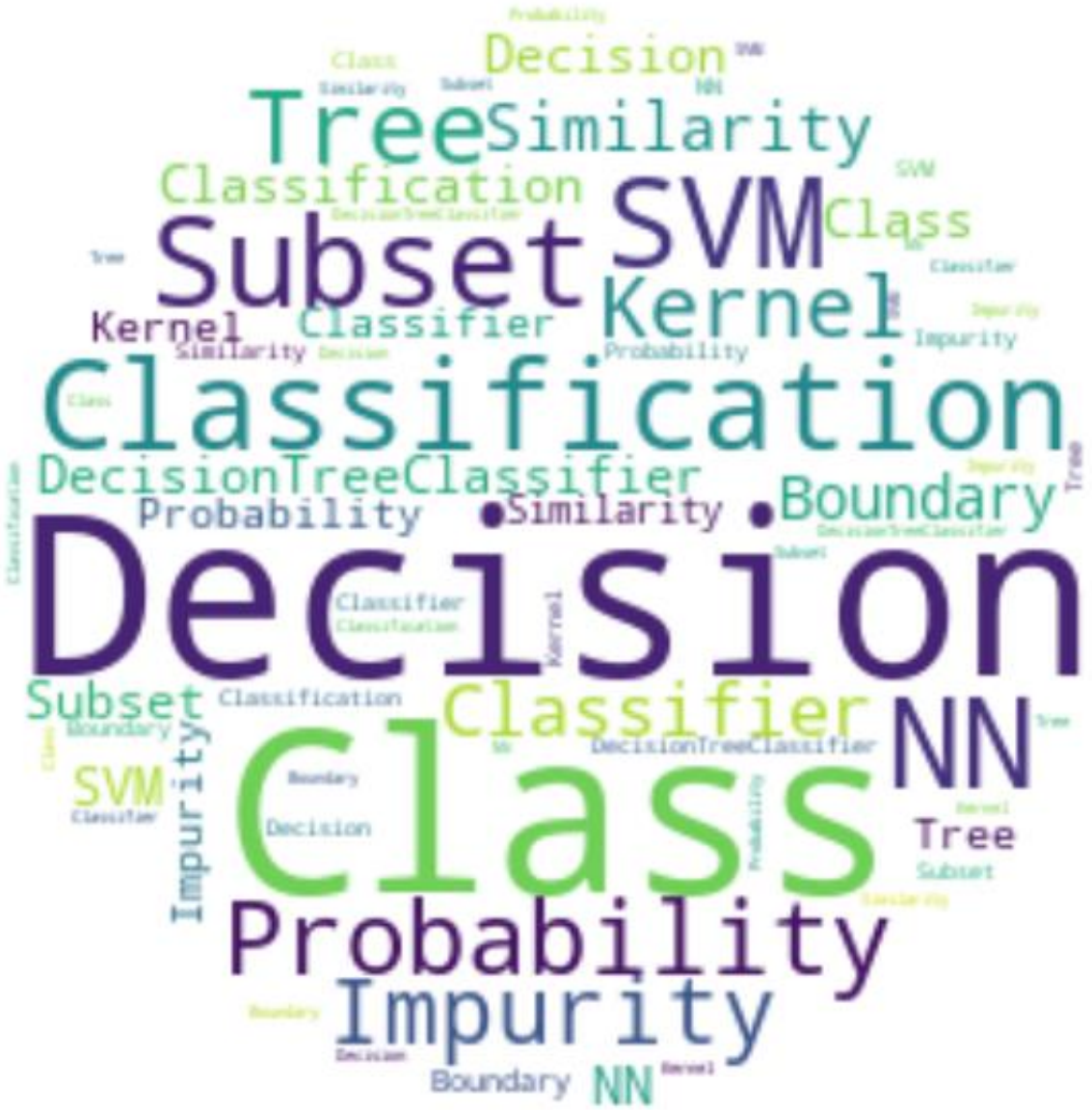

Decision
Tree
Similarity
Classification
SVM
Subset
Class
Kernel
Classifier
Probability
Boundary
DecisionTreeClassifier
Impurity
NN

2102470 Học máy

Bài giảng: 2.3 Bài toán phân lớp

Chương 2: Xấp xỉ và phân lớp

VTPham@IUH, 2024

Ôn lại bài học trước

- Bạn có nhớ ? % ?

Nội dung chính

- 2.3.1 Bài toán phân lớp
- 2.3.2 Hàm mục tiêu
- 2.3.3 K-nearest neighbors
- 2.3.4 Support vector machine
- 2.3.5 Decision tree
- 2.3.6 Ví dụ về bài toán phân lớp

2.3 Bài toán phân lớp

2.3.1 Mô tả bài toán

- Trong thực tế có nhiều bài toán có biến mục tiêu đầu ra là rời rạc => bài toán phân lớp/phân loại
  - Trong y học:
    - Chuẩn đoán hình ảnh => Chuẩn đoán: mắc bệnh hoặc bình thường
  - Trong kinh tế:
    - Giao dịch thẻ ngân hàng => Phát hiện: gian lận/bất thường hoặc bình thường
    - Định giá trị => Quyết định đầu tư: triển vọng cao, triển vọng trung bình, triển vọng thấp, tránh đầu tư
  - Trong nông nghiệp:
    - Thu hoạch nông sản => Quả: chín (có thể thu hoạch) hoặc chưa chín (để lại)

2.3.1 Mô tả bài toán

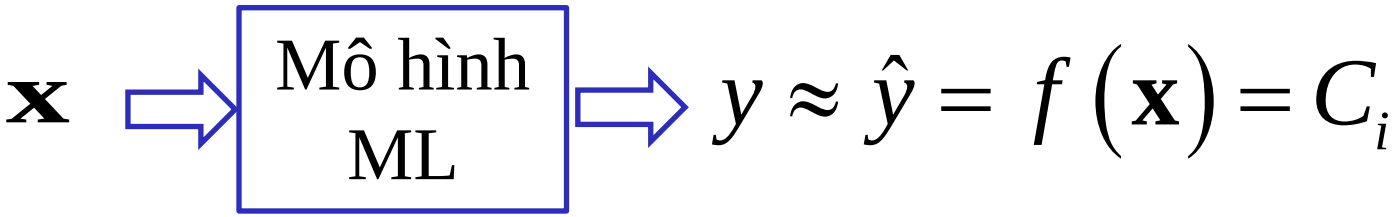
Có dữ liệu đầu vào, biểu diễn bởi vector đặc trưng (và nhãn)

Giá trị đầu ra là một nhãn hay lớp (giá trị rời rạc)

Nhãn thuộc tập hợp hữu hạn có *K* phần tử khác nhau

$$\{C_1, C_2, ..., C_K\}$$

$$f : \mathbb{R}^M \rightarrow \{C_1, C_2, ..., C_K\}$$



2.3.2 Hàm mục tiêu

y : giá trị đúng (mong đợi)

$\hat{y}$: giá trị phân loại

$e = y - \hat{y}$: sai số phân loại ?

Xét nhiều điểm dữ liệu $\left(\mathbf{x}_i, y_i\right)$ $i = 1, 2, ..., N$

N : số lượng điểm dữ liệu xem xét

Xây dựng hàm mục tiêu => tối ưu hóa

2.3.3 K-nearest neighbors

- Thuật toán k-NN: thuật toán học máy đơn giản
- Việc xây dựng mô hình chỉ bao gồm việc lưu trữ tập dữ liệu huấn luyện
- Để đưa ra dự đoán cho một điểm dữ liệu mới, thuật toán sẽ tìm các điểm dữ liệu gần nhất (~ độ tương đồng) trong tập dữ liệu huấn luyện (được coi như "hàng xóm gần nhất")

k-NN

- *k*: số “hàng xóm gần nhất”
- Cho điểm dữ liệu mới
  - Chọn ra được tập *k* “hàng xóm gần nhất”
  - Nhãn phân loại của điểm dữ liệu mới được quyết định dựa trên nhãn của tập k “hàng xóm gần nhất”
    - Ví dụ: bầu chọn theo đa số (max-voting)

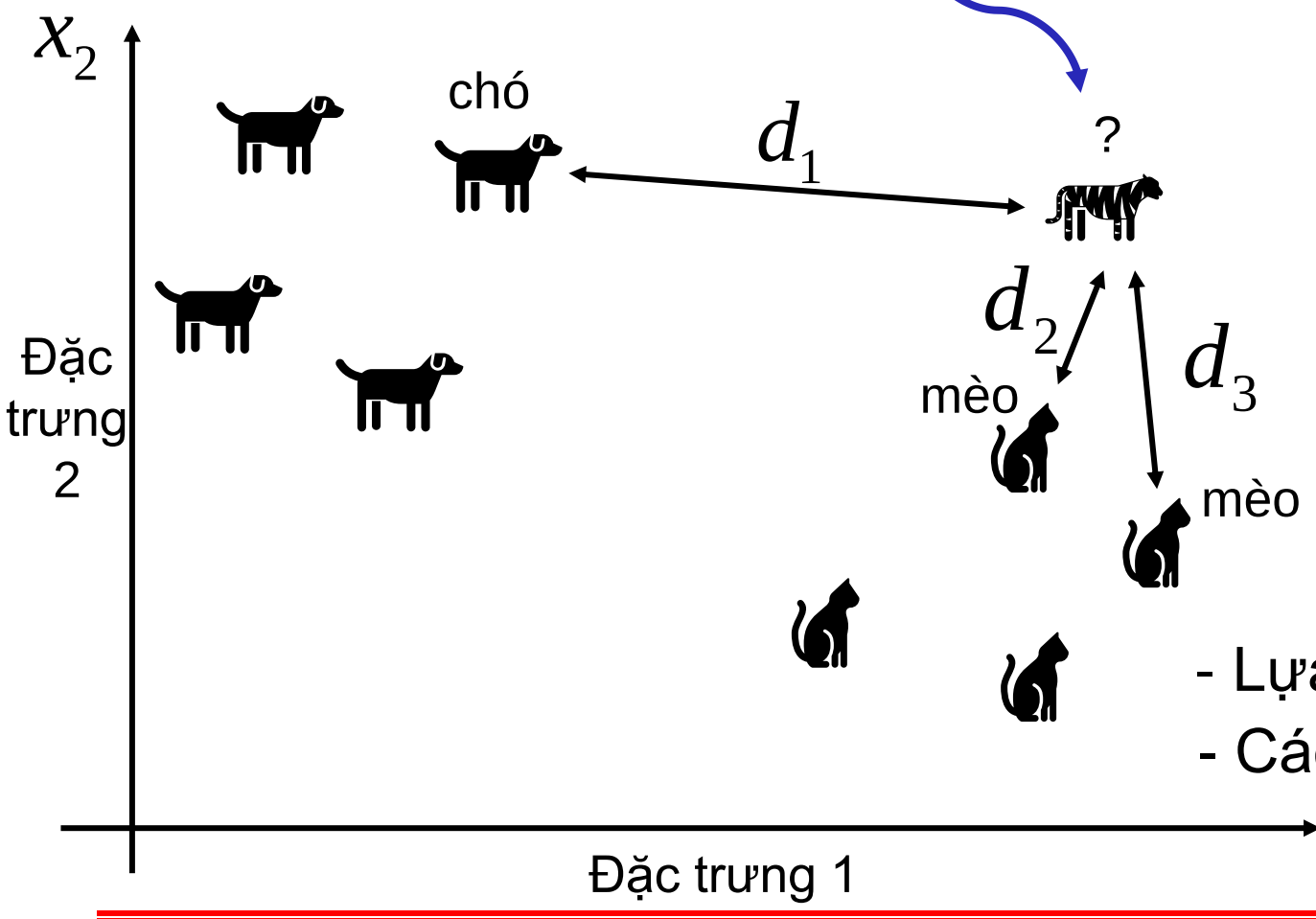
k-NN

- Chú ý
  - Lựa chọn *k*
  - Các bước tiền xử lý để chuẩn hóa vector đặc trưng
    - Có thể áp dụng: rescaling, standardization, …
  - Có thể đánh trọng số cho thuộc tính
  - Xem xét đánh trọng số cho *k* “hàng xóm gần nhất”
    - Scikit-learn:

      KNeighborsClassifier(n_neighbors = 5, p = 2, weights = ‘distance‘)

k-NN

$k = 3$ $\quad d_1 > d_3 > d_2$ $\quad d_i = \|\mathbf{z} - \mathbf{x}_i\|_2$



?

- Lựa chọn giá trị của *k*
- Cách để đo lường *d*

k-NN

Chuẩn của vector $\|\mathbf{x}\|_p$

$$\|\mathbf{x}\|_2 = \sqrt{x_1^2 + \ldots + x_n^2} \qquad \mathbf{x} \in \mathbb{R}^n$$

$$d = \|\mathbf{z} - \mathbf{x}\|_2$$

Hoặc chọn

$$d = \|\mathbf{z} - \mathbf{x}\|_2^2$$

$$= (\mathbf{z} - \mathbf{x})^T (\mathbf{z} - \mathbf{x}) = \|\mathbf{z}\|_2^2 + \|\mathbf{x}\|_2^2 - 2\mathbf{x}^T\mathbf{z}$$

Khoảng cách

- Euclidean Distance (l_2 norm)
- Manhattan Distance (l_1 norm)
- Minkowski Distance (l_p norm)

- Hamming Distance

https://www.turing.com/kb/how-to-decide-perfect-distance-metric-for-machine-learning-model

Ví dụ

- Smart house - Bkav

Ánh sáng thông minh

Tự động điều chỉnh hệ thống chiếu sáng theo các kịch bản và thói quen người dùng.

Chi tiết

Rèm cửa thông minh

Điều khiển rèm cửa từ xa, tự động đóng mở theo kịch bản, theo điều kiện ánh sáng và sở thích của người dùng.

Chi tiết

An ninh thông minh

Giám sát an ninh đa lớp thông minh. Cảnh báo cháy nổ, chập điện, an toàn cho trẻ nhỏ.

Chi tiết

Kịch bản điều khiển thông minh

Hệ thống Chiếu sáng, Rèm cửa, Môi trường ... cùng kết hợp hoạt động theo kịch bản chung.

Chi tiết

Trợ lý ảo, điều khiển giọng nói

Trải nghiệm tuyệt vời khi ra lệnh bằng giọng nói và ngôi nhà nghe theo yêu cầu của bạn

Chi tiết

Học thói quen người dùng

Chủ động hoạt động theo thói quen để đáp ứng nhu cầu, sở thích thường ngày của gia chủ

Chi tiết

Hệ thống giải trí âm thanh đa vùng

Giải trí âm thanh theo sở thích và nhu cầu, không bị giới hạn về địa điểm hay thời gian

Chi tiết

Hệ thống kiểm soát môi trường

Luôn đảm bảo trạng thái môi trường trong lành, tốt nhất cho sức khỏe của gia chủ

Chi tiết

https://smarthome.com.vn/?gad_source=1&gclid=CjwKCAjwoJa2BhBPEiwA0l0ImL2iffIzSB9nAjYm_hB3j5IgEvFdTpAeYdDoHnT6DVmdCOlHk22RARoCCzMQAvD_BwE

https://smarthomekit.vn/nha-thong-minh-tu-a-z/?srsltid=AfmBOooN4bSWU-8ikzma91ZjD64iWsL4y06LDkjqT-vDwgcY9-73Mnwj

Ví dụ

- Hệ thống kiểm soát môi trường

https://smarthome.com.vn/he-thong-kiem-soat-moi-truong

Ví dụ

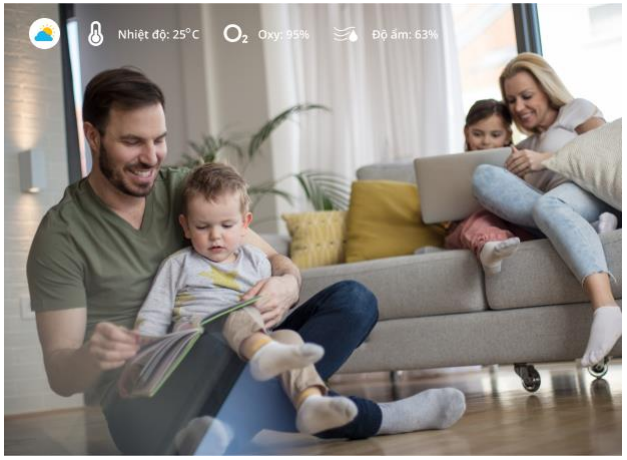
- Hệ thống kiểm soát môi trường

Cài đặt môi trường theo sở thích

Hệ thống tự động vận hành và cho phép gia chủ có thể tùy ý cài đặt kịch bản, thông số môi trường theo nhu cầu, sở thích của mình.

Tự điều chỉnh theo mùa và khung giờ

Mỗi mùa, mỗi cung giờ có các đặc tính riêng: hè khô, đông ẩm, ngày nóng, đêm lạnh... các thông số này được tính toán kỹ để đưa ra môi trường phù hợp theo mùa và khung giờ.



https://smarthome.com.vn/he-thong-kiem-soat-moi-truong

Ví dụ

Mẫu 1 : (22°C, 50%), Living Room

Mẫu 2 : (24°C, 45%), Living Room

Mẫu 3: (23°C, 47%), Living Room

Mẫu 4: (21°C, 49%), Living Room

Mẫu 5: (25°C, 44%), Living Room

Mẫu 6: (19°C, 55%), Bedroom

Mẫu 7: (20°C, 60%), Bedroom

Mẫu 8: (18°C, 58%), Bedroom

Mẫu 9: (17°C, 57%), Bedroom

Mẫu 10: (20°C, 62%), Bedroom

Mẫu 11: (23°C, 46%), Living Room

Mẫu 12: (24°C, 48%), Living Room

Mẫu 13: (19°C, 54%), Bedroom

Mẫu 14: (18°C, 56%), Bedroom

$(\mathbf{x}_i, y_i)$

Mẫu 1 : (22, 50), Living Room

(**x**(nhiệt_độ, độ_ẩm), kiểu_phòng)

Sử dụng *k*-NN với *k* = 5

Mẫu mới: (21°C, 52%)

Kiểu phòng ?

Mẫu 15: (°C, 48%), Living Room

Mẫu 16: (19°C, 54%),

Mẫu 17: (18°C, %), Bedroom

Ví dụ

Mẫu 1 : (22°C, 50%), Living Room

Mẫu 2 : (24°C, 45%), Living Room

Mẫu 3: (23°C, 47%), Living Room

Mẫu 4: (21°C, 49%), Living Room

Mẫu 5: (25°C, 44%), Living Room

Mẫu 6: (19°C, 55%), Bedroom

Mẫu 7: (20°C, 60%), Bedroom

Mẫu 8: (18°C, 58%), Bedroom

Mẫu 9: (17°C, 57%), Bedroom

Mẫu 10: (20°C, 62%), Bedroom

Mẫu 11: (23°C, 46%), Living Room

Mẫu 12: (24°C, 48%), Living Room

Mẫu 13: (19°C, 54%), Bedroom

Mẫu 14: (18°C, 56%), Bedroom

$(\mathbf{x}_i, y_i)$

Mẫu 1 : (22, 0.5), Living Room

(**x**(nhiệt_độ, độ_ẩm), kiểu_phòng)

Sử dụng *k*-NN với *k* = 5

Mẫu mới: (21°C, 52%)

(22, 0.52)

Kiểu phòng ?

Nhận xét ?

Mẫu 15: (°C, 48%), Living Room

Mẫu 16: (19°C, 54%),

Mẫu 17: (18°C, %), Bedroom

Ưu và nhược điểm

Ưu điểm

- Độ phức tạp tính toán cho quá trình huấn luyện thấp
- Việc dự đoán kết quả của dữ liệu đầu vào mới đơn giản
- Không cần giả sử về phân phối của các lớp

Nhược điểm

- Rất nhạy cảm với nhiễu khi *k* nhỏ
- Khi *k* càng lớn => việc tính toán khoảng cách tới từng điểm dữ liệu trong tập huấn luyện tăng
- Việc lưu trữ toàn bộ dữ liệu trong bộ nhớ ảnh hưởng tới hiệu năng của k-NN

k-NN

- Có thể dùng k-NN cho bài toán xấp xỉ hàm?

- Scikit-learn: KNeighborsRegressor( )

2.3.4 Support vector machine (SVM)

- Vapnik–Chervonenkis theory (VC theory) được phát triển trong khoảng 1960–1990 bởi Vladimir Vapnik và Alexey Chervonenkis

- SVM được phát triển từ AT&T Bell Laboratories bởi Vladimir Vapnik và các đồng nghiệp

Vladimir Vapnik

https://datascience.columbia.edu/people/vladimir-vapnik/

2.3.4 Support vector machine (SVM)

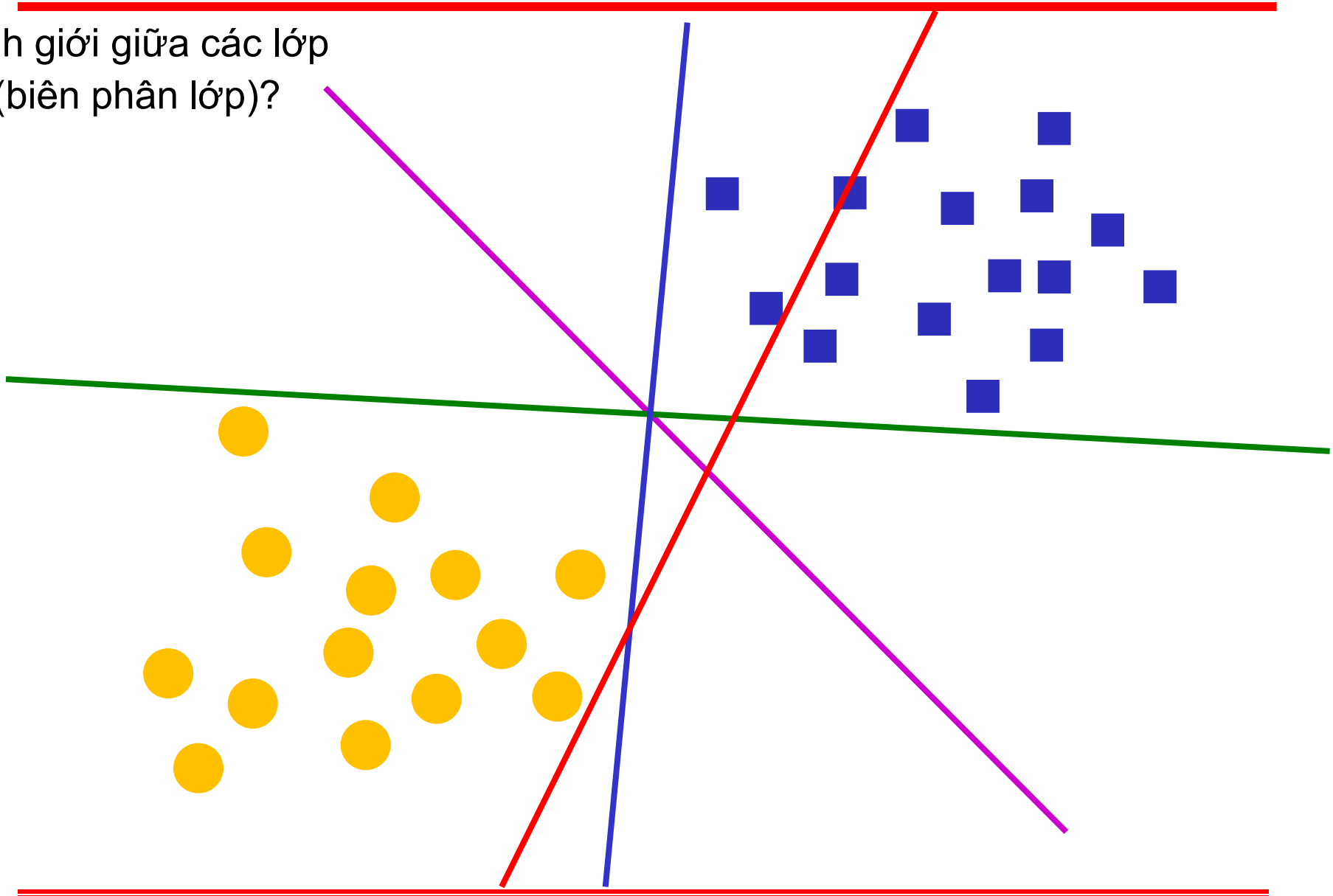
- Mạnh và đa năng
  - Có khả năng thực hiện phân loại (tuyến tính hoặc phi tuyến tính), hồi quy (tuyến tính hoặc phi tuyến tính) và thậm chí phát hiện giá trị ngoại lai/lệ (outlier detection)
- Một trong những mô hình phổ biến nhất trong ML
- Đặc biệt phù hợp để phân loại các tập dữ liệu phức tạp có quy mô nhỏ hoặc trung bình

Support vector machine (SVM)

- Thông thường chỉ có 1 số ít các điểm dữ liệu trong tập huấn luyện liên quan tới việc định ra đường bao giữa 2 lớp
  - Đó là những điểm nằm ở biên giữa 2 lớp => được gọi là các vector hỗ trợ (support vector)
- Tính khoảng cách của mỗi vector hỗ trợ để đưa ra quyết định đối với 1 điểm dữ liệu mới
  - Việc phân lớp được thực hiện dựa trên các khoảng cách tới các vector hỗ trợ và tầm quan trọng của các vector hỗ trợ được học thông qua quá trình huấn luyện

Support vector machine (SVM)

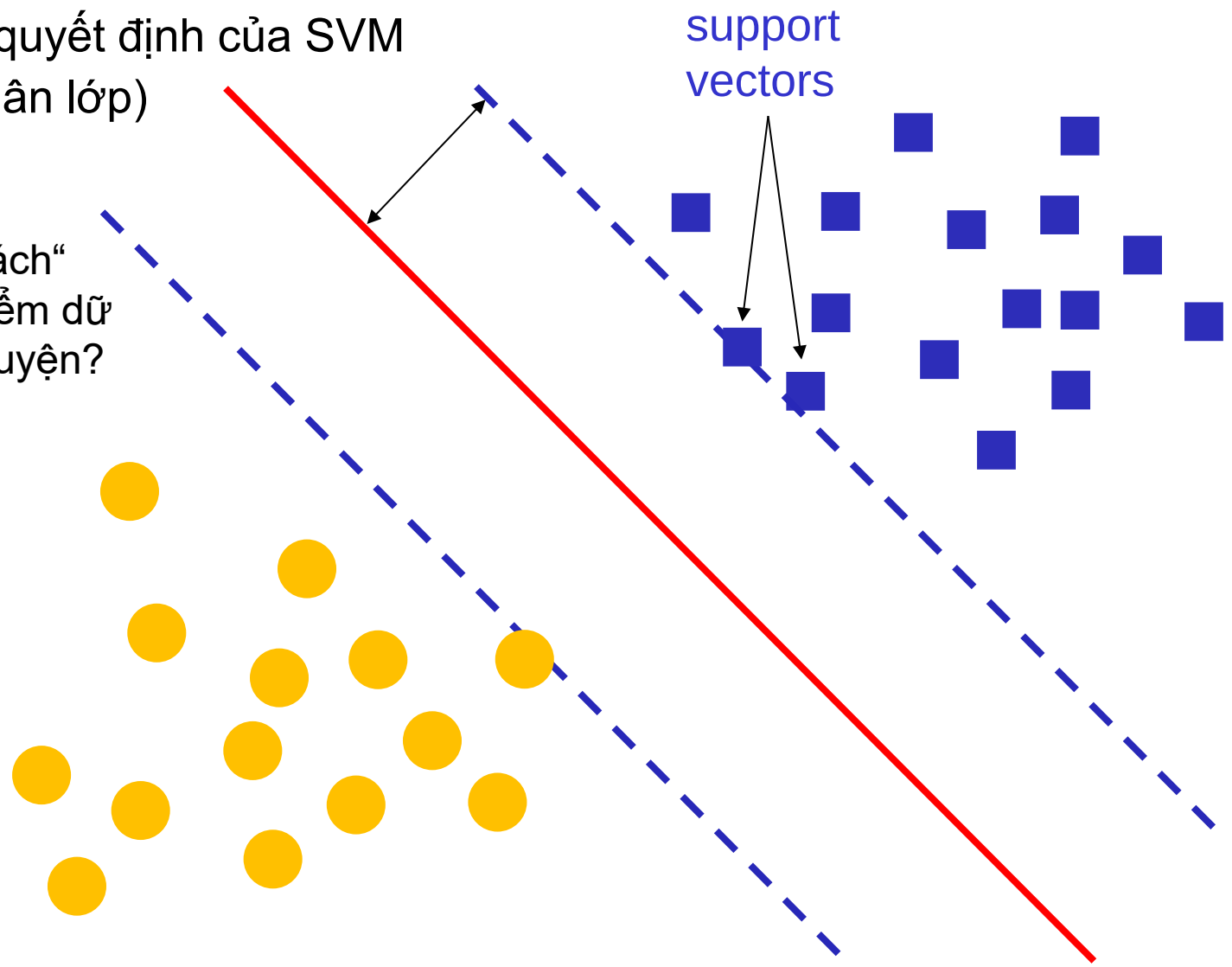
ranh giới giữa các lớp
(biên phân lớp)?



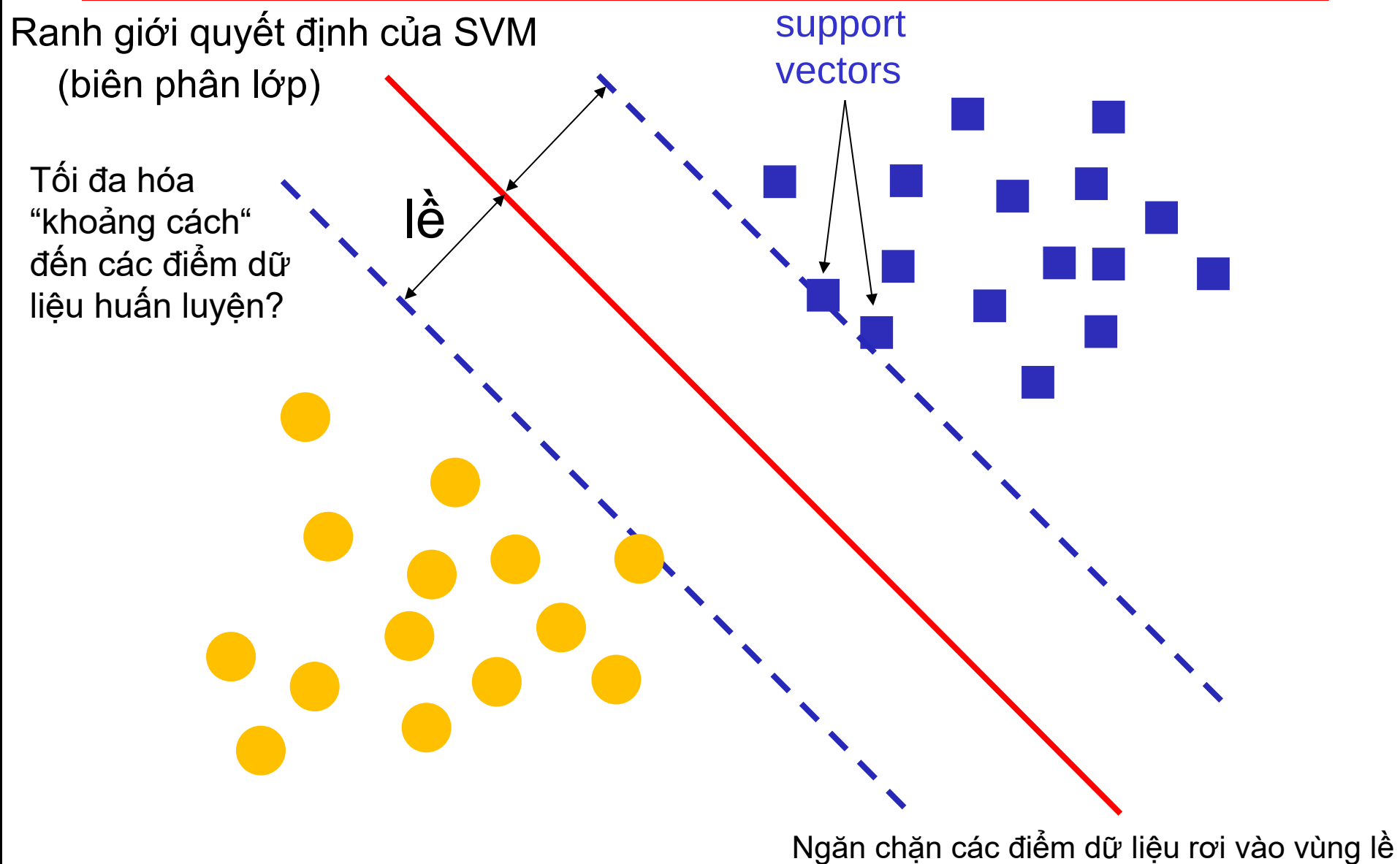
Support vector machine (SVM)

Ranh giới quyết định của SVM
(biên phân lớp)

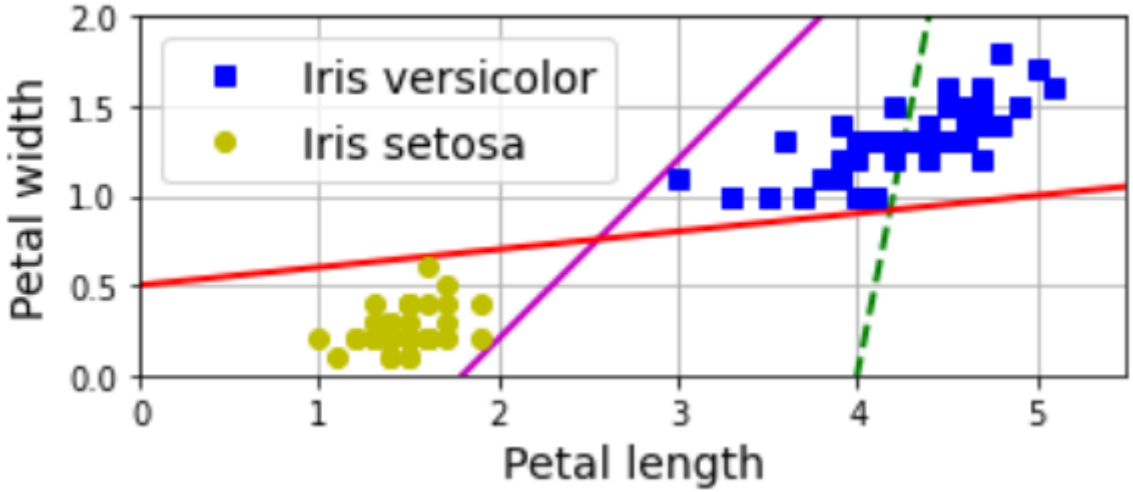
Tối đa hóa
"khoảng cách"
đến các điểm dữ
liệu huấn luyện?



Support vector machine (SVM)



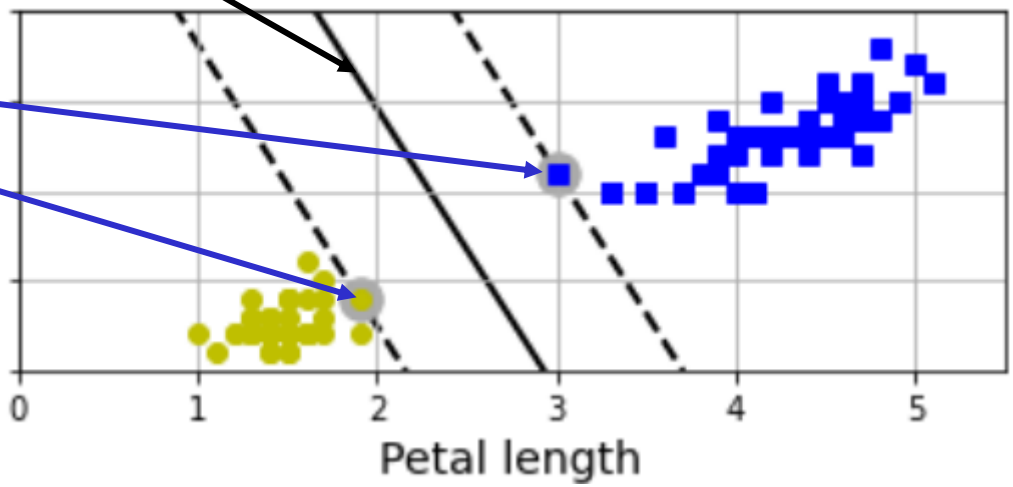
Phân lớp SVM tuyến tính



Ranh giới quyết định của SVM

large margin classification

support vectors



Phân lớp SVM tuyến tính

- Ôn tập lại

- Khoảng cách từ một điểm đến một mặt phẳng

$$A(x_A, y_A, z_A) \qquad w_1 x + w_2 y + w_3 z + b = 0$$

$$d = \frac{|w_1 x_A + w_2 y_A + w_3 z_A + b|}{\sqrt{w_1^2 + w_2^2}}$$

- Khoảng cách từ một điểm đến một siêu phẳng

$$A(x_{1A}, x_{2A}, \ldots, x_{MA}) \qquad w_1 x_1 + w_2 x_2 + \ldots + w_M x_M + b = 0$$

$$\mathbf{w}^T \mathbf{x} + b = 0$$

$$d = \frac{|w_1 x_{1A} + w_2 x_{1A} + \ldots + w_M x_{MA} + b|}{\sqrt{w_1^2 + w_2^2 + \ldots + w_M^2}} = \frac{|\mathbf{w}^T \mathbf{x}_A + b|}{\|\mathbf{w}\|_2}$$

Phân lớp SVM tuyến tính

?

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b = w_1x_1 + w_2x_2 + \ldots + w_Mx_M + b$$

Ràng buộc
$$\begin{cases} \mathbf{w}^T\mathbf{x} + b \geq 1, & y = 1 \\ \mathbf{w}^T\mathbf{x} + b \leq -1, & y = -1 \end{cases}$$

=> Ngăn chặn các điểm dữ liệu rơi vào vùng lề

$\hat{y}$: giá trị phân loại (dự đoán)

M : số lượng đặc trưng

b : hệ số bias

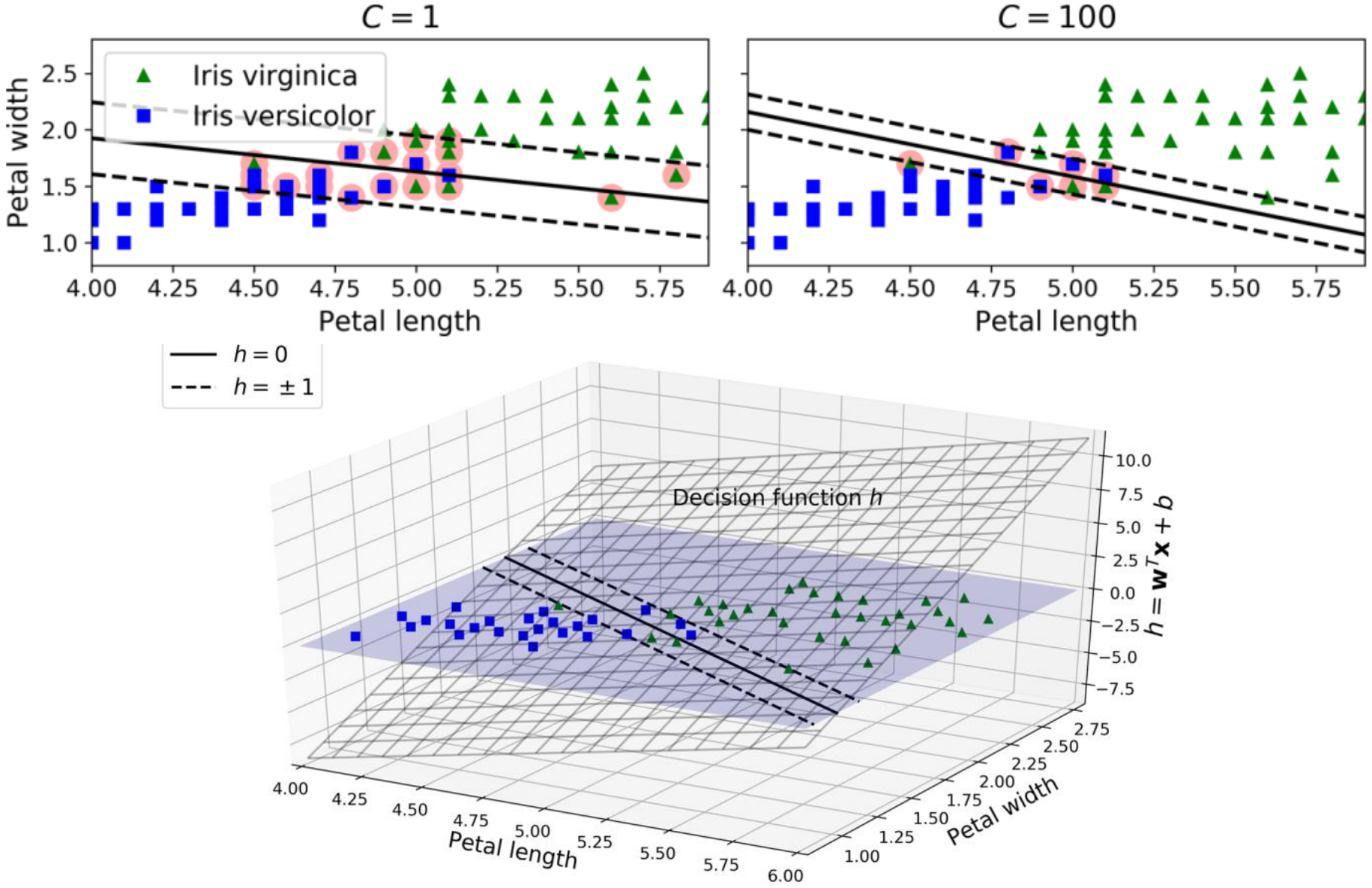
x_i : giá trị của đặc trưng thứ *i*

w_i : hệ số thứ *i* của mô hình

C_i : lớp thứ *i* (0,1)

Xét bài toán phân lớp nhị phân

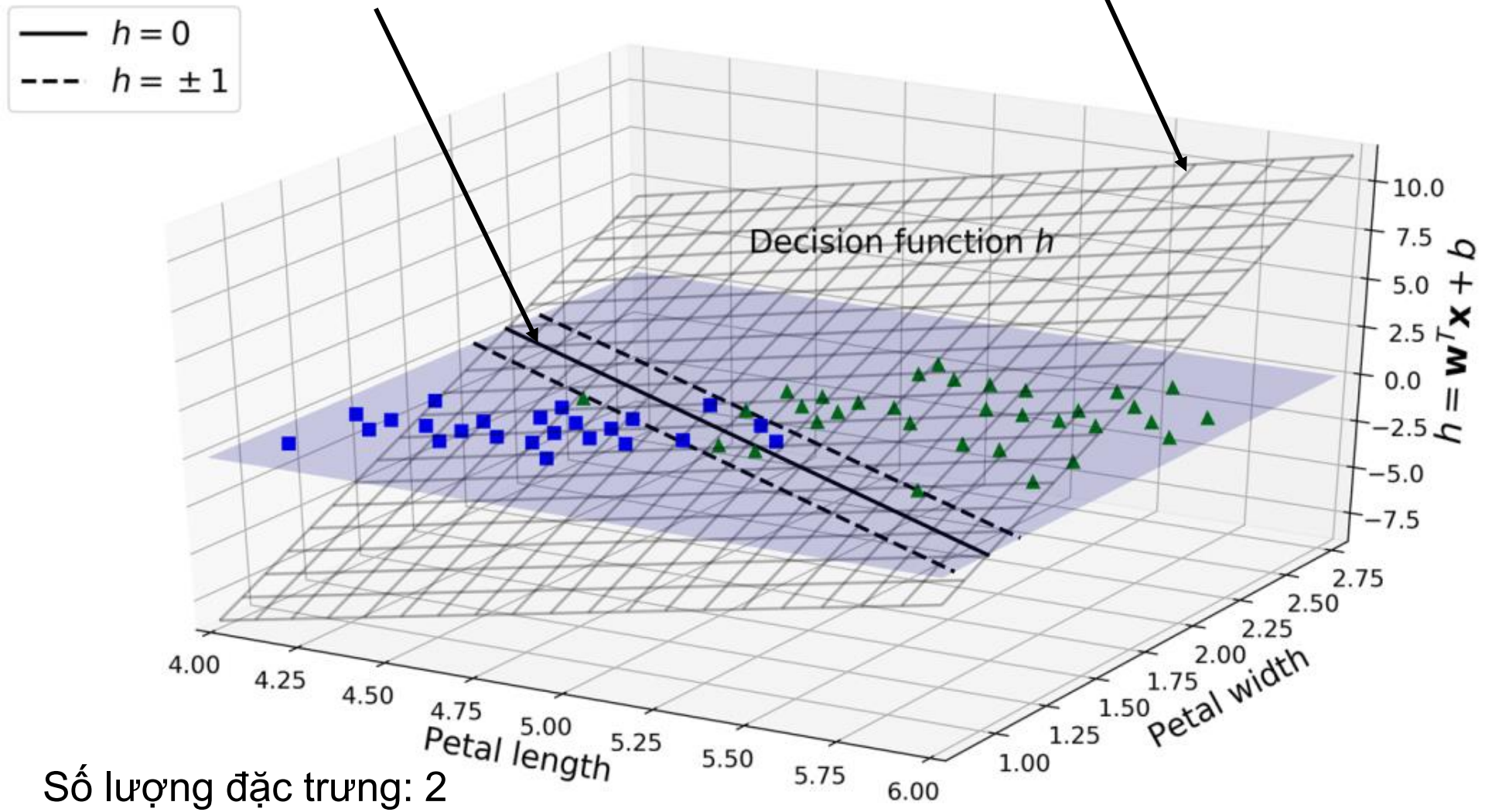
Phân lớp SVM tuyến tính



Phân lớp SVM tuyến tính

Ranh giới quyết định:
đường thẳng 1 chiều

Hàm quyết định *h*: mặt phẳng 2 chiều



Số lượng đặc trưng: 2

Phân lớp SVM tuyến tính

- Số lượng đặc trưng: *M*
- Hàm quyết định: siêu phẳng *M* chiều
- Ranh giới quyết định: siêu phẳng (*M*-1) chiều

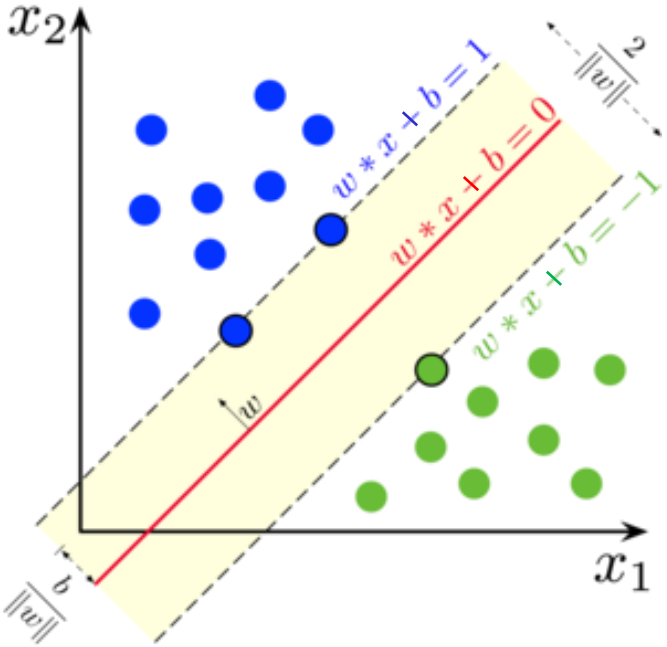
Huấn luyện: Tìm các giá trị của **w** và *b* làm cho

- Phần lề càng rộng càng tốt
- Trong khi cần tránh mẫu vi phạm (đối với lề cứng) hoặc hạn chế vi phạm (đối với lề mềm)

=> Tối ưu hóa có ràng buộc

Phân lớp SVM tuyến tính

Khoảng cách từ 1 điểm dữ liệu đến mặt phân lớp $\frac{y_i\left(\mathbf{w}^T\mathbf{x}_i+b\right)}{\|\mathbf{w}\|_2}$ => phần lề: $\frac{2}{\|\mathbf{w}\|_2}$



Chú ý: $y_i\left(\mathbf{w}^T\mathbf{x}_i+b\right)\geq 1$

Bài toán tối ưu có ràng buộc:

$$\left(\mathbf{w}^*,b^*\right)=\arg\min_{\mathbf{w},b}\frac{1}{2}\|\mathbf{w}\|_2^2$$

thỏa mãn: $y_i\left(\mathbf{w}^T\mathbf{x}_i+b\right)-1\geq 0,\ \forall i=1,2,\ldots,N$

https://upload.wikimedia.org/wikipedia/commons/thumb/7/72/SVM_margin.png/600px-SVM_margin.png

Phân lớp SVM tuyến tính

- Lề cứng

Giải bài toán tối ưu có ràng buộc ta sẽ tìm được siêu phẳng phân lớp

$$\left(\boldsymbol{\theta}^*, b^*\right) \qquad \mathbf{w}^*\mathbf{x} + b^*$$

Để áp dụng cho điểm dữ liệu mới ?

$$\hat{y} = \operatorname{sgn}\left(\mathbf{w}^*\mathbf{x} + b^*\right)$$

Ghi chú: Các em có thể đọc thêm tham khảo cách giải bài toán tối ưu có ràng buộc: tối ưu hóa lồi - convex optimization (thư viện CVXORT); quy hoạch tuyến tính – linear programming, quadratic programming, bài toán đối ngẫu (dual problem), điều kiện KKT – Karush-Kuhn-Tucker (KKT) condition

Bài toán đối ngẫu

- Xét một bài toán tối ưu hóa có ràng buộc (bài toán cơ bản)
  - có thể biểu diễn một bài toán khác nhưng có liên quan chặt chẽ, được gọi là bài toán đối ngẫu của nó
    - Bài toán đối ngẫu có thể dễ giải hơn
      - Có thể dùng cho SVM tuyến tính

Bài toán đối ngẫu

$$\boldsymbol{\alpha}^* = \arg\max_{\boldsymbol{\alpha}} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

thỏa mãn:
$$\begin{cases} \sum_{i=1}^{N} \alpha_i y_i = 0 \\ \alpha_i \geq 0, \forall i = 1, 2, \ldots, N \end{cases}$$

Các hệ số nhân Lagrange

Có thể áp dụng thủ thuật kernel

Từ nghiệm của bài toán đối ngẫu => nghiệm của bài toán cơ bản

$$\mathbf{w}^* = \sum_{i=1}^{N} \alpha_i^* y_i \mathbf{x}_i$$

$\alpha_i = 0, \forall \mathbf{x}_i \notin \text{SV}$

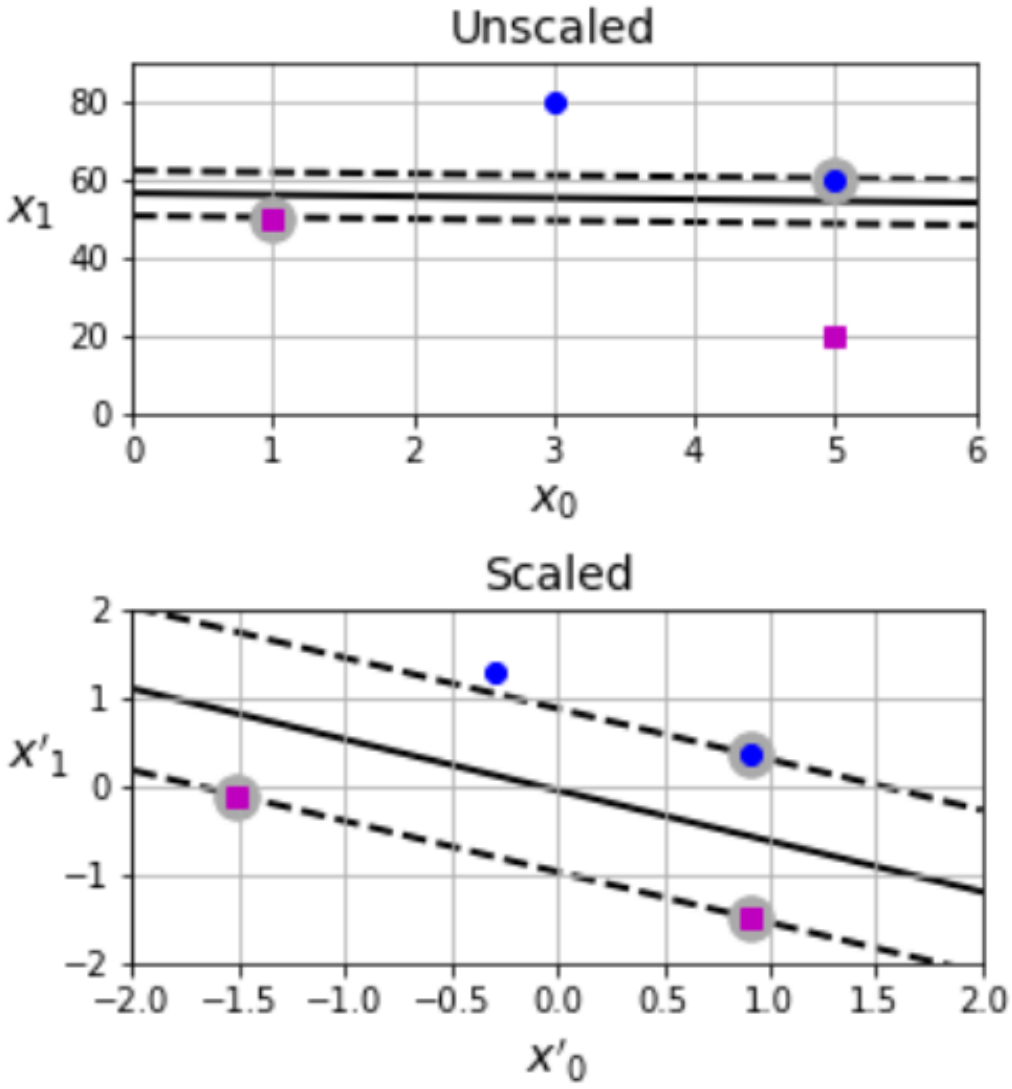
SV: tập các vector hỗ trợ

$$b^* = \frac{1}{N_{SV}} \sum_{\substack{i=1 \\ \alpha_i^* > 0}}^{N} y_i - \left(\mathbf{w}^*\right)^T \mathbf{x}_i$$

N_{SV} : số lượng vector hỗ trợ

Phân lớp SVM tuyến tính

- SVM nhạy cảm với tỷ lệ đặc trưng



Thay đổi tỷ lệ làm cho ranh giới quyết định tôt hơn

Phân lớp SVM tuyến tính

- Lề mềm (~ hạn chế vi phạm)

$$\begin{cases} \mathbf{w}^T\mathbf{x}+b \geq 1+\zeta_i, & y=1 \\ \mathbf{w}^T\mathbf{x}+b \leq -1-\zeta_i, & y=-1 \end{cases} \qquad \zeta_i \geq 0 \text{ : slack variable}$$

Bài toán tối ưu có ràng buộc:

$$\left(\mathbf{w}^*, b^*, \boldsymbol{\zeta}^*\right) = \arg\min_{\mathbf{w},b,\boldsymbol{\zeta}} \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{N}\zeta_i$$

thỏa mãn: $y_i\left(\mathbf{w}^T\mathbf{x}_i+b\right)-1+\zeta_i \geq 0, \quad \forall i=1,2,\ldots,N$

$$\zeta_i \geq 0, \quad \forall i=1,2,\ldots,N$$

$C \geq 0$: siêu hệ số (hyperparameter)

Phân lớp SVM tuyến tính

- Chú ý:
  - Có thể sử dụng Gradient descent để tối thiểu hóa hàm hingle loss hoặc squared hinge loss

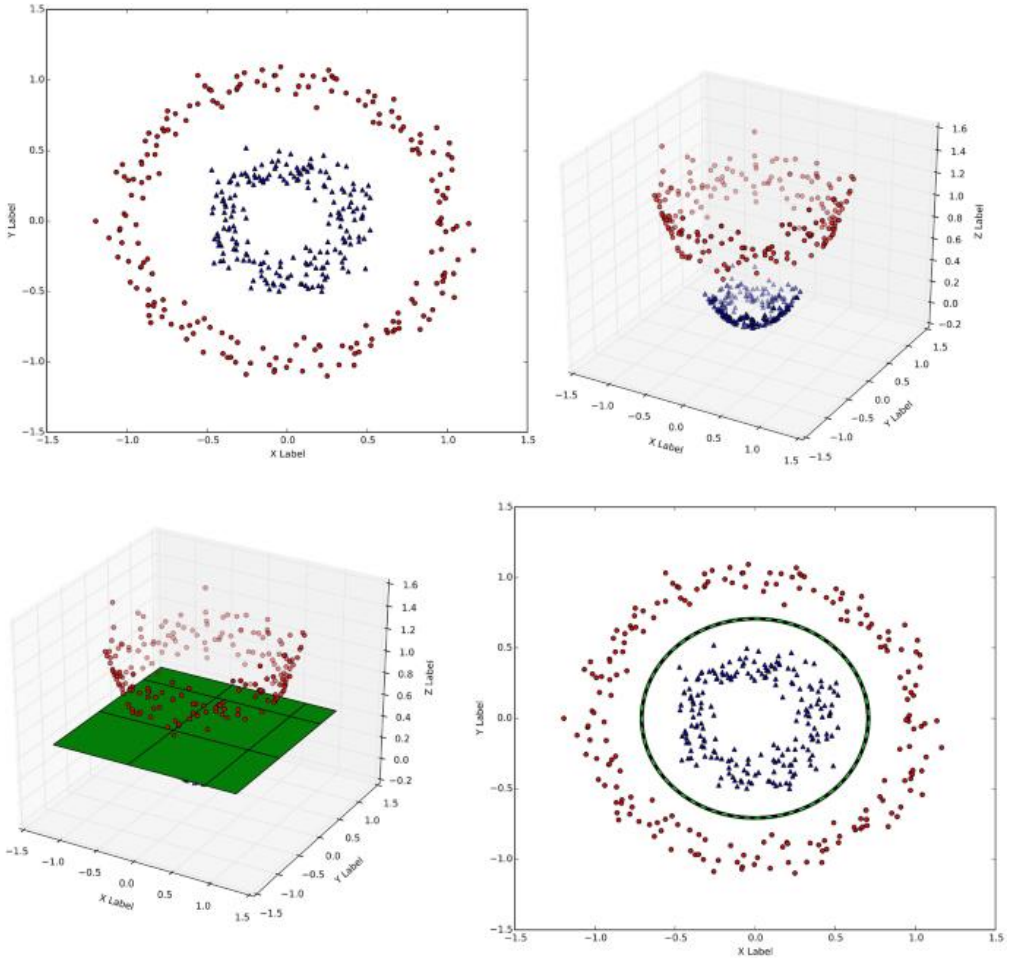
Phân lớp SVM phi tuyến

- Khi tập dữ liệu có thể phân lớp phi tuyến?
  - Chuyển sang không gian mới (thường nhiều chiều hơn), trong đó dữ liệu có thể phân lớp tuyến tính
  - Áp dụng lại như trong SVM tuyến tính

Phân lớp SVM phi tuyến

$$\mathbf{x} \mapsto \Phi(\mathbf{x})$$

Input space $\Longrightarrow$ Feature space



https://media.springernature.com/lw685/springer-static/image/art%3A10.1007%2Fs10115-019-01335-4/MediaObjects/10115_2019_1335_Fig13_HTML.png

Kernel

Xét một phép biến đổi: $\Phi(.)$

Hai vector là: $\mathbf{a}, \mathbf{b}$

Cần tính: $\Phi(\mathbf{a})^T \Phi(\mathbf{b})$

Kernel là một hàm giúp tính toán $\Phi(\mathbf{a})^T \Phi(\mathbf{b})$ dựa trên các vector gốc $\mathbf{a}, \mathbf{b}$ mà không cần phải tính toán phép biến đổi $\Phi(.)$ (thậm chí không cần biết $\Phi(.)$)

	Một số kernel phổ biến
Linear	$K(\mathbf{a},\mathbf{b}) = \mathbf{a}^T\mathbf{b}$
Polynomial	$K(\mathbf{a},\mathbf{b}) = (\gamma \mathbf{a}^T\mathbf{b} + r)^d$
Gaussian RBF	$K(\mathbf{a},\mathbf{b}) = \exp(-\gamma \lVert \mathbf{a}-\mathbf{b} \rVert^2)$
Sigmoid	$K(\mathbf{a},\mathbf{b}) = \tanh(\gamma \mathbf{a}^T\mathbf{b} + r)$

Kernel

- Ví dụ

Xét phép biến đổi $\Phi(\mathbf{x}) = \Phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \begin{bmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{bmatrix}$

với $\mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$

(a) Tính $\Phi(\mathbf{a})^T \Phi(\mathbf{b})$

(b) Áp dụng kernel *K* và so sánh với kết quả ở câu (a)

$$K(\mathbf{a}, \mathbf{b}) = (\mathbf{a}^T\mathbf{b})^2$$

Bài toán đối ngẫu

$$\boldsymbol{\alpha}^* = \arg\max_{\boldsymbol{\alpha}} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j)$$

$$K\left(\Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j)\right)$$

thỏa mãn:

$$\begin{cases} \sum_{i=1}^{N} \alpha_i y_i = 0 \\ \alpha_i \geq 0, \forall i = 1, 2, \ldots, N \end{cases}$$

Các hệ số nhân Lagrange

Có thể áp dụng thủ thuật kernel

Từ nghiệm của bài toán đối ngẫu => nghiệm của bài toán cơ bản

$$\mathbf{w}^* = \sum_{i=1}^{N} \alpha_i^* y_i \Phi(\mathbf{x}_i)$$

$\alpha_i = 0, \forall \mathbf{x}_i \notin \text{SV}$

SV: tập các vector hỗ trợ

$$b^* = \frac{1}{N_{SV}} \sum_{\substack{i=1 \\ \alpha_i^* > 0}}^{N} y_i - \left(\mathbf{w}^*\right)^T \Phi(\mathbf{x}_i)$$

N_{SV} : số lượng vector hỗ trợ

Kernelized SVM

- Ví dụ: Cho 1 mẫu mới $\mathbf{x}_n$, tính hàm quyết định *h*

$$h_{\mathbf{w}^*,b^*}\left(\Phi\left(\mathbf{x}_n\right)\right)=\left(\mathbf{w}^*\right)^T\Phi\left(\mathbf{x}_n\right)+b^*$$

$$=\left(\sum_{i=1}^{N}\alpha_i^* y_i\Phi\left(\mathbf{x}_i\right)\right)^T\Phi\left(\mathbf{x}_n\right)+b^*$$

$$=\sum_{i=1}^{N}\alpha_i^* y_i\left(\Phi\left(\mathbf{x}_i\right)^T\Phi\left(\mathbf{x}_n\right)\right)+b^*=\sum_{\substack{i=1\\ \alpha_i^*>0}}^{N}\alpha_i^* y_i K\left(\mathbf{x}_i,\mathbf{x}_n\right)+b^*$$

$\alpha_i^* \neq 0$ chỉ đối với các vector hỗ trợ

Kernelized SVM

- Ví dụ: Cho 1 mẫu mới $\mathbf{x}_n$, tính thành phần bias *b*

$$b^* = \frac{1}{N_{SV}} \sum_{\substack{i=1 \\ \alpha_i^* > 0}}^{N} y_i - \left(\mathbf{w}^*\right)^T \Phi\left(\mathbf{x}_i\right)$$

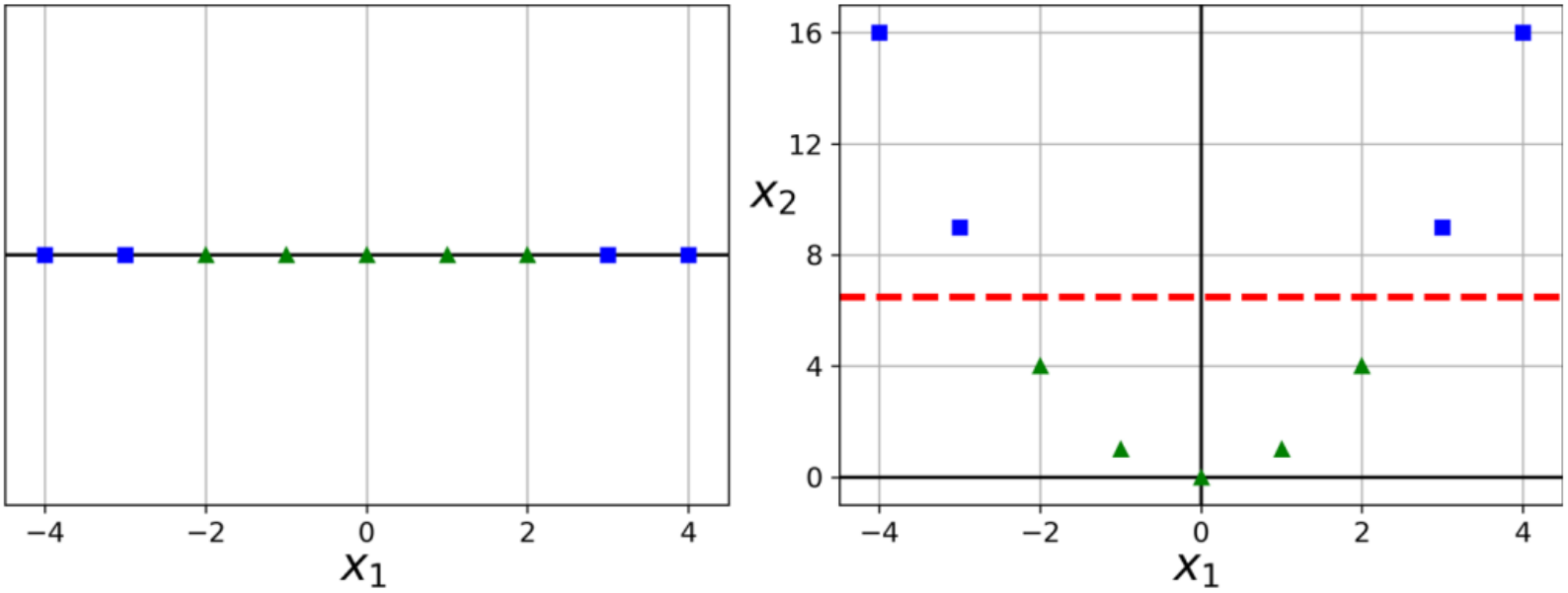
$$= \frac{1}{N_{SV}} \sum_{\substack{i=1 \\ \alpha_i^* > 0}}^{N} \left(y_i - \left(\sum_{j=1}^{N} \alpha_j^* y_j \Phi\left(\mathbf{x}_j\right) \right)^T \Phi\left(\mathbf{x}_i\right) \right)$$

$$= \frac{1}{N_{SV}} \sum_{\substack{i=1 \\ \alpha_i^* > 0}}^{N} \left(y_i - \sum_{\substack{j=1 \\ \alpha_j^* > 0}}^{N} \alpha_j^* y_j K\left(\mathbf{x}_i, \mathbf{x}_j\right) \right)$$

Phân lớp SVM phi tuyến

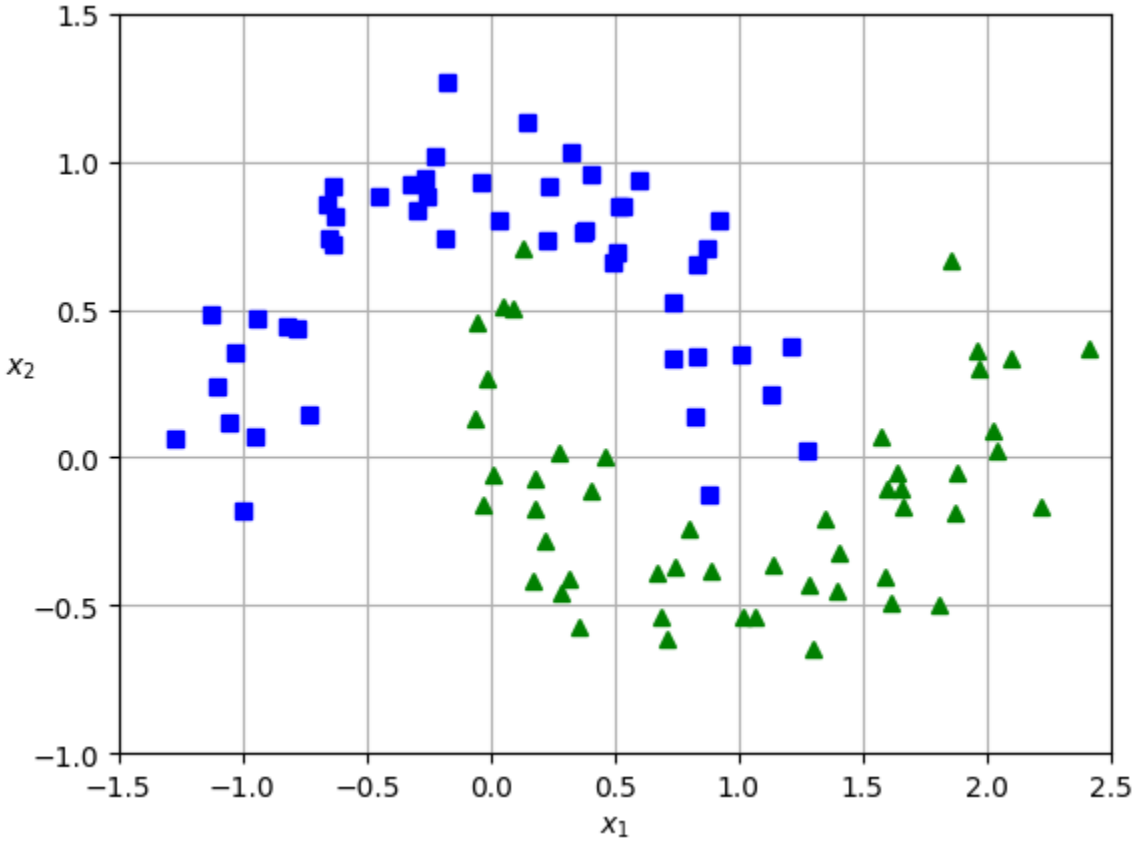
- Nhận xét:
  - Trong một số trường hợp có thể xử lý các tập dữ liệu phi tuyến thông qua việc thêm các đặc trưng, ví dụ đặc trưng đa thức => tập dữ liệu có thể phân tách tuyến tính

$$x_2 = x_1^2$$



Phân lớp SVM phi tuyến

- Ví dụ



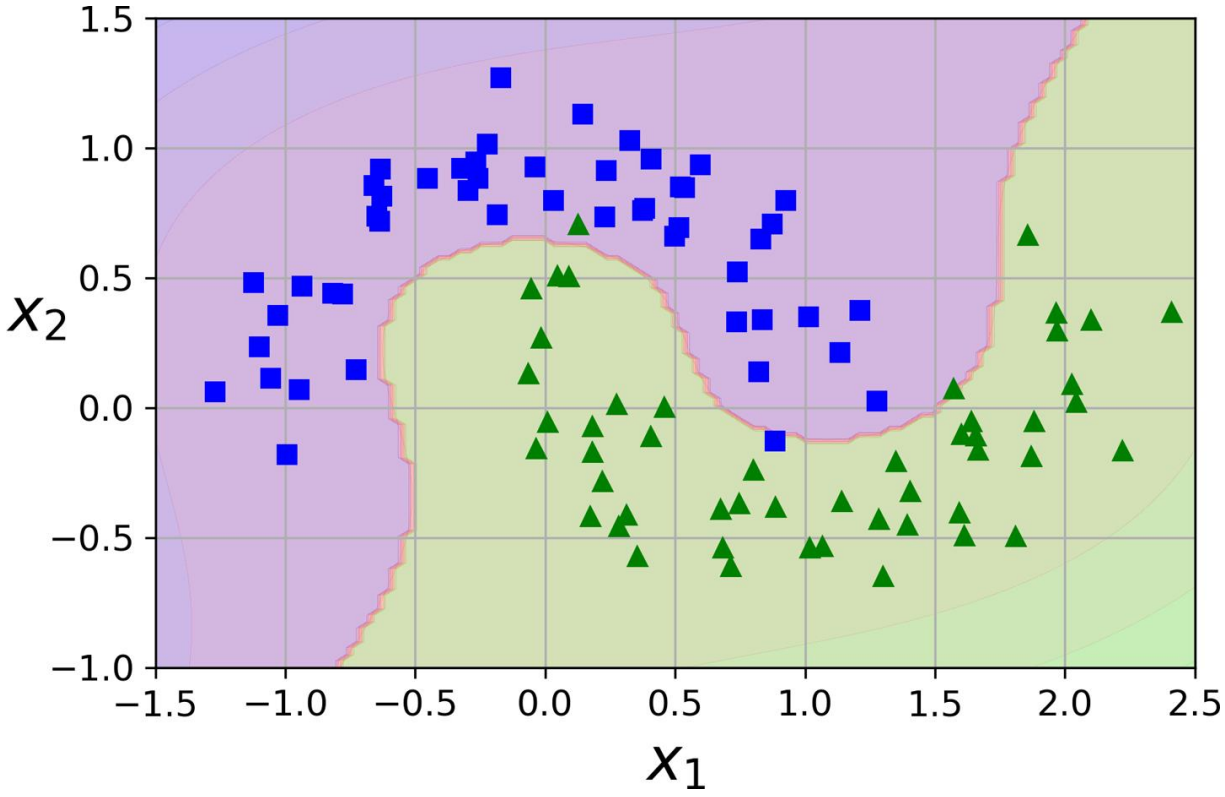
moons dataset

X, y = make_moons(n_samples=100, noise=0.15, random_state=42)

https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_moons.html

Phân lớp SVM phi tuyến

- Ví dụ

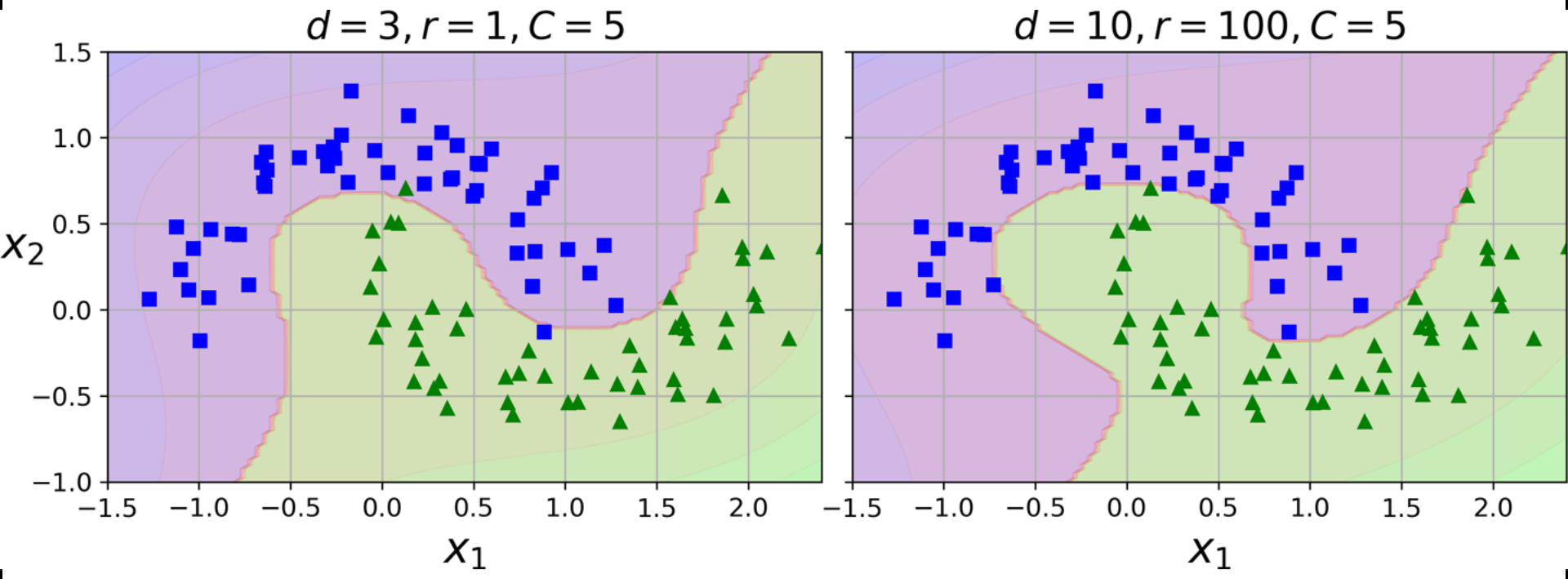


moons dataset

https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_moons.html

Phân lớp SVM phi tuyến

- Sử dụng Polynomial Kernel




```
from sklearn.svm import SVC
poly_kernel_svm_clf = Pipeline([
        ("scaler", StandardScaler()),
        ("svm_clf", SVC(kernel="poly", degree=3, coef0=1, C=5))
    ])
poly_kernel_svm_clf.fit(X, y)
```

Phân lớp SVM phi tuyến

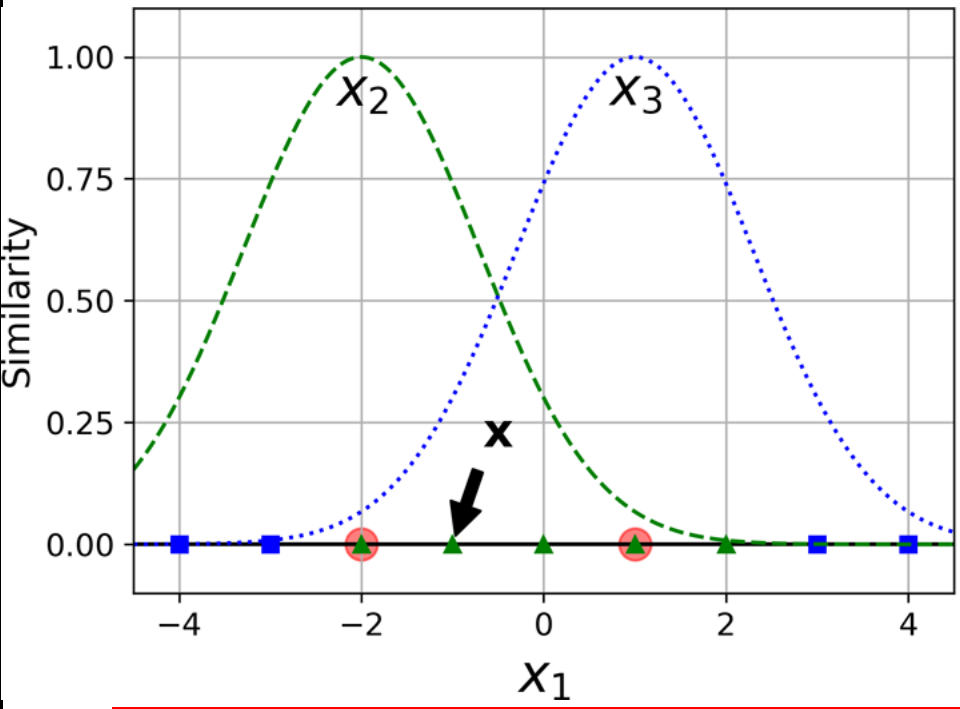
- Các thuộc tính tương tự
  - Một kỹ thuật khác để giải quyết các vấn đề phi tuyến tính là thêm các thuộc được tính toán bằng hàm tương tự (similarity function)
  - Similarity function
    - Hàm này đo mức độ giống nhau giữa mỗi trường hợp với một điểm cụ thể
    - Ví dụ: Gaussian Radial Basis Function (RBF)

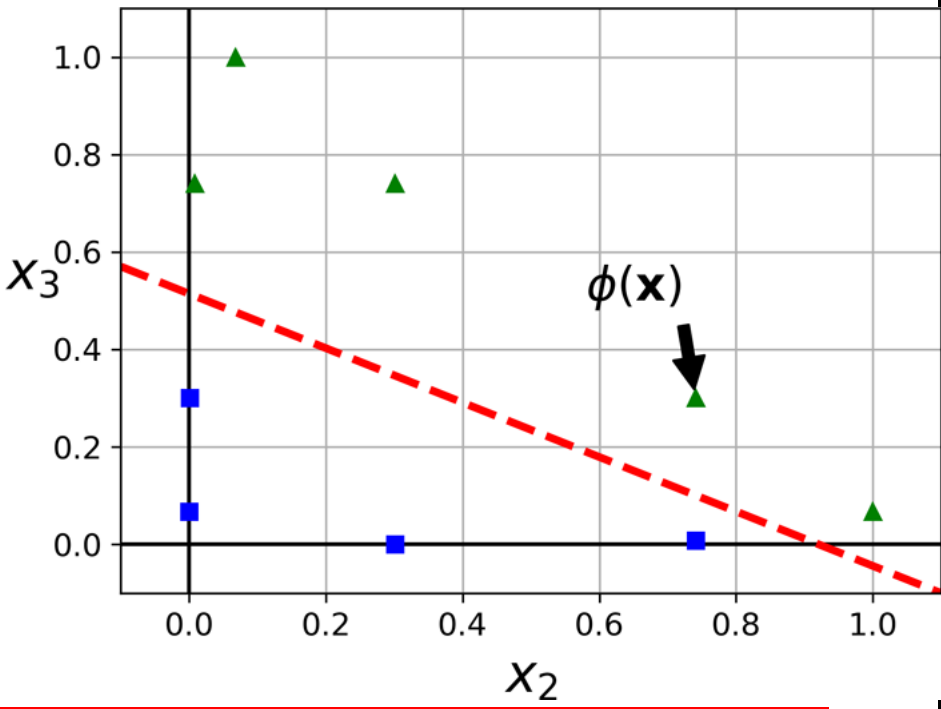
Phân lớp SVM phi tuyến

- Gaussian Radial Basis Function (RBF)

$$\phi_\gamma(\mathbf{x},\mathbf{l}) = \exp\left(-\gamma\|\mathbf{x}-\mathbf{l}\|^2\right)$$

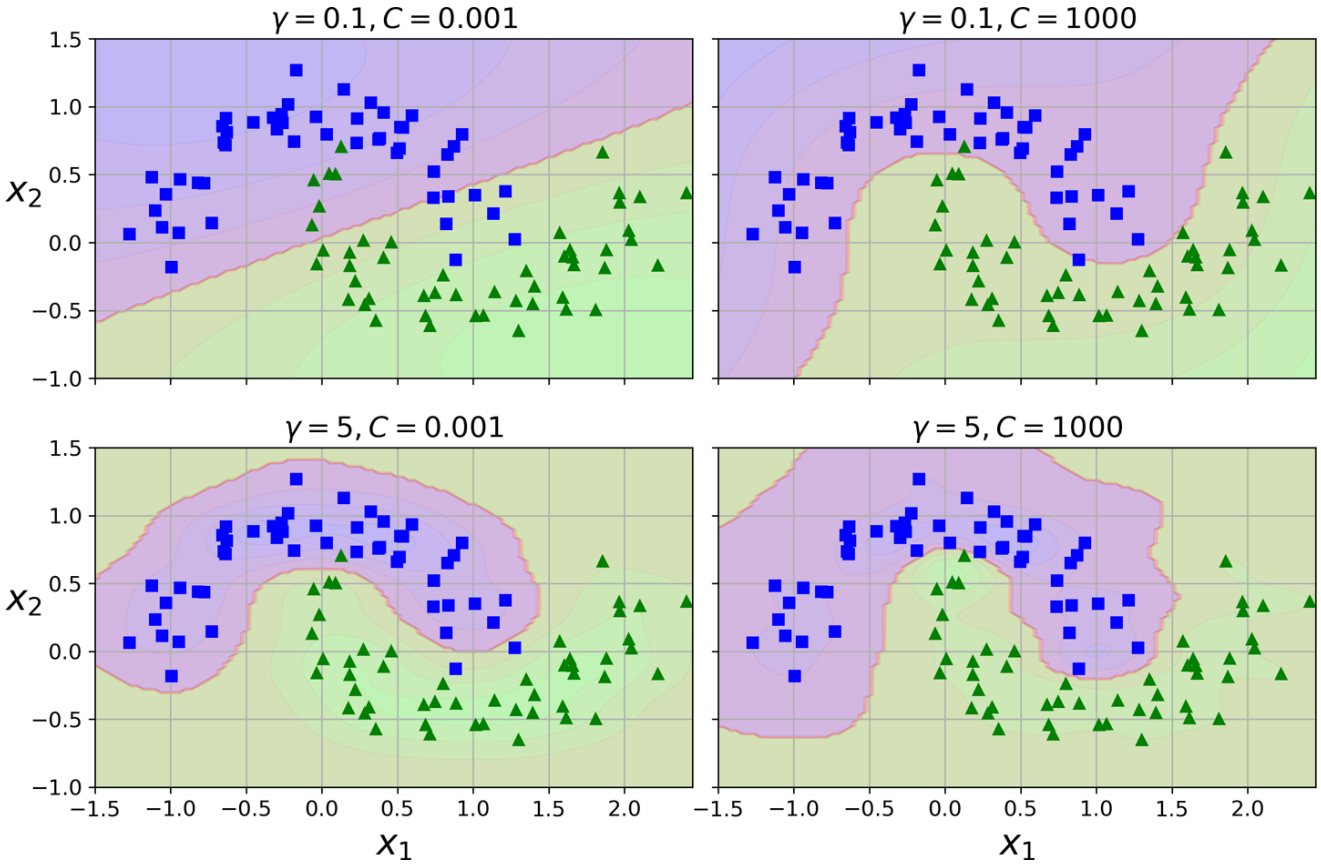
$\gamma = 0.3$





Phân lớp SVM phi tuyến

- Dùng Gaussian RBF kernel




```
rbf_kernel_svm_clf = Pipeline([
        ("scaler", StandardScaler()),
        ("svm_clf", SVC(kernel="rbf", gamma=5, C=0.001))
    ])
rbf_kernel_svm_clf.fit(X, y)
```

Phân lớp SVM phi tuyến

- Trong sklearn:
  - Sử dụng lớp SVC với các kernel khác nhau: linear, poly, rbf, sigmoid

Kernel trong sklearn

Chú ý việc thiết lập các hệ số khi sử dụng trong sklearn

sklearn	Một số kernel phổ biến	Hệ số
'linear'	$K(\mathbf{a},\mathbf{b}) = \mathbf{a}^T\mathbf{b}$	
'poly'	$K(\mathbf{a},\mathbf{b}) = \left(\gamma\mathbf{a}^T\mathbf{b} + r\right)^d$	*d*: degree, γ : gamma, *r:* coef0
'rbf'	$K(\mathbf{a},\mathbf{b}) = \exp\left(-\gamma\left\|\mathbf{a}-\mathbf{b}\right\|^2\right)$	γ : gamma
'sigmoid'	$K(\mathbf{a},\mathbf{b}) = \tanh\left(\gamma\mathbf{a}^T\mathbf{b} + r\right)$	γ : gamma, *r*: coef0

Ưu và nhược điểm của SVM

Ưu điểm

- Mô hình mạnh và hoạt động tốt trên nhiều bộ dữ liệu
- Có thể dùng cho các đường biên quyết định phức tạp
- Nên sử dụng nếu tập dữ liệu có tất cả các đặc trưng được biểu diễn cùng đơn vị và có tỷ lệ giống nhau

Nhược điểm

- Yêu cầu quá trình tiền xử lý và tinh chỉnh các hệ số phải được thực hiện cẩn thận
- Khó kiểm soát do khó hiểu được tại sao việc dự đoán cụ thể được thực hiện
- Khó giải thích mô hình cho một người không phải là chuyên gia

Chú ý

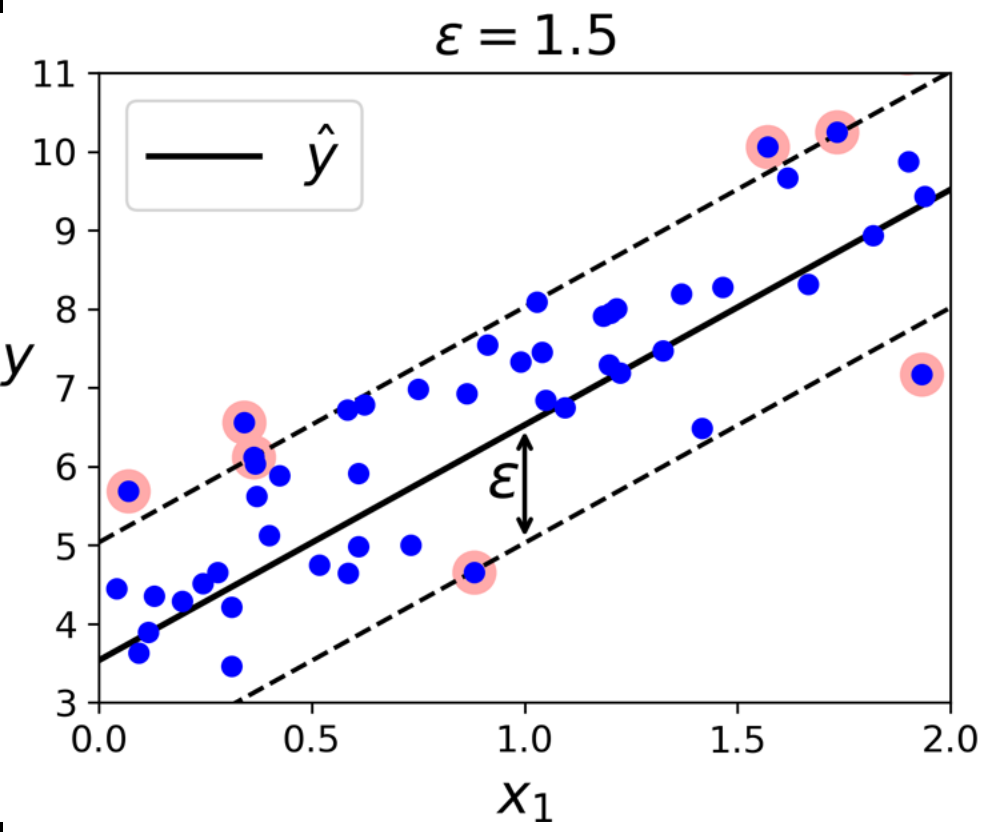
- Hiện nay thường xử dụng các mô hình dựa trên cây quyết định
  - Ví dụ như: random forest, gradient boosting
  - Do không yêu cầu hoặc yêu cầu ít việc tiền xử lý
- Các hệ số quan trọng trong SVM
  - Việc lựa chọn kernel và hệ số đặc biệt của kernel

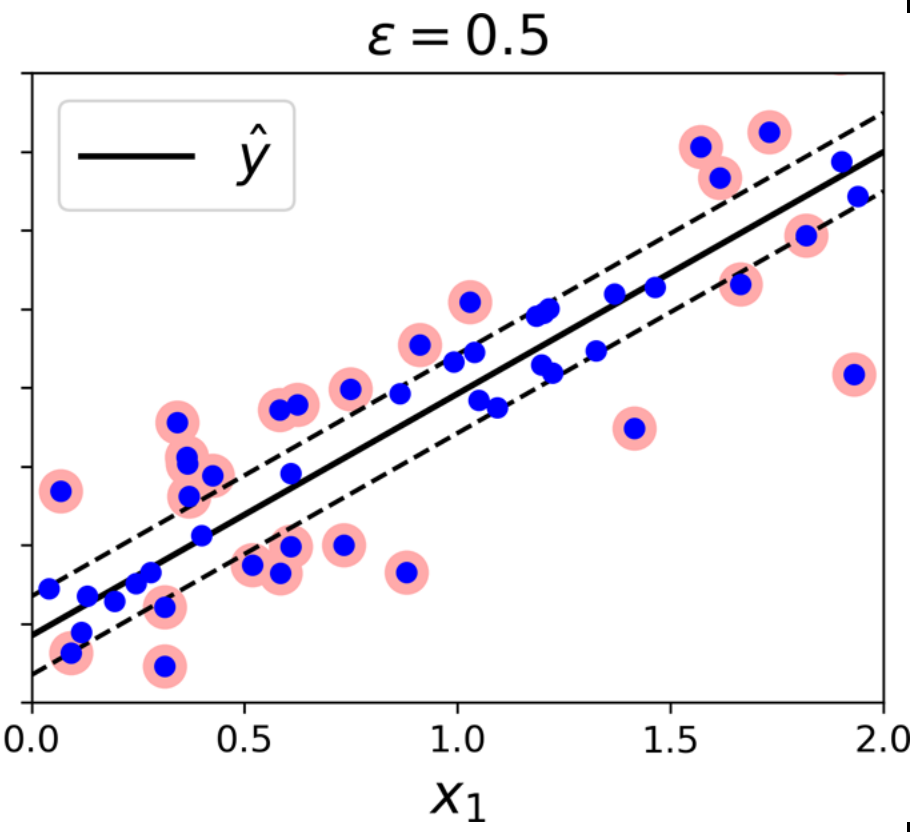
SVM Regression

- Hồi quy SVM
  - Cố gắng đưa càng nhiều điểm dữ liệu càng tốt vào phía trong hai lề, trong khi hạn chế các vi phạm vào lề
  - Độ rộng của lề được điều khiển bởi siêu hệ số (hyperparameter) ε

SVM Regression

- Hồi quy SVM tuyến tính



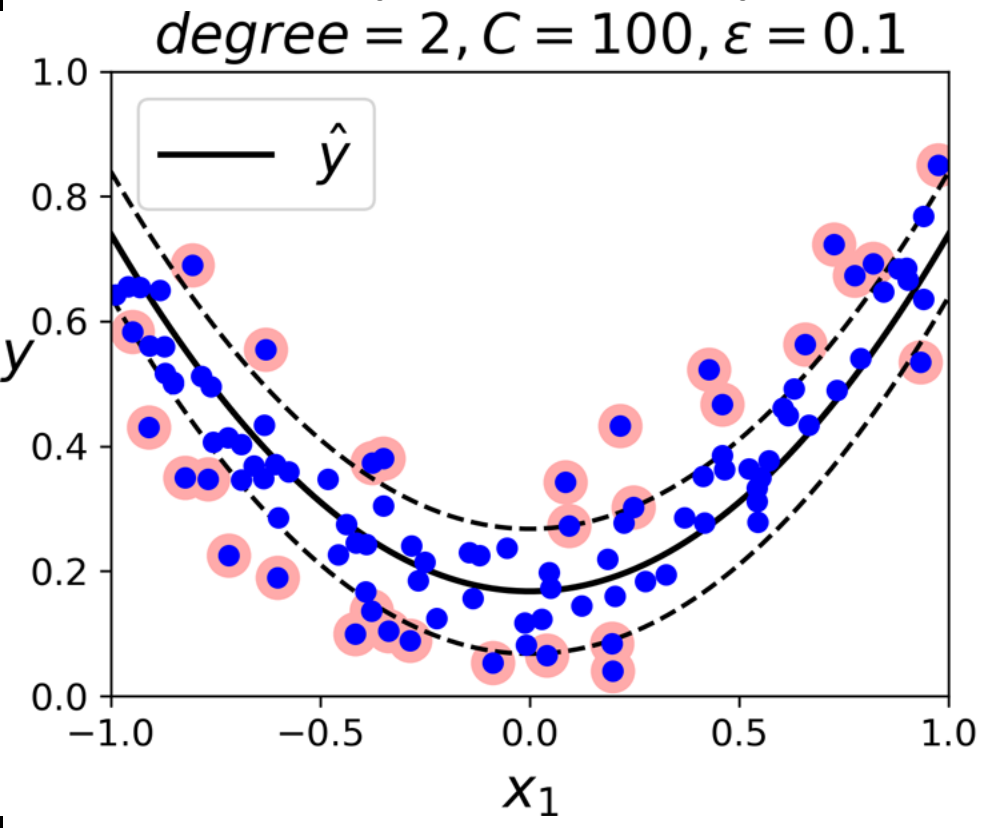


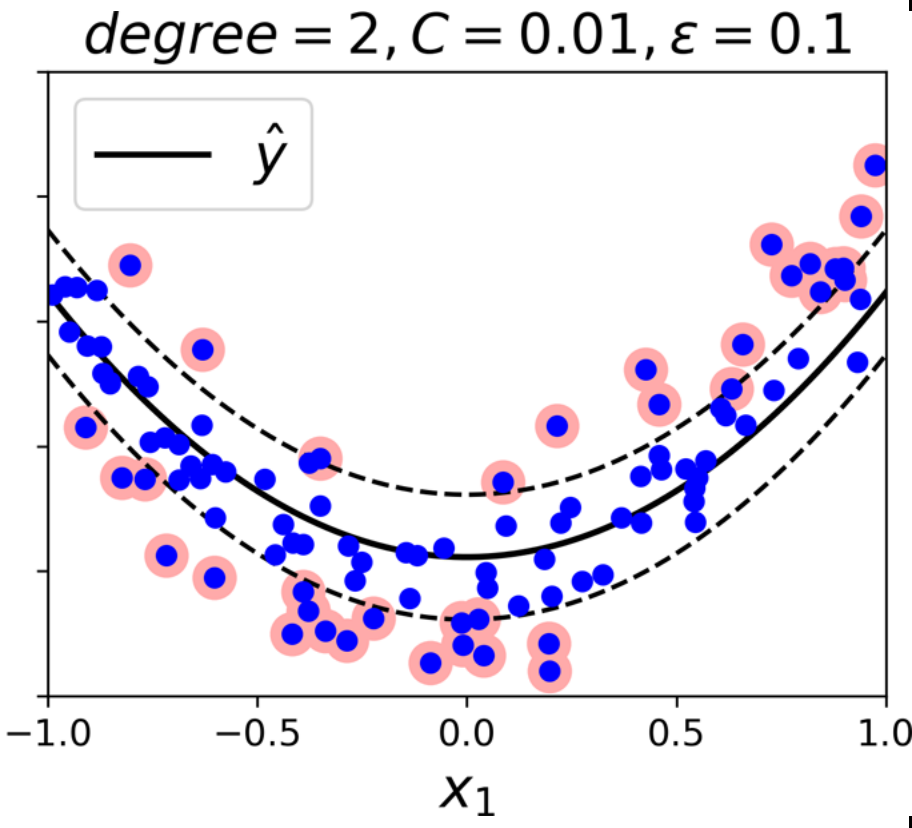

```
from sklearn.svm import LinearSVR

svm_reg = LinearSVR(epsilon=1.5)
svm_reg.fit(X, y)
```

SVM Regression

- Hồi quy SVM phi tuyến






```
from sklearn.svm import SVR

svm_poly_reg = SVR(kernel="poly", degree=2, C=100, epsilon=0.1)
svm_poly_reg.fit(X, y)
```

Tổng kết

- Có thể đưa vấn đề cần giải quyết về bài toán phân lớp
- Nắm được những điểm chính, những điểm cần chú ý liên quan tới bài toán phân lớp
- Hiểu và vận dụng được các thuật toán phân lớp cơ bản: k-NN và SVM

Hoạt động sau buổi học

- Làm bài tập về nhà
- Ôn lại kiến thức đã học về k-NN và SVM

Chuẩn bị cho buổi học tiếp theo

- Sinh viên tìm hiểu về các thuật toán phân lớp dựa trên cây quyết định (decision tree)
- Ôn tập, chuẩn bị cho việc Đánh giá thường xuyên 1 (tự luận) vào buổi học tiếp theo
  - Nội dung ôn tập: các kiến thức đã học cho tới buổi học này
  - Hình thức đánh giá: tự luận, 30 phút

Tài liệu tham khảo

- https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html
- https://en.wikipedia.org/wiki/Support_vector_machine