BÀI 9: SỰ PHÂN LỚP DỮ LIỆU

I. Mục tiêu:

Sau khi thực hành xong, sinh viên nắm được sự phân lớp dữ liệu thông qua:

- Support Vector Machines

II. Tóm tắt lý thuyết:

1. Ý tưởng:

a. Hard margin

Cho n điểm dữ liệu trong tập huấn luyện $\mathcal{D}$ được ký hiệu bởi $(\overline{X_1}, y_1) \dots (\overline{X_n}, y_n)$, với $\overline{X_i}$ là vector dòng d chiều tương ứng với điểm dữ liệu thứ i, và $y_i \in \{-1, +1\}$ là nhãn lớp nhị phân của điểm dữ liệu thứ i. Tìm một siêu phẳng sao cho nó có thể tách rời các điểm dữ liệu được dựa vào các lớp của chúng và cũng cực đại hóa khoảng cách nhỏ nhất của điểm dữ liệu tới siêu phẳng. Khi đó, siêu phẳng tách rời có dạng như sau:

$$\overline{W} \cdot \overline{X} + b = 0$$

với $\overline{W} = (w_1 \dots w_d)$ là vector dòng d chiều tương ứng với phương trực giao với siêu phẳng, và b là một vô hướng, cũng được biết như là nhiễu (bias). Vector $\overline{W}$ điều chỉnh sự định hướng của siêu phẳng và nhiễu b điều chỉnh khoảng cách của siêu phẳng từ gốc. Các hệ số $(d+1)$ tương ứng với $\overline{W}$ và b cần để được học từ dữ liệu huấn luyện để cực đại hóa lề của sự tách rời giữa 2 lớp. Bởi vì nó được giả sử rằng các lớp tách rời một cách tuyến tính, siêu phẳng như vậy cũng có thể được giả sử rằng tồn tại.

Khoảng cách từ một điểm tới mặt phẳng được tính bằng công thức:

$$Distance = \frac{|\overline{W} \cdot \overline{X} + b|}{\|\overline{W}\|}$$

Khi đó, chúng ta muốn siêu phẳng như vậy có thể tách 1 cách chính xác thành 2 lớp, mà nó thỏa ràng buộc theo sau:

$$\overline{W} \cdot \overline{X_i} + b > 0 \quad \text{nếu } y_i = 1$$

$$\overline{W} \cdot \overline{X_i} + b < 0 \quad \text{nếu } y_i = -1$$

Tương đương với

$$y_i(\overline{W} \cdot \overline{X_i} + b) > 0, \quad i = 1, \ldots, n.$$

Mục tiêu của chúng ta là tối đa hóa khoảng cách nhỏ nhất cho tất cả các quan sát tới siêu phẳng đó, nên chúng ta có bài toán toán tối ưu đầu tiên:

$$\max_{\overline{W},b} \min \left\{ \frac{|\overline{W} \cdot \overline{X_i} + b|}{\|\overline{W}\|} \right\}$$

sao cho

$$y_i(\overline{W} \cdot \overline{X_i} + b) > 0, \quad i = 1, \ldots, n.$$

Khi đó, chúng ta định nghĩa lề $M = \min \left\{ \frac{|\overline{W} \cdot \overline{X_i} + b|}{\|\overline{W}\|} \right\}$, và chúng ta scale $\overline{W}$ sao cho $\|\overline{W}\| = 1$.

$$\max_{\overline{W}, \|\overline{W}\|=1, b} M$$

sao cho

$$y_i(\overline{W} \cdot \overline{X_i} + b) \geqslant M, \quad i = 1, \ldots, n.$$

Ta định nghĩa $v = \dfrac{\overline{W}}{M}$ và chuẩn của $\|v\|$ là $\dfrac{1}{M}$. Thế v ngược lại bài toán tối ưu của chúng ta:

$$\max_{v,b} \frac{1}{\|v\|}$$

sao cho

$$y_i(\frac{\overline{W}}{M} \cdot \overline{X_i} + \frac{b}{M}) \geqslant \frac{M}{M} = 1, \quad i = 1, \ldots, n.$$

Ta thay đổi tên v thành $\overline{W}$, và để tối đa hóa $\dfrac{1}{\|v\|}$ tương đương với cực tiểu hóa $\dfrac{1}{2}\overline{W} \cdot \overline{W} = \dfrac{\|\overline{W}\|^2}{2}$. Nên chúng ta có được bài toán tối ưu cuối cùng như sau:

$$\min_{\overline{W},b} \frac{\|\overline{W}\|^2}{2}$$

sao cho

$$y_i(\overline{W} \cdot \overline{X_i} + b) \geqslant 1, \quad i = 1, \ldots, n.$$

Chúng ta muốn sử dụng Phương pháp của các hệ số nhân Lagrange để giải bài toán tối ưu. Chúng ta có thể viết ràng buộc của chúng như $g_i(\overline{W}) = y_i(\overline{W} \cdot \overline{X_i} + b) - 1 \geqslant 0$. Chúng ta hãy định nghĩa $L(\overline{W}, b, \lambda \geqslant 0) = \frac{\|\overline{W}\|^2}{2} - \sum_{i=1}^{n} \lambda_i \left[y_i \left(\overline{W} \cdot \overline{X_i} + b \right) - 1 \right]$. Chúng ta thấy rằng sự tối đa hóa của hàm L tương ứng với λ bằng $\frac{\|\overline{W}\|^2}{2}$. Nên chúng ta thay đổi bài toán ban đầu thành bài toán mới:

$$\min_{\overline{W},b} \max_{\lambda \geqslant 0} L(\overline{W}, b, \lambda) = \frac{\|\overline{W}\|^2}{2} - \sum_{i=1}^{n} \lambda_i \left[y_i \left(\overline{W} \cdot \overline{X_i} + b \right) - 1 \right]$$

với λ là hệ số nhân Lagrange.

Kiểm tra $\overline{W}, b, \lambda$ thỏa điều kiện Karush-Kuhn-Tucker.

$$\lambda_i \geqslant 0$$

$$\lambda_i [y_i(\overline{W} \cdot \overline{X_i} + b) - 1] = 0$$

Do đó, chúng ta có thể giải bài toán ban đầu bằng việc giải bài toán đối ngẫu của nó:

$$\max_{\lambda \geqslant 0} \min_{\overline{W},b} L(\overline{W}, b, \lambda) = \frac{\|\overline{W}\|^2}{2} - \sum_{i=1}^{n} \lambda_i \left[y_i \left(\overline{W} \cdot \overline{X_i} + b \right) - 1 \right]$$

Để giải bài toán đối ngẫu, đầu tiên chúng ta cần đặt đạo hàm riêng của $L(\overline{W}, b, \lambda)$ theo $\overline{W}$ và b bằng 0

$$\frac{\partial}{\partial_{\overline{W}}} L(\overline{W}, b, \lambda) = \overline{W} - \sum_{i=1}^{n} \lambda_i y_i \overline{X_i} = 0$$

$$\frac{\partial}{\partial_b} L(\overline{W}, b, \lambda) = \sum_{i=1}^{n} \lambda_i y_i = 0$$

Khi đó, chúng ta thế chúng trở ngược vào hàm $L(\overline{W}, b, \lambda)$

$$\begin{aligned} L(\overline{W}, b, \lambda \geqslant 0) &= \frac{\|\overline{W}\|^2}{2} - \sum_{i=1}^{n} \lambda_i \left[y_i \left(\overline{W} \cdot \overline{X_i} + b \right) - 1 \right] \\ &= \frac{\|\overline{W}\|^2}{2} - \sum_{i=1}^{n} \lambda_i y_i \overline{W} \cdot \overline{X_i} - \sum_{i=1}^{n} \lambda_i y_i b + \sum_{i=1}^{n} \lambda_i \\ &= \frac{\|\overline{W}\|^2}{2} - \sum_{i=1}^{n} \lambda_i y_i \overline{W} \cdot \overline{X_i} - b \sum_{i=1}^{n} \lambda_i y_i + \sum_{i=1}^{n} \lambda_i \\ &= \sum_{i=1}^{n} \lambda_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} (\lambda_i \lambda_j y_i y_j \overline{X_i} \cdot \overline{X_j}) \end{aligned}$$

Đơn giản bài toán đối ngẫu như sau:

$$\max_{\lambda \geqslant 0} \left(\sum_{i=1}^{n} \lambda_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} (\lambda_i \lambda_j y_i y_j \overline{X_i} \cdot \overline{X_j}) \right)$$

sao cho

$$\sum_{i=1}^{n} \lambda_i y_i = 0$$

Đây là một bài toán tối ưu lồi nên chúng ta có thể sử dụng thuật toán SMO để giải giá trị của hệ số nhân Lagrange λ. Sử dụng công thức ở trên, chúng ta có thể tính tham số $\overline{W}$

$$\overline{W} = \sum_{i=1}^{n} \lambda_i y_i \overline{X_i}$$

Hơn nữa, giá trị của b có thể được tính như sau:

$$y_r(\sum_{i=1}^{n}(\lambda_i y_i \overline{X_i} \cdot \overline{X_r}) + b) = +1, \quad \forall \lambda_r > 0$$

Giải phương trình này tìm giá trị b. Để giảm sai số, giá trị của b có thể được lấy trung bình trên tất cả các vector hỗ trợ với $\lambda_r > 0$.

Với một điểm $\overline{Z}$, các nhãn lớp của nó $F(\overline{Z})$ được xác định bởi

$$F(\overline{Z}) = sign\{\overline{W} \cdot \overline{Z} + b\} = sign\{\sum_{i=1}^{n}(\lambda_i y_i \overline{X_i} \cdot \overline{Z}) + b\}$$

b. Soft margin

Trong hầu hết dữ liệu thực, siêu phẳng không thể tách toàn bộ dữ liệu lớp nhị phân. Lề trong tình huống như vậy chúng ta gọi là lề mềm hoặc phân lớp vector hỗ trợ

$$\min_{\overline{W},b,\xi_i} \frac{\|\overline{W}\|^2}{2} + C\sum_{i=1}^{n} \xi_i$$

sao cho

$$y_i(\overline{W} \cdot \overline{X_i} + b) \geqslant 1 - \xi_i, \quad \xi_i \geqslant 0, i = 1, \ldots, n.$$

Với ξ_i là các biến yếu (slack) mà chúng cho phép phân lớp sai. Số hạng bất lợi $\sum_{i=1}^{l} \xi_i$ là một độ đo của tổng số phân lớp sai trong mô hình được xây dựng bằng việc huấn luyện tập dữ liệu, C (là tham số làm cộng hưởng) là chi phí phân lớp sai mà nó điều khiển sự cân bằng của việc tối đa hóa lề và cực tiểu hóa số hạng bất lợi.

Do đó, chúng ta có thể giải bài toán ban đầu bằng việc giải bài toán đối ngẫu của nó với α_i và β_i là hệ số nhân Lagrange:

$$\min_{\overline{W},b,\xi_i} \max_{\lambda \geqslant 0, \beta \geqslant 0} L(\overline{W}, b, \lambda, \beta, \xi) = \frac{\|\overline{W}\|^2}{2} + C\sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} \lambda_i \left[y_i \left(\overline{W} \cdot \overline{X_i} + b \right) - 1 + \xi_i \right] - \sum_{i=1}^{n} \beta_i \xi_i$$

Mặc dù tính chất không âm của các ràng buộc nguyên thủy và hệ số nhân Lagrange, phần $-\sum_{i=1}^{n}\lambda_i\left[y_i\left(\overline{W}\cdot\overline{X_i}+b\right)-1+\xi_i\right]-\sum_{i=1}^{n}\beta_i\xi_i$ nên là số âm do đó chúng ta cần cực tiểu hóa phần đó thành 0 để có được $\max_{\lambda\geqslant 0,\beta\geqslant 0} L(\overline{W},b,\lambda,\beta,\xi)=\frac{\|\overline{W}\|^2}{2}+C\sum_{i=1}^{n}\xi_i-\sum_{i=1}^{n}\lambda_i\left[y_i\left(\overline{W}\cdot\overline{X_i}+b\right)-1+\xi_i\right]-\sum_{i=1}^{n}\beta_i\xi_i$ tương ứng với $\overline{W}$, b và ξ. Tuy nhiên, các ràng buộc $\lambda_i\geqslant 0$ và $\beta_i\geqslant 0$ làm nó khó để tìm sự cực đại hóa. Do đó, chúng ta chuyển bài toán nguyên thủy thành bài toán đối ngẫu theo sau thông qua điều kiện KKT:

$$\max_{\lambda\geqslant 0,\beta\geqslant 0}\min_{\overline{W},b,\xi_i} L(\overline{W},b,\lambda,\beta,\xi)=\frac{\|\overline{W}\|^2}{2}+C\sum_{i=1}^{n}\xi_i-\sum_{i=1}^{n}\lambda_i\left[y_i\left(\overline{W}\cdot\overline{X_i}+b\right)-1+\xi_i\right]-\sum_{i=1}^{n}\beta_i\xi_i$$

Bởi vì bài toán con cho việc cực tiểu hóa phương trình tương ứng với nên chúng ta không có các ràng buộc trong chúng, do đó chúng có thể đặt gradient bằng 0 như sau để tìm $\min_{\overline{W},b,\xi_i} L(\overline{W},b,\lambda,\beta,\xi)$:

$$\begin{aligned}
\frac{\partial}{\partial_{\overline{W}}}L(\overline{W},b,\lambda,\beta,\xi)&=\overline{W}-\sum_{i=1}^{n}\lambda_i y_i\overline{X_i}=0\\
&\Rightarrow\overline{W}=\sum_{i=1}^{n}\lambda_i y_i\overline{X_i}\\
\frac{\partial}{\partial_b}L(\overline{W},b,\lambda,\beta,\xi)&=-\sum_{i=1}^{n}\lambda_i y_i=0\\
&\Rightarrow\sum_{i=1}^{n}\lambda_i y_i=0\\
\frac{\partial}{\partial_{\xi_i}}L(\overline{W},b,\lambda,\beta,\xi)&=C-\lambda_i-\beta_i=0\\
&\Rightarrow C=\lambda_i+\beta_i
\end{aligned}$$

Khi đó, ta thu được dạng Lagrange đối ngẫu của bài toán tối ưu này như sau:

$$\max_{\lambda\geqslant 0}\left(\sum_{i=1}^{n}\lambda_i-\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}(\lambda_i\lambda_j y_i y_j\overline{X_i}\cdot\overline{X_j})\right)$$

sao cho

$$\sum_{i=1}^{n}\lambda_i y_i=0,\quad 0\leqslant\lambda_i\leqslant C,\ \text{với } i=1,\ldots,n.$$

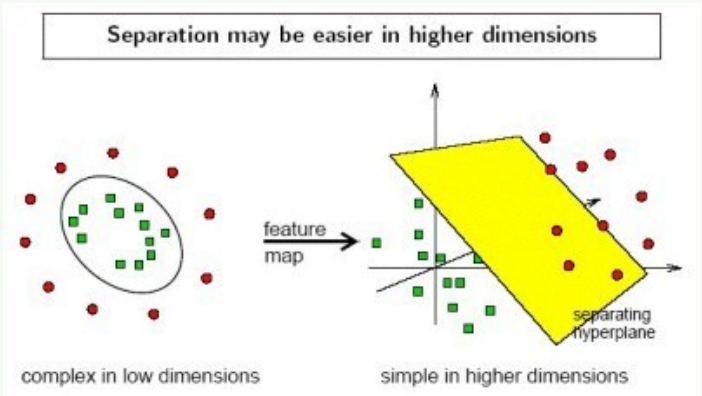
Khi đó,

- Nếu $\lambda_i=0$ và $\xi_i=0$ thì điểm $\overline{X_i}$ đã được phân lớp nằm ở bên phải.
- Nếu $0<\lambda_i<C$ thì chúng ta có thể kết luận rằng $\xi_i=0$ và $y_i=\overline{W}\cdot\overline{X_i}+b$, nghĩa là điểm $\overline{X_i}$ là một vector hỗ trợ.

- Nếu $\lambda_i = 0, \xi_i = 0$ và $\overline{X_i}$ là một vector hỗ trợ thì điểm $\overline{X_i}$ đã được phân lớp một cách chính xác khi $0 \leqslant \xi_i < 1$ và $\overline{X_i}$ phân lớp sai khi $\xi_i > 1$.

c. Phương pháp kernel

Ý tưởng: nếu các điểm dữ liệu là không tách rời trong không gian đặc trưng hiện tại thì chúng ta có thể sử dụng một hàm ánh xạ đặc trưng Φ và cố gắng để tách rời các điểm sau khi được ánh xạ.



Chúng ta định nghĩa hàm Kernel tương ứng như sau:

$$K(\overline{X_i}, \overline{X_j}) = \Phi(\overline{X_i}) \cdot \Phi(\overline{X_j})$$

Thay $\overline{X_i} \cdot \overline{X_j}$ bằng $K(\overline{X_i}, \overline{X_j})$, ta được

$$L(\overline{W}, b, \lambda) = \sum_{i=1}^{n} \lambda_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} (\lambda_i \lambda_j y_i y_j K(\overline{X_i}, \overline{X_j})$$

$$F(\overline{Z}) = sign\{\overline{W} \cdot \overline{Z} + b\} = sign\{\sum_{i=1}^{n} (\lambda_i y_i K(\overline{X_i}, \overline{Z}) + b\}$$

Một vài hàm kernel thông dụng

Function	Form
Gaussian radial basis kernel	$K(\overline{X_i}, \overline{X_j}) = e^{-\|\overline{X_i} - \overline{X_j}\|^2 / 2\sigma^2}$
Polynomial kernel	$K(\overline{X_i}, \overline{X_j}) = (\overline{X_i} \cdot \overline{X_j} + c)^h$
Sigmoid kernel	$K(\overline{X_i}, \overline{X_j}) = tanh(\kappa \overline{X_i} \cdot \overline{X_j} - \delta)$

2. Ví dụ:

Xét tập dữ liệu 2 chiều chứa 8 điểm huấn luyện như sau:

x_1	x_2	y	Hệ số nhân Lagrange
0.3858	0.4687	1	65.5261
0.4871	0.611	-1	65.5261
0.9218	0.4103	-1	0
0.7382	0.8936	-1	0
0.1763	0.05799	1	0
0.4057	0.3529	1	0
0.9355	0.8132	-1	0
0.2146	0.0099	1	0

Giải bài toán tối ưu $L(\overline{W}, b, \lambda \geqslant 0) = \sum_{i=1}^{n} \lambda_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} (\lambda_i \lambda_j y_i y_j \overline{X_i} \cdot \overline{X_j})$ để tìm các hệ số nhân Lagrange. Hai điểm dữ liệu đầu tiên khác 0 $\Rightarrow$ hai điểm dữ liệu này là vector hỗ trợ.

Sử dụng $\overline{W} = \sum_{i=1}^{n} \lambda_i y_i \overline{X_i}$ để tính $\overline{W} = (w_1, w_2)$ như sau:

$$w_1 = \sum_{i=1}^{n} \lambda_i y_i \overline{X_{i1}} = 65.5261 \times 1 \times 0.3858 + 65.5261 \times (-1) \times 0.4871 = -6.64$$

$$w_2 = \sum_{i=1}^{n} \lambda_i y_i \overline{X_{i2}} = 65.5261 \times 1 \times 0.4687 + 65.5261 \times (-1) \times 0.611 = -9.32$$

$$\Rightarrow \overline{W} = (-6.64, -9.32)$$

Số hạng b được tính bằng công thức $\lambda_i[y_i(\overline{W} \cdot \overline{X_i} + b) - 1] = 0$ cho mỗi vector hỗ trợ

$$b^{(1)} = 1 - \overline{W} \cdot \overline{X_1} = 1 - (-6.64) \times (0.3858) - (-9.32) \times (0.4687) = 7.9300$$

$$b^{(2)} = -1 - \overline{W} \cdot \overline{X_2} = -1 - (-6.64) \times (0.4871) - (-9.32) \times (0.611) = 7.9289$$

Tính trung bình của các giá trị này $\Rightarrow b = 7.93$

Vậy siêu phẳng cần tìm là $-6.64x_1 - 9.32x_2 + 7.93 = 0$

III. Nội dung thực hành:

1. SVM với tập dữ liệu tách rời tuyến tính

- Cài đặt ipywidgets

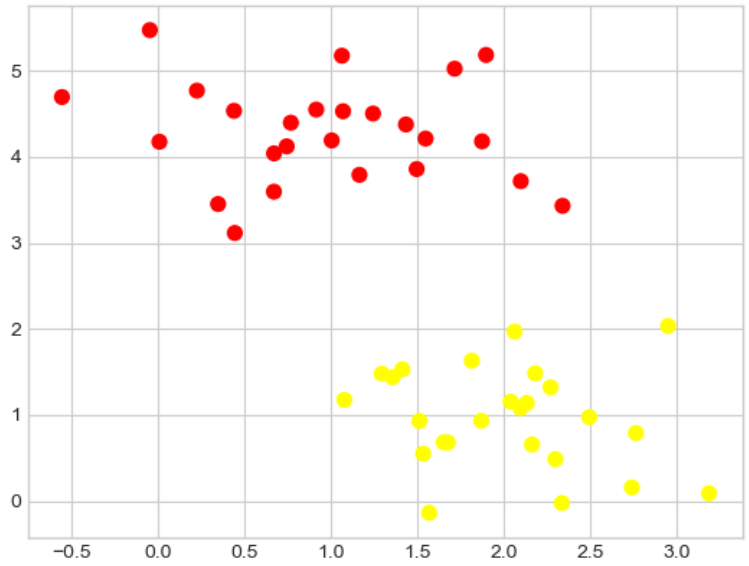
```
Microsoft Windows [Version 10.0.19045.2965]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Huynh>pip install ipywidgets
Collecting ipywidgets
  Downloading ipywidgets-8.0.6-py3-none-any.whl (138 kB)
     ---------------------------------------- 138.3/138.3 kB 2.1 MB/s eta 0:00:00
Collecting ipykernel>=4.5.1
  Downloading ipykernel-6.16.2-py3-none-any.whl (138 kB)
     ---------------------------------------- 138.5/138.5 kB 4.1 MB/s eta 0:00:00
Collecting traitlets>=4.3.1
  Downloading traitlets-5.9.0-py3-none-any.whl (117 kB)
     ---------------------------------------- 117.4/117.4 kB 6.7 MB/s eta 0:00:00
Collecting jupyterlab-widgets~=3.0.7
  Downloading jupyterlab_widgets-3.0.7-py3-none-any.whl (198 kB)
     ---------------------------------------- 198.2/198.2 kB 6.1 MB/s eta 0:00:00
Collecting ipython>=6.1.0
  Downloading ipython-7.34.0-py3-none-any.whl (793 kB)
     ---------------------------------------- 793.8/793.8 kB 3.6 MB/s eta 0:00:00
Collecting widgetsnbextension~=4.0.7
  Downloading widgetsnbextension-4.0.7-py3-none-any.whl (2.1 MB)
     ---------------------------------------- 2.1/2.1 MB 4.5 MB/s eta 0:00:00
Collecting debugpy>=1.0
  Downloading debugpy-1.6.7-cp37-cp37m-win_amd64.whl (4.8 MB)
     ---------------------------------------- 4.8/4.8 MB 5.3 MB/s eta 0:00:00
```

- Import thư viện

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.svm import SVC # "Support vector classifier"
plt.style.use('seaborn-whitegrid')
from scipy import stats
from ipywidgets import interact, fixed
from sklearn.datasets import make_blobs
```

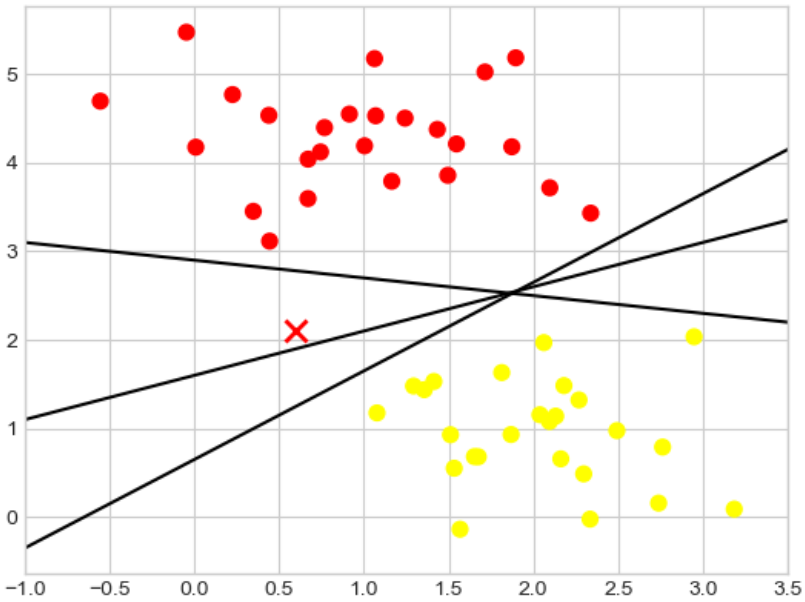
- Khởi tạo tập dữ liệu có 50 điểm và 2 cụm

```
X, y = make_blobs(n_samples=50, centers=2,
                  random_state=0, cluster_std=0.60)
plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='autumn');
plt.show()
```



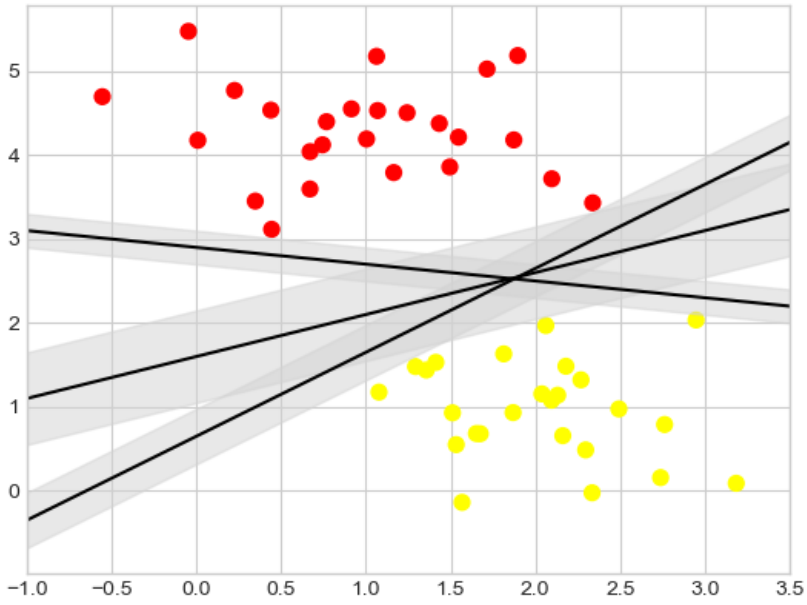
- Vẽ đường thẳng phân tách 2 cụm

```
xfit = np.linspace(-1, 3.5)
plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='autumn')
plt.plot([0.6], [2.1], 'x', color='red', markeredgewidth=2, markersize=10)

for m, b in [(1, 0.65), (0.5, 1.6), (-0.2, 2.9)]:
    plt.plot(xfit, m * xfit + b, '-k')

plt.xlim(-1, 3.5);
plt.show()
```



- Vẽ siêu phẳng lề lớn nhất

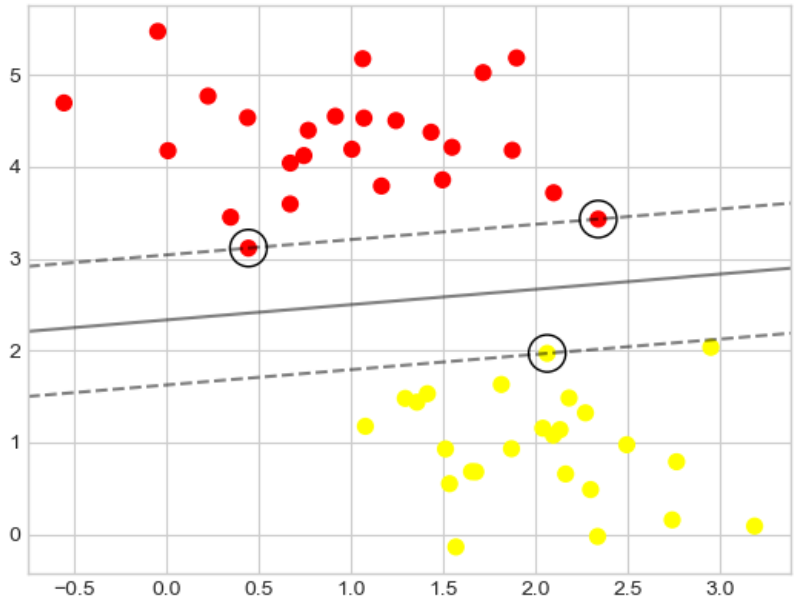
```
xfit = np.linspace(-1, 3.5)
plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='autumn')

for m, b, d in [(1, 0.65, 0.33), (0.5, 1.6, 0.55), (-0.2, 2.9, 0.2)]:
    yfit = m * xfit + b
    plt.plot(xfit, yfit, '-k')
    plt.fill_between(xfit, yfit - d, yfit + d, edgecolor='none',
                     color='lightgray', alpha=0.5)

plt.xlim(-1, 3.5);
plt.show()
```



- Sử dụng support vector classifier (SVC) của Scikit-Learn để huấn luyện mô hình SVM và vẽ các biên quyết định của SVM

```
model = SVC(kernel='linear', C=1E10)
model.fit(X, y)
def plot_svc_decision_function(model, ax=None, plot_support=True):
    """Plot the decision function for a 2D SVC"""
    if ax is None:
        ax = plt.gca()
    xlim = ax.get_xlim()
    ylim = ax.get_ylim()

    # create grid to evaluate model
    x = np.linspace(xlim[0], xlim[1], 30)
    y = np.linspace(ylim[0], ylim[1], 30)
    Y, X = np.meshgrid(y, x)
    xy = np.vstack([X.ravel(), Y.ravel()]).T
    P = model.decision_function(xy).reshape(X.shape)

    # plot decision boundary and margins
    ax.contour(X, Y, P, colors='k',
               levels=[-1, 0, 1], alpha=0.5,
               linestyles=['--', '-', '--'])

    # plot support vectors
    if plot_support:
        ax.scatter(model.support_vectors_[:, 0],
                   model.support_vectors_[:, 1],
                   s=300, linewidth=1, edgecolors='black',
                   facecolors='none');
    ax.set_xlim(xlim)
    ax.set_ylim(ylim)


plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='autumn')
plot_svc_decision_function(model);
plt.show()
model.support_vectors_
```

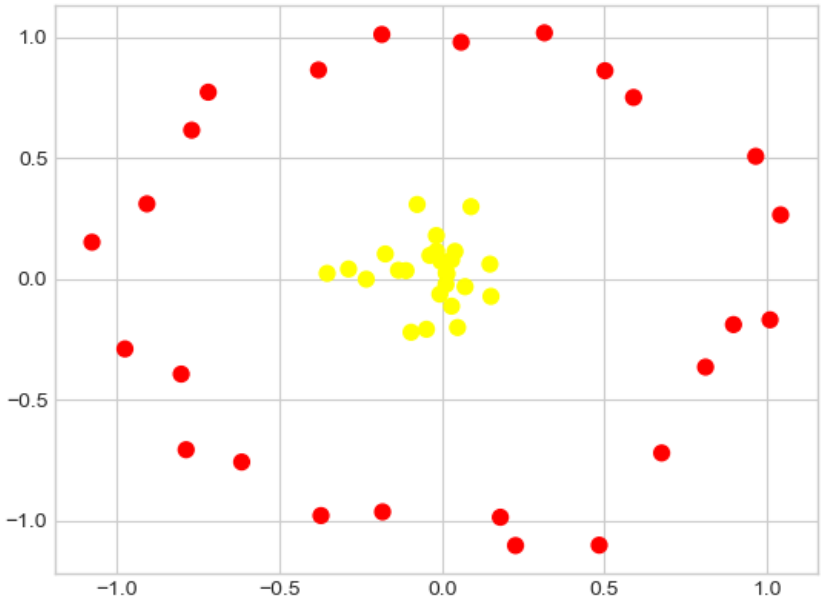
```
array([[0.44359863, 3.11530945],
       [2.33812285, 3.43116792],
       [2.06156753, 1.96918596]])
```



2. SVM với tập dữ liệu không tuyến tính

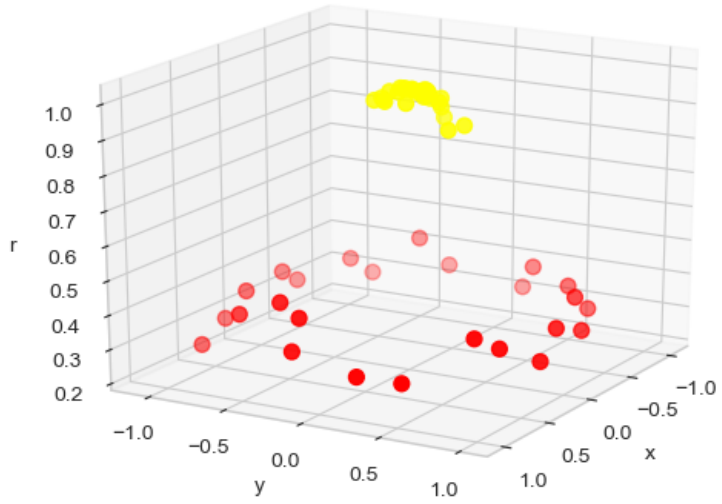
- Khởi tạo tập dữ liệu có 50 điểm.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.svm import SVC # "Support vector classifier"
plt.style.use('seaborn-whitegrid')
from scipy import stats
from ipywidgets import interact, fixed
from sklearn.datasets import make_blobs, make_circles
from mpl_toolkits import mplot3d

X, y = make_circles(50, factor=.1, noise=.1)
plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='autumn');
plt.show()
```



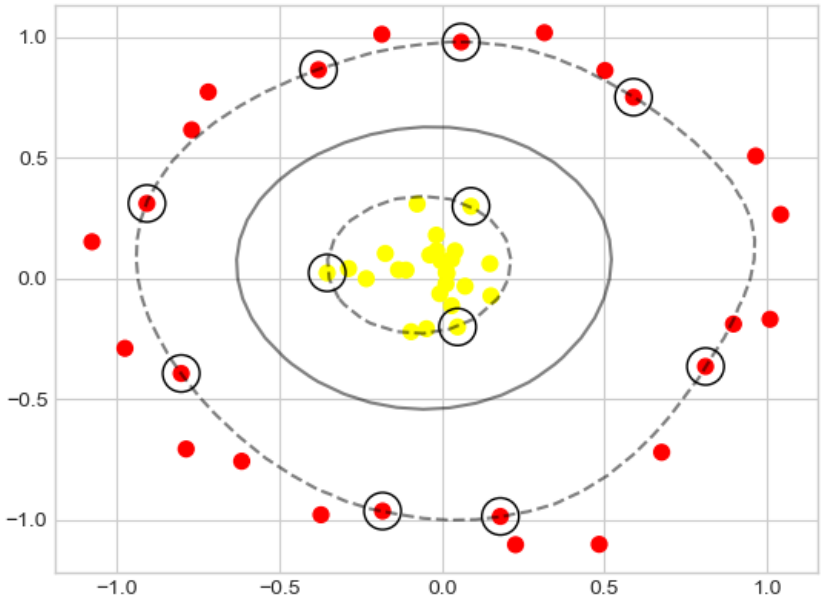
- Sử dụng hàm radial basis function và vẽ dữ liệu trong không gian 3 chiều

```
r = np.exp(-(X ** 2).sum(1))
ax = plt.subplot(projection='3d')
ax.scatter3D(X[:, 0], X[:, 1], r, c=y, s=50, cmap='autumn')
ax.view_init(elev=20, azim=30)
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('r');
plt.show()
```



- Sử dụng support vector classifier (SVC) của Scikit-Learn để huấn luyện mô hình SVM và vẽ các biên quyết định của SVM

```
model = SVC(kernel='rbf', C=1E6)
model.fit(X, y)
def plot_svc_decision_function(model, ax=None, plot_support=True):
    """Plot the decision function for a 2D SVC"""
    if ax is None:
        ax = plt.gca()
    xlim = ax.get_xlim()
    ylim = ax.get_ylim()

    # create grid to evaluate model
    x = np.linspace(xlim[0], xlim[1], 30)
    y = np.linspace(ylim[0], ylim[1], 30)
    Y, X = np.meshgrid(y, x)
    xy = np.vstack([X.ravel(), Y.ravel()]).T
    P = model.decision_function(xy).reshape(X.shape)

    # plot decision boundary and margins
    ax.contour(X, Y, P, colors='k',
               levels=[-1, 0, 1], alpha=0.5,
               linestyles=['--', '-', '--'])

    # plot support vectors
    if plot_support:
        ax.scatter(model.support_vectors_[:, 0],
                   model.support_vectors_[:, 1],
                   s=300, linewidth=1, edgecolors='black',
                   facecolors='none');
    ax.set_xlim(xlim)
    ax.set_ylim(ylim)


plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='autumn')
plot_svc_decision_function(model);
plt.show()
model.support_vectors_
```



3. Yêu cầu:

- Áp dụng với tập dữ liệu banknote authentication từ UCI Machine Learning Repository. Kiểm tra banknote là tuyến tính hay không tuyến tính rồi áp dụng tương tự trường hợp tương ứng như trên.
- Viết file báo cáo.