрягти» в одному з локальних мінімумів;

при великій розмірності простору ознак і/або малій довжині вибірки можливе перенавчання, тобто класифікація стає нестійкою, і ймовірність помилки збільшується;

якщо функція активації має горизонтальні асимптоти, то процес може потрапити в стан «паралічу» [6].

Існуючі способи усунення недоліків призводять або до збільшення кількості ітерацій, або до введення в модель гіперпараметрів, підбір яких породжує нову оптимізаційну задачу [2].

Генетичні алгоритми представляють собою відносно новий напрямок в обчислювальних методах [7, 8]. В останні роки з'явилося безліч публікацій, присвячених опису принципів побудови генетичних алгоритмів, заснованих на концепціях природного відбору і генетики. При цьому дуже часто їх можливості демонструються на прикладі вирішення завдань оптимізації та пошуку. Генетичний алгоритм має переваги перед іншими алгоритмами в умовах дуже великих

розмірностей задач і відсутності впорядкованості вихідних даних, альтернативою йому може стати лише метод повного перебору варіантів.

Зауважимо, що і генетичні алгоритми, і градієнтні методи мають тенденцію сходитися до локального оптимуму або навіть до довільної точки, а не до глобального оптимуму. Це означає, що вони «не знають», яким чином пожертвувати короткочасною високою придатністю для досягнення довгострокової придатності [9, 10]. Тому в порівняльному аналізі буде використаний алгоритм імітації відпалу, який є універсальним методом пошуку глобального мінімуму цільової функції. Перевагами методу є: можливість пошуку рішень для складних нелінійних задач, можливість роботи з даними, що мають велику кількість шумів і перешкод, здатність виходу з локальних мінімумів, універсальність методу, відносна легкість модифікації, адаптації та технічної реалізації [11].

Разом із вищезазначеними алгоритмами буде розглянутий дуже ефективний і простий алгоритм пошуку екстремуму функції багатьох змінних, який не накладає обмежень на гладкість функції та не використовує похідну – симплекс-метод Нелдера-Міда.

Порівняння декількох підходів до оптимізації функціоналу якості задач машинного навчання представляє науковий і практичний інтерес.

Постановка завдання. Існує проблема оптимізації параметрів $w_1, ..., w_N$ деякого алгоритму машинного навчання. Завдання оптимізації полягає в тому, щоб підібрати ці параметри таким чином, щоб алгоритм давав найкращий результат. Зокрема, якщо якість роботи алгоритму описувати функцією якості $Q(w_1, ..., w_N)$ від його параметрів – вагових коефіцієнтів моделі, то задача оптимізації набуває вигляду $Q(\alpha_1, ..., \alpha_N) \to \max\limits_{\alpha_1,...,\alpha_N}$. Розглянемо задачу навчання за прецедентами – задачу лінійної регресії

$$a(x) = w_0 + \sum_{j=1}^{d} w_j x^j,$$

де w_0 – вільний коефіцієнт, x – ознаки, w_j – вага x^j-ї ознаки, d – кількість ознак у вибірці.

Якщо додати (d+1)-у ознаку, яка на кожному об'єкті приймає значення 1, то лінійний алгоритм можна буде записати у більш компактному вигляді

$$a(x) = \sum_{j=1}^{d+1} w_j x^j = \langle w, x \rangle,$$

де використовується позначення $\langle w, x \rangle$ для скалярного добутку двох векторів.

Якість алгоритму оцінюється тим, наскільки точно отримана модель описує залежності даних у вибірці, тобто чим менша помилка (відхилення) на кожному об’єкті, тим вище якість алгоритму. В якості міри помилки не може бути вибрано відхилення від прогнозу (y) $Q(a, y) = a(x) - y$, оскільки в цьому випадку мінімум функціоналу не буде досягнутий при правильній відповіді $a(x) = y$. Найпростіший спосіб – розрахувати модуль відхилення $|a(x) - y|$. Тоді функціонал якості для регресійної моделі набуває вигляду

$$Q(a, x) = \frac{1}{l} \sum_{i=1}^{l} |a(x_i) - y_i|.$$

Запишемо цей функціонал у вигляді функції від вектора вагових коефіцієнтів

$$Q(w, x) = \frac{1}{l} \sum_{i=1}^{l} |\langle w, x_i \rangle - y_i|.$$

Необхідно підібрати вагові коефіцієнти w таким чином, щоб помилка алгоритму була найменшою

$$Q(w, x) = \frac{1}{l} \sum_{i=1}^{l} |\langle w, x_i \rangle - y_i| \to \min_w. \quad (1)$$

Оскільки функція (1) негладка, то використання градієнтних методів для оптимізації функціоналу якості, заданого таким чином, стає неможливим. Тому доцільно використовувати не модуль відхилення $|a(x) - y|$, а квадрат відхилення прогнозу $(a(x) - y)^2$. Запишемо функцію якості як середньоквадратичний критерій відхилення аналогічним чином через вектор вагових коефіцієнтів

$$Q(w, x) = \frac{1}{l} \sum_{i=1}^{l} (\langle w, x_i \rangle - y_i)^2 \to \min_w. \quad (2)$$

Тепер цільова функція (2) має неперервну похідну на всій множині визначення, і її можливо мінімізувати градієнтними методами.

Для порівняльного аналізу методів оптимізації кожного з видів функціоналів якості моделі навчання застосуємо стохастичні методи (генетичний та імітації відпалу), градієнтний та метод Нелдера-Міда на різних розмірах навчальної вибірки: у малій вибірці кількість об'єктів менше 100, у середній – до 5000, у великій – до 25000.

Викладення матеріалу та результати. Використання *градієнтних методів* зумовлено їх високою швидкістю збіжності і стабільністю, одержаних завдяки інформації про градієнт і кривизну досліджуваної функції. Алгоритм найшвидшого спуску реалізує ітераційну процедуру руху до мінімуму з довільно вибраної початкової точки в напрямку найсильнішого зменшення функції, визначеному біля поточного значення аргументу функції, що мінімізується. Такий напрям протилежний напряму, який задається вектором градієнта $\nabla f(x)$ функції, що мінімізується $f(x)$. Загальна формула для знаходження значення аргументу $x^{(k+1)}$ за значенням $x^{(k)}$, знайденому на k-му кроці роботи алгоритму найшвидшого спуску $x^{(k+1)} = x^{(k)} + \lambda^{(k)} \cdot s^{(k)}$, де $\lambda^{(k)}$ – крок градієнтної процедури, $s^{(k)}$ – вектор одиничної довжини в напрямку, протилежному напрямку градієнта $\nabla f(x^{(k)})$, визначеному в точці $x^{(k)}$

$$s^{(k)} = \frac{-\nabla f\left(x^{(k)}\right)}{\left\|\nabla f\left(x^{(k)}\right)\right\|},$$

де $\|\nabla f(x^{(k)})\|$ – норма вектора градієнта $\nabla f(x^{(k)})$.

Важливим фактором оцінки будь-якого алгоритму є обчислювальна складність – оцінка ресурсів, необхідних для виконання поставленого завдання засобами конкретної реалізації. Зазвичай цю характеристику відокремлюють від властивостей системи, адже головна ціль показника обчислювальної складності полягає не в визначенні часу виконання алгоритму для конкретної ЕОМ, а в загальній демонстрації залежності витрачених ресурсів від збільшення початкових даних [12]. До уваги не береться ані точний час виконання окремих інструкцій, ані число бітів, витрачених на представлення змінних, ані швидкість процесора – під час оцінки доволі складних алгоритмів усім іншим можна знехтувати (з точністю до постійного множника).

Складність певного алгоритму розраховується шляхом підрахунку кількості операцій над усіма початковими даними для виконання поставленого завдання. В загальній нотації використовується термін «велике O», що позначає функціональну залежність витраченого часу від збільшення об'єму опрацьованих даних. Важливо, що обчислювальна складність визначається для конкретної реалізації алгоритму, а не до загальних рекомендацій із його використання.

Потрібно зазначити, що немає прямої кореляції між найгіршим випадком застосування, та реальним використанням алгоритмів. Подібна оцінка методів дає розуміння того, що може бути при «незручних» початкових даних, але однозначною диференціацією по складності бути аж ніяк не повинна.

Для градієнтного методу обчислювальна складність становить $O(N)$, де N – кількість параметрів обраної для дослідження функції. Це пояснюється тим, що на кожній ітерації алгоритм використовує усі параметри, а при послідовному застосуванні N «велике O» накопичується алгебраїчно, тобто для теоретичної складності алгоритму, не суттєво. Залежність від збільшення параметрів сильно впливає на обчислювальну складність градієнтного метода.

Генетичні алгоритми моделюють процес природного відбору в ході еволюції та мають стадії генерації популяції, мутацій, схрещування і відбору. Порядок стадій залежить від конкретного алгоритму.

Для оптимізації функції $f(x)$ дійсних змінних $x \in R^n$ застосовується алгоритм диференціальної еволюції. Популяцією в алгоритмі диференціальної еволюції вважається безліч векторів з R^n, причому кожна змінна цього простору відповідає своїй ознаці. Параметрами цього алгоритму є: розмір популяції N, сила мутації $F \in [0, 2]$ і ймовірність мутації P.

В якості початкової популяції обирається набір з N випадкових векторів. На кожній наступній ітерації алгоритм генерує нове покоління векторів, комбінуючи вектори попереднього покоління. А саме: для кожного вектора x_i з поточного покоління виконуються наступні стадії:

Стадія мутації. Випадково з популяції вибираються нерівні x_i вектори v_1, v_2, v_3. На основі цих векторів генерується так званий мутантний вектор $v = v_1 + F \cdot (v_2 - v_3)$.

Стадія схрещування. Над мутантним вектором виконується операція «схрещування», в ході якої кожна координата з ймовірністю P заміщається відповідною координатою вектора x_i. Отриманий вектор називається пробним.

Стадія відбору. Якщо пробний вектор виявляється краще вихідного x_i (тобто значення досліджуваної функції на пробному векторі менше, ніж на вихідному), то в новому поколінні він займає його місце.

Якщо збіжність не була досягнута, починається нова ітерація.

Слід також враховувати наступні зауваження:

для вирішення завдань оптимізації функцій дискретних змінних досить перевизначити тільки стадію мутації. Решта кроків алгоритму залишаються без змін;

часто, щоб збільшити ефективність алгоритму, створюються кілька незалежних популяцій. Для кожної такої популяції формується свій початковий набір випадкових векторів. В такому випадку з'являється можливість використовувати генетичні алгоритми для вирішення задач глобальної оптимізації;

ефективність методу залежить від вибору операторів мутації і схрещування для кожного конкретного типу завдань.

Обчислювальна складність генетичного алгоритму складає $O(N^2)$. Це пояснюється тим, що при використанні генетичних алгоритмів, вся кількість параметрів викликається рекурсивно. Звісно, в певному сенсі на збільшення об'єму роботи впливає й кількість елементів у популяції [13]. Цей фактор можна позначити, як коефіцієнт біля поточного результату обчислювальної складності, але константами перед N можна знехтувати. Подібне значення «великого O» є одним із найгірших, адже квадратична функція, вочевидь, накопичується набагато швидше, ніж лінійна (значення обчислювальної складності для градієнтних методів).

Метод імітації відпалу є одним з методів глобальної оптимізації, для роботи якого не вимагається гладкість функції. Він є варіантом методу випадкового пошуку і відомий як алгоритм Метрополіса.

Алгоритм ґрунтується на імітації фізичного процесу, який відбувається при кристалізації речовини, в тому числі при відпалі металів. Відпал – вид термічної обробки металів, сплавів, який полягає в нагріванні до певної температури і наступному повільному охолодженні до кімнатної температури. Мета відпалу – привести систему, зразком якої є метал, в стан з мінімальної енергією. Атоми при відпалі можуть як потрапити в стан з меншою енергією, так і в стан з більшою. Причому ймовірність потрапити з поточного стану в стан з більшою енергією зменшується з температурою. Поступово температура зменшується до кімнатної і система потрапляє в стан з мінімальною енергією.

Для задач оптимізації імітація процесу відпалу може бути проведена в такий спосіб: вводиться параметр T, який має сенс температури, на початку йому встановлюється значення T_0. Набір змінних, за якими відбувається оптимізація, буде позначатися як x. В якості початкового стану системи вибирається довільна точка. Далі запускається ітераційний процес на кожному кроці з безліччю сусідніх станів з випадково вибраним новим x^*. Якщо значення функції в цій точці менше, ніж значення в поточній точці, то ця точка вибирається в якості нового стану системи. В іншому випадку (тобто якщо $f(x^*) > F(x)$) такий перехід відбувається з імовірністю P, залежної від температури T, поточного стану x і кандидата на новий стан x^* наступним чином

$$P=\exp(-(f(x^*) - f(x))/T).$$

Завдяки переходам до гіршого стану в методі імітації відпалу вдалося вирішити проблему локальних мінімумів і не застрягати в них. Поступово, при зменшенні температури, зменшується і ймовірність переходів в стан з великим значенням функції. Таким чином в кінці імітації відпалу в якості x виявляється шуканий глобальний мінімум.

Виявляється, що успішність цього алгоритму залежить від способу вибору кандидатів, іншими словами, того, як саме відбувається такий випадковий вибір. Питання, чи знаходить цей алгоритм мінімум або наскільки він ефективний для пошуку стану з меншим значенням функції f (в разі якщо мінімум не може бути знайдений), представляють окремий інтерес. Варто також відзначити, що практичну цінність можуть мати і алгоритми, які не гарантують досягнення мінімуму.

Однією з особливостей цього методу є найбільша обчислювальна складність $O(N^2\log 2N)$. Головною причиною підвищення складності є додаткова умова перевірки температури, що до-

зволяє навіть у найгіршому випадку (коли доведеться пройти максимальну кількість ітерацій) витратити більше часу, але гарантовано знайти глобальний мінімум [14].

Метод Нелдера-Міда, або метод деформованого багатогранника (симплекса), застосовується для знаходження рішення задачі оптимізації дійсних функцій багатьох змінних

$$f(x) \to \min, x \in R^n,$$

причому функція $f(x)$, у загальному випадку, не є гладкою і може бути зашумленою. Іншою особливістю методу є те, що на кожній ітерації обчислюється значення функції $f(x)$ не більше ніж у трьох точках. Це особливо важливо у випадку складно обчислюваної функції $f(x)$. Метод Нелдера-Міда простий в реалізації і корисний на практиці, але, з іншого боку, для нього не існує теорії збіжності – алгоритм може розходитися навіть на гладких функціях.

Основними параметрами методу є $\alpha > 0$, $\beta > 0$ і $\gamma > 0$ – коефіцієнти відображення, стиснення і розтягування відповідно. Нехай необхідно знайти безумовний мінімум функції n змінних.

Підготовка. Вибирається $n + 1$ точка, що утворюють симплекс n-мірного простору

$$x_i = \left(x_i^{(1)}, x_i^{(2)}, ..., x_i^{(n)}\right), \ i = 1, ..., n+1.$$

У цих точках обчислюється значення функції $f(x)$

$$f_1 = f(x_1), f_2 = f(x_2), ..., f_{n+1} = f(x_{n+1}).$$

Сортування. Серед вершин симплекса $\{x_i\}$ вибираються: точка x_h з найбільшим (зі значень на вершинах симплекса) значенням функції f_h, точка x_g з наступним за величиною значенням f_g і точка x_l з найменшим значенням функції f_l.

Центр тяжіння. Обчислюється центр ваги всіх точок, за винятком x_h (обчислювати значення функції в знайденої точці не обов'язково)

$$x_c = \frac{1}{n} \sum_{i \neq h} x_i \cdot$$

Віддзеркалення. Точка x_h відбивається щодо точки x_c з коефіцієнтом α. В отриманій точці x_r обчислюється значення функції

$$x_r = (1+\alpha)x_c - \alpha \cdot x_h, \qquad f_r = f(x_r).$$

Розгалуження. Цей крок залежить від значення f_r в порівнянні з $f_l < f_g < f_h$. Якщо $f_r < f_l$, то ймовірно напрямок було вибрано вдало. Можна зробити спробу збільшити крок. Проводиться розтягнення, знаходиться нова точка і значення функції в ній

$$x_e = (1-\gamma)x_c - \gamma \cdot x_r, \qquad f_e = f(x_e).$$

Якщо $f_e < f_r$, то точці x_h присвоюється значення x_e, а інакше – значення x_r. Після цього ітерація закінчується.

Якщо $f_l < f_r < f_g$, то вибір точки непоганий: точці x_h присвоюється значення x_r. Після цього ітерація закінчується.

Якщо $f_g < f_r < f_h$, то точки x_r і x_h міняються місцями (значення f_r і f_h теж). Після перейти на наступний крок.

Якщо $f_h < f_r$, то просто перейти на наступний крок.

Стиснення. Будується точка x_s і обчислюється значення функції в ній

$$x_s = \beta \cdot x_h + (1-\beta)x_c, \ f_s = f(x_s).$$

Розгалуження. Якщо $f_s < f_h$, то точці x_h присвоюється значення x_s. Після цього ітерація закінчується.

Глобальне стиснення. У випадку ($f_s > f_h$) має місце ситуація, коли початкові точки виявилися найвдалішими. Робиться перетворення (стиснення до точки x_l)

$$x_i \leftarrow x_l + (x_i - x_l)/2, \quad i \neq l.$$

Перевірка збіжності. Останній крок – перевірка збіжності, може виконуватися по-різному. Якщо потрібна точність не досягнута, слід перейти до етапу «Центр тяжіння».

Оцінка обчислювальної складності для цього методу є досить непростим питанням. Адже в загальному випадку невідомо, що кількість ітерацій скінченна [15], тим не менш в умовах поставленої задачі її можливо підрахувати. На кожній ітерації тричі використовуються усі початкові дані, тому можна зробити висновок, що обчислювальна складність становить $O(3N)$. Так, зазвичай для загальної демонстрації залежності коефіцієнтом біля N нехтують, але в випадку методу Нелдера-Міда доцільно показувати кількість «ітерацій в ітерації». Іншими словами,

початкові дані використовуються тричі на кожній ітерації, а загальна кількість операцій збільшиться в алгебраїчному порядку, тобто не суттєво для цього показника. Обчислювальна складність для цього методу не набагато більше від градієнтного, і суттєво менша за показники генетичного і алгоритму імітації відпалу, а це лише доповнює картину зручності алгоритму деформованого багатогранника (симплекса) для конкретних завдань.

Наведений на рис. 1 графік ілюструє обчислювальні складності кожного з наведених методів.

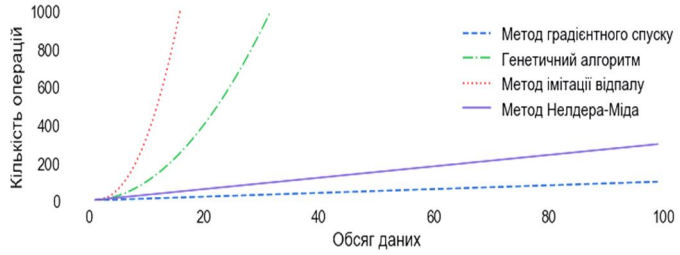

Рис. 1. Обчислювальна складність алгоритмів

В ході дослідження розроблено бібліотеку opti_methods методів багатовимірної оптимізації оціночного функціоналу якості моделей задач машинного навчання мовою Python 3.6, і проведено числовий експеримент на вибірках різного розміру для оптимізаційних функцій (1) і (2).

Результати застосування розглянутих алгоритмів мінімізації функції помилок регресійної моделі за критеріями середньоквадратичного відхилення і середнього значення відхилення наведено в табл. 1 і табл. 2 відповідно. Експеримент проведено на процесорі Intel Core i7-7500U CPU 2.70GHz, 2901 МГц, ядер: 2, логічних процесорів: 4, RAM 12 Гб.

Таблиця 1

Час роботи методів для гладкої оптимізаційної функції

Методи	Розмір вибірки		
	<100	5000	25000
	час, мс		
Градієнтний	59,2	384	1580
Нелдера-Міда	92,5	256	1230
Генетичний	342	869	3500
Імітації відпалу	3,64	10000	40800

Проілюструємо результати роботи алгоритмів на графіках: для малої вибірки – рис. 2, середньої – рис. 3 і великої – рис. 4 відповідно.

Як бачимо, градієнтний метод і метод Нелдера-Міда демонсрують достатньо красномовні результати – незалежно від обсягу вибірки, їх швидкість є прийнятною для оптимізації гладких функцій у задачах навчання за прецедентами.

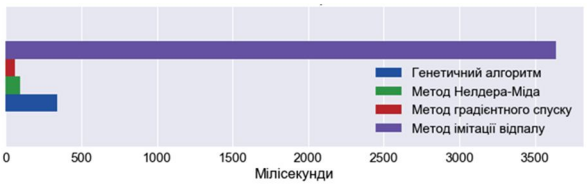

Рис. 2. Час роботи алгоритмів на малій вибірці

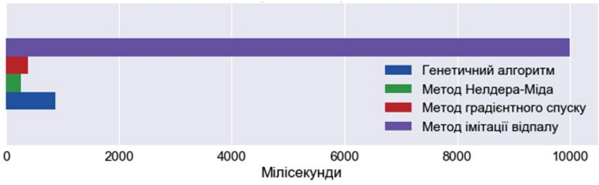

Рис. 3. Час роботи алгоритмів на середній вибірці

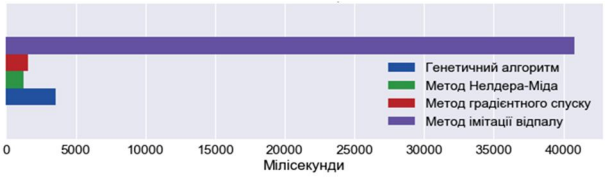

Рис. 4. Час роботи алгоритмів на великій вибірці

Таблиця 2

Час роботи методів для негладкої оптимізаційної функції

Методи	Розмір вибірки		
	<100	5000	25000
	час, мс		
Нелдера-Міда	105	266	12300
Генетичний	315	892	35500
Імітації відпалу	12400	10000	52700

Розглянемо роботу алгоритмів на негладкій оптимізаційній функції (1) і проілюструємо час виконання алгоритмів рисунками 5-7.

Зауважимо, що градієнтний метод не використовується для оптимізації негладкої функції, оскільки така функція є недиференційованою.

Для негладкого функціоналу якості моделі навчання є прийнятними методи оптимізації Нелдера-Міда і генетичний алгоритм. Причому, на великих обсягах даних генетичний алгоритм

працює швидше за метод Нелдера-Міда, що є важливим в умовах сучасного поширення використання великих даних (big data).

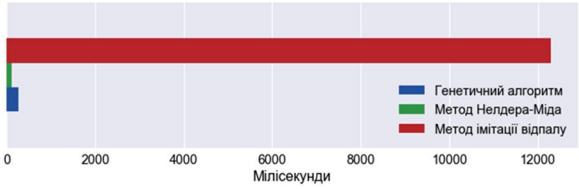

Рис. 5. Час роботи алгоритмів на малій вибірці

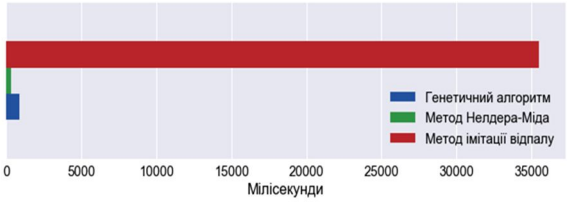

Рис. 6. Час роботи алгоритмів на середній вибірці

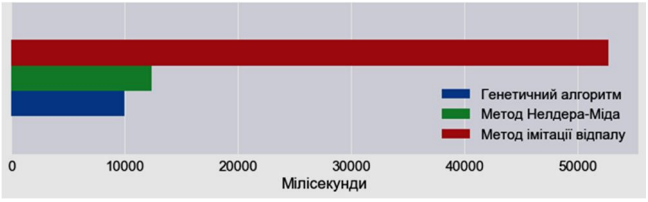

Рис. 7. Час роботи алгоритмів на великій вибірці

Висновки та напрямок подальших досліджень. Проаналізувавши роботу чотирьох методів оптимізації функцій (градієнтний метод, метод Нелдера-Міда, генетичний і алгоритм імітації відпалу) на вибірках різних розмірів і різних видах функціоналах якості моделей навчання, можна зазначити, що для гладких оптимізаційних функцій прийнятними є методи градієнтного спуску і Нелдера-Міда незалежно від розміру навчальної вибірки. Для негладких функціоналів зі зростанням обсягу об'єктів у вибірці кращі результати показав генетичний алгоритм, але за простотою викладення і швидкістю збігання на нескладних функціях може бути рекомендовано до застосування метод Нелдера-Міда. В ході подальших досліджень отримані висновки планується перевірити в ході експерименту глибокого навчання багатошарових штучних нейронних мереж.

Список літератури

1. Gartner's 2016 Hype Cycle for Emerging Technologies Identifies Three Key Trends That Organizations Must Track to Gain Competitive Advantage [Електронний ресурс] // Cycle for Emerging Technologies. – 2016. – Режим доступу до ресурсу: https://www.gartner.com/newsroom/id/3412017.

2. **Луис Педро Коэльо**, Вилли Ричарт. Построение систем машинного обучения на языке Python. 2-е издание / пер. с англ. Слинкин А. А. – М.: ДМК Пресс, 2016. – 302 с.

3. **Петер Флах**. Машинное обучение. Наука и искусство построения алгоритмов, которые извлекают знания из данных. Учебник / пер. с англ. Слинкин А. А. – М.: ДМК Пресс, 2015. – 408 с.

4. Оптимізаційні методи та моделі. Підручник. / [Л. В. Забуранна, Н. В. Попрозман, Н. А. Клименко, О. І. Попрозман, С. В. Забуранний]. – К.: __, 2014. – 372 с.

5. **Вітлінський В. В.**, Наконечний С. І., Терещенко Т. О. Математичне програмування: Навч.-метод. посібник для самост. вивч. дисц. – К.: КНЕУ, 2001. – 248 с.

6. Max Kuhn, Kjell Johnson. Applied Predictive Modeling // Springer, 2013. – 318 с.

7. **Кононюк А. Ю**. Нейроні мережі і генетичні алгоритми. – К.: «Корнійчук», 2008. – 446 с.

8. **Роберт Каллан**. Основні концепції нейроних мереж = The Essence of Neural Networks First Edition. — 1-е. – «Вильямс», 2001. – 248 с.

9. **Ясницкий Л. Н.** Введення в штучний інтелект. – видання 1-е. – Издательский центр «Академия», 2005. –176 с.

10. **Вороновский Г. К.**, Махотило К. В., Петрашев С. Н., Сергєєв С. А. Генетичні алгоритми, штучні нейроні мережі і проблеми віртуальної реальності. – Замовне. – Х.: ОСНОВА, 1997. – 112 с.

11. **Нейт Сильвер**. Сигнал и Шум. Почему одни прогнозы сбываются, а другие – нет // Азбука-Аттикус, КоЛибри, 2015. – 400 с.

12. **William M**. Bolstad. Introduction to Bayesian Statistics, 2nd Edition // Wiley-Interscience; 2nd edition.

13. **Бондаренко И. Б.**, Каляева Е. А., Кокшаров Д. Н. Адаптация параметров генетических алгоритмов для оптимизации сложных функций // Известия высших учебных заведений. Приборостроение. – 2011. – № 9. – С. 54 – 63.

14. **Боронихина Е. А.** Исследование эвристических методов решения задачи коммивояжера. Разработка интегрированного метода: дис. магистра инф. наук: 01.04.02 / Боронихина Елена Александровна – Томск, 2016. – 60 с.

15. **Singer S**. Complexity Analysis of Nelder – Mead Search Iterations [Текст] / Sanja Singer, Sasa Singer // Proceedings of the 1. Conference on Applied Mathematics and Computation – Dubrovnik, Croatia, September 13–18, 1999. – pp. 185–196.

Рукопис подано до редакції 19.02.2018