

SVM

☑ Trắc nghiệm 16 ⌵ PDF bài

Khi hai lớp dữ liệu tách được bằng một đường thẳng, thường có nhiều đường thẳng làm được việc đó. SVM (Support Vector Machine) chọn đường nằm xa các điểm gần nhất của cả hai lớp nhất, để một điểm mới lệch một chút vẫn được xếp đúng. Khoảng cách đó gọi là lề (margin). Bài này tính lề bằng tay trên ví dụ nhỏ, rồi dùng kernel để SVM tách được dữ liệu cần đường cong.

MỤC TIÊU

- Tính khoảng cách tới đường phân chia và hinge loss.
- Hiểu tham số C, gamma và vì sao cần chuẩn hóa.
- Dùng SVC với Pipeline và cross-validation.

Lề lớn nhất

Gán nhãn $y_i \in \{-1, +1\}$. Một siêu phẳng $\mathbf{w}^\top \mathbf{x} + b = 0$ phân loại đúng mọi điểm khi $y_i(\mathbf{w}^\top \mathbf{x}_i + b) > 0$. Khoảng cách từ $\mathbf{x}_i$ đến siêu phẳng là

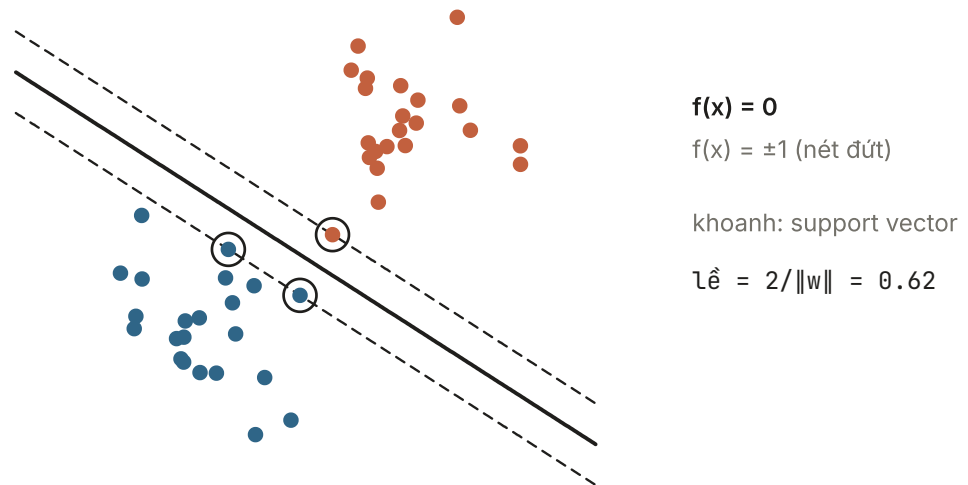
$$\frac{|\mathbf{w}^\top \mathbf{x}_i + b|}{\|\mathbf{w}\|}.$$

Vì nhân $\mathbf{w}$ và b với cùng một số dương không đổi siêu phẳng, ta chuẩn hóa sao cho các điểm gần nhất thỏa $y_i(\mathbf{w}^\top \mathbf{x}_i + b) = 1$. Khi đó độ rộng lề (margin) giữa hai phía là $2/\|\mathbf{w}\|$; chẳng hạn $\mathbf{w} = (3, 4)$ có $\|\mathbf{w}\| = 5$, nên lề rộng $2/5 = 0.4$. Cực đại lề tương đương

$$\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 \quad \text{thỏa} \quad y_i (\mathbf{w}^\top \mathbf{x}_i + b) \geq 1, \quad i = 1, \dots, n.$$

Đây là bài toán tối ưu lồi bậc hai: khi dữ liệu tách được thì có nghiệm, và không có cực tiểu địa phương nào để mắc kẹt. Các điểm nằm đúng trên biên lề là **vector hỗ trợ** (support vectors): chỉ chúng quyết định vị trí vạch, bỏ các điểm khác đi thì vạch không

đổi. (Với lề mềm ở mục sau, vector hỗ trợ còn gồm cả các điểm nằm trong lề hoặc bị phân loại sai.)



Chỉ các điểm nằm trên lề (khoanh tròn) quyết định vị trí vạch.

HỎI NHANH

Sau khi chuẩn hóa, hai biên lề $f(\mathbf{x}) = \pm 1$ cách nhau đúng 1 đơn vị. $\|w\|$ bằng bao nhiêu?

A 0.5

B 1

C 4

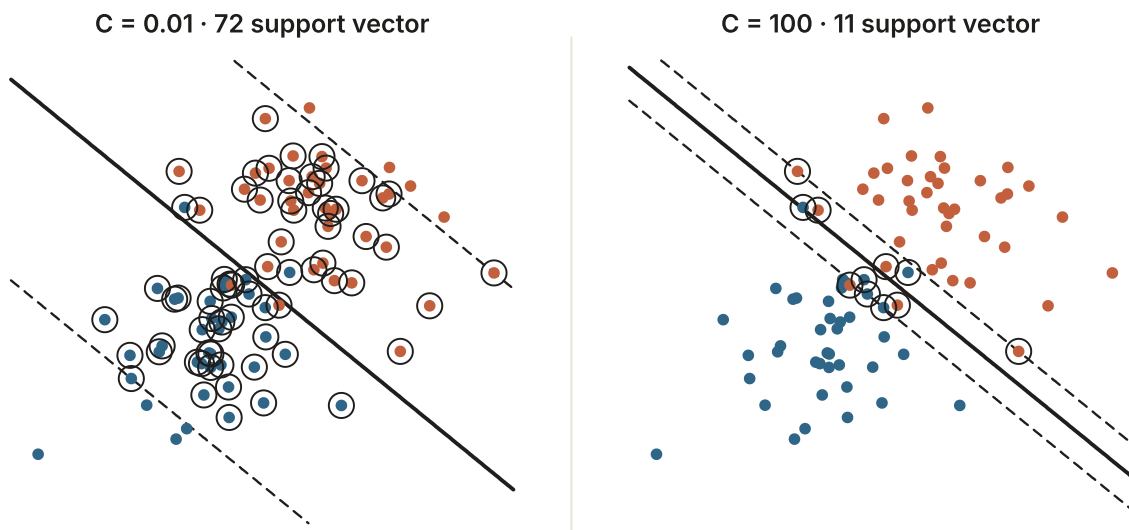
D 2

Lề mềm và hinge loss

Dữ liệu thật hiếm khi tách được hoàn toàn. Lề mềm (soft margin) cho phép vi phạm, có trả giá:

$$\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n \max(0, 1 - y_i f(\mathbf{x}_i)), \quad f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b.$$

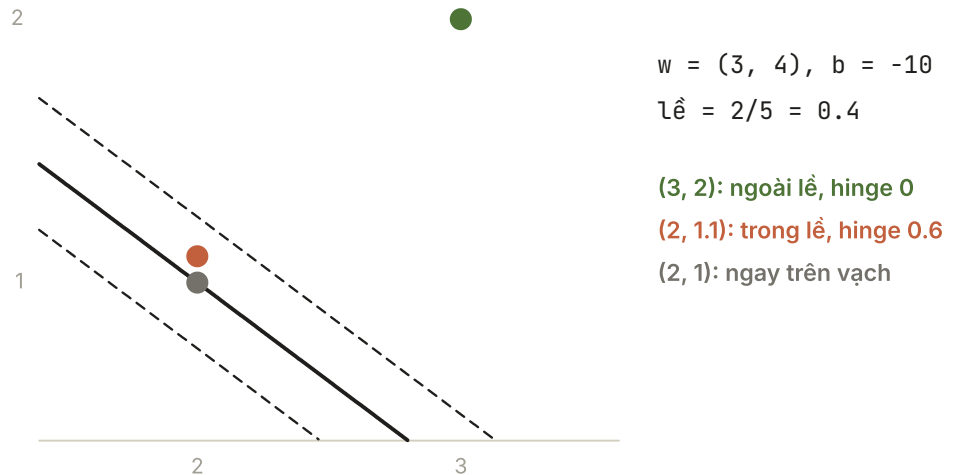
Số hạng thứ hai là hinge loss: bằng 0 nếu điểm nằm đúng phía và ngoài lề, tăng tuyến tính nếu lấn vào lề hoặc sang phía sai. Lại là “phạt + lỗi”, chỉ khác là hệ số đặt ở phía lỗi: C lớn thì ít chịu vi phạm, lề hẹp, dễ overfit; C nhỏ thì lề rộng, mô hình đơn giản hơn.



C nhỏ: lề rộng, chịu nhiều vi phạm. C lớn: lề hẹp.

Lấy siêu phẳng $\mathbf{w} = (3, 4)$, $b = -10$ và ba điểm:

- Điểm $(2, 1)$: $3 \cdot 2 + 4 \cdot 1 - 10 = 0$, nằm đúng trên siêu phẳng.
- Điểm $(3, 2)$ nhãn $+1$: $f = 7$, khoảng cách $7/5 = 1.4$, $yf = 7 \geq 1$, hinge loss bằng 0.
- Điểm $(2, 1.1)$ nhãn $+1$: $f = 0.4$, nằm đúng phía nhưng trong lề, hinge loss $= 1 - 0.4 = 0.6$.



Điểm (2, 1.1) đúng phía nhưng nằm trong lề nên vẫn chịu hinge 0.6.

So với logistic regression ở bài 2.3, hai mô hình chỉ khác hàm mất mát. Log loss không bao giờ bằng 0, nên mọi điểm đều góp tiếng nói. Hinge loss bằng 0 với điểm “an toàn”, nên chỉ các điểm gần ranh giới mới quan trọng.

HỎI NHANH

Cho C tiến dần về 0 trong bài toán lề mềm. Nghiệm có xu hướng thế nào?

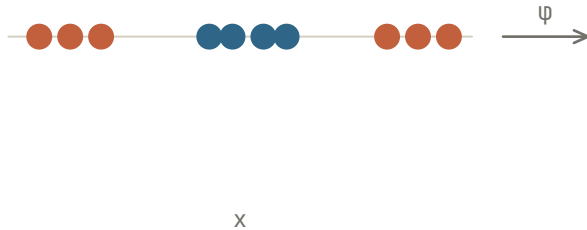
- A Lề hẹp lại, cố phân loại đúng mọi điểm train
- B w về gần 0, lề rất rộng, bỏ qua các vi phạm
- C Không đổi, vì C chỉ ảnh hưởng tốc độ tối ưu
- D Trở thành SVM lề cứng, không chấp nhận vi phạm

Mẹo kernel

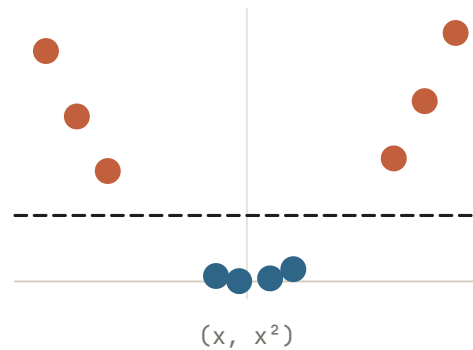
Muốn ranh giới cong, cách thô là ánh xạ $\mathbf{x}$ sang không gian đặc trưng lớn hơn $\phi(\mathbf{x})$ rồi chạy SVM tuyến tính ở đó. Điều đặc biệt là dạng đối ngẫu (dual) của bài toán SVM chỉ dùng dữ liệu qua tích vô hướng $\phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j)$. Nếu có hàm k tính thẳng tích đó, gọi là kernel, thì không bao giờ phải tạo ϕ :

$$k(\mathbf{x}, \mathbf{z}) = \phi(\mathbf{x})^\top \phi(\mathbf{z}).$$

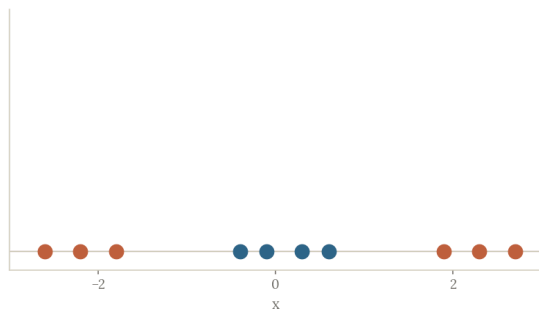
trên trục x: không vạch nào tách được



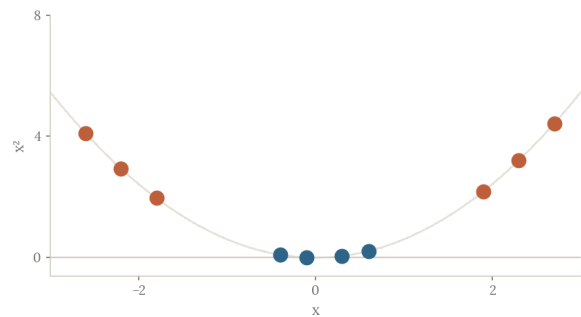
sau $\phi(\mathbf{x}) = (x, x^2)$: một đường ngang



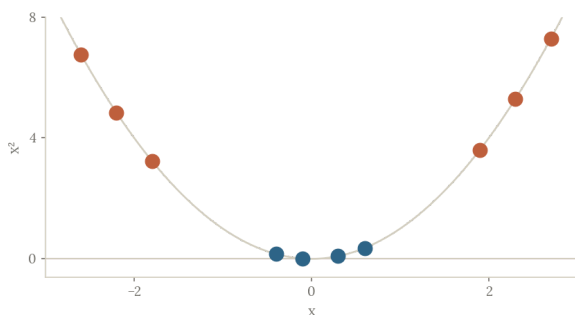
Thêm tọa độ x^2 thì một đường thẳng tách được hai lớp.



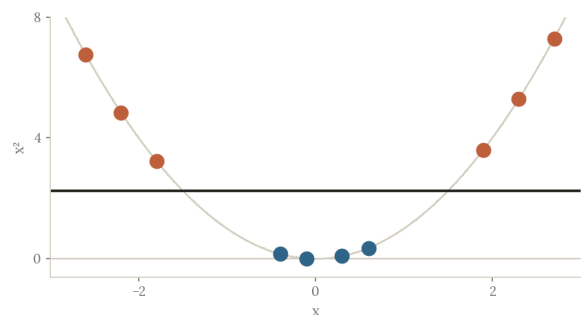
trên trục x: lớp cam nằm hai đầu, lớp xanh ở giữa



nâng mỗi điểm lên độ cao x^2



nâng mỗi điểm lên độ cao x^2



đường ngang $x^2 = 2.25$ tách hai lớp

Nâng mười điểm từ trục x lên (x, x^2) : hai lớp không tách được bằng một ngưỡng trên trục lại tách được bằng một đường thẳng.

Ví dụ với $\mathbf{x}, \mathbf{z} \in \mathbb{R}^2$ và $\phi(\mathbf{x}) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)$:

$$\phi(\mathbf{x})^\top \phi(\mathbf{z}) = x_1^2 z_1^2 + 2x_1 x_2 z_1 z_2 + x_2^2 z_2^2 = (\mathbf{x}^\top \mathbf{z})^2.$$

Một phép nhân vô hướng rồi bình phương thay cho việc lập và nhân hai vector ba chiều.
Với kernel RBF (Gaussian)

$$k(\mathbf{x}, \mathbf{z}) = \exp(-\gamma \|\mathbf{x} - \mathbf{z}\|^2),$$

không gian đặc trưng tương ứng có vô hạn chiều, vậy mà tính kernel chỉ tốn $O(d)$.

HỎI NHANH

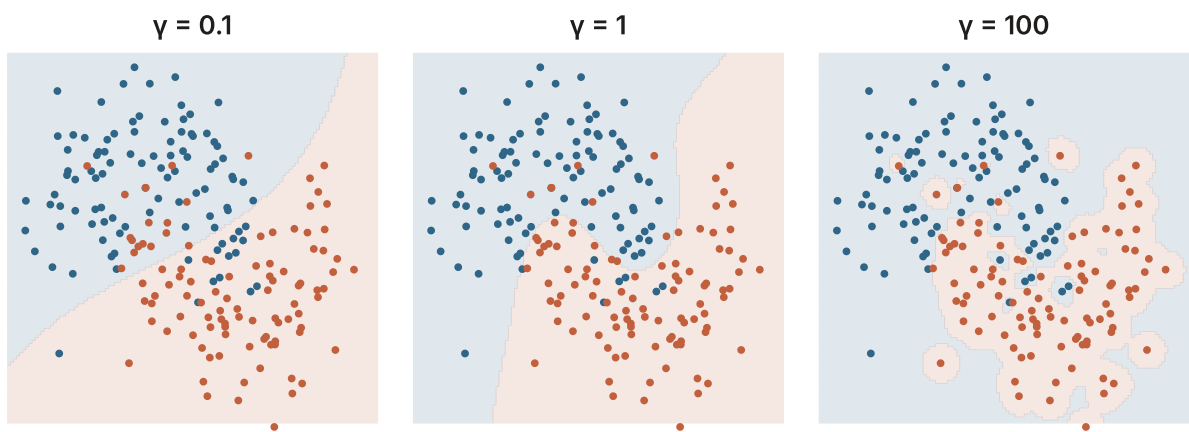
Với $\mathbf{x} \in \mathbb{R}^{1000}$, kernel $(\mathbf{x}^\top \mathbf{z})^2$ ứng với một ϕ có khoảng bao nhiêu chiều? Tính thẳng kernel thì tốn khoảng bao nhiêu phép nhân?

- A 2000 chiều, khoảng 2000 phép nhân
- B 1 000 000 chiều, khoảng 1 000 000 phép nhân
- C 500 500 chiều, khoảng 500 500 phép nhân
- D 500 500 chiều, khoảng 1000 phép nhân

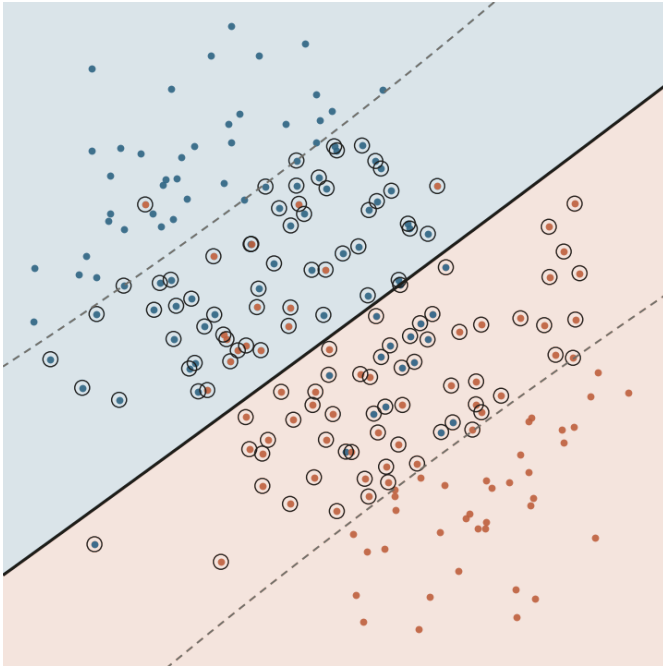
Hàm dự đoán khi đó có dạng $f(\mathbf{x}) = \sum_{i \in SV} \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}) + b$: một tổng có trọng số theo độ giống giữa điểm mới và các vector hỗ trợ. Nhìn kỹ thì giống kNN có trọng số, nhưng trọng số được học bằng tối ưu. Với RBF và $\gamma = 0.1$, hai điểm $(1, 2)$ và $(4, 6)$ có $\|\mathbf{x} - \mathbf{z}\|^2 = 25$ nên $k = e^{-2.5} \approx 0.082$. Một điểm test ở rất xa mọi vector hỗ trợ có mọi k gần 0, nên f gần bằng b và dấu của b quyết định nhãn.

Hai tham số C và γ

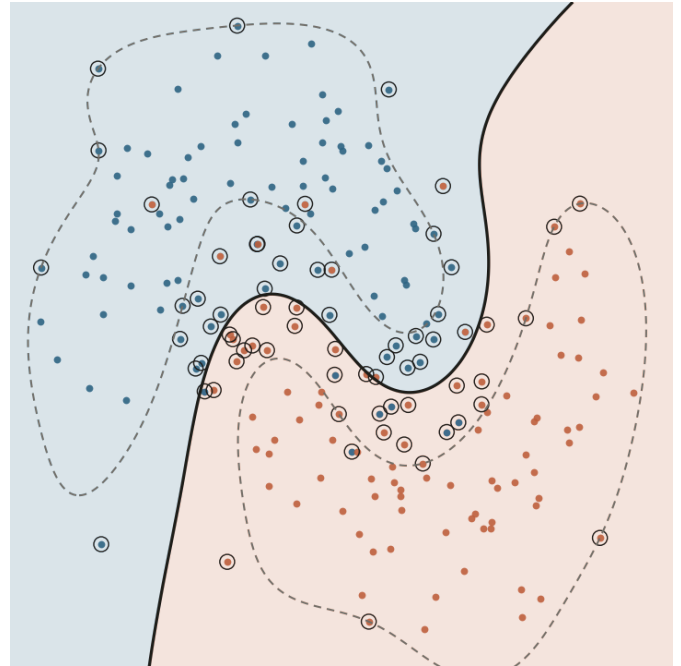
Với kernel RBF, γ quy định “tầm ảnh hưởng” của mỗi vector hỗ trợ. γ lớn thì mỗi điểm chỉ ảnh hưởng vùng rất nhỏ quanh nó, ranh giới uốn lượn theo từng điểm. γ nhỏ thì ranh giới mượt gần như tuyến tính. Cùng với C , đây là hai núm cần dò, luôn trên lưới logarit, ví dụ $C \in \{0.1, 1, 10, 100\}$ và $\gamma \in \{10^{-3}, \dots, 1\}$.



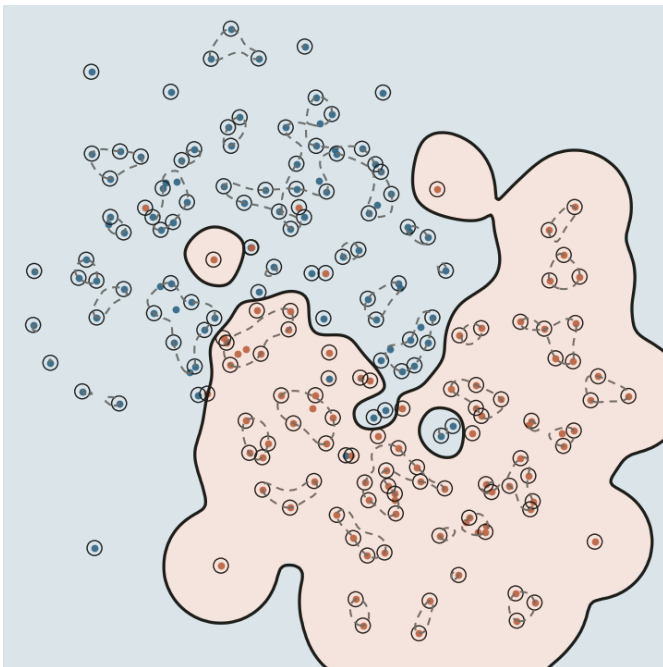
γ nhỏ: gần thẳng; $\gamma = 1$: theo dáng dữ liệu; $\gamma = 100$: khoanh từng điểm.



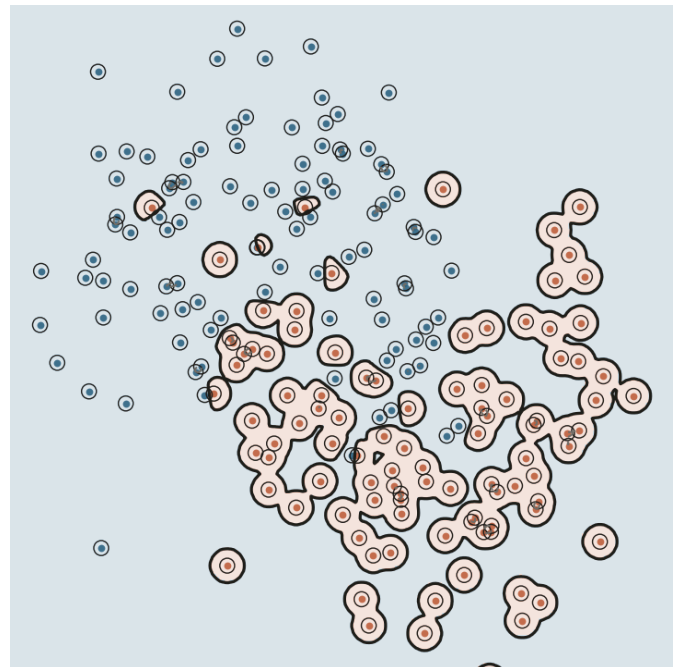
$\gamma = 0.01$: 123 support vector, accuracy 5-fold CV 0.830



$\gamma = 1$: 72 support vector, accuracy 5-fold CV 0.890



$\gamma = 30$: 177 support vector, accuracy 5-fold CV 0.845



$\gamma = 300$: 200 support vector, accuracy 5-fold CV 0.640

SVM kernel RBF, $C = 1$, trên dữ liệu trắng khuyết: tăng γ thì vùng ảnh hưởng của mỗi điểm hẹp lại, ranh giới uốn theo từng điểm (vòng tròn là support vector).

SVM RBF nhạy với khoảng cách; khi các cột khác đơn vị, hãy chuẩn hóa trong Pipeline và kiểm bằng validation.

HỎI NHANH

SVM với kernel RBF, $\gamma = 100$, đúng 100% trên train nhưng cross-validation chỉ được 70%. Ta nên thử gì trước?

- A Tăng γ lên 1000 cho ranh giới khít hơn
- B Tăng C để phạt vi phạm nặng hơn nữa
- C Giảm γ , và thử cả C nhỏ hơn
- D Bỏ chuẩn hóa để giữ đơn vị gốc của dữ liệu

Chi phí và giới hạn

- Huấn luyện SVM có kernel tốn khoảng $O(n^2)$ đến $O(n^3)$ theo số mẫu. Quá vài chục nghìn mẫu là bắt đầu đuối.
- Với dữ liệu lớn, thưa, nhiều chiều như văn bản, dùng SVM tuyến tính (LinearSVC): rất nhanh và rất tốt.
- SVM không cho xác suất. SVC(probability=True) hiệu chỉnh thêm bằng Platt scaling, chậm hơn và không hoàn hảo.

HỎI NHANH

Cần phân loại 2 triệu bình luận, mỗi bình luận là một vector TF-IDF 100 000 chiều, rất thưa. Nên thử mô hình nào trước?

- A LinearSVC hoặc logistic regression tuyến tính
- B SVC(kernel="rbf"), vì ranh giới cong thì mạnh hơn
- C SVC(kernel="poly", degree=5) cho đủ phức tạp
- D KNN với $k = 1$ trên toàn bộ vector TF-IDF

Đọc score trước khi đọc xác suất

Với $\mathbf{w} = (3, 4)$, $b = -10$, $\|\mathbf{w}\| = 5$. Mỗi điểm có score $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$; dấu quyết định phía của vạch, còn $|f|/5$ mới là khoảng cách hình học. Score 7 không có nghĩa xác suất 70%, vì score có đơn vị phụ thuộc cách đặt $\mathbf{w}$. Hinge loss lại dùng **nhãn có dấu** $y \in \{-1, +1\}$; nếu lấy nhãn 0/1 nhân vào công thức, mọi mẫu lớp 0 đều có loss 1, là một lỗi âm thầm.

Python

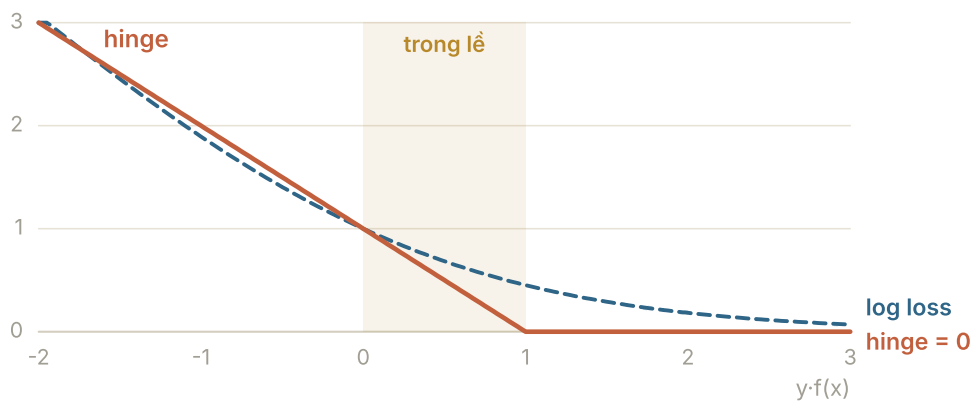
```
import numpy as np

w_margin = np.array([3., 4.])
b_margin = -10.
points_margin = np.array([[2., 1.], [3., 2.], [2., 1.1]])
labels_signed = np.array([1., 1., 1.])
scores = points_margin @ w_margin + b_margin
distances = np.abs(scores) / np.linalg.norm(w_margin)
hinge = np.maximum(0., 1. - labels_signed * scores)
print("score:", scores.round(3).tolist())
print("khoảng cách:", distances.round(3).tolist())
print("hinge:", hinge.round(3).tolist())
```

KẾT QUẢ

```
score: [0.0, 7.0, 0.4]
khoảng cách: [0.0, 1.4, 0.08]
hinge: [1.0, 0.0, 0.6]
```

Dòng thứ ba tách hai khái niệm: điểm score 0.4 được phân loại đúng phía nhưng vẫn chịu loss 0.6 vì nằm trong lề; điểm score 7 thì loss 0. Điểm nằm ngay trên vạch có score 0, cần quy ước phá hòa khi biến thành nhãn. Thư viện lo phần ấy, nhưng code tay cần ghi rõ.



Hinge loss bằng 0 khi $y \cdot f(x) \geq 1$; log loss thì luôn dương.

HỎI NHANH

Một mẫu có $y = -1$ và $f(\mathbf{x}) = -0.2$. Hinge loss bằng bao nhiêu?

A 1.2

B 0.8

C 0

D 0.2

Kiểm mọ kernel bằng số

Ta kiểm ví dụ polynomial trong bài bằng cả hai cách: dựng đặc trưng ba chiều rồi tính tích vô hướng, và bình phương tích vô hướng hai chiều.

Python

```
x_kernel = np.array([1., 2.])
z_kernel = np.array([3., 4.])
def phi2(v):
    a, b = v
    return np.array([a * a, np.sqrt(2.) * a * b, b * b])

print(round(float(phi2(x_kernel) @ phi2(z_kernel)), 6))
```

```
print(float((x_kernel @ z_kernel) ** 2))
assert np.allclose(phi2(x_kernel) @ phi2(z_kernel), (x_kernel @ z_kernel) ** 2)
```

KẾT QUẢ

```
121.0
121.0
```

Hai dòng cùng bằng 121. Ta tính được kernel mà không dựng từng tọa độ của không gian lớn hơn. Cái giá chuyển sang số mẫu: kernel SVM làm việc với quan hệ giữa các cặp mẫu, nên thời gian và bộ nhớ tăng nhanh theo n . Hãy đo trên một phần nhỏ dữ liệu trước khi fit cả bộ.

Dùng thư viện

Python

```
from sklearn.datasets import make_moons
from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC

X, y = make_moons(n_samples=500, noise=0.25, random_state=0)
model = make_pipeline(StandardScaler(), SVC(kernel="rbf"))
grid = GridSearchCV(model, {"svc__C": [0.1, 1, 10, 100], "svc__gamma": [0.01, 0.1, 1, 10]},
cv=5)
grid.fit(X, y)
print(grid.best_params_, round(grid.best_score_, 3))
print("số vector hỗ trợ:", grid.best_estimator_[1].n_support_.sum(), "trên", len(X))
```

KẾT QUẢ

```
{'svc__C': 1, 'svc__gamma': 10} 0.936
số vector hỗ trợ: 221 trên 500
```

HỎI NHANH

Mô hình tốt nhất dùng 221 trên 500 điểm làm vector hỗ trợ. Con số này nói lên điều gì?

- A Mô hình học thuộc 221 điểm nên chắc chắn overfit
- B 279 điểm còn lại bị bỏ đi vì gán nhãn sai
- C Hai lớp chồng lên nhau, nhiều điểm nằm trong lề
- D Lề cứng đã tách được hoàn toàn hai lớp

Khi đi thi

- Chuẩn hóa các cột trong từng fold trước SVM; dò C và γ trên thang log.
- Văn bản thì thử TF-IDF cộng LinearSVC làm baseline.
- Đề cần xác suất thì dùng logistic regression, hoặc hiệu chỉnh SVM trên validation.

Tóm tắt

- SVM chọn siêu phẳng có lề lớn nhất; chỉ các vector hỗ trợ quyết định ranh giới.
- Lề mềm cân bằng giữa lề rộng và vi phạm bằng C ; hàm mất mát là hinge loss.
- Kernel cho phép ranh giới cong mà không cần tạo đặc trưng; RBF điều khiển bằng γ .
- Chuẩn hóa đặc trưng; dò C , γ trên thang log; hạn chế với dữ liệu rất lớn.

Kiểm tra cuối bài

Mười câu kiểm cả cách hiểu, phép tính và quy trình. Câu sai chỉ ra phần cần đọc lại; hãy làm trước khi mở lời giải.

CÂU 1

$\mathbf{w} = (6, 8)$, $b = -10$. Khoảng cách từ điểm $\mathbf{x} = (1, 1)$ tới siêu phẳng $\mathbf{w}^\top \mathbf{x} + b = 0$ bằng bao nhiêu?

A 10

B 1.4

C 0.4

D 4

CÂU 2

Dữ liệu có bốn điểm: $(0, 1)$ và $(1, 1)$ mang nhãn -1 , còn $(3, 3)$ và $(4, 3.5)$ mang nhãn $+1$. SVM lề cứng chỉ có $(1, 1)$ và $(3, 3)$ là vector hỗ trợ. Lề rộng bao nhiêu?

A $2\sqrt{2} \approx 2.83$

B $\sqrt{2} \approx 1.41$

C 2

D 4

CÂU 3

Một bài phân loại có vài điểm bị gán nhãn nhầm, nằm sâu trong vùng của lớp kia. SVM lề cứng và SVM lề mềm xử lý thế nào?

- A Lề cứng tự bỏ qua các điểm nhầm, lề mềm thì không
- B Hai cách cho cùng một siêu phẳng
- C Lề cứng vô nghiệm, còn lề mềm chịu phạt các điểm nhầm theo C
- D Lề mềm không dùng được với kernel, nên phải dùng lề cứng

PHÒNG THEO NOAI 2026 Singapore · Vòng sơ loại · Câu 171

CÂU 4

Một mẫu có $y = -1$ và $f(\mathbf{x}) = 2$. Hinge loss bằng bao nhiêu?

- A 0
- B 3
- C 1
- D 2

CÂU 5

Một mẫu có $y = 1$ và $f(\mathbf{x}) = 0.4$. Nhân cả $\mathbf{w}$ và b với 2 thì siêu phẳng không đổi. Hinge loss của mẫu thay đổi thế nào?

- A Từ 0.6 xuống 0.2
- B Từ 0.6 lên 1.2, vì score tăng gấp đôi
- C Vẫn là 0.6, vì siêu phẳng không đổi
- D Từ 0.6 xuống 0, vì mẫu đã được phân loại đúng

CÂU 6

Trong mục tiêu $\frac{1}{2}\|\mathbf{w}\|^2 + C \sum_i \text{hinge}_i$, C lớn có nghĩa là gì?

- A Lẽ luôn rộng hơn, vì C nhân với $\|\mathbf{w}\|^2$
- B Bỏ hẳn hinge loss khỏi mục tiêu
- C Chỉ làm thuật toán chạy nhanh hơn
- D Vi phạm lề bị phạt nặng hơn so với phần còn lại

CÂU 7

Nhà máy có 10 000 bản ghi máy chạy bình thường, và gần như không có bản ghi máy hỏng. Vì sao one-class SVM hợp với việc phát hiện bất thường ở đây?

- A Vì nó cần đủ cả mẫu bình thường lẫn mẫu hỏng để học
- B Vì nó chỉ dùng được cho bài phân loại hai lớp
- C Vì nó là thuật toán nhanh nhất trên dữ liệu lớn
- D Vì nó chỉ cần học vùng của dữ liệu bình thường

ĐỀ THI NOAI 2026 Singapore · Vòng sơ loại · Câu 170

CÂU 8

SVM cho decision score bằng 2.0 ở một mẫu. Nếu đề yêu cầu xác suất để tính log loss, ta nên làm gì?

- A Học một phép hiệu chỉnh xác suất trên validation
- B Nộp thẳng 2.0 làm xác suất
- C Chia score cho 10 để mọi score đều nhỏ hơn 1
- D Đổi score thành nhãn theo dấu, rồi gọi đó là xác suất

CÂU 9

Tự viết hinge loss bằng $\text{np.maximum}(0, 1 - y*f)$, trong khi y nhận giá trị 0 hoặc 1. Ở các mẫu lớp 0, chuyện gì xảy ra?

- A Chỉ cần đổi dấu f , không cần đổi y
- B Loss luôn bằng 1, phải đổi y thành ± 1
- C Loss luôn bằng 0, nên lớp 0 đã được phân loại đúng
- D Loss vẫn đúng, vì 0 là nhãn âm

CÂU 10

Kernel $k(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^\top \mathbf{z} + 1)^2$ với $\mathbf{x} \in \mathbb{R}^2$ ứng với một ánh xạ ϕ sang không gian bao nhiêu chiều?

- A 3
- B 4
- C 6
- D 9

PHÒNG THEO NOAI 2026 Singapore · Vòng sơ loại · Câu 161

Đọc thêm

- Cortes, Vapnik (1995) — *Support-Vector Networks*, Machine Learning 20. Bài gốc của lễ mừng.
- James, Witten, Hastie, Tibshirani — [An Introduction to Statistical Learning](#), chương 9. Lễ, kernel và so sánh với logistic regression.
- Géron — *Hands-On Machine Learning*, chương 5. SVM tuyến tính, kernel và mẹo chọn C, γ bằng scikit-learn.

- [scikit-learn — Support Vector Machines](#). Kernel, scaling, complexity, support vectors và probability calibration.
- [ISLP — Support vector machines lab](#). Các thực nghiệm về C, gamma và validation.