1.1 Метод опорних векторів (Support Vector Machine)

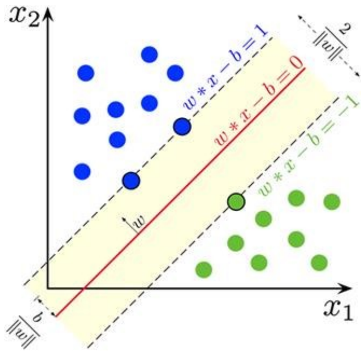
Розглянемо метод опорних векторів (англ. SVM, Support Vector Machine) для задач класифікації. Буде представлено основну ідею алгоритму, виведення налаштування його ваг і розібрано просту реалізацію своїми руками. На прикладі датасету $Iris$ буде продемонстровано роботу написаного алгоритму з лінійно розділеними/нерозділними даними у просторі R^2. Додатково будуть озвучені плюси та мінуси алгоритму, його модифікації.

1.1.1 Завдання

Вирішуватимемо завдання бінарної (коли класів всього два) класифікації. Спочатку алгоритм тренується на об'єктах з навчальної вибірки, котрим заздалегідь відомі мітки класів. Далі вже навчений алгоритм передбачає мітку класу для кожного об'єкта з відкладеної/тестової вибірки. Мітки класів можуть набувати значень $Y = \{-1, +1\}$. Об'єкт – вектор з N ознаками $X = (x_1, x_2, \ldots, x_n)$ у просторі R^n. При навчанні алгоритм повинен побудувати функцію $F(X) = Y$, яка приймає аргумент X – об'єкт з простору R^n і видає мітку класу Y. Головна мета SVM як класифікатора – знайти рівняння роздільної гіперплощини $w_1x_1 + w_2x_2 + \ldots + w_nx_n + w_0 = 0$ в просторі R^n, яка розділила б два класи якимось оптимальним чином.

1.1.2 Загальні відомості про алгоритм

Для простоти розглянемо вибірку об'єктів з двома ознаками x_1 та x_2 які належать до двох різних класів. Ілюстрацію наведено на рисунку 3.1. Загальний вид перетворення F об'єкта X на мітку класу Y: $F(X) = sign(w^TX - b)$, де $w = (w_1, w_2, \ldots, w_n), b = -w_0$. Після налаштування ваг алгоритму w та b(навчання), всі об'єкти, що потрапляють по одну сторону від побудованої гіперплощини, передбачатимуться як перший клас, а об'єкти, що потрапляють по інший бік – другий клас.



Робота алгоритму SVM

Сині кружки – об'єкти першого класу; зелені – об'єкти другого класу; кружки з чорним контуром – опорні об'єкти; пряма $w * x - b = 0$ оптимально розділяє два класи; прямі $w * x - b = \pm 1$, які проходять через опорні вектори x_+ та x_- визначають ширину розділяючої полоси, як проекцію вектора $(x_+ - x_-)$ на вектор нормалі до розділяючої прямої (гіперплощини) w.

Усередині функції $sign()$ стоїть лінійна комбінація ознак об'єкта з вагами алгоритму, саме тому SVM відноситься до лінійних алгоритмів. Розділяючу гіперплощину можна побудувати різними способами, але в SVM ваги w і b налаштовуються таким чином, щоб об'єкти класів лежали якнайдалі від роздільної гіперплощини. Іншими словами, алгоритм максимізує зазор (англ. margin) між гіперплощиною та об'єктами класів, які розташовані найближче до неї – опорні вектори x_+ та x_- (див. рис.3.1).

1.1.3 Правила налаштування ваг SVM

Щоб розділяюча гіперплощина знаходилась якнайдалі від точок вибірки, ширина смуги повинна бути максимальною. Знайдемо проекцію вектора, кінцями якого є опорні вектори різних класів, на вектор w. Тут і далі позначатимемо скалярний добуток двох векторів як $\langle a, b \rangle$ або $a^T b$. Ця проекція і буде показувати ширину смуги, що розділяє два класи:

$$\frac{\langle (x_+ - x_-), w \rangle}{||w||} = \frac{(\langle x_+, w \rangle - \langle x_-, w \rangle)}{||w||} = \frac{(b+1) - (b-1)}{||w||} = \frac{2}{||w||}.$$

Ширина смуги буде максимальною за умови:

$$\frac{2}{||w||} \to max \quad \Rightarrow \quad ||w|| \to min \quad \Rightarrow \quad \frac{(w^T w)}{2} \to min.$$

Відступом (англ. margin) об'єкта X від межі класів називається величина $M = Y(w^T X - b)$. Алгоритм припускається помилки на об'єкті тоді і тільки тоді, коли відступ M негативний (коли Y і $(w^TX - b)$ різних знаків). Якщо $M \in (0, 1)$ об'єкт потрапляє всередину розділяючої смуги. Якщо $M > 1$, то об'єкт X класифікується правильно, і знаходиться на деякому віддаленні від смуги, що розділяє. Тобто алгоритм правильно класифікуватиме об'єкти, якщо виконується умова:

$$Y(w^TX - b) \geq 1.$$

Якщо об'єднати два виведені вирази, то отримаємо дефолтне налаштування SVM з жорстким зазором (hard-margin SVM), коли жодному об'єкту не дозволяється потрапляти на смугу поділу. Для класів, які лінійно розділяються задача вирішується аналітично через теорему Куна-Таккера. Задача, що отримується, еквівалентна двоїстої задачі пошуку сідлової точки функції Лагранжа:

$$\begin{cases} (w^Tw)/2 \to min \\ y(w^Tx - b) \geq 1 \end{cases} \tag{1.1}$$

Щоб алгоритм зміг працювати і з лінійно нероздільними даними, необхідно дозволити алгоритму припускатися помилок на навчальних об'єктах, але при цьому цих помилок має бути якнайменше. Введемо набір додаткових змінних $\xi_i > 0$, що характеризують величину помилки на кожному об'єкті x_i. Введемо в функціонал, що мінімізується, штраф за сумарну помилку:

$$\begin{cases} (w^Tw)/2 + \alpha \sum \xi_i \to min \\ y(w^Tx_i - b) \geq 1 - \xi_i \\ \xi_i \geq 0 \end{cases} \tag{1.2}$$

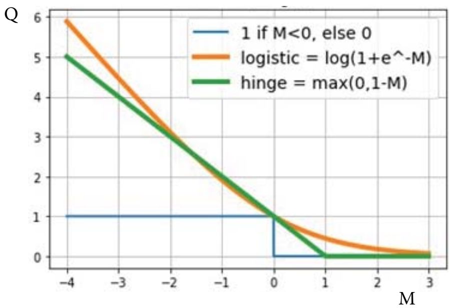
Еквівалентна задача безумовної мінімізації набуває вигляду:

$$\frac{1}{2}(w^Tw) + \alpha \sum (1 - M_i(w, b))_+ \to min. \tag{1.3}$$

Вважатимемо кількість помилок алгоритму (коли $M < 0$). Назвемо це штрафом (*Penalty*). Тоді штраф для всіх об'єктів дорівнюватиме сумі штрафів для кожного об'єкта x_i, де $[M_i < 0]$ – порогова функція:

$$Penalty = \sum [M_i < 0]$$

$$M_i < 0] = \begin{cases} 1, & \text{якщо } M_i < 0 \\ 0, & \text{якщо } M_i \geq 0 \end{cases}$$



Функція втрат (Loss function)

Далі зробимо штраф чутливим до величини помилки (чим сильніше M "йде в мінус" – тим більше штраф) і заразом введемо штраф за наближення об'єкта до кордону класів. Для цього візьмемо функцію, яка обмежує граничну функцію помилки:

$$Penalty = \sum[M_i < 0] \leq \alpha \sum(1 - M_i)_+ = \alpha \sum max(0, 1 - M_i)$$

При додаванні до виразу штрафу доданка $(w^T w)/2$ отримуємо класичну фукцію втрат SVM з м'яким зазором (*soft-margin SVM*) для одного об'єкта:

$$Q = \alpha max(0, 1 - M_i) + (w^T w)/2 = \alpha max(0, 1 - yw^T x) + (w^T w)/2$$

Q – Функція втрат (*loss function*). Саме її ми і мінімізуватимемо за допомогою градієнтного спуску. Правила зміни ваг набуває вигляду:

$$w = w - \eta \nabla Q,$$

де η – крок спуску (крок навчання). Похідна від функції втрат дає:

$$\nabla Q(w) = \frac{dQ}{dw} = \frac{1}{n} \sum_{i=1}^{n} \begin{cases} w, & \text{якщо } max(1 - y_i \langle w_i, x_i \rangle)_+ = 0 \\ w - \alpha y_i x_i, & \text{у іншому випадку} \end{cases}$$

Цей алгоритм дозволить провести класифікацію об'єктів для лінійно-роздільної вибірки.

1.1.4 Плюси, мінуси та модифікації

Плюси:

- добре працює із простором ознак великого розміру;
- добре працює з даними невеликого обсягу;
- алгоритм максимізує смугу, що розділяє, яка, як подушка безпеки, дозволяє зменшити кількість помилок класифікації;
- оскільки алгоритм зводиться до розв'язання задачі квадратичного програмування у опуклій області, то таке завдання завжди має єдине рішення (роздільна гіперплощина з певними гіперпараметрами алгоритму завжди одна).

Мінуси:

- довгий час навчання (для великих наборів даних);
- нестійкість до шуму: викиди у навчальних даних стають опорними об'єктами-порушниками і безпосередньо впливають на побудову роздільної гіперплощини;
- не описані загальні методи побудови ядер і спрямовуючих просторів, що найбільш підходять для конкретного завдання у разі лінійної нероздільності класів. Підбирати корисні перетворення даних – мистецтво.

Модифікації алгоритму:

- Метод релевантних векторів (Revance Vector Machine, RVM)
- 1-norm SVM (LASSO SVM)
- Doubly Regularized SVM (ElasticNet SVM)
- Support Features Machine (SFM)
- Relevance Features Machine (RFM)

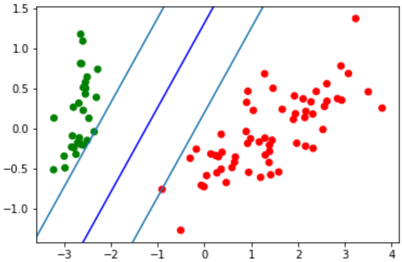
1.1.5 Постановка задачі та приклад реалізації

Провести класифікацію даних з використанням методу опорних векторів. Етапи розв'язання

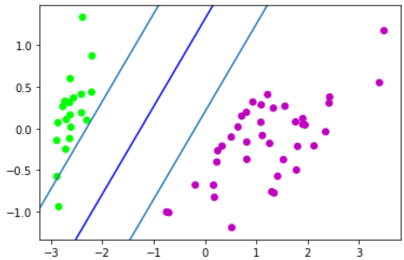
1. Імпортувати вибірку для проведення навчання
2. Розділити всю вибірку на навчальну та тестову
3. Провести підготовку даних до класифікації
   (а) визначити дві головних ознаки за якими буде проводитись класифікація
   (б) додати ще один стовбчик до матриці ознак та встановити всі значення рівним 1
   (в) встановити значення цільового вектора $+1$ та -1
4. Побудувати алгоритм навчання на навчальній вибірці
   (а) встановити значення кроку навчання η, коефіцієнта змінення ваг α
   (б) визначити умову завершення навчання (встановити кількість епох навчання або мінімально допустиму точність алгоритму)
   (в) в залежності від величини зазору змінювати ваги за допомогою градієнта функції втрат Q
   (г) рахувати кількість помилок (неправильно класифікованих об'єктів) у процесі навчання
5. Подати графічно результат класифікації для навчальної вибірки та залежність помилок від номеру епохи навчання
6. Перевірити точність роботи алгоритму на тестовій вибірці
7. Подати графічно результат класифікації для тестової вибірки та залежність помилок від номеру епохи навчання
8. Порівняти результати з SVM з sklearn
9. Оформити результати у вигляді звіту.

1.1.6 Приклад подання результатів

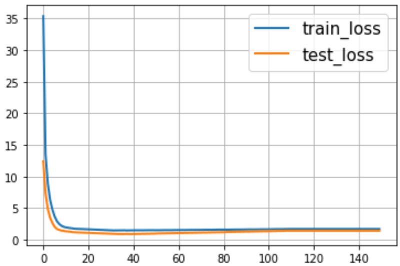
Класифікація лінійно роздільної вибірки



Навчальна вибірка

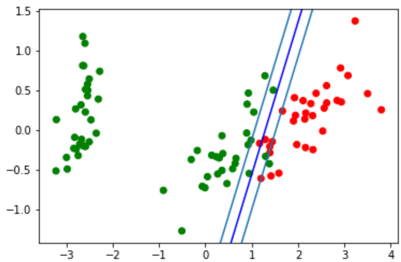


Тестова вибірка

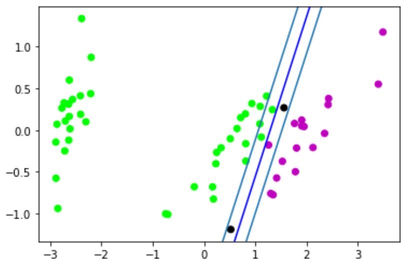


Помилки класифікації

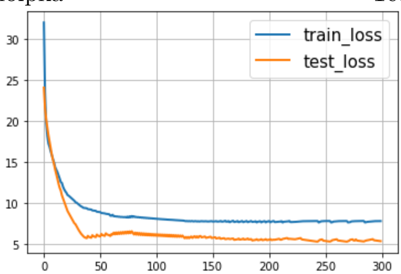
Класифікація лінійно не роздільної вибірки



Навчальна вибірка



Тестова вибірка



Помилки класифікації