КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет комп'ютерних наук та кібернетики
Кафедра інтелектуальних програмних систем

Кваліфікаційна робота

на здобуття ступеня бакалавра

за спеціальністю
121 Інженерія програмного забезпечення
на тему:

РОЗРОБКА PYTHON-ПАКЕТУ ДЛЯ НАВЧАННЯ МОДЕЛЕЙ З ВИКОРИСТАННЯМ МЕТОДУ SVM ТА СУБГРАДІЄНТНОГО ПІДХОДУ

Виконав

студент 4-го курсу
Дмитро ПАЩЕНКО _____
(підпис)

Науковий керівник:

асистент, доктор філософії з прикладної математики
Віктор СТОВБА _____
(підпис)

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

Студент _____
(підпис)

Роботу розглянуто й допущено до захисту на засіданні кафедри інтелектуальних програмних систем «25» травня 2022 р., протокол № 10

Завідувач кафедри
Олександр ПРОВОТАР _____
(підпис)

Київ – 2022

РЕФЕРАТ

Обсяг роботи 30 сторінок, 13 ілюстрацій, 2 таблиці і 14 джерел посилання.

Ключові слова: СУБГРАДІЄНТНИЙ МЕТОД, КЛАСИФІКАЦІЯ, МЕТОД ОПОРНИХ ВЕКТОРІВ, SVM, HINGE LOSS, SCIKIT-LEARN.

Об’єктом роботи є субградієнтний метод, недиференційовна опукла функція втрат hinge loss, яка фігурує в контексті задачі бінарної класифікації методом опорних векторів, а також scikit-learn API для створення власних Python-пакетів для машинного навчання.

Метою роботи є розробка власного Python-пакету для навчання моделей бінарній класифікації з використанням методу опорних векторів (SVM) та субградієнтного підходу.

Засоби реалізації: мова програмування Python, бібліотека NumPy для швидких алгебраїчних обчислень, бібліотека Pandas для роботи з даними статистичних вибірок, бібліотека машинного навчання scikit-learn, середовище тестування Pytest.

У ході роботи переконалися в коректності реалізованого пакету, а також дослідили його відмінності від аналогічних інструментів у scikit-learn. Обґрунтували доцільність використання функції втрат hinge loss.

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 hinge loss. Ця робота описує реалізацію зручного інструменту, сумісного з бібліотекою scikit-learn, для навчання моделей з використанням методу опорних векторів (SVM) та субградієнтного підходу.

Актуальність. Класифікація є однією з найпопулярніших задач машинного навчання, метод опорних векторів (SVM) – одним із найкращих лінійних класифікаторів. Субградієнтний метод дозволяє розширити спектр функцій втрат придатних для оптимізації. Деякі з таких функцій вдосконалюють властивості методу SVM. Тож, виникає необхідність у зручному та інтуїтивно зрозумілому інструменті для навчання моделей вказаним методом.

Мета й завдання роботи. Розробити власний Python-пакет для навчання моделей бінарній класифікації з використанням методу опорних векторів (SVM) та субградієнтного підходу.

Об’єктом дослідження є субградієнтний метод та недиференційовна опукла функція втрат hinge loss, її аналоги, метод опорних векторів (SVM), його параметри, бібліотека scikit-learn.

Предметом дослідження є субградієнтний метод, метод опорних векторів (SVM) та розробка Python-пакетів

1. СУБГРАДІЄНТНИЙ МЕТОД

1.1. Опуклі функції та опукла оптимізація. Градієнтний спуск

Теореми та означення для опуклих функцій візьмемо з посібника Моклячука [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 [3].

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. 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 полягає 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

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 бік негативного субградієнта.

Може статися так, що $-g^{(k)}$ не є напрямком спуску для f у $x^{(k)}$, тобто ітерація субградієнтного методу може збільшити цільову функцію. Проте ключовий аспект алгоритму полягає в тому, що при достатньо малому кроці субградієнтний метод прямує до мінімізатора [14]. 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.

Запишемо алгоритм на псевдокоді.

```
procedure SubgradientDescent(T):
    x_0 = 0_n
    f_min = +∞
    x_min = x_0
    for t = 0, ..., T − 1 do
        Compute g_t = ∇f(x_t)
        Update x_{t+1} = x_t − α_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
```

1.4. Стратегії вибору розміру кроку

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 не залежить від k.

- Постійна довжина кроку.

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

- 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 $\lim_{k\to\infty} f\left(x^{(k)}\right) = f^*$, зі швидкістю $O(1/\varepsilon^2)$ [10].

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.

- Поляк пропонує розмір кроку, який можна використовувати, коли відоме оптимальне значення f^*, і в певному сенсі є оптимальним [4]. Такий крок обчислюється за формулою:

$$\alpha_k = \frac{f\left(x^{(k)}\right) - f^*}{\|g^{(k)}\|_2^2}.$$

Ми можемо досягти схожого результату, наближено обчисливши оптимальне значення f^* у вигляді $f_{best} - \gamma^k$, де $\gamma^k > 0$ та $\gamma^k \to 0$. Маємо наступний крок:

$$\alpha_k = \frac{f\left(x^{(k)}\right) - f_{best}^{(k)} + \gamma_k}{\|g^{(k)}\|_2^2}.$$

Кроки Поляка збігаються до оптимального значення з тією ж швидкістю збіжності: $O(1/\varepsilon^2)$ [10].

1.5. Стохастичний субградієнтний метод

Стохастичні методи корисні для оптимізації великої суми функцій замість однієї функції. Наприклад, у випадку мінімізації емпіричного ризику. Розглянемо таку задачу мінімізації:

$$\min_x \sum_{i=1}^{m} f_i(x).$$

Стохастичний субградієнтний метод повторює наступні оновлення кроку:

$$x^{(k)} = x^{(k-1)} - t_k g_{i_k}^{(k-1)}, \qquad k = 1, 2, 3, \dots$$

де $i_k \in \{1, \dots, m\}$ – певний випадковий індекс на k-ій ітерації, що обирається випадково або за циклічним правилом, і $g_i^{(k-1)} \in \partial f_i(x^{(k-1)})$. Різниця між таким оновленням і методом субградієнта полягає в тому, що ми уникаємо обчислення повної суми $\sum_{i=1}^m g_i^{(k-1)}$ на кожній ітерації. Також зауважимо, що коли кожна функція f_i, $i = 1, \dots, m$ диференційовна, метод зводиться до звичайного стохастичного градієнтного спуску (SGD).

Як згадувалося, існує два правила для вибору індексу i_k на ітерації k:

- Циклічне: обираємо $i_k = 1, 2, \dots, m, 1, 2, \dots, m, \dots$.
- Рандомізоване: обираємо $i_k \in \{1, \dots, m\}$ випадковим чином (розподіл рівномірний).

Рандомізоване правило використовується частіше, оскільки воно захищає від найгіршого випадку або протилежного сценарію [10].

Чим стохастичний субградієнтний метод відрізняється від пакетного (batch) субградієнтного методу? З точки зору обчислень ми знаємо, що m стохастичних кроків приблизно відповідають одному пакетному кроку, але головна перевага полягає в тому, що нам не потрібно зачіпати всі дані при застосуванні стохастичного кроку (різницю див. на рис. 2 [10]).

Зазначимо, що всі кроки, наведені в пункті 1.4, дійсні й для стохастичного субградієнтного методу.

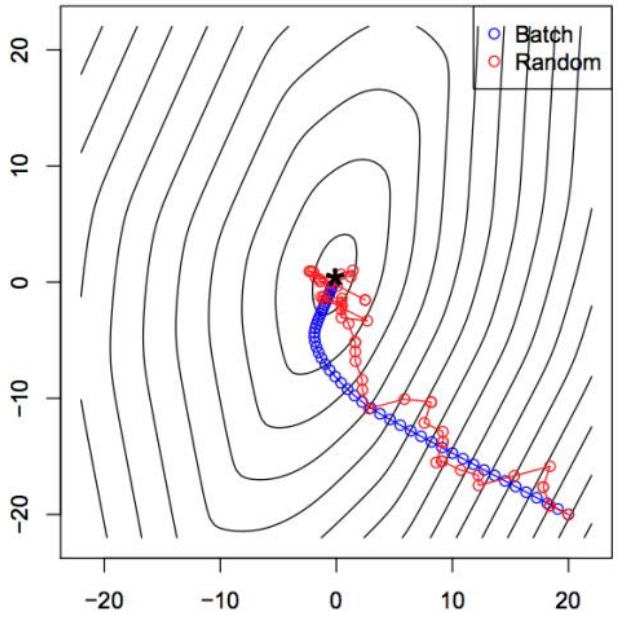


Рис. 2. Порівняння роботи пакетного (batch) та стохастичного (random) субградієнтних методів.

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 обираємо 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. 3. Важливо відмітити, що в цій роботі термін зміщеності b прирівняли до нуля.

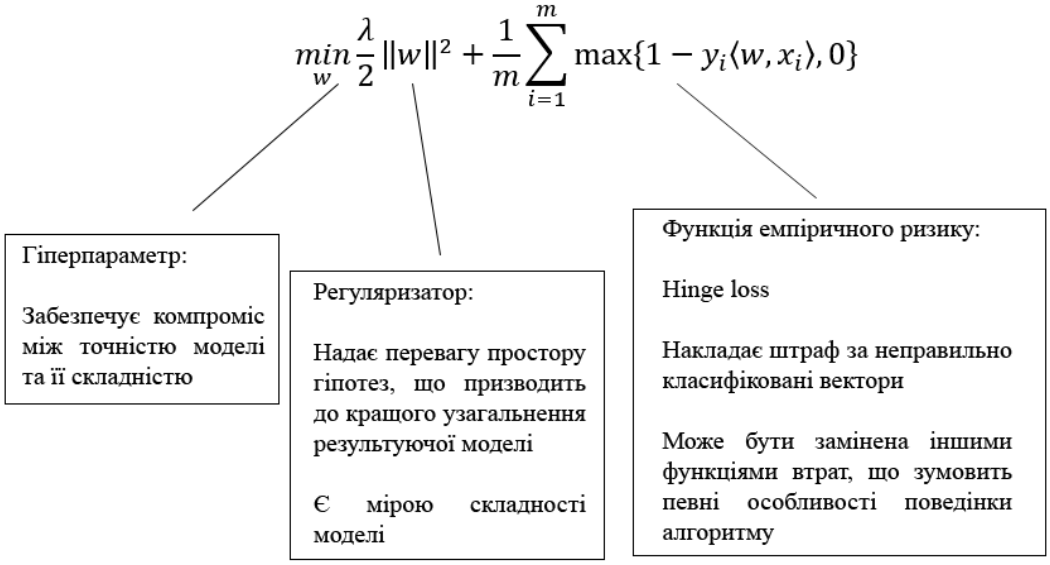


Рис. 3. Пояснення моделі лінійного методу опорних векторів (soft margin).

Завдання методу опорних векторів зазвичай сприймається як проблема обмеженого квадратичного програмування. Однак такий підхід дуже повільний на великих даних [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) [13]. 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 [13]. 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. 4):

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

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).

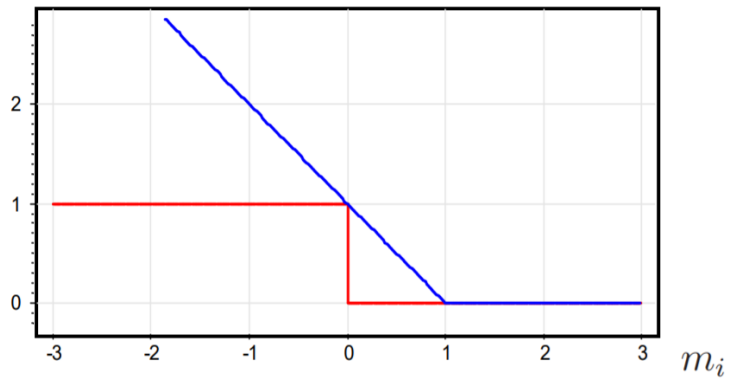


Рис. 4. Функції 0-1 loss (червоним) та hinge loss (синім)

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

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 (рис. 5 [13]), зокрема:

- logistic loss

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

- square loss

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

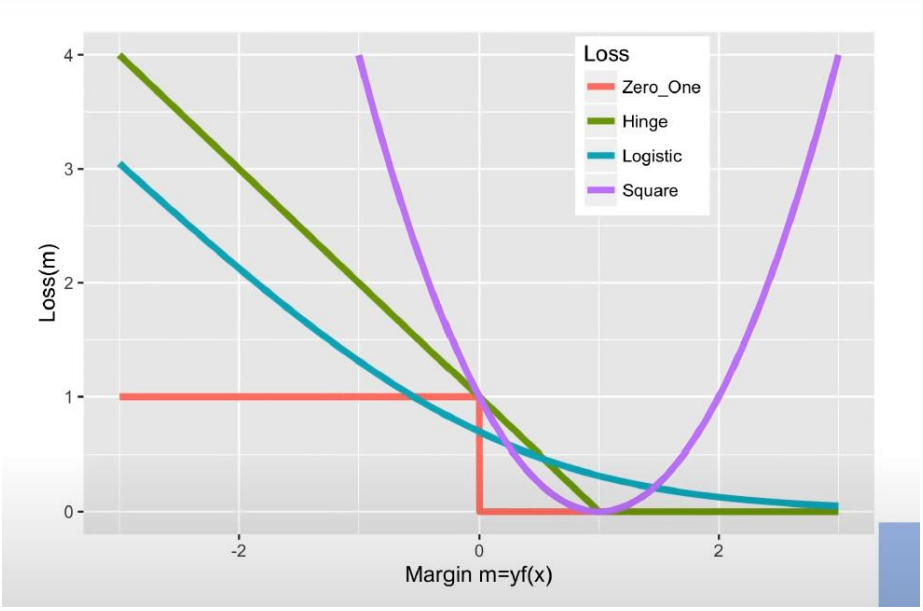


Рис. 5. Графіки різних функцій втрат.

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. Усі ці функції однаково застосовні до задачі бінарної класифікації. Наведемо їх похідні, значення яких обчислюються в процесі мінімізації:

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

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

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

2.3. Ядровий метод опорних векторів

Лінійний метод опорних векторів дозволяє точно класифікувати дані, які є лінійно роздільними. Що робити у випадку класифікації нелінійних даних? Для цього у машинному навчанні існує ядровий метод, який дозволяє нам застосовувати лінійні класифікатори до нелінійних задач шляхом відображення нелінійних даних у простір більшої розмірності.

Однією з головних переваг SVM є те, що його можна використовувати з ядрами замість прямого доступу до вектора ознак x. Оптимальний розв'язок рівняння (6) можна виразити у вигляді лінійної комбінації навчальних екземплярів. Тому можна навчати та використовувати SVM без прямого доступу до навчальних екземплярів, а натомість отримати доступ лише до їх внутрішніх продуктів через оператор ядра. Тобто замість того, щоб розглядати класифікатори, які є лінійними функціями навчальних екземплярів x, ми розглядатимемо класифікатори, які є лінійними функціями деякого неявного відображення $\varphi(x)$ екземплярів. Тоді навчання передбачає вирішення проблеми мінімізації [12]:

$$\min_{w} \frac{\lambda}{2}\|w\|^2 + \frac{1}{m}\sum_{i=1}^{m} \ell\big(w; (\varphi(x_i), y_i)\big), \tag{8}$$

де

$$\ell\big(w; (x_i, y_i)\big) = max\{0, 1 - y_i\langle w, \varphi(x_i)\rangle\}. \tag{9}$$

Однак відображення $\varphi(\cdot)$ ніколи не вказується явно, а за допомогою оператора ядра $K(x, x') = \langle\varphi(x), \varphi(x')\rangle$, який дає внутрішні добутки після відображення $\varphi(x)$.

Приклади ядрових функцій:

- $K(x, y) = (x^T y + c)^d$ – поліноміальне ядро;
- $K(x, y) = e^{-\frac{1}{2\sigma^2}\|x-y\|^2}$ – гауссове ядро;

- $K(x, y) = e^{-\gamma\|x-y\|^2}$ – RBF ядро.

Алгоритм SVM можна легко реалізувати, підтримуючи вектор α. На кожній ітерації t нехай $\alpha_{t+1} \in \mathbb{R}^m$ буде таким вектором, що $\alpha_{t+1}[j]$ підраховує, скільки разів було вибрано приклад j на даний момент, і ми мали ненульові втрати на ньому. Замість зберігання в пам'яті вагового вектора w_{t+1} ми виражатимемо його через α_{t+1} [12]:

$$w_{t+1} = \frac{1}{\lambda t}\sum_{j=1}^{m} \alpha_{t+1}[j]\, y_j \varphi(x_j)$$

Оскільки ітерації w_t залишаються колишніми (змінюється лише їх подання), гарантії точності після ряду ітерацій зберігають дійсність. Псевдокод цієї ядрової реалізації SVM наведено на рис. 6 [12].

INPUT: S, λ, T

INITIALIZE: Set $\boldsymbol{\alpha}_1 = 0$

FOR $t = 1, 2, \ldots, T$

Choose $i_t \in \{0, \ldots, |S|\}$ uniformly at random.

For all $j \neq i_t$, set $\alpha_{t+1}[j] = \alpha_t[j]$

If $y_{i_t} \frac{1}{\lambda t} \sum_j \alpha_t[j] y_{i_t} K(\mathbf{x}_{i_t}, \mathbf{x}_j) < 1$, then:

Set $\alpha_{t+1}[i_t] = \alpha_t[i_t] + 1$

Else:

Set $\alpha_{t+1}[i_t] = \alpha_t[i_t]$

OUTPUT: $\boldsymbol{\alpha}_{T+1}$

Рис. 6. Псевдокод ядрового методу опорних векторів.

3. ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1. Scikit-learn API

Розроблений інструмент для навчання моделей класифікації є сумісним з бібліотекою машинного навчання scikit-learn. Для цього були виконані вимоги відповідного API [11]. Основні об'єкти scikit-learn, інтерфейси яких було імплементовано, вказані в табл. 1.

Estimator	Базовий об’єкт реалізує метод fit для вивчення даних (навчання з учителем): `estimator = estimator.fit(data, targets)`

Predictor	Для навчання з учителем реалізує:
	`prediction = predictor.predict(data)`
	Алгоритми класифікації зазвичай також пропонують спосіб кількісної оцінки впевненості прогнозу, використовуючи функцію decision_function або predict_proba:
	`probability = predictor.predict_proba(data)`

Model	Модель, яка може виміряти якість передбачення на нових даних, реалізує (чим вище, тим краще): `score = model.score(data)`

Табл. 1. Основні об’єкти scikit-learn API, інтерфейси яких було імплементовано.

Клас-оцінювач (Estimator) створеного пакету SubgradientSVMClassifier наслідує бібліотечні класи BaseEstimator та ClassifierMixin. Окрім надання необхідних інтерфейсів, завдяки цьому, деякі методи, як-от get_params() та set_params(), а також метод клонування base.clone(), реалізуються автоматично. Для повної сумісності було задоволено безліч інших вимог рекомендаційного характеру, повний список яких можна переглянути за посиланням [11].

Коректність створеного класу було перевірено бібліотечним методом sklearn.utils.estimator_checks.check_estimator(). Ця функція запускає великий набір тестів для перевірки вхідних даних, розмірностей тощо, щоб переконатися, що об’єкт оцінювача відповідає умовам scikit-learn, які детально описано в главі «Rolling your own estimator» [11]. Також запускаються додаткові тести для класифікаторів як результат наслідування від відповідного міксину sklearn.base (для тестування лише бінарної класифікації до класу був доданий тег «binary_only»).

Інтеграція пакету з scikit-learn дозволяє широкий спектр застосування інструментів бібліотеки з класом оцінювача. Наприклад, за допомогою модуля sklearn.utils.multiclass можна здійснювати мультикласову класифікацію (OneVsOneClassifier або OneVsRestClassifier), передаючи оцінювач як параметр. Також створений клас може виступати елементом sklearn.Pipeline, завершуючи ланцюг трансформацій вхідних даних.

3.2. Структура пакету

Пакет написано на базі шаблона scikit-learn для створення розширень, сумісних з бібліотекою. Він допомагає розробити оцінювачі, які можна використовувати в конвеєрах scikit-learn (pipelines) і для пошуку (гіпер)параметрів, одночасно полегшуючи тестування (включаючи певну відповідність API), документацію, пакування (packaging) та подальшу інтеграцію. Для збірки пакету необхідно запустити файл setup.py.

Головним класом, класом оцінювача, є SubgradientSVMClassifier. При його створенні налаштовуються параметри SVM (для кожного задано значення за замовчуванням). Перелічимо їх (див. табл. 2).

Параметр	Опис	Допустимі значення
loss	Функція втрат	"hinge", "logistic", "quadratic"
iterations	Кількість ітерацій субградієнтного методу	int
batch_size	Розмір тренувального пакету (при batch_size=1 маємо стохастичну версію субградієнтного методу)	int
regularizer	Регуляризатор	float
step_size_rule	Правило вибору довжини кроку	"constant" (α), "diminishing" (α/k), "polyak" (крок Поляка з параметром α/k)
alpha	Параметр кроку	float
kernel	Ядрова функція	"linear", "quadratic" (поліноміальне ядро з $d = 2$), "rbf"
gamma	Параметр ядрової функції (RBF)	float

Табл. 2. Параметри класу SubgradientSVMClassifier.

Клас SubgradientSVMClassifier містить наступні методи (рис. 7): fit(X, y), predict(X), predict_proba(X), score(X, y, sample_weight=0), decision_function(X).

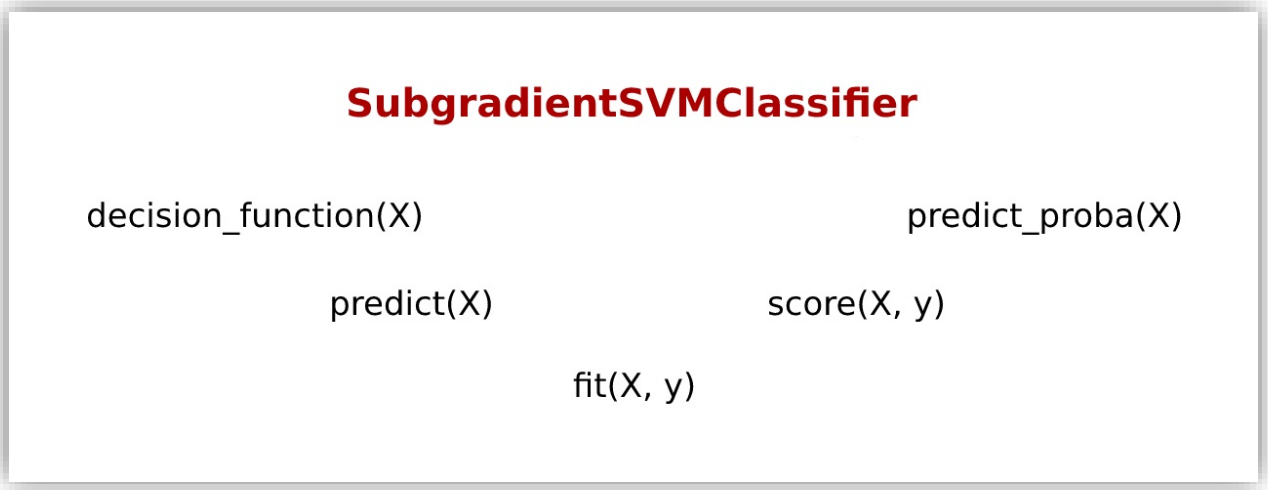


Рис. 7. Методи класу SubgradientSVMClassifier.

3.3. Порівняння з аналогами. Демонстрація роботи

Створений оцінювач порівняли з оцінювачами з scikit-learn: sklearn.svm.SVC (працює на основі алгоритму libsvm) та sklearn.svm.LinearSVC (працює на основі алгоритму liblinear). У якості тестового набору даних було обрано «adult dataset» (більше 30 тисяч спостережень). Оцінювачі перевірялися, починаючи з вибірки розміром 1000, і далі з кроком 1000 до розміру 30000. Розбиття вибірки на тренувальну та тестову, а також усереднення значень досягли за допомогою sklearn.model_selection.cross_validate() з параметром cv=5. Результат підбору параметрів оцінювачів наступний:

- SubgradientSVMClassifier: loss="hinge", iterations=1000, batch_size=100, regularizer=1e-2, step_size_rule="diminishing", alpha=1e-3.
- SVC: параметри за замовчуванням.
- LinearSVC: max_iter=1000, C=1e-2, інші параметри за замовчуванням.

Результати вимірювань можна побачити на рис. 8-10.

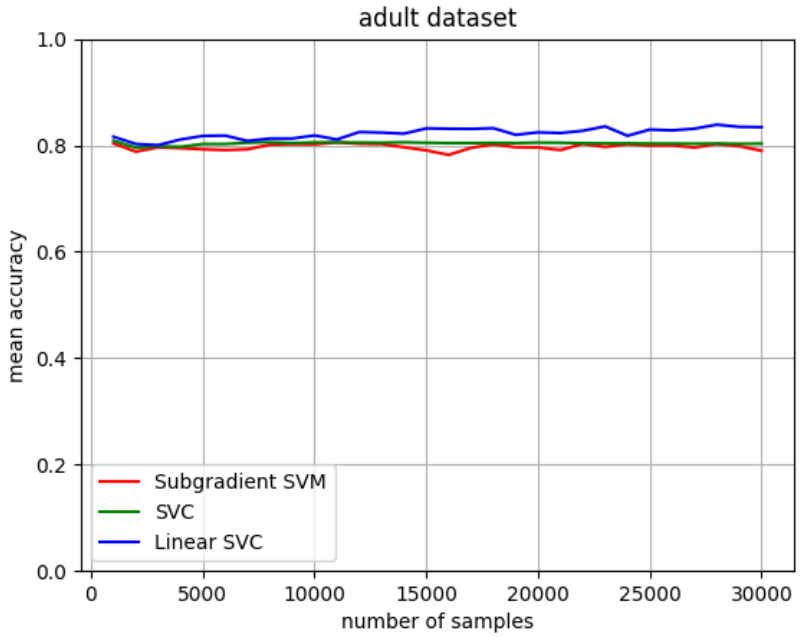


Рис. 8. Залежність середньої точності тренування моделей від розміру вибірки.

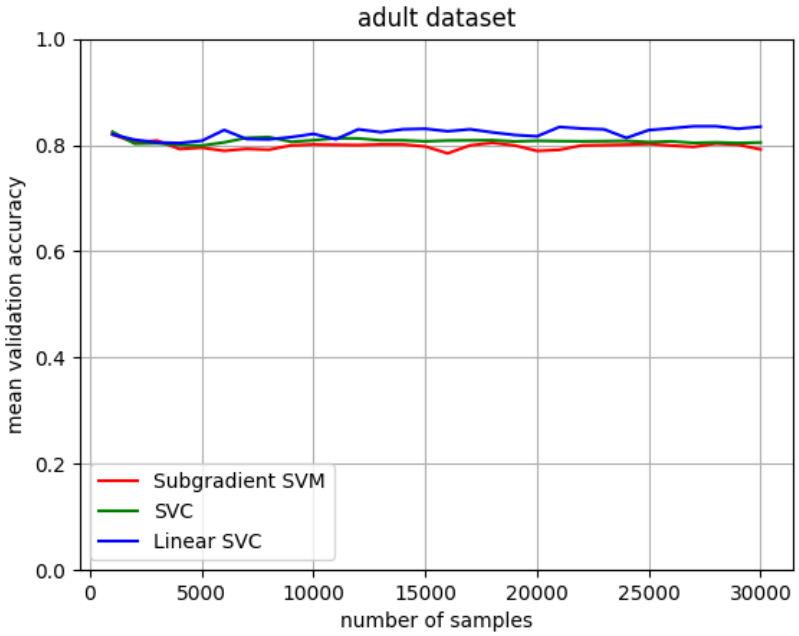


Рис. 9. Залежність середньої точності тестування моделей від розміру вибірки.

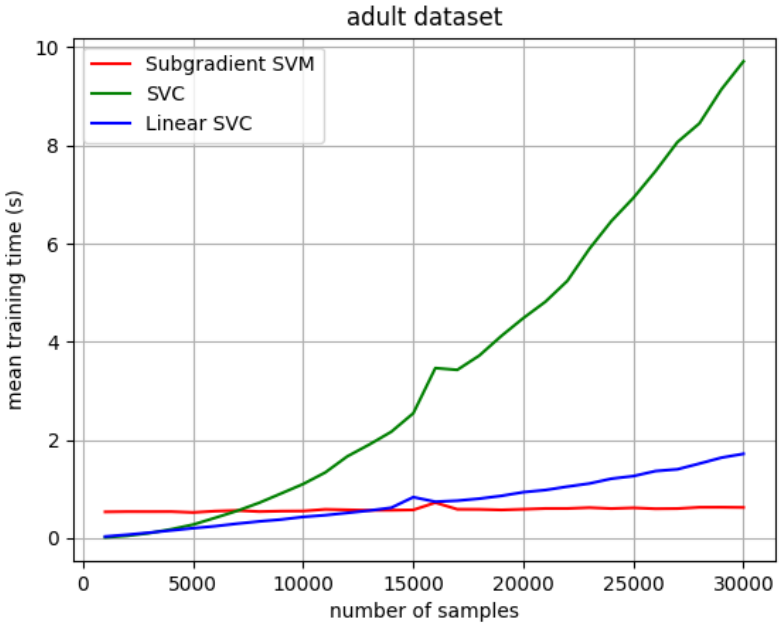


Рис. 10. Залежність середнього часу тренування моделей від розміру вибірки.

Бачимо, що за майже однакової точності тренування і тестування середній час тренування дуже відрізняється. Чим більше даних у вибірці, тим повільнішими стають точні алгоритми, як-от libsvm та liblinear. Натомість наближений субградієнтним методом mini-batch алгоритм SVM показав чудові результати на великих наборах даних, оскільки не є чутливим до розміру вибірки.

Також в рамках створеного пакету можна порівняти різні функції втрат, обґрунтувавши доцільність використання саме недиференційовної функції втрат hinge loss. Для дослідів використали датасет «breast-cancer-wisconsin» об’ємом у 683 спостереження. Результат підбору параметрів наступний: regularizer=1e-4, step_size_rule="diminishing", alpha=1. Результати вимірювань можна побачити на рис. 11.

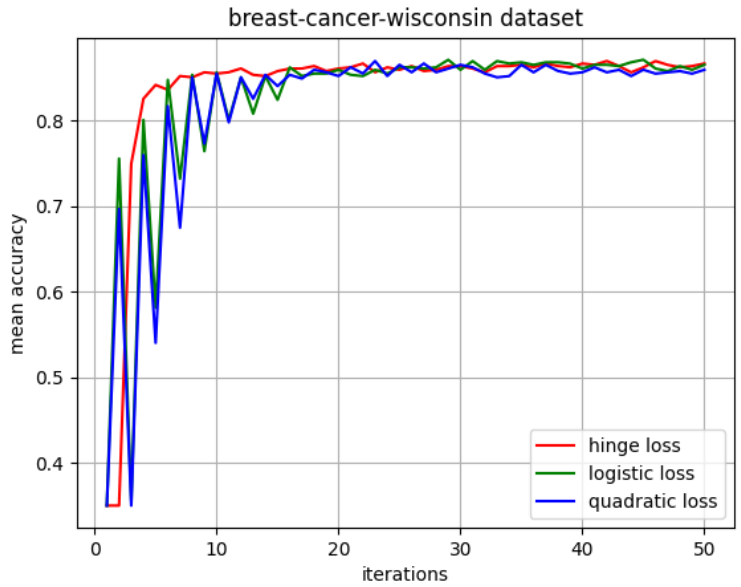
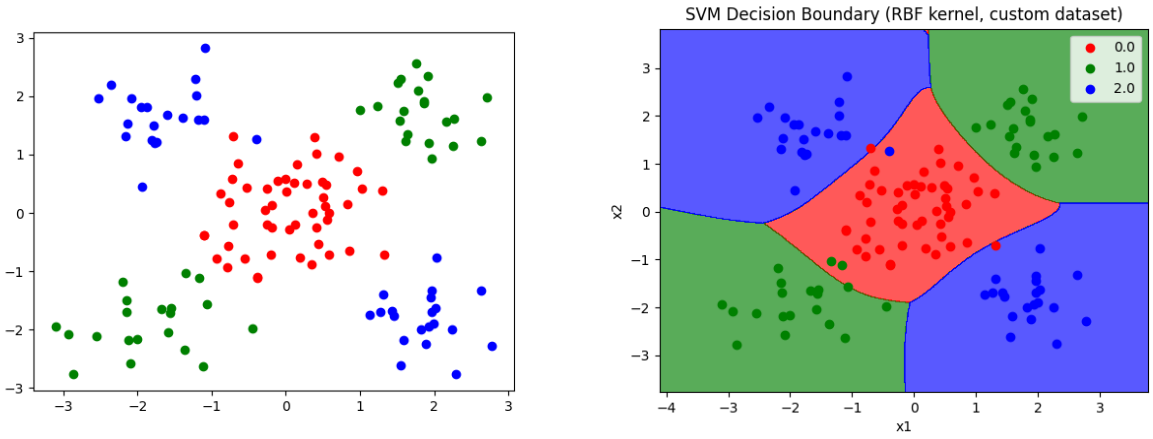


Рис. 11. Залежність середньої точності тестування моделі від кількості ітерацій оптимізаційного методу.

Переконалися у кращій швидкості збіжності та точності моделі саме при використанні функції втрат hinge loss, що чудово поєднується з арифметичною простотою її обчислення.

Насамкінець розглянули можливості ядрового SVM (з RBF-ядром), застосувавши класифікатор до власного набору даних. Дані належать до трьох класів і не є лінійно роздільними (рис. 13а). Створили об’єкт оцінювача з наступними параметрами: iterations=30, regularizer=0.1, alpha=0.1, kernel="rbf", gamma=1. Отриманий оцінювач передали в конструктор класу sklearn.multiclass.OneVsRestClassifier, що призначений для мультикласової класифікації на базі бінарної за допомогою стратегії «один проти інших». Результат роботи створеного класифікатора можна побачити на рис. 13б.



а) б)

Рис. 13. Використання SVM з RBF-ядром для мультикласової класифікації даних з власного набору: а) розподіл даних на координатній площині, кожному класу відповідає свій колір; б) результат класифікації – графік порогу прийняття рішень.

ВИСНОВКИ

Розроблено Python-пакет для розв’язання задачі класифікації методом опорних векторів (SVM) з використанням субградієнтного підходу. Реалізована інтеграція з scikit-learn, що дозволяє виконувати мультикласову класифікацію на базі бінарної, використовувати оцінювач як елемент scikit-learn pipeline тощо. Проведено тестування створеного пакету на базі методу sklearn.utils.estimator_checks.check_estimator(). Досліджено аналогічні інструменти в scikit-learn та знайдено умови доречності використання саме нашого описаного в роботі пакету. Проведено порівняльний аналіз різних функцій втрат. Обґрунтовано доцільність використання hinge loss та субградієнтного підходу. Продемонстровано можливості ядрового методу опорних векторів на основі субградієнтного методу.

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, 464
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: http://gr.xjtu.edu.cn/c/document_library/get_file?folderId=2466728&name=DLFE-110532.pdf
10. [Електронне джерело] Режим доступу до ресурсу: https://www.stat.cmu.edu/~ryantibs/convexopt-F16/scribes/prox-grad-scribed.pdf

11. [Електронне джерело] Режим доступу до ресурсу: https://scikit-learn.org/stable/developers/develop.html
12. [Електронне джерело] Режим доступу до ресурсу: https://home.ttic.edu/~nati/Publications/PegasosMPB.pdf
13. [Електронне джерело] Режим доступу до ресурсу: https://www.youtube.com/watch?v=1oi_Mwozj5w&list=PLnZuxOufsXnvftwTB1HL6mel1V32w0ThI&index=9
14. [Електронне джерело] Режим доступу до ресурсу: https://www.youtube.com/watch?v=jYtCiV1aP44&list=PLnZuxOufsXnvftwTB1HL6mel1V32w0ThI&index=11