Метод опорних векторів

Метод опорних векторів (Support Vector Machine, SVM) — це алгоритм машинного навчання для класифікації та регресії, який працює шляхом знаходження гіперплощини, що найкраще розділяє два або більше класи даних.

У SVM зображення даних представляються у вигляді векторів у n-вимірному просторі (де n — кількість ознак у даних), і для розділення класів шукається гіперплощина (тобто $n-1$-вимірна площина у n-вимірному просторі), що максимально відділяє класи один від одного. Оскільки може бути декілька таких гіперплощин, то SVM знаходить ту, яка максимізує мінімальну відстань між гіперплощиною та найближчими до неї точками обох класів (це відстань називається «мінімальною шириною розділяючої смуги» або «margin»).

SVM може використовуватися для класифікації даних, які мають два або більше класів, а також для регресії, тобто для передбачення числового значення залежної змінної на основі інших ознак даних. SVM є потужним алгоритмом машинного навчання, що зазвичай дає дуже хороші результати для класифікації даних у випадку, коли класи добре розділені гіперплощиною.

1. Математична модель

1.1. Жорстке розділення

Сформулюємо попереднє визначення більш строго та математично. В своїй основі ми маємо знову лінійну модель, і позначення ми використовуємо як і в попередніх модулях.

$$\begin{cases} \vec{w} \cdot \vec{x}^{(i)} + b \geqslant 1, & \text{якщо } y^{(i)} = 1, \\ \vec{w} \cdot \vec{x}^{(i)} + b \leqslant -1, & \text{якщо } y^{(i)} = 0 \end{cases} \tag{1}$$

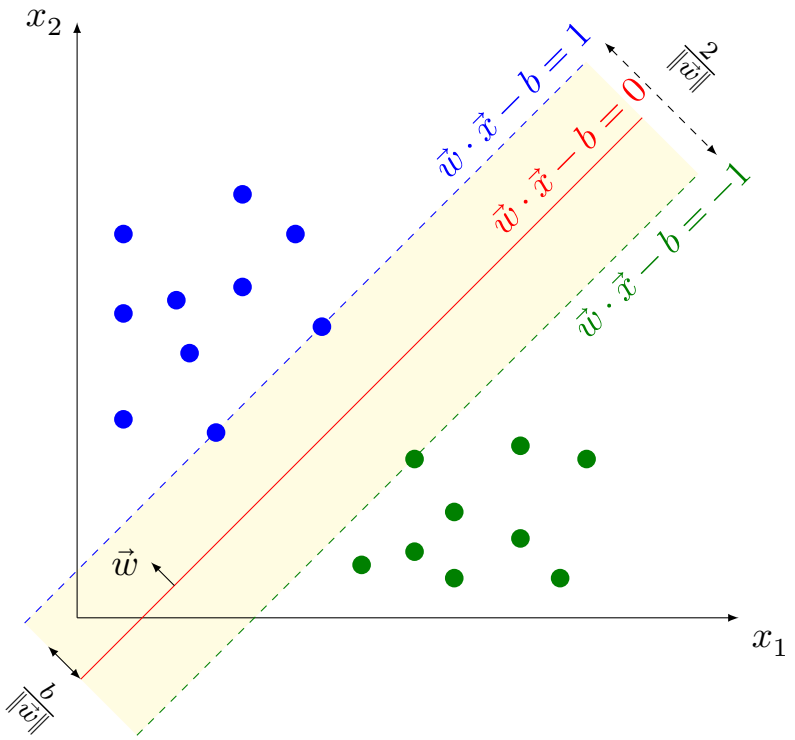


Рис. 1. Ілюстрація методу опорних векторів

Таке розділення класів гіперплощиною називається жорстким, оскільки така модель не дозволяє робити помилок, і різні класи чітко лежать по різні сторони розділяючою гіперплощини. Геометрично ширина розділяючою смуги (margin) рівна $\frac{2}{\|\vec{w}\|}$, де $\|\vec{w}\|$ норма вектору вагів, що рівна кореню зі скалярного добутку вектора вагів самого на себе, тобто:

$$\|\vec{w}\| = \sqrt{\vec{w} \cdot \vec{w}} = \sqrt{\sum_{i=1}^{n} w_i^2}. \tag{2}$$

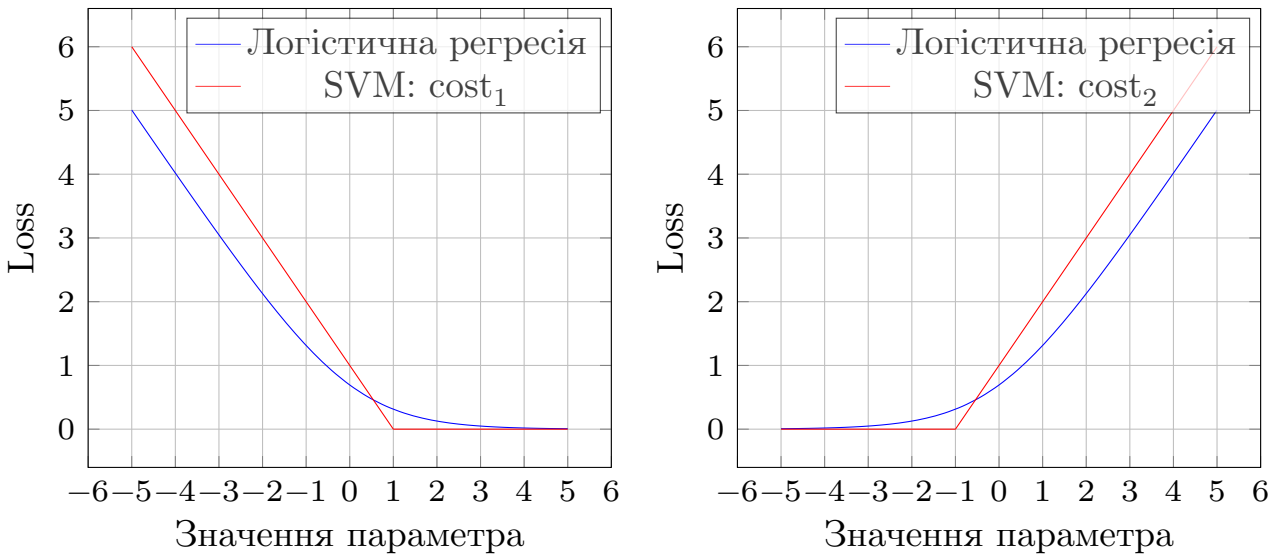
І оскільки ми хочемо максимізумати ширину розподіляючої полоси, і такий тип класифікаторів називають maximum margin, то величина норми, чи довжини, вектора вагів повинна бути мінімальною. А значить ми можемо використати алгоритм градієнтного спуску і знайти оптимальні значення коефіцієнтів w_i. І, до речі, останнє визначення довжини вектору може нам дещо нагадати — а саме доданок L_2-регуляризації.

1.2. М'яке розділення

З попереднього розділу ми знаємо, що при жорсткому розділенні модель не дозволяє робити помилок. Проте у наших даних можуть бути якісь аномалії, чи неправильно розмічені приклади, в такому випадку якщо ми будемо спиратись на такі дані, модель навчиться неоптимально, і буде дуже чутлива до таких аномалій. Щоб уникнути цього ми дозволимо нашому класифікатору робити помилки, проте нам потрібно оцінити наскільки сильно помиляється класифікатор і добавити їх у функцію втрат. В результаті ми отримаємо таку функцію втрат:

$$\begin{cases} \text{cost}_1 = \max\left(0, 1 - \left[\vec{w} \cdot \vec{x}^{(i)} + b\right]\right), & \text{якщо } y^{(i)} = 1, \\ \text{cost}_2 = \max\left(0, 1 + \left[\vec{w} \cdot \vec{x}^{(i)} + b\right]\right), & \text{якщо } y^{(i)} = 0 \end{cases} \tag{3}$$

Давайте поглянемо на графіки й порівняємо функцію втрат для логістичної регресії та методу опорних векторів:



(а) (б)

Рис. 2. Порівняння функції втрат для логістичної регресії та методу опорних векторів

Що якраз відповідає нашим вимогам. Напишемо повну формулу функції втрат:

$$\text{Loss} = C \sum_{i=1}^{m} \left[y^{(i)} \cdot \max\left(0, 1 - \left[\vec{w} \cdot \vec{x}^{(i)} + b\right]\right) + (1 - y^{(i)}) \cdot \max\left(0, 1 + \left[\vec{w} \cdot \vec{x}^{(i)} + b\right]\right) \right]$$

2. Ядровий трюк

Ось ми розібрались, як побудувати лінійний класифікатор за допомогою методу опорних векторів. Проте, як ми знаємо, часто ми не можемо лінійно розділити наші дані, як наприклад на ілюстрації 3.

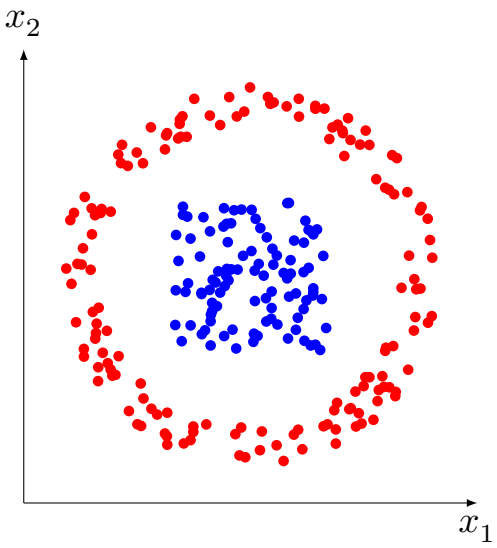


Рис. 3. Не роздільні ознаки

Для цього ми повинні перейти до нового простору ознак, раніше ми займались створенням нових фіч, таких як x_j^2 чи x_j^3, у такому випадку ми повинні перевести всі наші дані у новий векторний простір доповнений новими штучно створеними ознаками. Та це може бути досить дорого з точки зору обчислень, а ядровий трюк полягає у тому, що ми не робимо трансформацію даних, щоб відобразити їх у новому векторному просторі, а лише розраховуємо відстані між векторами так, наче ми вже перебуваємо у цьому просторі.

Це досягається за допомогою ядерної функції, яка обчислює скалярний добуток між векторами у вищорозмірному просторі, не потребуючи безпосереднього перетворення даних у цей простір. Це дозволяє SVM працювати зі складними нелінійно-роздільними даними, зберігаючи при цьому швидкість роботи алгоритму.

Застосування ядерного трюку в SVM дозволяє підвищити точність класифікації та зменшити ризик перенавчання моделі. Проблеми, які можуть бути вирішені за допомогою SVM з ядерним трюком, включають класифікацію текстових даних, розпізнавання образів та прогнозування фінансових показників.

Існує кілька типів ядерних функцій, які можна використовувати з методом опорних векторів (SVM), залежно від характеристик даних та завдання класифікації. Ось деякі з найпоширеніших ядерних функцій:

- Лінійна функція ядра: $K(x, y) = xy$;

- Поліноміальна функція ядра: $K(x, y) = (xy + r)^d$, де r — константа зміщення. d — степінь полінома;
- Радіальна базисна функція (RBF) ядра: $K(x, y) = e^{-\gamma \cdot \|x-y\|^2}$, де γ — параметр ширини гаусівської функції;
- Сигмоїдальна функція ядра: $K(x, y) = \tanh(\alpha \cdot xy + r)$, де α — параметр швидкості зростання, r — константа зміщення.

Кожен тип ядерної функції має свої унікальні властивості та може бути корисним для різних завдань класифікації. Наприклад, лінійна функція ядра добре підходить для лінійно роздільних даних, тоді як RBF функція може допомогти з розділенням складних нелінійних даних.

2.1. Гаусівське ядро

Розглянемо більш детально радіально базисну функцію, котра також називається ще й гаусівським ядром. Напишемо формулу густини ймовірності гаусівського розподілу:

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}.$$

Де символи μ та σ позначають математичне очікування та стандартне відхилення відповідно. Тоді як ядрова функція радіального базису має такий вигляд:

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

Ну і якщо ми зробимо такі заміну $\gamma = \dfrac{1}{2\sigma^2}$, то побачимо що дані функції відрізняються лише множенням на скаляр.

Проте як саме працює ядровий трюк? Як ми бачимо з визначенням ядрових функції в розрахунках ми використовуємо два вектори $\vec{x}$ та $\vec{y}$ і ми розраховуємо відстань між ними у новому лінійному просторі. А далі, у якості нових ознак для нашої моделі, описаної вище, ми будемо брати нові ознаки — відстані між нашими приклади розраховані за допомогою нашої ядрової функції. Формалізуємо вище сказане твердження:

$$f_j^{(i)} = K(\vec{x}^{(i)}, \vec{x}^{(j)}) = e^{-\gamma \cdot \|\vec{x}^{(i)} - \vec{x}^{(j)}\|^2}.$$

Отже тепер ми маємо замість вектора $\vec{x}^{(i)} \in \mathbb{R}^n$ новий $f^{(i)} \in \mathbb{R}^m$. І тепер будемо використовувати нові ознаки і в новому просторі шукати розподільчу смугу і максимізувати її ширину.

3. Реалізація методу опорних векторів у бібліотеці `scikit-learn`

Вся вищенавадена математична модель вже є реалізована у знайомій нам бібліотеці, і ви можете ознайомитись більше з різними версіями даного методу за посиланням. А тут наведемо приклад використання SVM в елементарному прикладі:

```
from sklearn import svm

X = [[0, 0], [1, 1]]
y = [0, 1]

clf = svm.SVC()
clf.fit(X, y)

print(clf.predict([[2., 2.]]))
```

Або більш складніший приклад:

```
# Import the necessary libraries
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC

# Generate synthetic data (two point classes)
X, y = datasets.make_classification(
    n_samples=100,
    n_features=2,
    n_informative=2,
    n_redundant=0,
    n_clusters_per_class=1,
```

```
    random_state=42)

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42)

# Create and train the SVM model
svm_classifier = SVC(kernel='linear', C=1)
svm_classifier.fit(X_train, y_train)

# Output the classification accuracy on the test set
accuracy = svm_classifier.score(X_test, y_test)
print(f'Classification accuracy: {accuracy:.2f}')

# Create a grid to visualize the solution boundary
xx, yy = np.meshgrid(np.linspace(X[:, 0].min() - 1,
                                 X[:, 0].max() + 1, 100),
                     np.linspace(X[:, 1].min() - 1,
                                 X[:, 1].max() + 1, 100))
Z = svm_classifier.decision_function(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)

# Data visualization and decision boundaries
plt.figure(figsize=(8, 6))
plt.contourf(xx, yy, Z,
             levels=[-1, 0, 1],
             colors=['r', 'g', 'b'], alpha=0.2)
plt.scatter(X[:, 0], X[:, 1], c=y,
            cmap=plt.cm.coolwarm,
            edgecolors='k')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.title('Reference vector method (SVM)')
plt.show()
```

Висновок

SVM (англ. Support Vector Machine) — це метод машинного навчання, який знайшов широке застосування в задачах класифікації та регресії. Основною ідеєю методу є побудова гіперплощини, яка розділяє два класи даних з максимальною можливою шириною проміжку між ними.

При роботі з SVM ми спочатку перетворюємо наші дані до більш високої розмірності за допомогою ядерної функції, що дозволяє лінійно нероздільні дані перетворювати в лінійно роздільні. Потім ми побудовуємо гіперплощину, максимізуючи відстань між гіперплощиною і найближчими до неї точками (векторами опору). Ці точки називаються опорними векторами і вони є ключовими елементами моделі SVM.

Серед переваг методу SVM можна виділити високу точність класифікації в задачах з великою кількістю ознак та можливість використання різних ядерних функцій для перетворення даних, які підходять для різних типів даних та задач.

Однак, метод SVM може бути чутливий до викидів, тобто до даних, які значно відрізняються від інших даних у вибірці. Крім того, вибір оптимальних параметрів для SVM може бути часо- та ресурсомістким завданням.

У загальному, SVM є потужним та ефективним методом машинного навчання, який може бути застосований у багатьох різних задачах. Використання SVM може залежати від характеру даних та конкретних умов задачі, тому варто розглянути його як один з варіантів при вирішенні конкретної задачі.