КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет комп'ютерних наук та кібернетики

Кафедра інтелектуальних програмних систем

Курсова робота

на тему:

ЗАСТОСУВАННЯ СУБГРАДІЄНТНОГО МЕТОДУ ДЛЯ МІНІМІЗАЦІЇ ФУНКЦІЇ ВТРАТ HINGE LOSS В ЗАДАЧІ БІНАРНОЇ КЛАСИФІКАЦІЇ МЕТОДОМ ОПОРНИХ ВЕКТОРІВ

Виконав студент 3-го курсу

Дмитро ПАЩЕНКО _____

(підпис)

Науковий керівник:

асистент кафедри інтелектуальних програмних систем,

Віктор СТОВБА _____

(підпис)

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

Студент _____

(підпис)

Київ – 2021

РЕФЕРАТ

Обсяг роботи 27 сторінок, 12 ілюстрацій, 1 таблиця і 11 джерел посилання.

Ключові слова: СУБГРАДІЄНТНИЙ МЕТОД, БІНАРНА КЛАСИФІКАЦІЯ, МЕТОД ОПОРНИХ ВЕКТОРІВ, HINGE LOSS.

Об’єктом роботи є субградієнтний метод та недиференційовна опукла функція втрат hinge loss, яка фігурує в контексті задачі бінарної класифікації методом опорних векторів.

Метою роботи є застосування субградієнтного методу для мінімізації функції втрат hinge loss в задачі бінарної класифікації методом опорних векторів та обґрунтування доцільності субградієнтного підходу.

Засоби реалізації: мова програмування Python, бібліотека NumPy для швидких алгебраїчних обчислень, бібліотека Pandas для роботи з даними статистичних вибірок.

У ході роботи переконалися в ефективності застосування саме субградієнтного методу та функції втрат hinge loss у задачі бінарної класифікації методом опорних векторів. Субградієнтний метод довів свою швидкість та високу точність за достатньої кількості ітерацій, хоча для нього й не існує гарантій досягнення оптимуму чи простих правил зупинки, як у випадку градієнтного спуску.

ЗМІСТ

ВСТУП

У задачах оптимізації далеко не завжди цільова функція є диференційовною. Проте якщо вона опукла, це відкриває простір для багатьох особливих методів оптимізації, одним із яких є субградієнтний метод. Однією з найбільш актуальних областей застосування оптимізаційних методів є машинне навчання. Наприклад, задача бінарної класифікації. Зокрема канонічна первинна задача методу опорних векторів визначається опуклою, але недиференційовною функцією. Ця робота обґрунтовує як доцільність використання субградієнтного методу, так і загалом можливість звернення до недиференціовних опуклих функцій у машинному навчанні.

Актуальність. Оптимізація недиференційовних опуклих функцій є актуальною проблемою, що знаходить різні сфери застосування – у тому числі в задачах машинного навчання.

Мета й завдання роботи. Застосувати субградієнтний метод для мінімізації функції втрат hinge loss в задачі бінарної класифікації методом опорних векторів та обґрунтувати доцільність субградієнтного підходу.

Об’єктом дослідження є субградієнтний метод та недиференційовна опукла функція втрат hinge loss, яка фігурує в контексті задачі бінарної класифікації методом опорних векторів.

Предметом дослідження є оптимізація недиференційовних опуклих функцій та використання для цього субградієнтних методів.

1. СУБГРАДІЄНТНИЙ МЕТОД

1.1. Опуклі функції та опукла оптимізація. Градієнтний спуск

Теореми та означення для опуклих функцій візьмемо з посібника Моклярчука [1].

Функція $f\colon \mathbb{R}^n \to \mathbb{R}$, що визначена на опуклій множині $X \subset \mathbb{R}^n$, називається опуклою, якщо справджується нерівність

$$f(\lambda x^1 + (1-\lambda)x^2) \leq \lambda f(x^1) + (1-\lambda) f(x^2) \qquad (1)$$

для всіх $x^1, x^2 \in X$ та всіх $\lambda \in [0,1]$. Якщо для всіх $x^1, x^2 \in X$, $x^1 \neq x^2$, та всіх $\lambda \in (0,1)$ справджується строга нерівність, то функція f називається строго опуклою на X. З геометричної точки зору, якщо побудувати дотичну гіперплощину до опуклої функції в будь-якій точці, то вона завжди лежатиме нижче графіка функції.

Зокрема опуклими є функції:

$$f(x) = x,$$
$$f(x) = x^2,$$
$$f(x) = \max(0, x).$$

Це спостереження надалі буде корисним.

Опуклі функції відіграють важливу роль в теорії оптимізації. Історично спочатку задачі програмування розглядали в опозиції лінійності (лінійна цільова функція, лінійні обмеження) та нелінійності, але згодом прийшли до більш широкої класифікації, а саме поділу задач оптимізації на опуклі та неопуклі. Для задач математичного програмування, що задаються опуклими функціями, вдається отримати найбільш змістовні умови оптимальності, а також розробити ефективні алгоритми їх рішення [2].

Нехай множина X опукла і функція f опукла на X. Тоді локальний розв'язок задачі на мінімум

$$f(x) \to min, x \in X,$$

є також глобальним розв'язком задачі. Отже, для опуклих задач поняття локального і глобального розв'язків не відрізняються і можна говорити просто про розв'язок задачі.

Якщо функція f опукла і диференційовна на X, для розв'язку задачі оптимізації можна використати метод градієнтного спуску. Відомо, що градієнт функції вказує напрямок її найшвидшого зростання. Градієнтний спуск використовує цю властивість для поступового наближення до локального мінімуму (а оскільки функція опукла, то локальний мінімум є глобальним). Типовий алгоритм наведено на Рис. 1.

given a starting point $x \in \mathbf{dom}\, f$.

repeat

1. $\Delta x := -\nabla f(x)$.
2. *Line search.* Choose step size t via exact or backtracking line search.
3. *Update.* $x := x + t\Delta x$.

until stopping criterion is satisfied.

Рис. 1 [3]. Алгоритм градієнтного спуску.

Тобто градієнтний спуск – це спуск по функції в напрямку, протилежному до градієнта, із заданим кроком, допоки не виконається критерій зупинки. Типовим критерієм зупинки є обмеження евклідової норми градієнта [3]:

$$\|\nabla f(x)\|_2 \leq \eta,$$

де η – мале позитивне.

1.2. Поняття субградієнта

Проте в задачах оптимізації далеко не завжди цільова функція є диференційовною. Тоді такі звичні методи, як градієнтний спуск або метод Ньютона, не діють. На щастя, існує певний еквівалент градієнта, який володіє потрібними властивостями, – субградієнт.

Для диференційовної в точці $\hat{x}$ опуклої функції f виконується нерівність

$$f(x) - f(\hat{x}) \geq \langle f'(\hat{x}), x - \hat{x} \rangle \quad x \in X, \tag{2}$$

яка означає, що графік функції f лежить не нижче дотичної до нього гіперплощини в точці $\big(\hat{x}, f(\hat{x})\big)$ [1]. Визначальна особливість субградієнта в тому, що його можна підставити у нерівність (2) замість $f'(\hat{x})$. Дамо означення.

Нехай f – функція на множині $X \subset \mathbb{R}$. Вектор $a \in \mathbb{R}^n$ називається субградієнтом функції f в точці $\hat{x}$, якщо

$$f(x) - f(\hat{x}) \geq \langle a, x - \hat{x} \rangle \quad x \in X. \tag{3}$$

Множина всіх субградієнтів називається субдиференціалом функції f в точці $\hat{x}$ і позначається $\partial f(x)$. Якщо функція диференційовна в певній точці, то її субдиференціал у цій точці містить лише її градієнт.

Гіперплощина H називається опорною до множини $X \subset \mathbb{R}^n$ в точці a, якщо X міститься в одному з півпросторів, породжених цією гіперплощиною, а сама вона містить точку a. Для функції числового аргумента субградієнт – це тангенс кута нахилу опорної прямої (тобто опорної гіперплощини), так само як похідна є тангенсом кута нахилу дотичної [2]. З цього випливає, що опорні прямі у точці $\big(\hat{x}, f(\hat{x})\big)$ охоплюють усі проміжні значення між лівосторонніми і правосторонніми дотичними в цій точці. Для опуклої функції f на опуклій числовій множині $X \subset \mathbb{R}$ справедлива формула:

$$\partial f(\hat{x}) = [f'_-(\hat{x}), f'_+(\hat{x})], \tag{4}$$

де $f'_-(x)$ – це лівостороння похідна, а $f'_+(x)$ – правостороння.

Субградієнт складної функції можна обчислити, знаючи субградієнти для базового набору функцій. Для цього необхідно застосувати правила субградієнтного числення [5]:

a) Масштабування.

$$\forall a > 0 \quad \partial(af) = a * \partial f$$

b) Додавання.

$$\partial(f_1 + f_2) = \partial f_1 + \partial f_2$$

c) Афінна композиція.

Якщо $g(x) = f(Ax) + b$, то

$$\partial g(x) = A^T \partial f(Ax + b)$$

d) Скінченний поточковий максимум.

Якщо $f(x) = \max\limits_{i=1,\dots,m} f_i(x)$, то

$$\partial f(x) = conv\left(\bigcup_{i:f_i(x)=f(x)} \partial f_i(x)\right)$$

e) Загальний поточковий максимум.

Якщо $f(x) = \max\limits_{s\in S} f_s(x)$, то

$$\partial f(x) \supseteq cl\left\{conv\left(\bigcup_{s:f_s(x)=f(x)} \partial f_s(x)\right)\right\}$$

1.3. Субградієнтний метод

Виявляється, що субградієнти можна використати для адаптування методу градієнтного спуску для мінімізації недиференційовних функцій, проте з деякими особливостями:

- довжина кроку зазвичай фіксується завчасно (у звичайному методі градієнта довжина кроків вибирається за допомогою лінійного пошуку);
- на відміну від звичайного методу градієнта, субградієнтний метод не є методом спуску: значення функції може зростати;
- не існує загального ефективного критерію зупинки, тому субградієнтний метод зазвичай використовується без формального критерію зупинки [4].

Субградієнтні методи можуть буди набагато повільнішими (невисока швидкість збіжності), ніж методи внутрішніх точок (або метод Ньютона у необмеженому випадку) [1]. Однак субградієнтні методи мають певні переваги. Вони можуть бути застосовані до набагато більш широкого кола проблем. Вимоги до пам’яті субградієнтних методів можуть бути набагато меншими ніж у методах описаних вище, що означає, що він може бути використаний для надзвичайно великих проблем.

Розглянемо випадок необмеженої мінімізації опуклої функції $f: \mathbb{R}^n \to \mathbb{R}$ на області $\mathbb{R}^n$. Субградієнтний метод, як і градієнтний, полягає у виконанні простої ітерації

$$x^{(k+1)} = x^{(k)} - \alpha_k g^{(k)}. \qquad (5)$$

де, $x^{(k)}$ – значення аргументу на k-тій ітерації, $g^{(k)}$ – будь-який субградієнт f у точці $x^{(k)}$, та $\alpha_k > 0$ – k-ий розмір кроку. Таким чином, на кожній ітерації субградієнтного методу ми робимо крок у бук негативного субградієнту.

Може статися так, що $-g^{(k)}$ не є напрямком спуску для f у $x^{(k)}$, тобто ітерація субградієнтного методу може збільшити цільову функцію. Проте є й гарна новина: при достатньо малому кроці субградієнтний метод прямує до мінімізатора [10]. Тобто якщо x^* – мінімізатор функції f, а $x = x_0 - tg$, де g – субградієнт, то при достатньо малому кроці $t > 0$

$$\|x - z\|_2 < \|x_0 - z\|_2.$$

Зрозуміло, що виникає необхідність на кожному кроці перевіряти поточне значення функції на мінімальність і запам'ятовувати аргумент у разі успішної перевірки. Таким чином, метод повертає найкращий знайдений мінімізатор.

Як вибирати розмір кроку в розглянутому методі? У субградієнтному методі вибір розміру кроку сильно відрізняється від стандартного методу градієнта. Використовується багато різних типів правил розміру кроків. Ось деякі з основних правил, наведених і обґрунтованих Стівеном Бойдом [4]:

- Постійний розмір кроку.

  $\alpha_k = \alpha$ додатна константа, яка незалежить від k.

- Постійна довжина кроку.

  $\alpha_k = \gamma / \|g^{(k)}\|_2$, де $\gamma > 0$. Це означає, що $\|x^{(k+1)} - x^{(k)}\|_2 = \gamma$.

- Несумовна, проте квадратично сумовна послідовність:

$$a_k \geq 0, \qquad \sum_{k=1}^{\infty} \alpha_k^2 < \infty, \qquad \sum_{k=1}^{\infty} \alpha_k = \infty.$$

- Несумовна, проте спадна послідовність:

$$a_k > 0, \qquad \lim_{k\to\infty} \alpha_k = 0\,, \qquad \sum_{k=1}^{\infty} \alpha_k = \infty.$$

Для останніх двох кроків алгоритм гарантовано збігається до оптимального значення, тобто $\lim_{k\to\infty} f\left(x^{(k)}\right) = f^*$.

Особливістю цих варіантів є те, що вони визначаються перед запуском алгоритму, вони не залежать від даних, обчислених під час виконання. Це сильно відрізняється від правил вибору розміру кроків, які існують у стандартних методах спуску, які дуже залежать від поточної точки та напрямку пошуку.

Запишемо алгоритм на псевдокоді.

$procedure\ SubgradientDescent(T)$:
    $x_0 = 0_n$
    $f_{min} = +\infty$
    $x_{min} = x_0$
    $for\ t = 0, \dots, T-1\ do$
        $Compute\ g_t = \nabla f(x_t)$
        $Update\ x_{t+1} = x_t - \alpha_t g_t$
        $if\ f(x_{t+1}) < f_{min}\ then$
            $f_{min} = f(x_{t+1})$
            $x_{min} = x_{t+1}$
    $return\ x_{min}$
$end\ procedure$

2. МЕТОД ОПОРНИХ ВЕКТОРІВ ТА ФУНКЦІЯ HINGE LOSS

2.1. Лінійний метод опорних векторів

Класифікація даних – поширене завдання машинного навчання. Припустимо, задані точки даних, про які відомо, що належать до одного з двох класів, і мета полягає в тому, щоб вирішити, в якому класі буде знаходитися нова точка даних. Якщо метод для вирішення задачі шукає гіперплощину, що розділить точки, то такий метод називається лінійним класифікатором. Існує багато гіперплощин, що здатні розділити дані. Один із найкращих підходів – побудувати гіперплощину, що максимально розділить дані, тобто максимізує відступ (margin) [7] між двома класами. Тому ми вибираємо гіперплощину так, щоб відстань від неї до найближчої точки даних з кожної сторони була максимальною. Якщо така гіперплощина існує, вона відома як гіперплощина максимального відступу, а лінійний класифікатор, який вона визначає, відомий як класифікатор максимального відступу (maximum-margin classifier).

Метод опорних векторів (support vector machine, SVM) – метод машинного навчання з учителем та один із найпопулярніших лінійних класифікаторів. Нехай $X \subseteq \mathbb{R}^n$ – простір ознак, $Y = \{1, -1\}$ – множина виходів (класів), $M = \{(x_1, y_1), \dots, (x_n, y_n)\}$ – тренувальна вибірка. Метод опорних векторів шукає оптимальну гіперплощину вигляду:

$$\{x: f(x) = \langle w, x \rangle = 0\}.$$

Поточною стандартною інкарнацією методу є soft-margin SVM [8]. Для знаходження вектора невідомих параметрів w він мінімізує наступний функціонал:

$$\min_w \frac{\lambda}{2}\|w\|^2 + \frac{1}{m}\sum_{i=1}^{m} \ell\big(w; (x_i, y_i)\big), \qquad (6)$$

де

$$\ell\big(w;(x_i,y_i)\big) = max\{0,1 - y_i\langle w,x_i\rangle\}. \qquad (7)$$

Функція (7) називається hinge loss, її ми розглянемо пізніше. $\frac{\lambda}{2}\|w\|^2$ у формулі (6) – l_2-регуляризатор, він забезпечує компроміс між мінімальністю результату та складністю гіпотези. Складна гіпотеза може призвести до проблеми «перенавчання» моделі, тому регуляризація сприяє кращому узагальненню результату. Опис параметрів моделі наведений на Рис. 2. Важливо відмітити, що в цій роботі для простоти термін зміщеності b прирівняли до нуля.

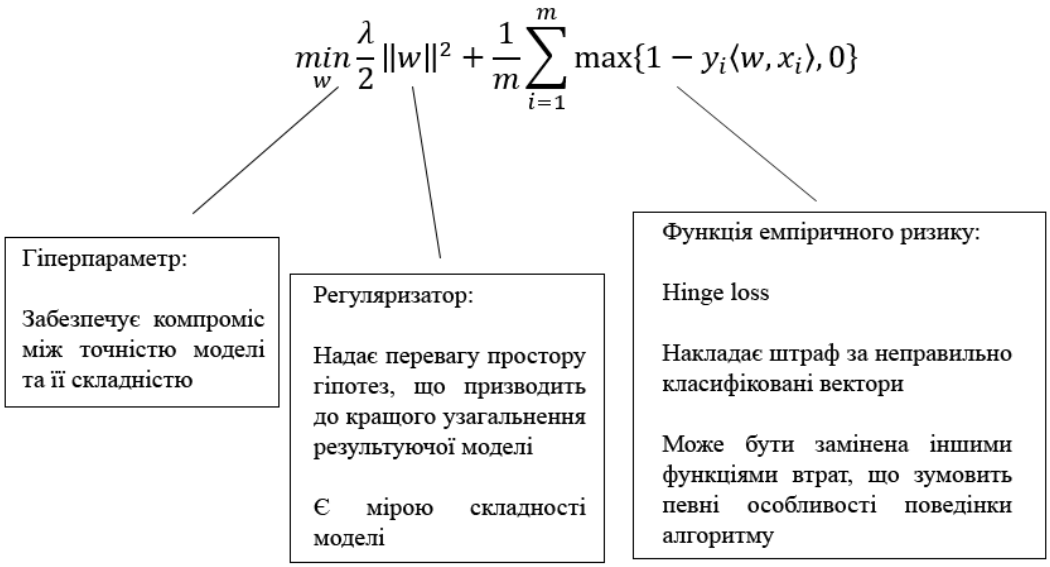

Рис. 2.Пояснення моделі лінійного методу опорних векторів (soft margin).

Завдання вивчення методу опорних векторів зазвичай сприймається як проблема обмеженого квадратичного програмування. Однак такий підхід дуже повільний [8]. Виникає проблема, оскільки функція, яку ми назвали hinge loss, очевидно недиференційовна, тобто градієнтні методи оптимізації застосувати неможливо.

2.2. Функція втрат hinge loss

Функція $f(x) = \langle w, x \rangle$ відображає вектор x з X у дійсне число $\hat{y}$. Це число $\hat{y}$ називають оцінкою параметра y, а величину $m = y\hat{y}$ – функціональним відступом (functional margin) [9]. Вважаємо, що точка даних була правильно класифікована, якщо $f(x) > 0$. Звідси випливає:

- якщо $m > 0$, то об'єкт було класифіковано правильно, і чим більше m, тим більша впевненість;
- якщо $m \leq 0$, то об'єкт було класифіковано неправильно, і чим менше m, тим більша похибка класифікації.

Тож тепер стає ясною мета методу опорних векторів – максимізувати відступ. Для цього існує декілька варіантів емпіричного ризику. Наприклад, 0-1 loss:

$$\hat{R}_n(f) = \frac{1}{n}\sum_{i=1}^{n} 1(m_i \leq 0).$$

Проте така функція не придатна для обчислень: вона недиференційовна (більше того, розривна) і неопукла. Її оптимізація – NP-складна проблема [9]. Тому був знайдений кращий підхід: hinge loss.

$$l_{hinge} = \max\{1 - m, 0\} = (1 - m)_+.$$

Ця функція втрат є кусково-лінійною апроксимацією 0-1 loss та обмежує її зверху (Рис. 3):

$$1(m_i \leq 0) \leq (1 - m)_+.$$

Незважаючи на те, що hinge loss недиференційовна в точці $m = 1$, вона є опуклою, а значить, її можна оптимізувати субградієнтним методом. Для цього обрахуємо субдиференціал функції (7) за змінною w, користуючись формулою (4).

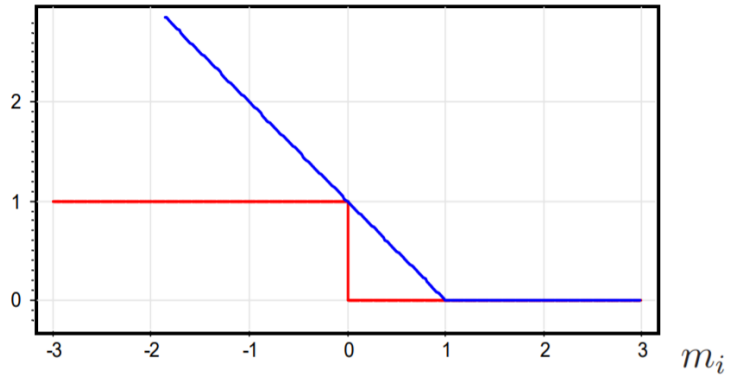

Рис. 3. Функції 0-1 loss (червоним) та hinge loss (синім)

$$\partial\ell(\mathrm{w};(\mathrm{x},\mathrm{y})) = \begin{cases} \{-yx\}, & m < 1 \\ [-yx, 0], & m = 1 \\ \{0\}, & m > 1, \end{cases}$$

де $m = y\langle w, x\rangle$. Для зручності будемо вважати, що при $\mathrm{m} = 1$ субградієнт рівний 0.

Залишилося обчислити субградієнт функції (6). За правилом додавання, для знаходження субградієнта суми функцій достатньо просумувати субградієнти цих функцій.

$$\nabla_{\mathrm{w}}\left(\frac{\lambda}{2}\|w\|^2 + \frac{1}{m}\sum_{i=1}^{m}\ell(w;(x_i,y_i))\right) = \begin{cases} 2\lambda w - \frac{1}{m}\sum_{i=1}^{m} y_i x_i\,, & w < 1 \\ 0, & w \geq 1 \end{cases}$$

Існують також інші функції втрат, зокрема:

- logistic loss

$$l_{logistic} = \log(1 + e^{-m})$$

- square loss

$$l_{square} = (1 - m)^2$$

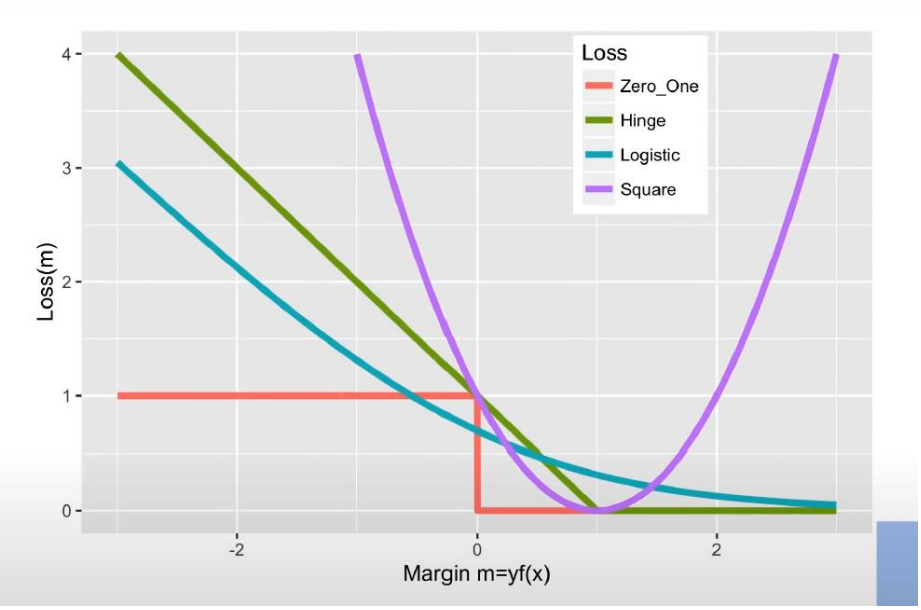

Рис. 3 [9]. Графіки різних функцій втрат.

Логістична функція досить близько апроксимує hinge loss, але найголовніше – є гладкою, тобто її можна оптимізувати звичайним методом градієнтного спуску. Те ж саме стосується й квадратичної функції, але вона вводить штрафи для великих додатних відступів. Усі ці варіації мають свої особливості, що можуть виявитися зручними для вирішення певних нішевих задач. Наприклад, logistic loss є основою логістичної регресії. Проте чи всі ці функції однаково застосовні до задачі бінарної класифікації? Порівняємо оптимізацію досліджуваної функції втрат субградієнтним методом та оптимізацію інших – градієнтним. Для цього спершу обчислимо градієнти зазначених функцій.

$$\nabla_w l_{logisitc} = -\frac{y_i}{1 + e^{y_i \langle w, x_i \rangle}} x_i$$

$$\nabla_w l_{square} = -2(1 - y_i \langle w, x_i \rangle) y_i x_i$$

Звернемо також увагу, що ці функції містять багато арифметичних операцій (логістична, наприклад, взагалі містить піднесення у степінь) і є потенційно схильними до переповнення типу на великих наборах даних.

3. ОБЧИСЛЮВАЛЬНІ ЕКСПЕРИМЕНТИ

3.1. Структура програми. Підбір параметрів

Основним елементом програми є клас SuportVectorMachine, що має стандартний у машинному навчанні набір методів:

- fit(train_x, train_y)

  Шукає невідомі параметри моделі за тренувальною вибіркою.

- evaluate(test_x, test_y)

  Рахує точність моделі за тестовою вибіркою.

Поведінку моделі можна налаштовувати через параметри конструктора. Зокрема можна обрати функцію втрат (HINGE, LOGISTIC, QUADRATIC), кількість ітерацій оптимізаційного методу та розмір пакету (batch) даних. Шляхом перебору було підібрано множник регуляризації $\lambda = 0.0001$.

Для оптимізації hinge loss використовується субградієнтний метод з кроком $\frac{1}{k}$, де k – номер поточної ітерації («square summable but not summable»). Запам’ятовування останнього оптимального значення не проводилося для спрощення алгоритму (у цьому не виникало потреби на обраних даних). Інші функції оптимізуються тим же методом, але з іншим кроком, характерним для градієнтного спуску (у структурі програми ці методи узагальнено об’єднані в один метод спуску та легко налаштовуються). Градієнтний спуск виконує вже згаданий backtracking line search (з параметрами $\alpha = 0.3$ і $\beta = 0.8$).

Для тестування було використано два набори даних: один згенерований самостійно – «custom.data», складається зі 100 тривимірних точок; інший – «breast-cancer-wisconsin.data» – є популярним датасетом для бінарної класифікації, має об’єм 699 зразків, кожен з яких містить по 10 атрибутів та класову ознаку.

3.2. Порівняння робастності

Робастність у стастистиці – це нечутливість до різних відхилень і неоднорідностей у вибірці. Для порівняння реакції відповідних функцій втрат на аномалії було використано власні згенеровані дані. Зокрема було згенеровано вибірку розміру 100: по 50 точок на кожен клас. Вибірка сконцентрована в точках $(-1,1)$ та $(1,-1)$ для класів -1 і 1 відповідно. Були зроблені заміри точності та нахилу розділяючої прямої. Потім поступово додавав аномальні точки в околі $(10,-5)$, що належить класу -1 і робив заміри точності та нахилу розділяючої прямої. Результати можна побачити на Рис. 4-6 та в Табл. 1.

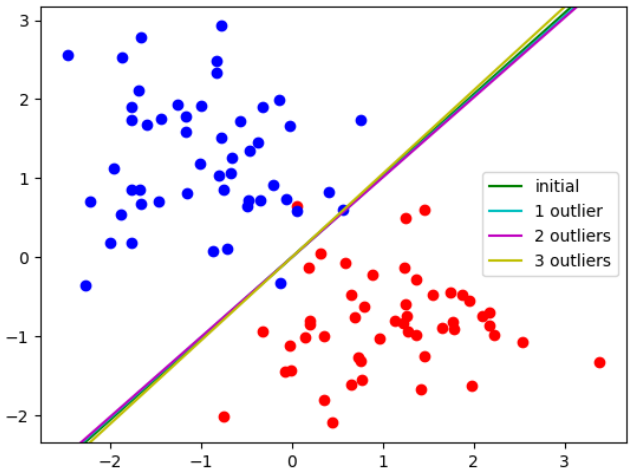

Рис. 4. Різні положення розділяючої прямої в залежності від кількості викидів для hinge loss.

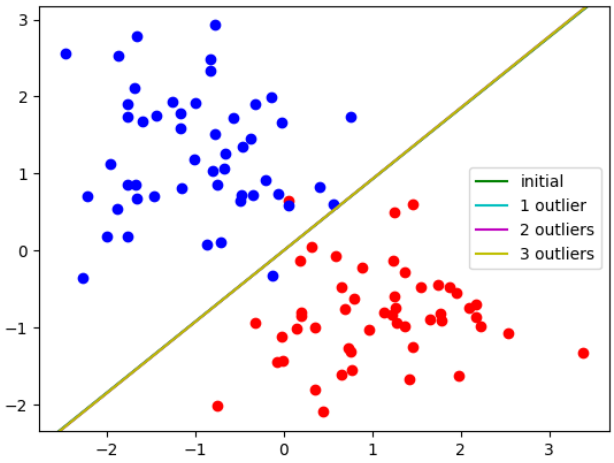

Рис. 5. Різні положення розділяючої прямої в залежності від кількості викидів для logistic loss.

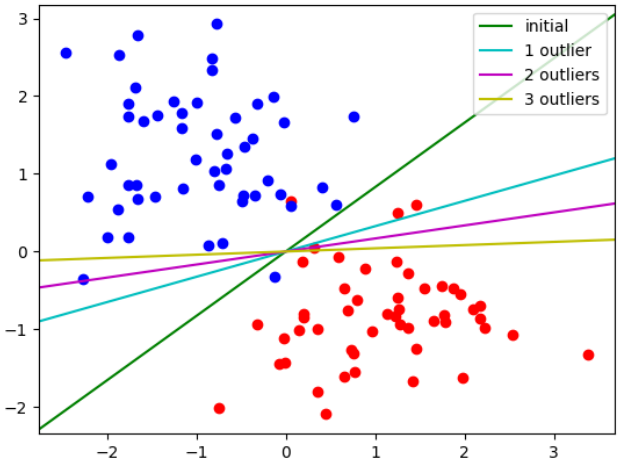

Рис. 6. Різні положення розділяючої прямої в залежності від кількості викидів для quadratic loss.

Кі-сть викидів / Функція втрат	1	2	3
Hinge loss	0.01268	0.01980	0.02523
Logistic loss	0.00242	0.00341	0.00311
Quadratic loss	0.50293	0.66076	0.78715

Табл. 1. Абсолютні відхилення тангенсів кутів нахилу розділяючої прямої від початкового значення.

Тести були проведені з єдиним уточненням: для функцій втрат logistic та quadratic у градієнтному методі використовувався не лінійний пошук кроку, а те ж правило, що й для hinge loss. Такий крок підвищив точність моделі на цих даних, а також поставив усі алгоритми в рівні умови.

Бачимо, що найкраще себе проявила логістична функція втрат. Це витікає з форми функцій (Рис. 3). При збільшенні відступу в негативний бік вона накладає найменший штраф (але все ж продовжує накладати невеликий штраф навіть на додатний відступ, прагнучи його максимізувати). Трохи гірше проявила себе функція втрат hinge loss. Із попереднього опису зрозуміло, чому. Нарешті найбільш нестійкою до викидів виявилася квадратична функція втрат. За негативне та позитивне відхилення від гіперплощини вона накладає найбільш стрімко зростаючий штраф.

3.3. Порівняння збіжності та перфомансу

Для порівняння перфомансу використав датасет «breast-cancer-wisconsin». Для кожного випадку робив по 5 замірів і шукав усереднене значення. Процедура, яку я використовував, називається 5-fold cross-validation. Наш масив даних розміру 683 (після видалення неповних даних) розбивався на 5 частин і кожна з них по черзі використовувалася для тестування (20% даних).

Спочатку дослідив збіжність методів (Рис. 7). Усі вони, і градієнтний, і субградієнтний, зійшлися до сталого значення приблизно однаково швидко. Можна звернути увагу на певну флуктуаційність субградієнтного методу на малій кількості ітерацій, це пов’язано насамперед із негарантованою збіжністю до оптимуму (ми наближаємося лише до мінімізатора, але не завжди до мінімуму).

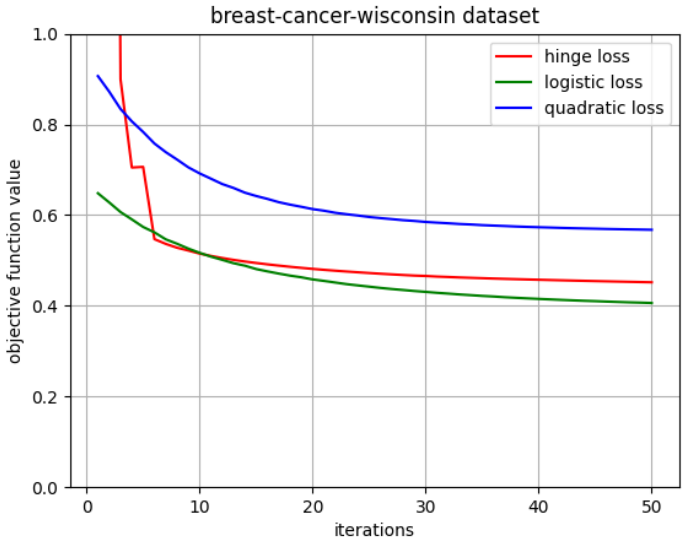

Рис. 7. Залежність значення цільової функції (функції втрат) від кількості ітерацій оптимізаційного методу.

Також провів дослідження залежності точності тестування моделі від кількості ітерацій методів оптимізації (Рис. 8). Виявив, що найбільш швидко та стабільно високої точності досягає саме hinge loss (приблизно за 4 ітерації), оптимізований субградієнтним методом.

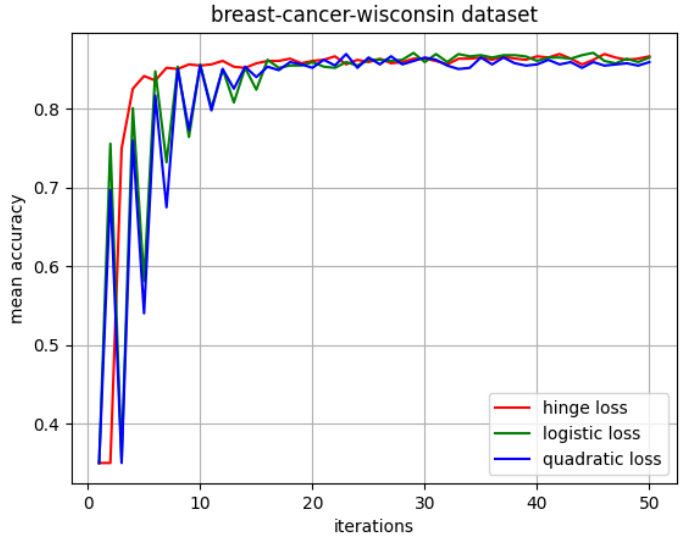

Рис. 8. Залежність середньої точності тестування моделі від кількості ітерацій оптимізаційного методу.

Це, у тому числі, пояснюється вдалим вибором кроку, іншим методам на малій кількості ітерацій недостатньо лінійного пошуку. Проте приблизно за 20 ітерацій усі методи досягають майже однакової сталої точності 86-87%.

Нарешті провів виміри середнього часу тренування моделі в залежності від кількості ітерацій оптимізаційного методу (Рис. 9). Виявив, що субградієнтний метод виконується вагомо швидше, ніж градієнтний аналог.

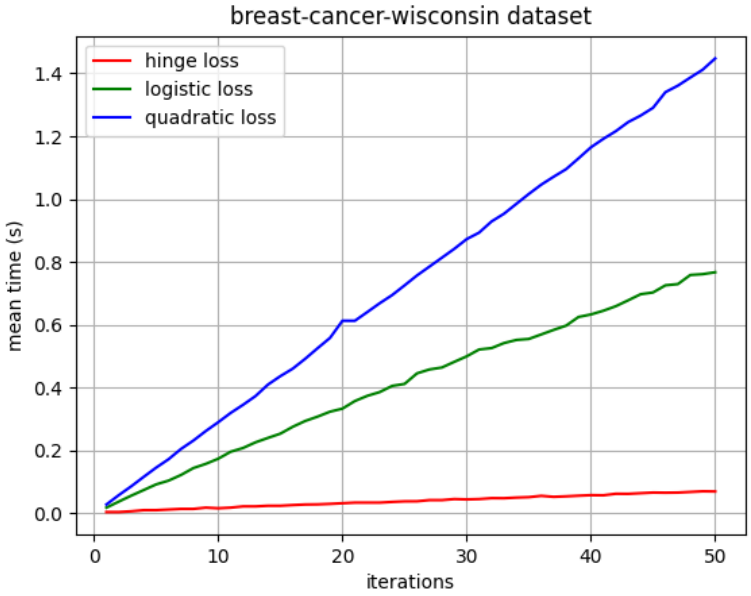

Рис. 9. Залежність середнього часу тренування моделі від кількості ітерацій оптимізаційного методу.

Повільність роботи градієнтного методу пов’язана з використанням лінійного пошуку кроку – також ітеративної процедури. Тому вирішив протестувати результати роботи ще раз, але замінивши лінійний пошук на просте правило $\frac{1}{k}$, яке використовує субградієнтний пошук (Рис. 10). Тепер бачимо, що час роботи алгоритмів відрізняється вже дещо менше. Остаточна різниця також пояснюється складністю обчислень. Hinge loss є найбільш простою функцією, що містить лише одну операцію множення (коли рахуємо похідну

безпосередньо від функції втрат без усереднення і терміну регуляризації), тоді як quadratic loss містить аж п’ять, а logistic loss – взагалі піднесення у степінь.

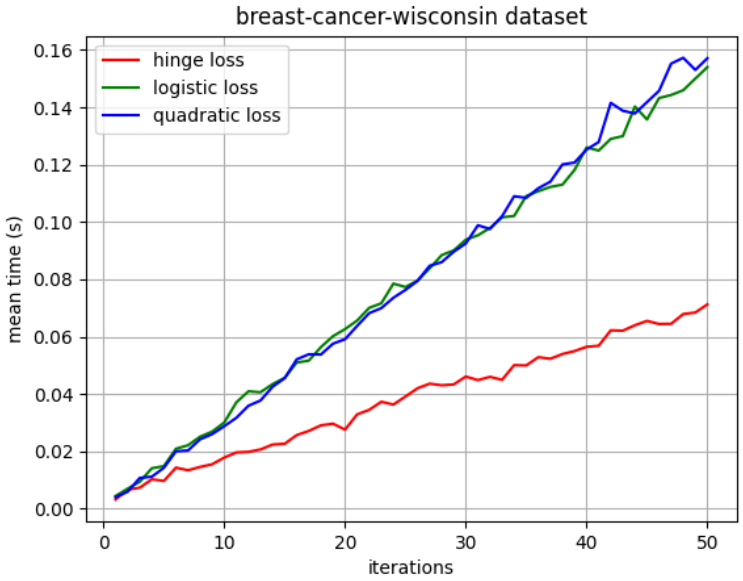

Рис. 10. Залежність середнього часу тренування моделі від кількості ітерацій оптимізаційного методу (з кроком $\frac{1}{k}$).

Крім того, виявляється, що у випадку квадратичної функції втрат градієнтний метод не збігається (Рис. 11). Логістична функція втрат при такому кроці досягає високої точності майже одночасно з hinge loss (Рис. 12).

Отже, що субградієнтний спуск, що hinge loss гарно проявили себе на практиці, володіючи достатньою простотою та ефективністю.

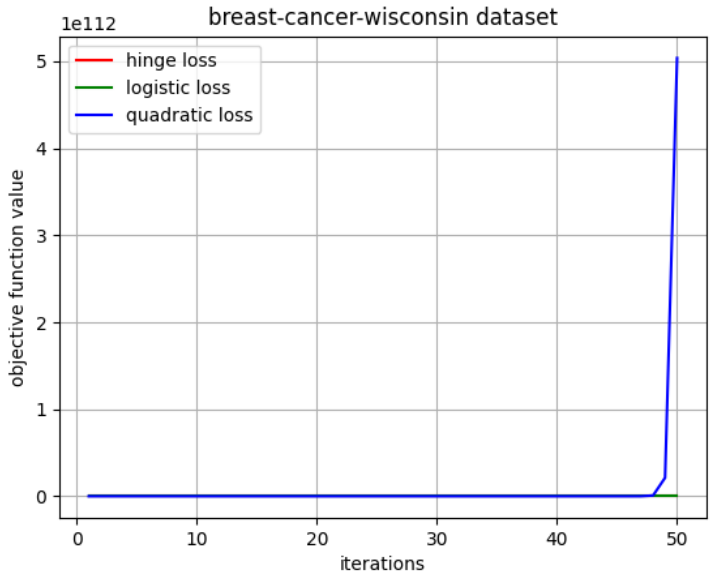

Рис. 11. Залежність значення цільової функції (функції втрат) від кількості ітерацій оптимізаційного методу (з кроком $\frac{1}{k}$).

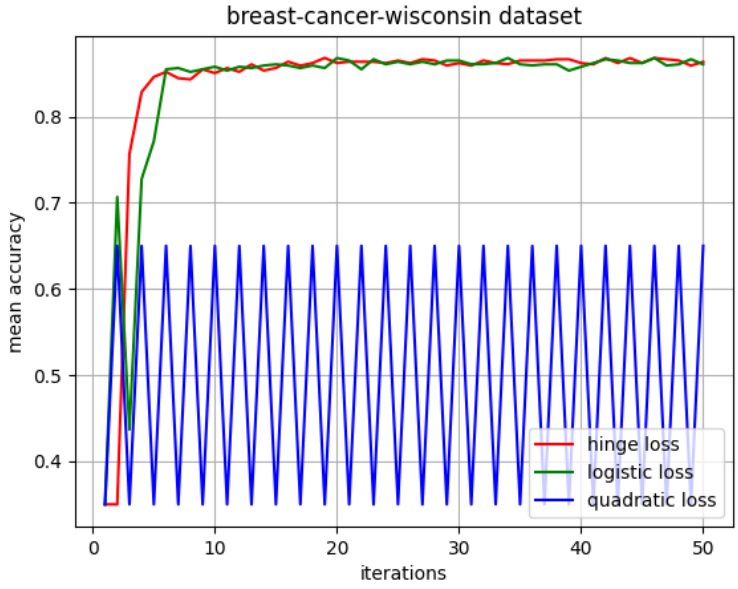

Рис. 12. Залежність середньої точності тестування моделі від кількості ітерацій оптимізаційного методу (з кроком $\frac{1}{k}$).

ВИСНОВКИ

В даній курсовій роботі було розглянуто поняття субградієнта, його практична роль в методах машинного навчання, а саме в оптимізаційних задачі класифікації. Більш детально розібрали метод опорних векторів, в основі якого лежить мінімізація деякої математичної функції відносно наявного набору даних. Метод опорних векторів зводить навчання класифікатора до оптимізаційної задачі, яка розв’язується евристичними алгоритмами.

Провели ряд експериментів, де порівняли різні функції втрат для первинної задачі методу опорних векторів. Переконалися в ефективності застосування саме субградієнтного методу та функції втрат hinge loss. Незважаючи на очевидну ефективність досліджуваної функції, вона все ж програє логістичній функції втрат у робастності, але різниця досить невелика. Субградієнтний метод довів свою швидкість та високу точність за достатньої кількості ітерацій, хоча для нього й не існує залізних гарантій досягнення оптимуму чи простих правил зупинки, як у випадку градієнтного спуску.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Моклячук М. П. Основи опуклого аналізу. Навчальний посібник. – Київ, Видавництво ТВіМС, 2004, 236 с. – С. 161-162
2. Методы оптимизации. Ч. I. Введение в выпуклый анализ и теорию оптимизации: учебное пособие / В. Г. Жадан – М.: МФТИ, 2014. – 271 с. ISBN 978-5-7417-0514-8 (Ч. I) – С. 88
3. Boyd, Stephen P. Convex Optimization / Stephen Boyd & Lieven Vandenberghe p. cm. Includes bibliographical references and index. ISBN 0-521-83378-7 – С. 480
4. [Електронне джерело] Режим доступу до ресурсу: https://web.stanford.edu/class/ee364b/lectures/subgrad_method_notes.pdf
5. [Електронне джерело] Режим доступу до ресурсу: https://www.stat.cmu.edu/~ryantibs/convexopt-F13/scribes/lec6.pdf
6. [Електронне джерело] Режим доступу до ресурсу: https://www.cs.utah.edu/~zhe/pdf/lec-19-2-svm-sgd-upload.pdf
7. [Електронне джерело] Режим доступу до ресурсу: http://www.ccas.ru/voron/download/SVM.pdf
8. [Електронне джерело] Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Support-vector_machine
9. [Електронне джерело] Режим доступу до ресурсу: https://www.youtube.com/watch?v=1oi_Mwozj5w&list=PLnZuxOufsXnvftwTB1HL6mel1V32w0ThI&index=9
10. [Електронне джерело] Режим доступу до ресурсу: https://www.youtube.com/watch?v=jYtCiV1aP44&list=PLnZuxOufsXnvftwTB1HL6mel1V32w0ThI&index=11